A Deep Learning Pipeline Using Few Shot Learning Approaches for Retail Product Recognition

Arun Naranayan
an572@cornell.edu

Karan Yang
ky393@cornell.edu

Hortense Gimonet
hg332@cornell.edu

Franck Kengne
fmk28@cornell.edu

Philippe El Asmar
pye4@cornell.edu

Camile El Dahdah
aae57@cornell.edu

Abstract

In this paper, we test the performance and fit of Prototypical Networks (ProtoNet), a low-shot image classification method proposed by Snell et al. [8] to a real world fine-grain dataset of grocery store products. We evaluated the performance of ProtoNet with different hyper parameters, embedders, and augmentation methods on the fine-grained Caltech-UCSD Birds Dataset (CUB) [11], a Grocery Store Dataset proposed by Klasson, et al. [4], as well as a combination of the two datasets. We found that Prototypical Networks combined with pre-trained ResNets and state-of-art auto augmentation techniques a promising approach to tackle few-shot image classification problems in the retail domain.

1 Introduction

In order for many existing image detection and segmentation pipelines to perform well they assume both abundant data in the form of numerous labeled examples and a set of coarse-grain object classes. Recently, the problem of learning new concepts from only a few labelled examples referred to as ’few-shot learning’ has received considerable attention. More concretely, K-shot N-way classification is the task of classifying a data point into one of N classes, when only K examples of each class are available to inform the decision. Generic few shot learning methods are mainly tested on coarse-grain datasets (Omniglot [5] , miniImageNet [7]) which do not extend well to real world product recognition scenarios particularity in the grocery domain. Given the fine-grain nature of items sold in retail stores (the average grocery store has 40M distinct items alone), annotating numerous instances of raw item image data can be costly, time consuming and is nearly impossible. In this paper we aim to extend and improve the performance of a popular few shot learning approach proposed by Snell et al. called "Prototypical Networks" [8] to a real world fine-grain dataset. Initially we aimed to test our amended few shot learning pipeline on our own custom fine-grain NYC grocery store dataset. However, due to limitations from COVID-19 as a proxy we have combined an existing coarse-grain grocery dataset with an existing standard fine-grain dataset. We believe that our combined dataset successfully mimics the large inter-product category differences and small but fine-grain intra-product category differences that are inherent in the diverse array of objects that exist on store shelves. Our larger aim is to use what we learn in this paper as a first step to enable the creation of a robust, low cost, commercial inventory management system for small brick and mortar retailers.

1.1 Business Problem Overview

With online shopping rapidly gaining popularity, many industry experts have decried the death of traditional “brick and mortar” storefronts. Yet, as the US census bureau reports, e-commerce sales in

Preprint. Work in progress.

the fourth quarter of 2019 accounted for just 11.4 percent of total sales. A closer look reveals that small retailers [6] (those having less than 50 employees, represent 98 percent of total retail firms and nearly 40 percent of all retail workers) are remaining competitive by creatively engaging and listing products online to bring customers into their stores. A key necessary condition for successful online listing is the real-time tracking and management of inventory for sale. While larger big box brick and mortars and e-commerce companies have access to sophisticated proprietary inventory management (IM) systems; smaller retailers do not. Many retailers still go through a labor intensive process of manual inventory counts in store fronts and warehouses. This activity can cripple store operations for days and can be detrimental for small retailers in the face of intensifying online competition. For small retailers, there exist two main problems related to online listing:

The long tail of stock keeping units (SKU) Recognition of products in store shelves poses distinct challenges. The task mandates the recognition of an extremely high number of different items with many of them featuring small intra and large inter class variability. This specificity is important as customers care about specific brands and flavors. For instance, “Orange Juice” can be decomposed into many unique brands and permutations i.e. pulp / no pulp each with a distinct SKU, as illustrated in Figure 1. Any candidate IM algorithm would need to be robust enough to correctly classify each distinct sku with the limitation that this specificity leads to sparse or less data i.e. the dataset changes from having 1 class with three samples to 3 classes with one sample each. In practice, product databases usually include just one or a few studio quality images as well.

Updated Availability Products that are only available for sale (i.e. on a shelf) need to be counted and aggregated in order for them to be listed online. IM algorithms need to be sophisticated enough to recognize multiple instances of the same fine grain class in spite of real world challenges relating to occlusion and lighting changes.

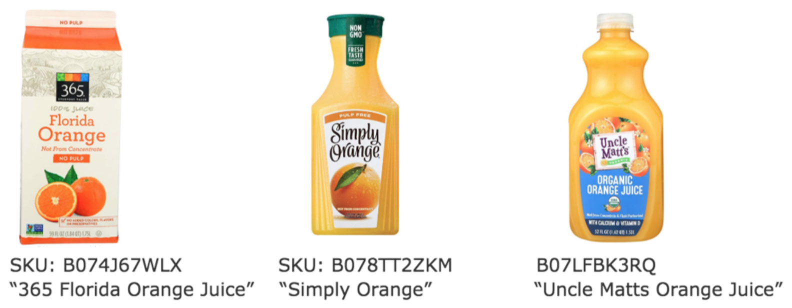


Figure 1: Orange Juice SKU variety

1.2 Scope of Solution

Deep convolutional neural networks (CNNs) have produced strong results in visual recognition tasks such as object recognition and scene classification. A CNN learns to recognize a large quantity of visual categories by training on a large collection of annotated images using a gradient-descent technique. Yet, even after a long training period, the model is limited in that it can only recognize a fixed set of classes. To learn to recognize novel categories, one has to collect many instances of new training data and re-train the CNN model with further adjustments. Unfortunately, as in the context of the retail industry there is often not enough quality labeled data available and annotating fine grain novel categories is expensive for training.

Our solution addresses the first retail product recognition challenge "long tail of stock keeping units" described above using few-shot learning approaches. Few-shot or Low-shot learning is based on the concept that reliable algorithms can be created to make predictions from minimalist datasets which conform well to information available within existing commercial product databases, i.e. at most a few high-quality images per SKU.

Although there has been work adapting various few shot learning techniques to different domains with coarse-grain classes we take a novel approach by focusing our attention on extending few shot learning to a complex real world fine-grain dataset.

1.3 Related Works

1.3.1 State of the Art for Grocery Image Classification

Being able to detect products in a grocery store aisle is invaluable, both for digitized shopping experiences like Amazon Go and for universal accessibility purposes such as assisting the visually impaired [12]. Classical methods for grocery image classification have been divided into two categories: macro-class classification and instance recognition. While macro-class classification simply aims to classify grocery products into broad categories such as Food/Bakery, instance recognition involves classifying product vary specifically into categories like Orange Juice or Toothpaste. Instance recognition is much more complex due to inter-class variance and intra-class similarity, and requires good fine-grained datasets [2].

1.3.2 Few Shot Image Classification

Low or few shot learning designates machine learning approaches in which the training dataset has very little supervised information for the target task [10]. In few shot classification, the classifier must be able to classify new classes it has not seen in training based off of only a few images. The naive approach, which is to simply retrain the model, is both costly and can cause severe over-fitting [8]. There exist 3 broad families of popular low-shot learning approaches: distance metric learning, model-agnostic meta-learning, and semantic-based learning. In this paper, we focus on Snell et al.'s distance metric learning approach, Prototypical Networks [8].

Distance metric learning methods addresses the few-shot image classification problem by "learning to compare." This model attempts to classify an unseen image based on its similarity to labeled images in the training data set. During the training process, its meta-learning based methods leverage distances between images to learn sophisticated comparison models. In contrast to the metric learning approach, model-agnostic meta-learning approaches deal with data deficiency by learning how to learn [10]. This technique combines deep networks and probabilistic methods to handle ambiguity in few-shot classification problems. Semantic-based methods are fairly recent and rely on meta data in the form of category name, textual description or other attributes to make classifications decisions.

2 Method

2.1 Data

Initially we aimed to test our few shot learning pipeline on our own custom fine-grain store dataset taken from actual shelves in various NYC grocery stores. Images would have been taken via low resolution iphone cameras which would mimic the actual actions / technology available to store inventory clerks at query time. In addition, real world "in the wild" images would subject our training and test optimization to object occlusion and photometric variability. Most importantly, a custom real world store dataset would also mimic the peculiar distribution of products that exist on store shelves. In particular, objects have large inter-product category differences but small fine-grain intra-product category differences i.e. in general the difference between orange juice vs. cereal is greater than the difference between orange juice vs. 1000 other orange juice brands. Due to limitations from COVID-19 as a proxy we have combined an existing coarse-grain grocery dataset with a widely used fine-grain dataset. We believe that our combined dataset successfully mimics the real life distribution which combine instances of coarse and fine-grain classes as we describe below:

Caltech-UCSD Birds-200-2011 The Caltech-UCSD Birds Dataset (CUB) [11] is a widely used dataset for fine-grain classification. The CUB dataset contains 200 sub-classes of birds with approximately 60 samples per subclass. It is easy for a classifier to recognize basic-level categories such bird, flower, and car but harder to recognize 200 or more subcategories. The sub-categories are generally the same in global appearance and distinguished by subtle and local differences such color, shape, texture. This extends well to the fine-grain nature of intra-class SKU variation within classes of grocery products. As a first step to extend the Protoptyical Network architecture to fine-grain classes we ran preliminary experiments on the CUB dataset.

A Hierarchical Grocery Store Image Dataset We used a Grocery Store Dataset used by Klasson, et al. [4]. The dataset contains natural images of grocery items taken with a smartphone camera in different grocery stores. In total the dataset contains 2,640 images from 44 different classes of fruits, vegetables, and carton items (e.g. juice, milk, yoghurt). Of the 44 classes, 12 can be further decomposed into 50 fine-grain classes where for example the fine-grained classes 'Royal Gala' and 'Granny Smith' belong to the same coarse-grained class 'Apple'.

Combined Dataset (1+2) Since the Grocery Store Dataset contains only a few fine-grain classes mixed with coarse grain classes, we experimented combining a more fine-grain dataset to our domain specific grocery dataset. The goal was to train the embedder to be more robust to real-life scenarios where product catalogues can have extremely long fine-grain class tails for certain product categories versus others i.e. chocolate store that contains 200 different types of chocolate but also sells other items such as water / fruit at checkout. Although the combined datasets span different domains (grocery store products vs. birds) we believe the differences between milk vs. lime and milk vs. bird are "similarly different" and mimic the large inter-product category differences that exist in retail stores. Given data collection limitations we felt this was a justifiable proxy and grounds for experimentation. We also experimented with taking a fine-grain subset of the Grocery Store Dataset (only 12 fine-grain classes) and combining that with the CUB dataset.

We ran experiments on each dataset separately but mainly report experiment results and takeaways on the Combined Dataset.

2.2 Performance Benchmarks

As summarized in appendix A, there has been considerable interest in few shot learning with some novel approaches achieving impressive results. These approaches use the miniImageNet and Omniglot datasets as baselines which do not extend well to real world product recognition scenarios and as such do not provide logical benchmark for our problem.

Given the fine-grain nature and novelty of our dataset profile (large inter-product category differences and small but fine-grain intra-product category differences) we have decided to create our own performance benchmarks.

Upper-Bound We trained a CNN using Resnet-18 architecture from scratch and achieved 83% top-1 accuracy on our combined dataset as an upper-bound benchmark. Although state of the art CNNs have achieved top 1 accuracy of approximately 90% on coarse-grain ImageNet 1000 and approximately 80% accuracy on the fine-grain CUB bird data set, we believe our combined dataset upper-bound to be a more appropriate baseline for our problem given that it includes a more domain focused dataset. For the above upper-bound we do not place 5-shot data limitation constraints for each class. Our logic is that a theoretical upper-bound for a few shot architecture would be for the model to perform just as well as a CNN with no data limitations for the same dataset.

Lower-Bound Similarly, as a lower bound benchmark we trained a CNN using VGG16 architecture from scratch and achieved 6% top-1 accuracy on our combined dataset. For the lower-bound we placed a 5-shot data limitation constraint for each class i.e each class had only 5 images in the training set. We felt this was an appropriate lower bound benchmark for few shot approaches as even if only 5 samples were available for a classification task a CNN with limited data could be used a naive last resort.

Snell Prototypical Network Architecture As an additional baseline we ran the original Snell Prototypical architecture used for the Omniglot dataset with no changes on our combined dataset and achieved a 35%-20 way test accuracy (56%-5 way test accuracy).

Our aim in this paper is to attempt to achieve better few shot classification performance for our combined dataset with Prototypical Networks within our upper and lower bounds.

2.3 Model

Our model replicates the Snell Prototypical Network approach [8] with some key changes we describe below. The goal of the model is to learn a feature extractor function from a base set that is then applied to a novel data set of unseen images (not present in the base set). The training and test set are two disjoint set of classes. Prototypical Networks are based on the idea that each class can be represented by the mean of its examples in a representation space learned by a neural network [8]. The approach is based on the idea that there exists an embedding in which points cluster around a single "prototype" representation for each class.

We mimic the Snell Prototypical Network procedure of ascertaining a non-linear mapping of the input (images) into an embedding space using a neural network, then using the embedding function to classify a point by simply finding the nearest class prototype. Items in a store and the initial character images the Snell Prototypical Network were trained on share few characteristics, and the number of common features which can be exploited to map the image on the corresponding metric space is negligible. As such we decided to re-train our embedding function from scratch on the datasets we described above.

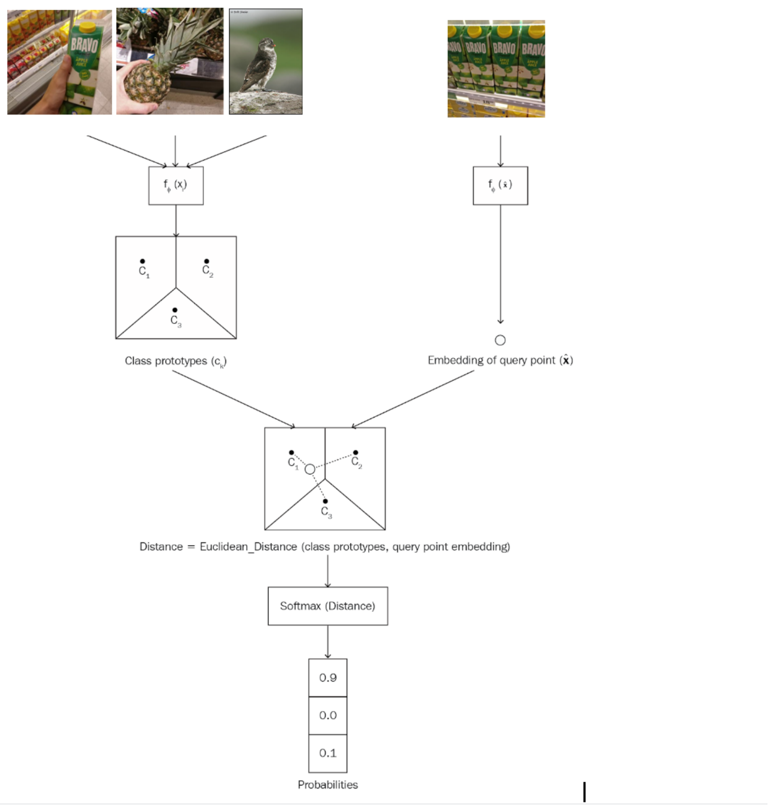


Figure 2: The overall flow of the Prototypical Network with our sample images

We pre-processed the raw images in our dataset by resizing, using custom data augmentation and auto-augmentation to generate batches of tensor image data. Initially, our embedding architecture mirrored that used by Vinyals et al. [9] and Snell et al. [8] and was composed of four convolutional blocks (see appendix B). Each block comprised a 64-filter 3×3 convolution, batch normalization layer, a ReLU nonlinearity and a 2×2 max-pooling layer. After all the blocks, the remaining output is flattened and returned as a result. Since the Prototypical Network few shot procedure is independent of the actual embedding architecture, we choose an embedding architecture that best fit our dataset. We experimented with Conv-4, Conv-5 (one more CNN block), a ResNet-18 trained from scratch and finally chose an architecture that used a pre-trained ResNet-18 [3].

3 Experiments & Results

3.1 Varying 5 Shot N Ways Test/Train and Hyperparamters

The authors of Prototypical Networks observed that training the embedder on higher N-way classes produced better results at test time. We hypothesize that this is the case because the increased difficulty of higher N-way classification helps the network to generalize better. Conversely, in our experiments we observed the opposite and assume that there may be a point where higher N-way past 20 forces the model to maximize fine-grained decisions in the embedding space and ironically make it less able to generalize (see appendix C). We also observed that training the model for longer than 8000 training episodes (approximately 65% train accuracy) causes it to over fit the training data causing poor test performance. As we expected due to the increased difficulty and fine-grain nature of the test set increasing N-way classes decreases classifier performance at test time.

3.2 Architecture Changes

Since the Prototypical Network few shot procedure is independent of the actual embedding architecture, we experimented with various embedding architectures that best fit our dataset. We first tried modifying the number of CNN blocks and layers in Snell et al. initial architecture to get better performance. After testing multiple times, having five convolution blocks led to the best 20 way testing accuracy of 37% (5-way test accuracy of 63%). We then trained a ResNet-18 model from scratch, but due to the low image resolution of our initial 84*84 image size we did not achieve better results. We also used a pre-trained ResNet-18 model but due to CUDA memory constraints, we were only able to to resize our images to 164*164. While not ideal, this was still relatively close to the image size that ResNet-18 was trained on and our model achieved 73% top 1 20-way test accuracy (85% for 5-way).

We also tested multiple distance functions such as pairwise distance and cosine similarity. However, the cosine similarity performed poorly with only 11% accuracy (5-way accuracy of 33%). The pairwise similarity performed best.

3.3 Data Augmentation & Pre-processing

Custom data augmentation We tried various custom data augmentation strategies to increase top 1, 5-shot accuracy. First, we applied transforms by rotating, resizing, cropping and randomly flipping each training image. We hoped this would force the embedder to find the salient features that differentiate between classes and make it more robust at test time. The added generalizaibility from these transforms increased test accuracy. Second, we attempted to increase the size of our training set by applying the same augmentation methods on a copy of the training set, increasing the size of the test set by 2. This increased our average 5 shot 20-way accuracy to 37% (5-way accuracy of 64%).

Transfer Auto Augmentation We also explored ways of transferring beneficial data augmentation techniques from other real world image datasets analogous to transferring weights from pre-trained models using transfer learning. We searched for image auto-augment policies (optimally searched data augmentation procedures) that were suitable for our specific dataset and used the ImageNet Auto-Augment Policy [1]with the following procedure:

- Resize input image size is 164x164
- Pre-trained ResNet-18 weights
- The ImageNet Auto-Augment policy is applied after center cropping.
- 5-shot (support images) and 10-query images for both training and testing stages.
- With these chosen hyperparameters the final model is trained on the combination of training and validation set and tested on the test set

As we can see from the sample outputs generated by the augmentation in Figure 3.3, geometric transformation, rotations, and solarization are commonly used data augmentation techniques used in the ImageNet Auto-Augment policy.

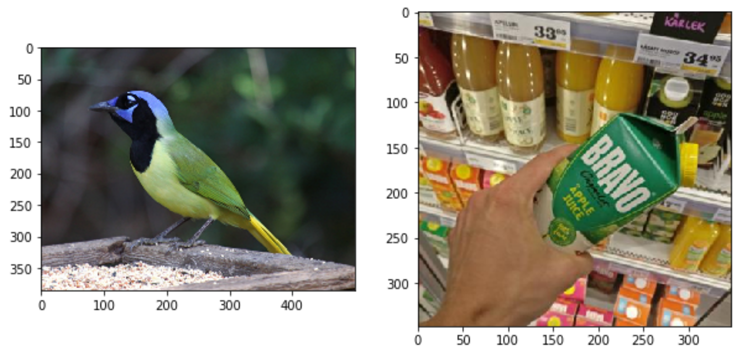


Figure 3: Original Image Samples

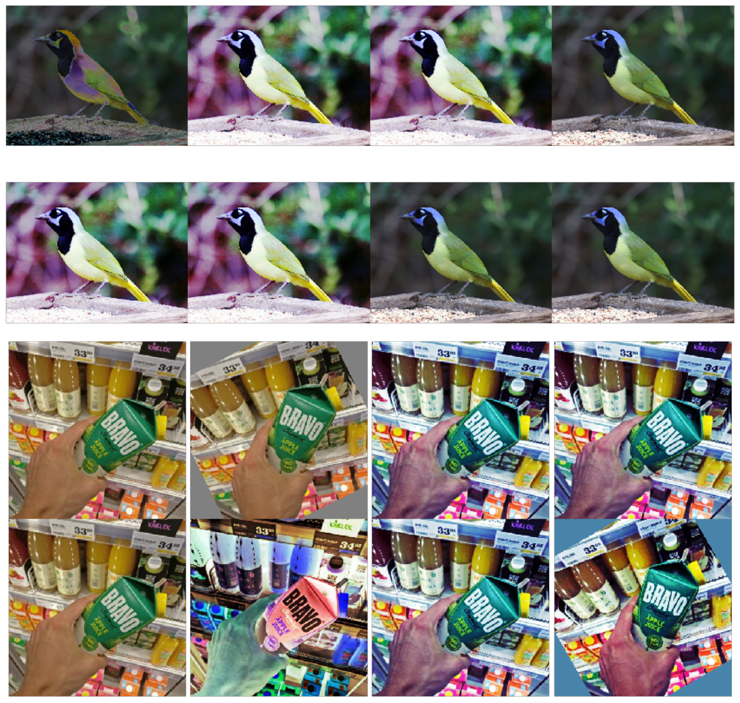


Figure 4: Output samples using ImageNet augmentation policy

Average test accuracy results with and without ImageNet Auto-Augment policy are shown in table 1. We observed that training on a higher N way than test helps to improve test accuracy with a ResNet-18 embedding function. Although the increased accuracy of combining ImageNet Auto-Augment policies with a pre-trained Resnet-18 provide similar accuracy results in the 20-way case, the combination did achieve higher (5% more) test accuracy in the 5-way case. Further experiments needs to be done to understand why this is the case.

	ImageNet Augmentation		No Augmentation
	Trained 20-way*	Trained 30-way*	Trained 20-way*
Tested 5-way*	90.23%	91.36%	85.00%
Tested 20-way*	72.87%	74.45%	73.17%

* By n-way, we mean that n images were provided to create each class prototype.

Table 1: Test Accuracy results using pretrained ResNet-18

3.4 Hyperparameters

For our two best performing models (Resnet-18 with and without ImageNet Auto-Augment), we tuned our hyperparameters based on performance on a separate validation set. We used Adam optimizer with a learning rate scheduler. We set the initial learning rate to be 1e-3 and it decreased by gamma=0.5 per 20 steps.

4 Conclusion & Future Work

While Prototypical Networks reflect a simpler inductive bias and achieve excellent results (approximately 98% top 1 20-way accuracy) on the simple coarse-grain Omniglot dataset, we have found that it does not achieve the same performance on our combined dataset. Our final top 1 20-way test accuracy of 74.5%, compares well to few shot performance on miniImageNet (see appendix A). In spite of the fact that our dataset is inherently more complicated than miniImageNet combining fine-grain and course grain classes with real world natural images. Through our numerous experiments we modified Snell et al.'s original paper architecture and improved accuracy by 42.5%. As expected, our results generally fall between our upper and lower bound benchmarks.

One main learning is that although Prototypical Networks successfully classify classes it has only seen 5 times before, it is still very sensitive to the underlying data distribution of the training and test classes. The goal of the algorithm is to learn a feature extractor function from a base set that is then applied to a novel data set of unseen images (not present in the base). We expected that Protypical Networks would only perform well when there is no domain shift between the base and novel classes but it is also important that the general feature distribution of the test and train sets need to be similar so that the extractor can exploit common features in both sets. In addition, the algorithm needs to run for numerous episodes (<8000) in order to appropriately sample most of the training data.

Main takeaways:

- Much effort in recent years has been put in finding better network architectures for image classification tasks, while data augmentation techniques remain largely the same; auto-augment [1] data augmentations show a lot of promise.
- Compared to vanilla CNNs, Prototypical Networks are less sensitive to class imbalances as the training algorithm always picks equal number of samples from k-classes.
- Choice of embedding function architecture is highly dependent on the dataset used and is very important for few shot performance.
- Number of classes Prototypical Networks can train is mainly limited by memory size. A single RTX 2070 can only handle up to 20 classes for 5 shot with 164x164 images for example.
- Pre-trained ResNet-18 was the most resilient to a few number of examples per class for our dataset. We speculate that this is due to the fact that the pre-trained feature extraction does not need to learn from a large amount of samples to be able to classify the query images using few shot support images.
- Combining ImageNet Auto-Augment policy with a pre-trained ResNet-18, we achieved our highest test accuracy of 90.23% for 5-way test/20-way train and 72.87% for 20-way test/20-way train. Without using ImageNet auto-augment policy, we achieved 85% test accuracy for 5-way test/20-way train and 73.17% for 20-way test/20-way train.

Our larger aim is to use what we have learnt in this paper as a first step to enable the creation of a robust, low cost, commercial inventory management system for small brick and mortar retailers. Any commercially viable IM algorithm should address the following performance concerns:

- Should perform well given a small corpus of annotated fine-grain images (<5).
- If data augmentation is unavoidable, it must be cheap.
- At test time recognition is performed by associates on the floor - query performance must be fast, cheap and robust with low resolution cameras on mobile devices.

- Items on sale in a store as well as their appearance change frequently over time (i.e. seasonal packages) and sometimes pictures contain only a portion of full products due to the nature of visibility on shelves.

While our paper attempts to evaluate Prototypical Networks for the first two challenges, we acknowledge that future work should also be focused on addressing challenges relating to low cost data augmentation, occlusion, image segmentation and mobile performance. In particular, investigating methods that marry cropping / drawing bounding boxes around images before inputting into a few shot learning pipeline could dramatically increase performance. Given our results, we also plan to explore how we can better use cheap data augmentation, use a fully custom NYC grocery dataset, train auto-augmentation policies from scratch, and experiment combining data augmentation with existing CNN architectures & meta-learning methods to increase performance.

References

[1] Ekin D. Cubuk. Autoaugment: Learning augmentation strategies from data. 2019.

[2] Weidong Geng, Feilin Han, Jiangke Lin, Liuyi Zhu, Jieming Bai, Suzhen Wang, Lin He, Qiang Xiao, and Zhangjiong Lai. Fine-grained grocery product recognition by one-shot learning. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, page 1706–1714, New York, NY, USA, 2018. Association for Computing Machinery.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[4] Marcus Klasson, Cheng Zhang, and Hedvig Kjellström. A hierarchical grocery store image dataset with visual and semantic labels. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 491–500. IEEE, 2019.

[5] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[6] PricewaterhouseCoopers LLP. The economic impact of the us retail industry, Sep 2014.

[7] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[8] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. 2017.

[9] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[10] YAQING Wang, J Kwok, LM Ni, and Q Yao. Generalizing from a few examples: A survey on few-shot learning. *arXiv preprint arXiv:1904.05046*, 2019.

[11] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[12] T. Winlock, E. Christiansen, and S. Belongie. Toward real-time grocery detection for the visually impaired. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 49–56, 2010.

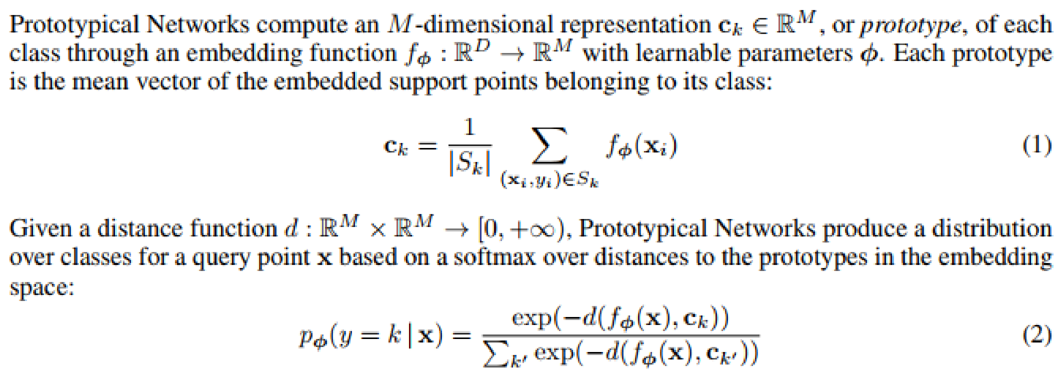
Appendix

A Comparison of low-shot models

Paper	1-shot	5-shot
Edge-Labeling Graph Neural Network for Few-shot Learning	NA	80.10
Task Agnostic Meta-Learning for Few-Shot Learning	49.40	66.00
Spot and Learn: A Maximum-Entropy Patch Sampler for Few-Shot Classification	51.03	67.96
Few-Shot Learning with Localization in Realistic Settings	51.10	69.45
Revisiting Local Descriptor based Image-to-Class Measure for Few-shot Learning	51.24	71.02
Few-shot Learning via Saliency-guided Hallucination of Samples	57.45	72.01
Image Deformation Meta-Networks for One-Shot Learning	59.14	74.63
Meta-Transfer Learning for Few-Shot Learning	61.20	75.50
Dense Classification and Implanting for Few-Shot Learning	61.26	79.01
Generating Classification Weights with GNN Denoising Autoencoders for Few-Shot	62.96	78.85
Meta-Learning with Differentiable Convex Optimization	64.10	80.00
Finding Task-Relevant Features for Few-Shot Learning by Category Traversal	64.12	80.51
Baby steps towards few-shot learning with multiple semantics	67.20	74.80

Table 2: Comparison of low-shot models

B "Image2Vector" model Architecture for Prototypical Network


Prototypical Networks compute an M-dimensional representation $\mathbf{c}_k \in \mathbb{R}^M$, or *prototype*, of each class through an embedding function $f_\phi : \mathbb{R}^D \to \mathbb{R}^M$ with learnable parameters ϕ. Each prototype is the mean vector of the embedded support points belonging to its class:

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\phi(\mathbf{x}_i) \tag{1}$$

Given a distance function $d : \mathbb{R}^M \times \mathbb{R}^M \to [0, +\infty)$, Prototypical Networks produce a distribution over classes for a query point $\mathbf{x}$ based on a softmax over distances to the prototypes in the embedding space:

$$p_\phi(y = k \mid \mathbf{x}) = \frac{\exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_{k'}))} \tag{2}$$

Figure 5: Prototypical Networks: Classification Decision

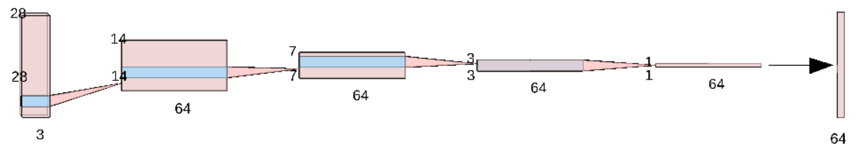
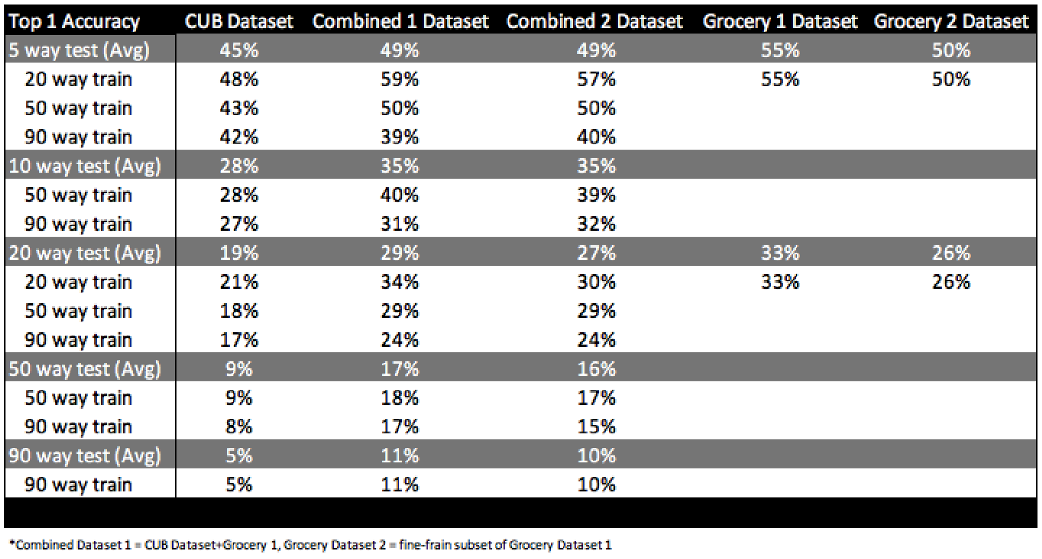


Figure 6: Embedding CNN architecture used in the paper by Snell et al

C Experiments



Top 1 Accuracy	CUB Dataset	Combined 1 Dataset	Combined 2 Dataset	Grocery 1 Dataset	Grocery 2 Dataset
5 way test (Avg)	45%	49%	49%	55%	50%
20 way train	48%	59%	57%	55%	50%
50 way train	43%	50%	50%		
90 way train	42%	39%	40%		
10 way test (Avg)	28%	35%	35%		
50 way train	28%	40%	39%		
90 way train	27%	31%	32%		
20 way test (Avg)	19%	29%	27%	33%	26%
20 way train	21%	34%	30%	33%	26%
50 way train	18%	29%	29%		
90 way train	17%	24%	24%		
50 way test (Avg)	9%	17%	16%		
50 way train	9%	18%	17%		
90 way train	8%	17%	15%		
90 way test (Avg)	5%	11%	10%		
90 way train	5%	11%	10%		

*Combined Dataset 1 = CUB Dataset+Grocery 1, Grocery Dataset 2 = fine-frain subset of Grocery Dataset 1

Table 3: Varying 5 Shot N Ways Test/Train and Hyperparameters

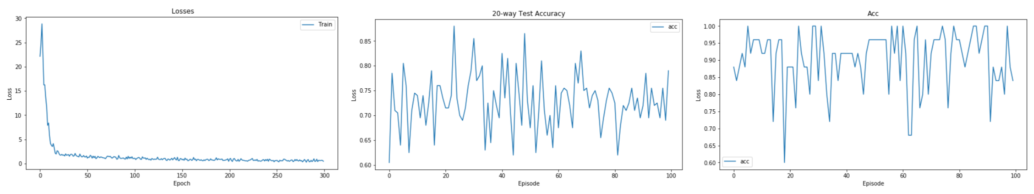


Figure 7: 20-Way Training Loss and Test Accuracy with ImageNet AutoAugment+ResNet-18

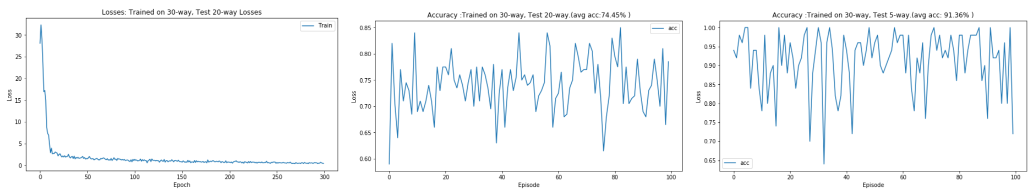


Figure 8: 30-Way Training Loss and Test Accuracy with ImageNet AutoAugment+ResNet-18