Prototypical Clustering Networks for Dermatological Image Classification

Viraj Prabhu [*,1] Anitha Kannan[3] Murali Ravuri[3] Manish Chablani[3]
David Sontag[2] Xavier Amatriain[3]
[1]Georgia Tech [2]MIT [3]Curai
virajp@gatech.edu dsontag@mit.edu {anitha, murali, manish, xavier}@curai.com

Abstract

We consider the problem of image classification for the purpose of aiding doctors in dermatological diagnosis. Dermatological diagnosis poses two major challenges for standard off-the-shelf techniques: First, the data distribution is typically extremely long tailed. Second, intra-class variability is often large. To address these issues, we formulate the problem as low-shot learning, where once deployed, a base classifier must rapidly generalize to diagnose novel conditions given very few labeled examples. To model diverse classes effectively, we propose Prototypical Clustering Networks (PCN), an extension to Prototypical Networks [22] that learns a mixture of prototypes for each class. Prototypes are initialized for each class via clustering and refined via an online update scheme. Classification is performed by measuring similarity to a weighted combination of prototypes within a class, where the weights are the inferred cluster responsibilities. We demonstrate the strengths of our approach in effective diagnosis on a realistic dataset of dermatological conditions. We extensively benchmark our approach and demonstrate gains over competing methods.

1. Introduction

Globally, skin disease is one of the most common human illnesses that affects 30% to 70% of individuals, with even higher rates in at-risk subpopulations where access to care is scarce [17, 2, 13, 10, 1]. Untreated or mistreated skin conditions often lead to detrimental effects including physical disability and death [1].

A large fraction of skin conditions are diagnosed and treated at the first point of contact, *i.e.* by primary care and general practitioners. While this makes access to care faster, recent studies indicate that general physicians, especially those with limited experience, may not be well-trained for diagnosing many skin conditions [5, 8]. In addition, people with no or little access to health care systems often depend on their own search and 'image recognition capabilities' to self (mis-)diagnose and treat. While there is a recent surge in online services and telemedicine for closing the gap of healthcare access, these services also have similar problems [20]. The need to find effective solutions to *aid* doctors in accurate diagnosis motivates this work.

Why is diagnosis of skin conditions hard for doctors? One important factor is the sheer number of dermatological conditions. The International Classification of Disease 10 (ICD 10) classification of human disease[1] enumerates more than 1000 skin or skin-related illnesses. However, most general physicians are trained on a few tens of common skin ailments under the assumption that this will enable accurate diagnoses in most cases. Recent studies indicate that this assumption may be flawed [27]. To make an accurate diagnosis, the knowledge of all possible diseases becomes important, especially to workup and eliminate possible life-threatening conditions. The difficulty of diagnosis is further compounded by the large intra-class variability within several conditions. To motivate the scale of this problem, see Figure 1, where we show the class distribution of Dermnet[2], a publicly available large-scale dataset of dermatological conditions. The plot shows examples illustrating the intra-class variability found in the dataset. This makes accurate diagnosis challenging even for experienced dermatologists.

These issues create an opportunity for incorporating automated machine learning systems as part of the doctor's workflow, aiding them in sieving through possible skin conditions. AI systems have shown promising results in many applications in computer vision (see *c.f.* [11],[12] and citations within) that have access to large balanced datasets with huge numbers of classes and significant intra-class variability. These advances have started to impact the healthcare domain, with early applications on automated classification of skin lesions using images [4] and diagnosis based on radiology data [15].

Inspired by these recent successes, this paper tackles the problem of fine-grained skin disease classification. We conjecture that a high fidelity AI system can serve as a dermato-

[*] Work done while V.P. was an intern at Curai.

[1] http://www.who.int/classifications/icd/en/

[2] http://www.dermnet.com/

logical diagnostic decision support system to general physicians. By suggesting candidate diagnoses, it can greatly reduce effort and compensate for the possible lack of experience or time at the point of care. In the context of teledermatology with a store-and-forward approach that involves asynchronous evaluation by dermatologists, such a system can aid in triaging the right doctor resource in a timely manner, especially when acute conditions need immediate care [8].

Learning a model for dermatological image classification poses two major challenges:

- Access to large amounts of data may not always be possible. As dermatology images are collected as part of Electronic Health Records (EHR), access is usually strictly controlled for privacy reasons. For a new healthcare platform that wants to build a dermatological classifier, starting with a small set of conditions and rapidly increasing the scope of predictable diagnoses is often the only practical alternative.

- The data distribution is invariably long tailed (see Figure 1[3]). Some skin conditions are rare and may not have many recorded examples. Others may be common but are so easily diagnosable that they are simply not recorded in EHR. In Figure 1, notice how common conditions such as flea bites and rarer diseases such as melanoma both end up in the tail of the dataset.

Despite these issues, we need robust mechanisms to make correct diagnoses. Our approach pursues the following objectives: First, the model needs to be able to handle the long-tail in the data and perform well on classes in both the head and the tail. Second, once deployed, it needs to be easily extensible to novel classes that it encounters given very few labeled examples (potentially labeled by a physician). With these objectives, we pose dermatological image classification as a few-shot learning problem. Our proposed model, that we call Prototypical Clustering Networks (PCN), extends prior work on Prototypical Networks [22] to represent a class as a mixture of prototypes instead of a single prototype. Training this classifier involves learning an embedding space while simultaneously learning to represent each class as a mixture of prototypes. Prototypes are initialized for each class via clustering and refined via an online update scheme. Classification is performed by measuring similarity to a weighted combination of prototypes within a class, where the weights are the inferred cluster responsibilities. The examples shown in Figure 1 are in fact, nearest neighbors to prototypes of the classes learned using the proposed approach. We extensively compare the performance of the algorithm to Prototypical Networks and other strong baselines on Dermnet.

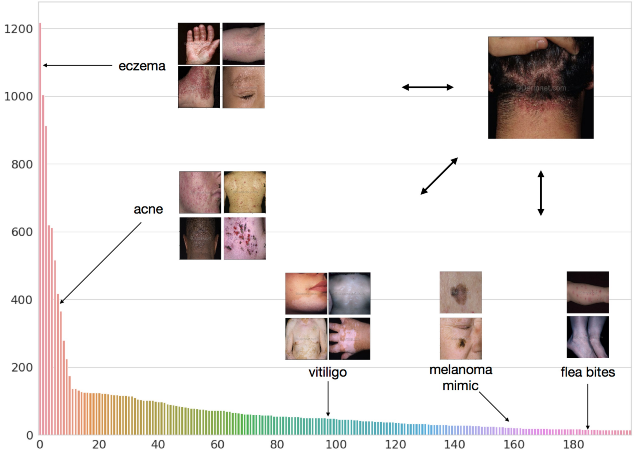

Figure 1: Long-tailed class distribution of Dermnet (shown here for the top-200 classes). Also shown are nearest neighbors to four of the many prototypes learned for select classes using the proposed Prototypical Clustering Network approach. This is illustrative of the huge intra-class variability in the data. For a novel test image, shown at the right upper corner, the model predicts the correct class by measuring weighted similarity to per-class clusters in the embedding space learned through a deep convolutional neural network.

2. Related Work

Dermatological Classification. A few prior works address the problem of dermatological classification. In [4], authors focus specifically on diagnosing skin cancer, and establish a benchmark on a large closed-source dataset of skin lesions by finetuning a pretrained deep convolutional neural network (CNN). In [16], authors study the problem of skin disease diagnosis on the Dermnet dataset but focus on coarse 23-way classification. In this work, we study the problem of fine-grained recognition of skin conditions on the Dermnet dataset in a few-shot setup, and propose a method to model multimodal classes and generalize effectively to unseen novel classes with very little data.

Class-imbalanced datasets. Real world datasets typically possess long tails [24, 26, 29], and learning robust CNN representations from such data is a topic of active research. Conventional training methods typically lead to poor generalization on tail classes as class-prior statistics are skewed towards the head of the distribution. Simple techniques such as random oversampling (or undersampling) by repeating (or removing) tail instances are found to help mitigate this issue to a degree [3]. Alternative approaches include [26], that proposes a meta learning algorithm to transfer knowledge from data-rich head classes to the tail. In this work, we propose a few-shot learning approach on a real-world imbalanced dataset of dermatological conditions, and demonstrate strong generalization capabilities even in the presence of very few training examples.

[3] Note that we have only plotted the 200 largest classes.

Few-shot learning. Few-shot learning aims to learn good class representations given very few training examples [22, 25, 23, 9, 21]. Main paradigms of approaches include simulating data starved environments at training time, and including non-parametric structures in the model as regularizers. Matching networks[25] learn an attention mechanism over support set labels to predict query set labels for novel classes. Prototypical networks[22] jointly learn an embedding and centroid representations (as class *prototypes*), that are used to classify novel examples based on euclidean distance. In both [25] and [22], embeddings are learned end-to-end and training employs episodic sampling. In an incremental few-shot learning context, [19] propose a class-centroid based representation in an embedding space learned using a generative model. In [9], authors study few-shot learning on an imbalanced dataset, treating tail classes as novel, and propose a method to "hallucinate" additional samples for such data-starved classes. In this work, we focus on a similar setup on the real-world long-tailed Dermnet dataset. We propose an extension to [22] to model the multimodal nature of diverse classes, and demonstrate how this also helps generalize better to data-starved novel classes.

Prototypical Networks. Prior extensions to protoyipical networks exist in the literature, and here we distinguish our contributions [23, 6]. [23] propose extending prototypical networks to a semi-supervised setting by using unlabeled examples while producing prototypes. [6] propose additionally predicting a covariance estimate for each embedding and using a direction and class dependent distance metric instead of Euclidean distance. In this work, we extend prototypical networks to model multimodal classes in an automated diagnostic setting by learning multiple prototypes per class, that are initialized via clustering and refined via an "online" update scheme.

3. Approach

We formulate dermatological image classification as a low-shot learning problem. During training time, we have access to a labeled dataset corresponding to dermatological examples $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ where each x_i is an observation and $y_i \in \{1, ..., K_{base}\}$ is the label that assigns the observation to one of the *base* classes known at training time. At test time, we are also provided with a small labeled dataset corresponding to K_{novel} novel classes, (i.e) classes distinct from K_{base} classes. The objective is to learn a classifier that can predict labels for examples that may come from a combined $K_{base} + K_{novel}$ label space.

3.1. Model

Prototypical Clustering Networks (PCN) builds upon recent work in Prototypical Networks [22]. PCN represents each class using a set of prototypical representations learned from the data. Let $\{\mu_{z,k}\}_{z=1}^{M_k}$ be the collection of M_k prototypes for class k. Then, at test time, we measure similarity to these representations to derive its corresponding class label. In particular,

$$p(y = k|\mathbf{x}; \phi) = \frac{\exp(-\sum_z q(z|k, x)d(f_\phi(x), \mu_{z,k}))}{\sum_{k'} \exp(-\sum_{z'} q(z'|k', x)d(f_\phi(x), \mu_{z',k'}))} \tag{1}$$

where $f_\phi(x)$ is the embedding function with learnable parameters ϕ that maps input x to a learned representation space, d is a distance function and $q(z|k, x)$ (eq. 3) is soft assignment of examples to clusters from the class. Note that when $M_k = 1$ for all classes, we revert to standard prototypical networks.

3.1.1 Model training

The goal is to learn a model with parameters θ so as to maximize the likelihood of the correct class:

$$\phi^* = \arg\max_\phi \sum_{(\mathbf{x},y)} \log p(y|\mathbf{x}; \phi), \tag{2}$$

and minimize its corresponding loss function L_ϕ. We use episodic training [22, 25, 18] to learn the embedding function by optimizing the loss and updating the cluster prototypes for each class. In particular, a training epoch consists of E episodes. Algo. 1 provides the details of computing the loss for one episode that is used in learning the function. We describe some key components of the algorithm below:

Class-specific cluster responsibilities: The assignment of an example within each class is given by:

$$q(z|k, x) = \frac{\exp(-d(f_\phi(x), \mu_{z,k})/\tau)}{\sum_{z'} \exp(-d(f_\phi(x), \mu_{z',k})/\tau)}, \tag{3}$$

where τ is temperature parameter that controls the variance of the distribution. As we decrease the temperature, the distribution becomes more peaky, and becomes flatter as we increase it.

This has implications in model training; The importance of τ can be understood by studying the loss function L_ϕ in line 15 of Algo. 1. During training, if clusters are well-separated, $q(z|k, x)$ will be peaky so that each example effectively contributes to the update of a single cluster in a class. In contrast, if clusters are overlapping then $q(z|k, x)$ will be diffused and the corresponding example will contribute to multiple prototypes through the loss function. Therefore, during training, we typically set τ to favor peaky distributions so that clusters are learned to focus on different regions of the input space.

Class-specific cluster prototypes: In episodic training, an epoch corresponds to a fixed number of episodes. In each

Algorithm 1 Training episode loss computation for Prototypical Clustering Networks. N is the number of examples in the training set, K_{base} is the number of base classes for training, M_k is the number of clusters for class k, $N_C \leq K_{base}$ is the number of classes per episode, N_S is the number of support examples per class, N_Q is the number of query examples per class. RANDOMSAMPLE(S, N) denotes a set of N elements chosen uniformly at random from set S, without replacement. *Differences from Algorithm 1 in [22] in blue*

```
Input:Training set $\mathcal{D} = \{(x_1, y_1), \cdots, (x_N, y_N)\}$, where each $y_i \in \{1, \cdots, K\}$. $\mathcal{D}_k$ denotes the subset of $\mathcal{D}$ containing
    all class prototypes, i.e. elements $(x_i, y_i) = \{\mu_{z,k}\}_{z=1}^{M_k} \forall k \in \{1, \cdots, K\}$
Output: The loss J for a randomly generated training episode

V ← RANDOMSAMPLE($\{1, \cdots, K\}$, $N_C$)                                          ▷ Select class indices for episode
for $k \in \{1, \cdots, N_C\}$ do
    $S_k \leftarrow$ RANDOMSAMPLE($\mathcal{D}_{v_k}$, $N_S$)                                   ▷ Select support examples
    $Q_k \leftarrow$ RANDOMSAMPLE($\mathcal{D}_{v_k \setminus S_k}$, $N_Q$)                               ▷ Select query examples
    for $(x, y) \in S_k$ do                            ▷ Compute probabilistic assignment of x to y's clusters
        $q(z|k, x) = \frac{\exp(-d(f_\phi(x), \mu_{z,k})/\tau)}{\sum_{z'} \exp(-d(f_\phi(x), \mu_{z',k})/\tau)}$
    for $z \in \{1, \cdots, M_k\}$ do
        $\mu_{z,k}^{new} \leftarrow \alpha \mu_{z,k}^{old} + (1 - \alpha) \frac{\sum_{(x,y) \in S_k} q(z|k,x) f_\phi(x)}{\sum_{(x,y) \in S_k} q(z|k,x)}$
$L_\phi \leftarrow 0$
for $k \in \{1, \cdots, N_C\}$ do
    for $(x, y) \in Q_k$ do
        $L_\phi \leftarrow L_\phi + \frac{1}{N_C N_Q} \left[ \sum_z q(z|k,x) d(f_\phi(x), \mu_{z,k}) + \log \sum_{k'} \exp(-\sum_{z'} q(z'|k',x) d(f_\phi(x), \mu_{z',k'})) \right]$
```

episode, classes are sampled uniformly, which makes an inherent assumption that classes are balanced. In our setting, with huge class imbalances this translates to examples from tail classes being oversampled, while examples from the head may be undersampled within an epoch. This can adversely affect subsequent model training. One approach to mitigate this is to sample classes according to class priors, which has the disadvantage of tail classes being poorly represented. Instead, we take the approach of refreshing the prototypes. In particular, at the start of an epoch, for every class, the cluster prototypes are initialized using k-means on the learned embedding representation of examples from that class.

Subsequently, in each episode, we use an *online* update scheme that balances between the local estimate of the prototype computed from embeddings of the current support set, and the prototypes learned so far:

$$\mu_{z,k}^{new} \leftarrow \alpha \mu_{z,k}^{old} + (1 - \alpha) \frac{\sum_{(x,y) \in S_k} q(z|k, x) f_\phi(x)}{\sum_{(x,y) \in S_k} q(z|k, x)}, \tag{4}$$

where α balances the episodic memory, using trade-off between memory from previous episodes and its current estimate. In particular, when $\alpha = 0$, the prototypes are memory-less while with $\alpha = 0.5$, current estimate of prototypes are as important as their previous values.

3.2. Understanding the role of multiple clusters

We can derive insights about the role of multiple clusters by interpreting PCN also as a linear model ([22]), but with data dependent factors. Using Euclidean distance in eqn. 1, we expand the term in the exponent so that:

$$\begin{aligned} & -\sum_z q(z|k, x) ||f_\phi(x) - \mu_{z,k}||^2 \\ & = -\sum_z q(z|k, x) f_\phi(x)^T f_\phi(x) - \sum_z q(z|k, x) \mu_{z,c}^T \mu_{z,k} \quad (5) \\ & \quad + 2 \sum_z q(z|k, x) f_\phi(x)^T \mu_{z,k} \\ & = \text{constant for k} + w_{k,x}^T f_\phi(x) - b_k \quad (6) \end{aligned}$$

$$\text{where} \qquad w_{k,x} = 2 \sum_z q(z|k, x) \mu_{z,k} \tag{7}$$

$$b_{k,x} = \sum_z q(z|k, x) \mu_{z,k}^T \mu_{z,k} \tag{8}$$

We can see from eqn. 6 that PCN also has a linear form as in prototypical networks. While the model has a linear form, the non-linearity required is captured through the embedding using the neural network.

The functional forms of the linear factors, namely $w_{k,x}$ and $b_{k,x}$, also sheds light on the advantage of using multiple clusters per class. In particular, unlike in prototypical networks, $w_{k,x}$ is an *example-specific* "prototypical" representation for class k, obtained by using a convex combination of all prototypes for the class, weighted by posterior probability over the cluster assignments within the class. When $q(z|k, x)$ is confident with a peaky posterior, the model behaves like a regular prototypical network. In contrast, when the posterior has uncertainty, PCN interpolates between the prototypes by modulating through $q(z|k, x)$.

4. Results

4.1. Experimental setup

Dataset: We construct our dataset from the Dermnet Skin Disease Atlas[4], one of the largest public photo dermatology sources containing over 23,000 images of dermatological conditions. Images are annotated at a two level hierarchy – a coarse top-level containing parent 23 categories, and a fine-grained bottom-level containing more than 600 skin conditions. We focus on the more challenging bottom-level hierarchy for our experiments. First, we remove duplicates from the dataset based on name, and also based on collisions found using perceptual image hashing [28].

Figure 1 presents a histogram of the resulting class distribution, filtered to the top-200 classes. We can see that the dataset has a long tail with only the 100 largest classes having more than 50 images; beyond 200 classes, the number of images reduces around tens and with 300 classes in single digits. Unless otherwise stated, for experimental comparisons, we focus on the top-200 classes so that $K_{base+novel} = 200$, which contains 15507 images. Similar to [9], we treat the largest 150 classes as base classes ($K_{base} = 150$) and the remaining 50 classes as novel ($K_{novel} = 50$). This helps in ensuring reasonably sized splits for training, validation and evaluation. In particular, we sample $max(5, 20\%)$ without replacement for each base class to get validation and test splits (3163 images each). The remaining is used for training (9181 images). For the low-shot learning phase, following the procedure used in [9], we sample 5 examples each for training and testing, respectively. We report mean and standard deviation of metrics over 10 cross validation runs.

Metrics: We report mean of per-class accuracy (mca), treating each class as equally important. For a dataset consisting of C classes, with T_c examples in each class, mean accuracy is the average of per-class accuracies:

$$\text{mca} = \frac{1}{C}\sum_{c}\frac{\sum_{t=1}^{T_c}[\hat{y}^{(t)}[0] = y^{(t)}]}{T_k}, \tag{9}$$

where, for t^{th} example, $\hat{y}^{(t)}[j]$ is the j^{th} top class predicted from a model and $y^{(t)}$ is its corresponding ground truth label. $[a = b]$ is the Iverson notation that evaluates to one only if a=b or else to zero.

We use $\text{mca}_{\text{base+novel}}$ to report combined mca performance of examples from all classes. mca_{base} corresponds to evaluation of classifier on $K_{base+novel}$ classes but restricted to only test examples from base classes. Similarly, $\text{mca}_{\text{novel}}$ corresponds to evaluation of test examples in novel classes while performing $K_{base+novel}$ way classification.

We also report recall@k (k $\in \{5, 10\}$). Over a test set of size T:

$$\text{recall@k} = \frac{\sum_{t=1}^{T}\sum_{j=1}^{j=K}[\hat{y}^{(t)}[j] = y^{(t)}]}{T}, \tag{10}$$

This metric is valuable in deployment contexts that involve aiding doctors in diagnosis. As a matter of fact, it is equivalent to the medical measure of *sensitivity*. This metric is not as strict as mca but it ensures that the relevant disease condition is considered within a small range of false positives.

Baselines

- Prototypical Network (PN): We train a prototypical network [22] using the training set from base classes. We use episodic batching with 10-way 10-shot classification at train time, while computing the prototypes using the entire training set, at validation time. The model is then used to adapt to do $K_{base+novel}$ way classification involving both base and novel classes.
- Finetuned Resnet with nearest neighbor (FT-*NN): Here, we finetune an ImageNet-pretrained ResNet-v2 convolutional neural network with 50 layers [11] on training data from K_{base} classes. The model is trained as a softmax classifier with a standard cross entropy objective. Then, we obtain embeddings for the entire training set consisting of $K_{base+novel}$ classes. This is used to perform $*$-nearest neighbor classification on test set from all of $K_{base+novel}$ classes
- Finetuned ResNet (FT-CE): This is the same ResNet model as the above but trained for $K_{base+novel}$ way classification using training data from both base and novel classes, and validated using the corresponding validation set on the base classes (due to lack of data in novel classes). We train the model with class balancing. This is a strong baseline as we use all $K_{base+novel}$ during training, and also due to class balancing, which has shown to improve generalization [3]

Hyperparameters: Unless otherwise reported, all experiments with PCN use 10 clusters per class for base classes and 4 clusters per class for novel classes. We pick the number of clusters for base classes via grid search over a small range of values. For novel classes, we pick the cluster size corresponding to the smallest test support set we have. Following [9], low shot experiments are performed with train and test shots of 5 unless reported otherwise. We use 200

[4] http://www.dermnet.com/

Approach	$n=5$ mca$_{\text{base+novel}}$	 mca$_{\text{base}}$	 mca$_{\text{novel}}$	$n=10$ mca$_{\text{base+novel}}$	 mca$_{\text{base}}$	 mca$_{\text{novel}}$
FT-1NN	46.18 +/- 0.81	55.32 +/- 0.30	18.76 +/- 3.30	49.51 +/- 0.34	54.86 +/- 0.50	33.44 +/- 1.35
FT-3NN	44.28 +/- 0.32	54.77 +/- 0.47	12.80 +/- 1.50	47.01 +/- 0.56	54.13 +/- 0.43	25.64 +/- 1.51
FT-CE	**47.82 +/- 0.46**	**55.75 +/- 0.71**	24.00 +/- 3.22	**51.51 +/- 0.41**	**55.21 +/- 0.26**	40.40 +/- 2.36
PN	43.92 +/- 0.40	48.71 +/- 0.37	29.56 +/- 2.35	44.93 +/- 0.79	47.55 +/- 0.37	37.08 +/- 3.39
PCN (ours)	**47.79 +/- 0.71**	53.70 +/- 0.18	**30.04 +/- 2.77**	**50.92 +/- 0.63**	51.38 +/- 0.34	**49.56 +/- 2.76**

Table 1: MCA on top 200 classes. FT-CE is trained on all 200 classes while the other models are trained with 150 classes. We report recall metrics in the appendix.

episodes per epoch for PCN and PN. The embedding function for PCN and PN produces 256-dimensional embeddings, and uses the same architecture as in FT-CE but with one less fully connected layer and without the last softmax layer projection. Models are trained with early stopping using Adam [14], with a learning rate of 10^{-4} and L2 weight decay of 10^{-5}.

4.2. Main results

Table 4 highlights our main results. The table shows test set MCA over the 200 classes available during test time for two different low shot settings: train shots of 5 and 10 with test set of 5. In both low shot settings, we observe the following trends:

- FT-CE and PCN shares similar performance on combined MCA. However, their performance on the base and novel classes are quite distinct. Much of the performance gains for FT-CE come from the base classes that have a lot more training examples than novel classes. In contrast, PCN, through episodic training aims at learning discriminative feature representations that are generalizable to novel classes with highly constrained number of examples; this is evident by its significantly better performance (9% absolute gains) in generalizing to novel classes. At the same time, PCN ensures that performance on novel classes doesn't come at the cost of lower accuracy on the base classes. Also note that the FT-CE model requires re-training for adding novel classes while PCN only requires a single forward pass to learn prototypes for novel classes.

- FT-*NN models learn robust representations for base classes, but are unable to generalize to novel classes, outperforming a regular PN model on top-200 MCA but underperforming against PCN. Interestingly, we find that increasing the number of nearest neighbors leads to poor performance, especially on novel classes. This could be due to sparsity of training data.

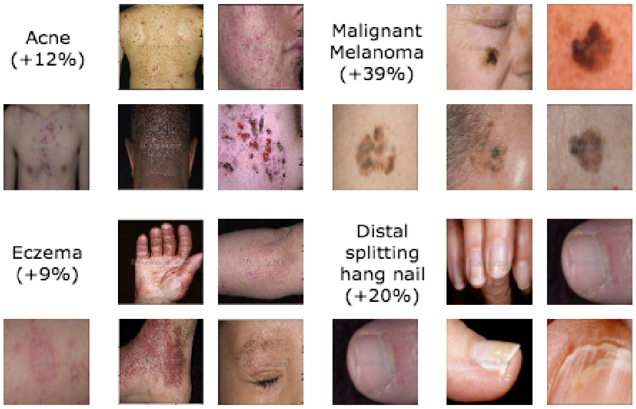

Figure 2: Learned prototypes are shown using their nearest neighbors in the training set. Each skin condition is in a 2×3 grid; The image below the name of the skin condition corresponds to PN while the 2×2 grid corresponds to nearest neighbors of four cluster prototypes. +X% below the name denotes improvement of PNC over PN for that class for mca$_{base+novel}$,. Note that novel classes such as 'Distal splitting hang nail' can also be diverse, as shown by clusters identified with PCN.

- PCN outperforms PN on combined base and novel classes by a large margin. This demonstrates that representing classes with multiple prototypes leads to better generalization on both base and novel classes. In Figure 2, we show the nearest neighbor to class prototype for PN and to four of the PCN prototypes, for select classes. We can see that PCN has learned to model intra-class variability much more effectively. As an example, for eczema and acne classes we can see that PCN learns clusters corresponding to these skin conditions in different anatomical regions. We provide a more in-depth comparison in the next section.

4.3. Comparison between PCN and PN

PCN or PN with post-hoc clustering? To understand the effectiveness of PCN, we compare it to a PN model in which

Model	Eval CPC (base / novel)	$\text{mca}_{\text{base+novel}}$	mca_{base}	$\text{mca}_{\text{novel}}$	recall@5	recall@10
PN	1 / 1	43.92 +/- 0.40	48.71 +/- 0.37	29.56 +/- 2.35	70.88 +/- 0.36	80.19 +/- 0.26
PN	1 / 4	44.35 +/- 0.53	50.35 +/- 0.42	26.36 +/- 2.34	74.16 +/- 0.21	83.45 +/- 0.25
PN	10 / 4	43.78 +/- 0.78	50.30 +/- 0.21	24.20 +/- 3.02	75.58 +/- 0.23	84.03 +/- 0.19
PCN (ours)	10 / 4	**47.79 +/- 0.71**	**53.70 +/- 0.18**	**30.04 +/- 2.77**	**77.76 +/- 0.19**	**85.96 +/- 0.38**

Table 2: Does post-hoc clustering on PN help?

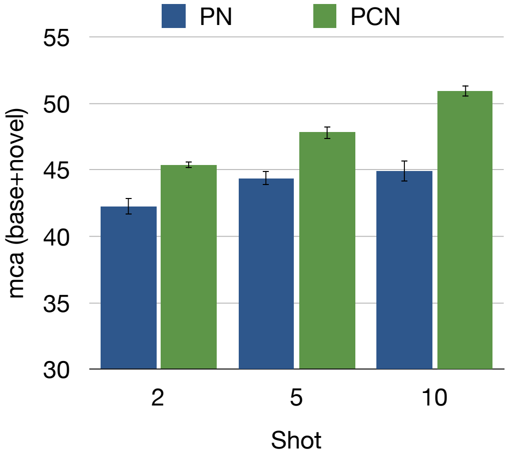

Figure 3: Comparison between PN and PCN as a function of training shot size for novel classes.

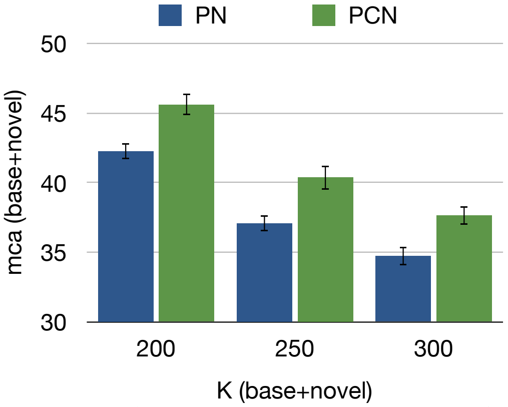

Figure 4: Comparison between PN and PCN as a function of number of novel classes. Due to lack of sufficient data, we compare using a train shot of 2 and test shot of 5.

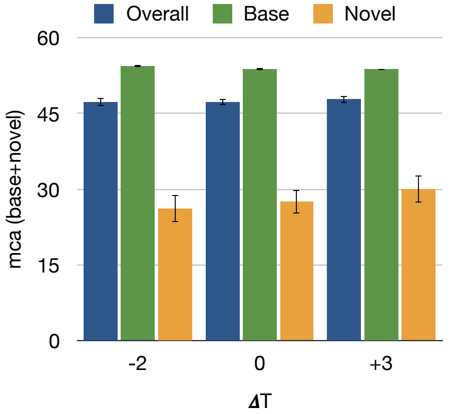

Figure 5: Effect of temperature on PCN

Table 3: Importance of episodic memory

Approach	α	$\text{mca}_{\text{base+novel}}$
PCN	0	45.62 +/- 0.89
PCN	0.5	47.49 +/- 0.71
PN	0	44.35 +/- 0.53
PN*	0.5	45.84 +/- 0.46

we perform "post-hoc" clustering: (a) cluster novel class representations using the PN model's learned embeddings (with cluster size of 4) (b) cluster both base and novel class representations (with cluster size of 10 and 4, as in PCN). Rows 1-3 in Table 2 compare the performance between different post-hoc clustering variants of PN against PCN. We see that PCN leads in all metrics across the board; thus such post-hoc clustering does not lead to improved performance. A reason for this is that the PN model is optimized to learn representations assuming a projection to a single cluster for each class, and hence clustering on such learned representation does not improve performance. This further validates the importance of training with multiple clusters.

Role of shot in novel classes: Figure 3 highlights the effect of number of support examples (shot). As we increase the shot, the performance improves on both methods, but that improvement is larger for PCN than for PN. Because of this, the performance gap between the two methods drastically increases. PCN is better at utilizing the availability of more data by partitioning the space with clusters.

Effect of increasing the novel classes: In this experiment, we study the performance as we vary the number of novel classes at test time from 50 to 150, bringing the total number of classes up from 200 to 300. Fig 4 provides the comparison. We used a train and test shot of 2 and 5, respectively since most classes in these additional 100 novel classes in the long tail have less than 10 examples. Results are reported with 10-fold cross validation. While there is a drop in performance for both models due to very small shot sizes, we can see that the performance gap between PCN and PN continues to hold.

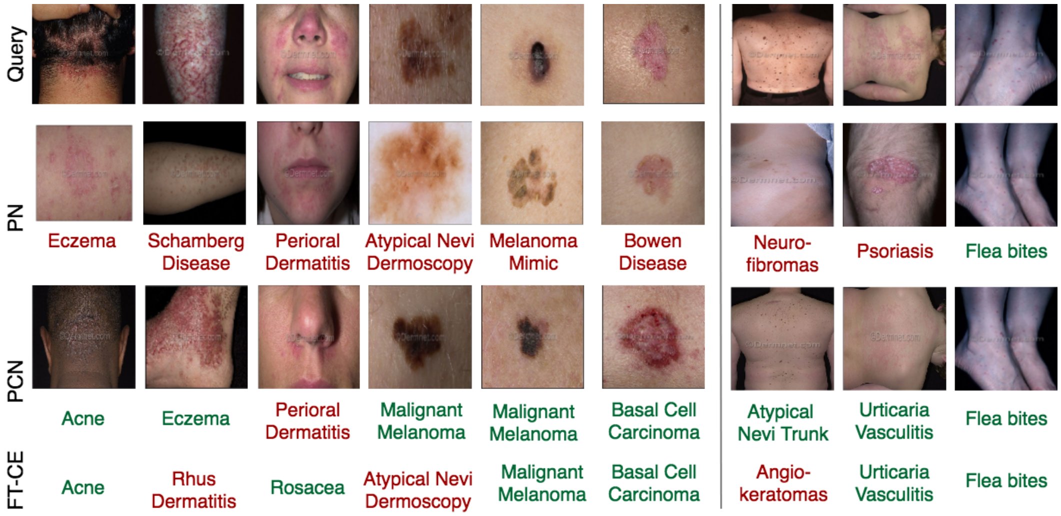

Figure 6: For each query image in test set, we compare PCN with PN and FT-CE. For each image, we color code correct label with green and incorrect with red. For PN, we show the nearest neighbor to the prototype of the *predicted* class. For PCN, we show the nearest neighbor of the top cluster according to $q(z|c, x)$ of the predicted class. The last three columns correspond to examples from novel classes.

4.4. Role of Hyperparameters

Importance of Temperature: Fig. 5 presents the performance by varying $\Delta_\tau = \tau_{test} - \tau_{train}$, the difference in temperature used in test versus train time. We can see that the performance is agnostic for the base classes, as these classes have been used in the training phases. However, for the novel classes, higher interpolation through an increased temperature leading to $\Delta_\tau > 0$ leads to improved performance. Conversely, when $\Delta_\tau < 0$, performance drops. This means that at test time, the model requires interpolating between the cluster prototypes to effectively predict a class label, as described in § 3.2.

Does episodic memory help? Table 3 shows that we can get improvements even with a simple online update rule that blends prototypes computed using the support set in the current episode with the past, using $\alpha = 0.5$. This trend is also seen for prototypical networks (denoted by PN*). We leave as future work the task of modeling adaptive α.

4.5. Qualitative Results

Figure 6 provides qualitative examples comparing the three methods. Acne is one of the largest classes in the base classes with large intra-class variability. Both FT-CE and PCN can diagnose this example correctly. However, PN due to its limited capacity to represent the huge variability in the class is confused with another large class, namely, eczema. PCN, due to having access to multiple clusters can learn a better representation and correctly diagnose acne.

In column 4, we present a case in which both PN and FT-CE identified the query image as atypical nevi dermoscopy, while PCN correctly classified it as malignant melanoma. Atypical nevi are 'funny-looking' moles that are precursors to melanoma. It has been recently studied that dermoscopic features discriminating between atypical naevi and melanoma require expert interpretation through longitudinal monitoring, but are often ignored as simple moles [7]. In contrast, consider Column 7: FT-CE misdiagnose atypical nevi as Angiokeratoma, a benign skin lesion of capillaries, resulting in small marks of red to blue color. In the data-starved setting, FT-CE and PN are unable to differentiate the two skin conditions while PCN can better match up to the support set.

Effectiveness of multiple clusters. In Sec 3, we show how PCN can interpolate between the learned prototypes by modulating $q(z|c, x)$. In Figure 7 we show some qualitative examples to illustrate this. We show query images from the test set for various classes, with a mix of correct and incorrectly classified examples. Below the class label, we show the nearest neighbor image from the training set to each of the learned prototypes for the class *predicted* by PCN, and below each query image, cluster responsibilities placed by the model on each of these prototypes. As an example, consider examples corresponding to acne in Figure 7(a). For this class, we show three examples, all of which are correctly classified by PCN. We can see that the $q(z|c, x)$ distribution varies quite a bit across examples, being a lot more diffuse in some cases than others. It can also be seen that the model learns to accurate place probability mass on similar prototypes. For instance, in column 2 of 7(a), the model appears to interpolate between two prototypes that are similar to the query image in pose and skin texture respectively, to make a correct prediction. Similarly for 7(b) (eczema), the model is accurately able to identify eczemas on the face, arm, and hand, by combining the most

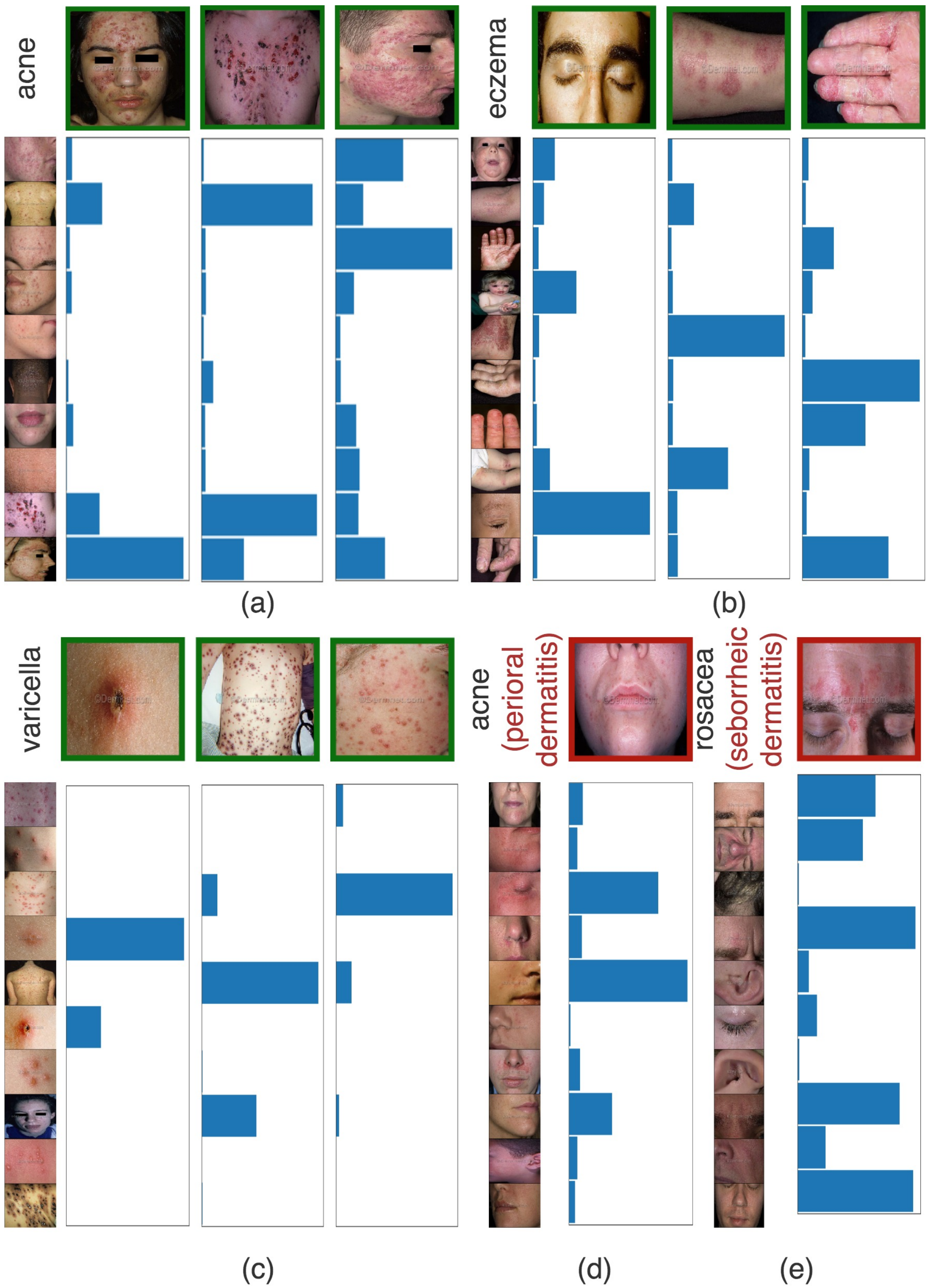

Figure 7: Effectiveness of using multiple clusters. Shown for base classes. (a)-(c): Examples from test set that are correctly classified by PCN. For each class, we show the nearest neighbor to the learned prototypes. We also present three examples (columns) whose labels are correctly predicted and the inferred cluster responsibilities $q(z|c, x)$ conditioned on the correct class. (d)-(e): Examples from the test set that are incorrectly classified by PCN. Correct label is shown in black, while the incorrect prediction is shown in red. We show the nearest neighbors to the learned cluster prototypes of the *predicted* (incorrect) class, and the corresponding cluster responsibilities. Note that green outlines around query images denote correct classification while red denotes incorrect classification.

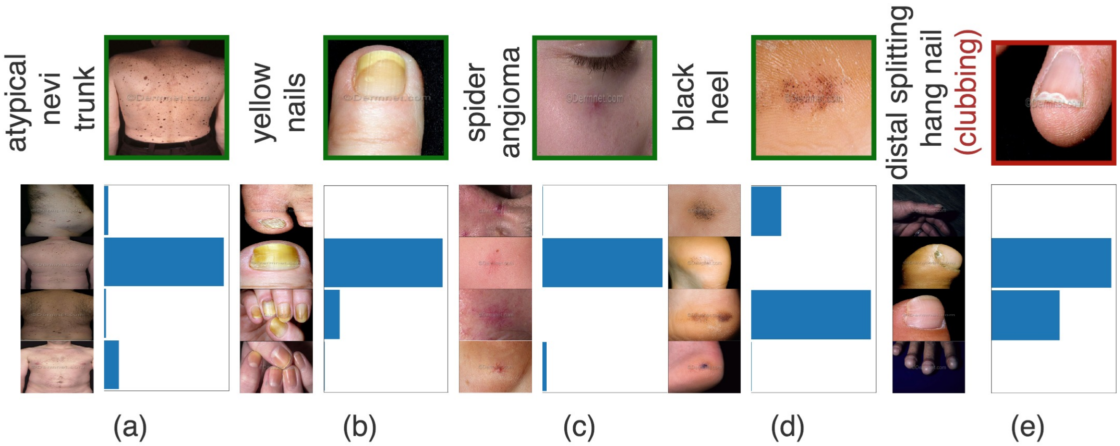

Figure 8: Effectiveness of using multiple clusters for novel classes. (a)-(d) examples from test set that are correctly classified by PCN. For each class, we show the nearest neighbor to the prototypes of the four clusters that are learned for each novel class. For each example, the distribution of inferred cluster responsibilities $q(z|c, x)$ conditioned on the correct (predicted) class is shown. (e): Query example that is incorrectly classified by PCN. Correct label is shown in black, while incorrect prediction is in red. Shown alongside are the nearest neighbors to the cluster prototypes of the *predicted* (incorrect) label, and the corresponding cluster responsibilities.

relevant prototypes. While these classes see relatively diffuse responsibility distributions, the distribution is far more peaked for the varicella class in (c). Figure 7(d)-(e) shows incorrectly predicted examples; Even for these examples, the model seems to interpolate, albeit incorrectly, to make predictions.

In Figure 8, we show similar examples for novel classes. Again, we can see how example specific interpolation is quite helpful to accurately predict the class.

5. Conclusion and Future Work

We propose Prototypical Clustering Networks: a few shot learning approach to dermatological image classification. This method is scalable to novel classes, and can effectively capture intra-class variability. We observe that our approach outperforms strong baselines on this task, especially on the long tail of the data distribution.

There are a number of future directions worth pursuing. The true effectiveness and utility of our system is in aiding the physician, and this requires studies that include such a deployment. Another interesting direction would be to incorporate additional modalities of data for more robust prediction. Dermatologists use symptoms that patient experience, such as itchiness of the skin, in disambiguating skin conditions [20]. Incorporating these medical symptoms as part of the classification task will be an interesting direction to pursue. Finally, while this approach has been developed in the context of the very specific needs of dermatological diagnosis, we believe that similar requirements exist in other domains. It would be interesting to experiment with a similar configuration on other settings and datasets.

A. Appendix

In this appendix we first provide a per-class performance comparison of our proposed approach against the FT-CE baseline. Next, we report additional recall@k metrics for all studied approaches.

B. Per-class Accuracy

In Figure 9 we provide a class-wise performance comparison between the PCN and FT-CE models as a scatter plot, in order to demonstrate their efficacies (shown here for the best performing PCN model evaluated with a train shot of 10 with $\text{mca}_{\text{base+novel}} = 50.92$, details in Table 1 of main paper).

We make the following observations. Overall metrics indicate that FT-CE demonstrates slightly stronger average performance on base classes. For a large fraction of the base classes, both methods have similar performance. For the ones in which PCN performance is lower, there is usually a reasonable lower bound on the classification accuracy. In contrast, for novel classes, PCN performs better on average. Importantly, when FT-CE performance is lower than PCN, it is usually significantly lower. As an example is the novel class 'distal splitting hang nail' for which PCN performs significantly better.

C. Additional Metrics

In table 4 we provide recall@5 and recall@10 metrics for PN, PCN, and FT-CE approaches, for train shot $n = 5$ and $n = 10$. PCN performs on par with the FT-CE baseline and outperforms PN on these metrics. We note that since our

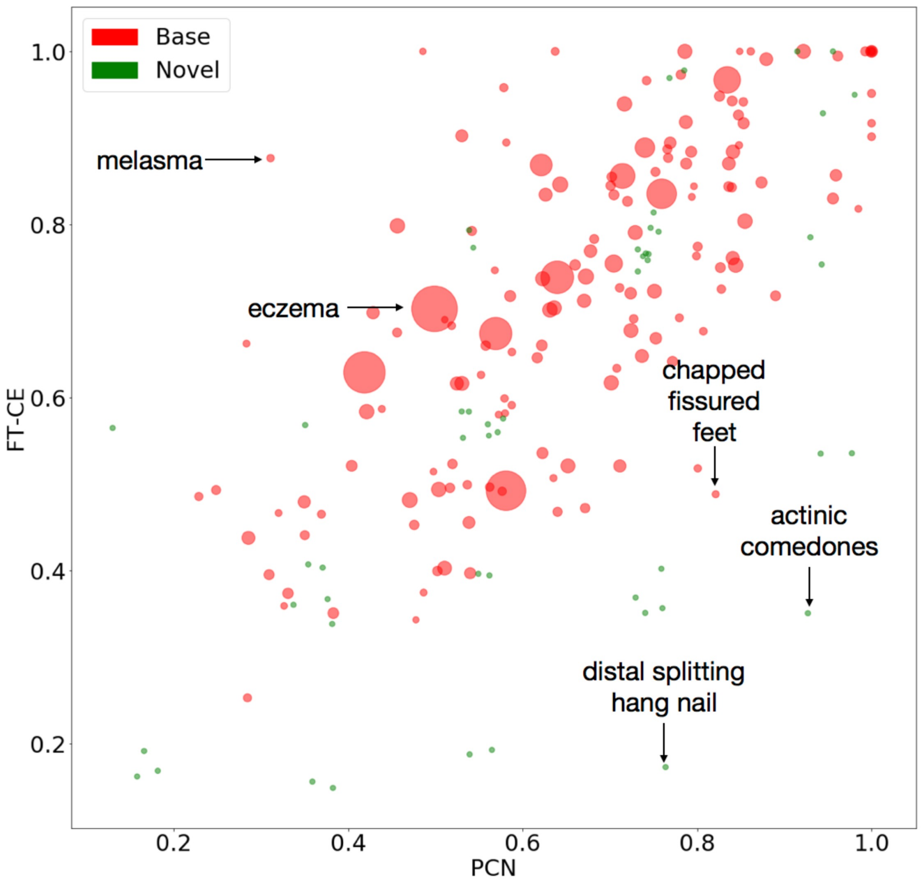

Figure 9: Comparison between FT-CE and PCN: Per-class accuracy. Each class is denoted by a dot and the area of dot is proportional to the number of training examples for the class.

Table 4: Recall@k on top 200 classes.

	Approach	r@5$_{\text{base+novel}}$	r@5$_{\text{base}}$	r@5$_{\text{novel}}$	r@10$_{\text{base+novel}}$	r@10$_{\text{base}}$	r@10$_{\text{novel}}$
n=5	FT-CE	77.7 +/- 0.79	80.84 +/- 0.83	38.0 +/- 2.14	84.92 +/- 0.47	88.06 +/- 0.44	45.33 +/- 1.47
	PN	70.88 +/- 0.36	71.48 +/- 0.33	63.24 +/- 2.34	80.19 +/- 0.26	80.50 +/- 0.18	76.28 +/- 2.13
	PCN (ours)	77.76 +/- 0.19	79.23 +/- 0.22	59.24 +/- 2.57	85.96 +/- 0.38	87.05 +/- 0.20	72.16 +/- 3.59
n=10	FT-CE	78.59 +/- 0.13	80.22 +/- 0.33	58.0 +/- 3.22	86.42 +/- 0.25	88.05 +/- 0.12	65.87 +/- 2.62
	PN	69.36 +/- 0.29	69.14 +/- 0.31	72.16 +/- 1.66	78.59 +/- 0.25	78.11 +/- 0.26	84.68 +/- 2.22
	PCN (ours)	76.29 +/- 0.22	76.43 +/- 0.23	74.52 +/- 2.62	85.03 +/- 0.23	85.04 +/- 0.25	85.04 +/- 2.00

test set is imbalanced, recall@k metrics unfairly reward strong performance on the head classes (which is observed with FT-CE). However, it is clear that PCN and PN models dominate in recall@k metrics on novel classes.

To provide a fairer comparison, in Table 5 we report *balanced* (or macro) recall@k metrics, wherein we compute recall@k for each class and average, treating each class as equally important. Here we clearly find PCN to outperform all baselines owing to strong performance across the board on base and novel classes.

References

[1] M. Basra and M. Shahrukh. Burden of skin diseases. *Expert Review Pharmacoeconomics Outcomes Research*, 2009.

[2] D. Bickers, H. Lim, and D. Margolis. The burden of skin diseases: 2004 a joint project of the american academy of dermatology association and the society for investigative dermatology. *Journal of American Academy Dermatology*, 2006.

Table 5: Balanced Recall@k on top 200 classes.

	Approach	br@$5_{\text{base+novel}}$	br@5_{base}	br@5_{novel}	br@$10_{\text{base+novel}}$	br@10_{base}	br@10_{novel}
n=5	FT-CE	65.44 +/- 0.65	74.57 +/- 0.16	38.0 +/- 2.14	73.08 +/- 0.54	82.33 +/- 0.24	45.33 +/- 1.47
	PN	66.47 +/- 0.58	67.55 +/- 0.15	63.24 +/- 2.34	75.28 +/- 0.54	74.94 +/- 0.13	76.28 +/- 2.13
	PCN (ours)	70.66 +/- 0.64	74.47 +/- 0.18	59.24 +/- 2.57	79.10 +/- 1.04	81.41 +/- 0.29	72.16 +/- 3.59
n=10	FT-CE	69.86 +/- 0.46	73.81 +/- 0.6	58.0 +/- 3.22	77.9 +/- 0.60	81.91 +/- 0.18	65.87 +/- 2.62
	PN	67.51 +/- 0.39	65.96 +/- 0.28	72.16 +/- 1.66	75.87 +/- 0.57	72.94 +/- 0.27	84.68 +/- 2.22
	PCN (ours)	71.41 +/- 0.66	70.37 +/- 0.16	74.52 +/- 2.62	79.93 +/- 0.47	78.23 +/- 0.21	85.04 +/- 2.00

[3] M. Buda, A. Maki, and M. A. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

[4] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115, 2017.

[5] D. Federman, J.Concato, and R. Kirsner. Comparison of dermatologic diagnoses by primary care practitioners and dermatologists. a review of the literature. *Archives of Family Medicine*, 8(2), 1999.

[6] S. Fort. Gaussian prototypical networks for few-shot learning on omniglot. *arXiv preprint arXiv:1708.02735*, 2017.

[7] S. R. Fuller, G. M. Bowen, B. Tanner, S. R. Florell, and D. Grossman. Digital dermoscopic monitoring of atypical nevi in patients at risk for melanoma. *Dermatologic Surgery*, 33(10):1198–1206, 2007.

[8] E. E. Goldman. Skin diseases get misdiagnosed in primary care. *Family Practice News*, 2007.

[9] B. Hariharan and R. B. Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *ICCV*, pages 3037–3046, 2017.

[10] R. Hay and L. Fuller. The assessment of dermatological needs in resource-poor regions. *International Journal of Dermatology*, 2012.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017.

[13] J.K.Scholfield, D. Grindlay, and H. Williams. Skin conditions in the uk: A health needs assessment. *University of Nottingham, Centre of Evidence Based Dermatology UK; Nottingham, UK*, 2009.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Z. Li, C. Wang, M. Han, Y. Xue, W. Wei, L. Li, and F. Li. Thoracic disease identification and localization with limited supervision. *IEEE Computer Vision and Pattern Recognition*, 2018.

[16] H. Liao. A deep learning approach to universal skin disease classification. *University of Rochester Department of Computer Science, CSC*, 2016.

[17] NHANES. Skin conditions and related need for medical care among persons 174 years. *NHANES: United State*, 1978.

[18] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. 2016.

[19] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Proc. CVPR*, 2017.

[20] J. Resneck, M. Abrouk, M. Steuer, and et al. Choice, transparency, coordination, and quality among direct-to-consumer telemedicine websites and apps treating skin disease. *JAMA Dermatology*, 152(7), 2016.

[21] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*, 2016.

[22] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[23] E. Triantafillou, H. Larochelle, J. Snell, J. Tenenbaum, K. J. Swersky, M. Ren, R. Zemel, and S. Ravi. Meta-learning for semi-supervised few-shot classification. 2018.

[24] G. Van Horn and P. Perona. The devil is in the tails: Fine-grained classification in the wild. *arXiv preprint arXiv:1709.01450*, 2017.

[25] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

[26] Y.-X. Wang, D. Ramanan, and M. Hebert. Learning to model the tail. In *Advances in Neural Information Processing Systems*, pages 7029–7039, 2017.

[27] E. Wilmer, C. Gustafson, C. Ahn, S. Davis, S. Feldman, and W. Huang. Most common dermatologic conditions encountered by dermatologists and nondermatologists. *Cutis*, 94(6).

[28] C. Zauner. Implementation and benchmarking of perceptual image hash functions. 2010.

[29] X. Zhu, D. Anguelov, and D. Ramanan. Capturing long-tail distributions of object subcategories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 915–922, 2014.