Soft Weight Networks for Few-Shot Learning

Samuel Hess and Gregory Ditzler, *Senior Member, IEEE*

***Abstract*—Neural network design has accomplished remarkable advances in complex tasks previously thought to be reserved solely for higher cognitive ability. Neural networks can now accurately classify images, play sophisticated games, translate and speak naturally, and even mimic artistic styles. Unfortunately, neural networks have performed sub-par where human performance often excel, which is true for few-shot learning tasks where the objective is to classify a set of query samples based on only a few support examples. Recent neural networks for few-shot learning have found that they can learn pairwise similarity metrics via stochastic gradient descent when a sub-set of support and query samples are repeatedly and randomly selected from the training set (known as episodic training). In this work, we propose a novel Soft Weight Network (SWN) that allow the network to generate embedded features *and* soft weights to cross compare every query-support pairing. The benchmarks of SWNs against the state-of-the-art few shot learning algorithms show that the performance of SWNs on Omniglot and miniImageNet are more favorable when compared to the current/recent few shot learning approaches.**

***Index Terms*—Few-Shot Learning; Deep Learning**

I. Introduction

LARGE-SCALE classification with neural networks has made promising strides in reducing a given task's error over a large amount of data that are sampled from a distribution [1]–[3]. For example, residual networks have surpassed human-level classification accuracy on image recognition for the ImageNet database (1.28 million training images sampled from 1,000 classes) [1], [4]. Value and policy networks have defeated professional humans in the game of Go [2], and WaveNet can automatically generate natural human-sounding speech from text [3]. Unfortunately, many of these types of networks still suffer from poor accuracy when presented with slightly more abstract tasks that humans often find trivial. One such task is known as one-shot learning, where the goal is to categorize (with arbitrarily high accuracy) a set of unlabeled query samples into specific classes after being presented with a set of labeled support samples that contain precisely one sample from each class. The problem is extended to few-shot learning where the support set includes only a few support samples from each class.

Recent neural network approaches to few-shot learning generally fall into two separate categories: meta-learning methods or methods that learn a similarity metric. Meta-learning methods often focus on using the training data repeatedly to learn how best to optimize the base learner parameters for the few-shot task. On the other hand, methods that learn a similarity metric commonly sub-sample the training data into small random batches of support and query samples. This sub-sampling is done repeatedly and a similarity metric is minimized between the support and query samples based on pairwise comparisons within the batch; a technique called episodic training.

S. Hess and G. Ditzler are with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, 85721.
E-mail: {shess,ditzler}@email.arizona.edu

Manuscript received April 19, 2005; revised August 26, 2015.

In this work, we present a *Soft Weight Network* (SWN) that is a nonparametric method inspired by the classification layer on the output of a conventional neural network. For a conventional neural network, the layer before the classification layer is considered to have the information of the nonlinear feature embeddings. These feature embeddings are fed into the classification layer that – after training – uses a hard set of weights (plus a bias) to linearly weight and sum the feature embeddings. The result is a score for each of the classes at the output nodes that are often passed through a softmax function to estimate a posterior.

SWNs use the concept of training a classification layer to approach the few-shot problem (see Figure 1). That is, the network learns, via episodic training, a set of nonlinear feature embeddings for each sample *and* a set of embedded weights. Unlike a conventional neural network with hard set weights, the weights of the SWM are nonlinear functions of the input and differ from sample to sample (i.e. soft weights). Furthermore, the network obtains two scores for the classification of each query-class combination: (1) the features of the query sample with the weights of the support set; and (2) the features of the support set with the weights of the query sample. The result is a simple network design that can better use the information in the episodic batches during training. Our empirical findings show that SWNs can achieve significantly better performance on the Omniglot data set as well as comparable performance on the miniImageNet data set.

II. Related Work

Approaches that focus on performance with few-shot learning can be decomposed into parametric and nonparametric methods. Generally, parametric models that perform well on few-shot data sets use a meta-learning approach that optimizes training parameters over a repeated set of training instances on a base model. For example, Ravi and Larochelle use a recurrent neural network (LSTM) as a meta-learner that learns to optimize the learning rate of stochastic gradient descent (SGD) on the base learner [5]. Li et al. use a similar approach to Meta-LSTM but showed better performance using an SGD algorithm rather than LSTM for the meta-learner [6]. Finn et al. designed Model Agnostic Meta-Learning (MAML) which also learns to optimize the learning rate of SGD on the base learner but avoids the use of additional meta-learning parameters by using repeated batches of k-shot tasks to find the base learner's SGD parameters that produce the largest improvements [7].

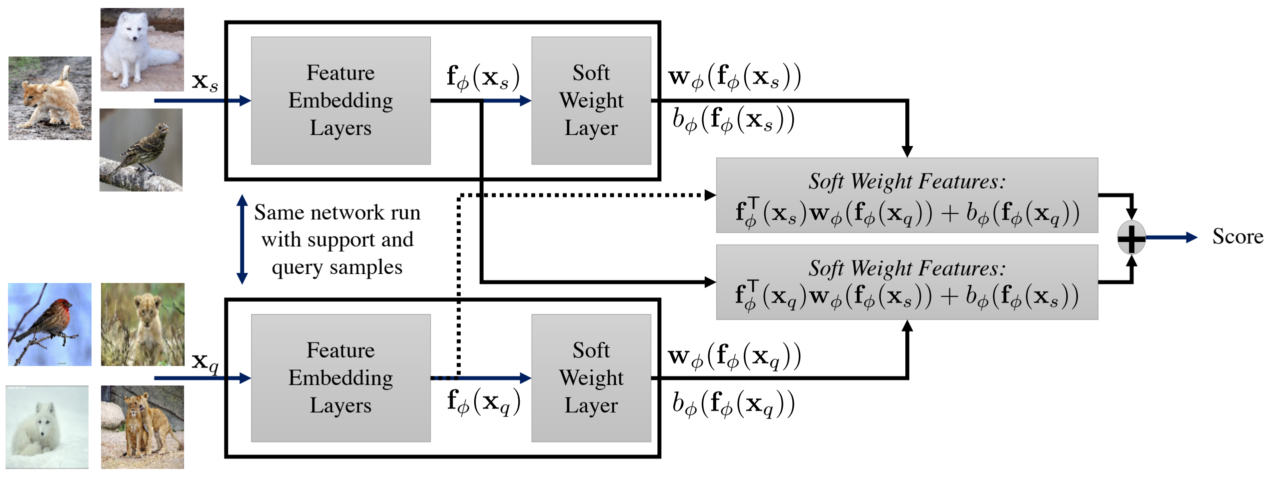
$\mathbf{x}_s$
Feature Embedding Layers
$\mathbf{f}_\phi(\mathbf{x}_s)$
Soft Weight Layer
$\mathbf{w}_\phi(\mathbf{f}_\phi(\mathbf{x}_s))$
$b_\phi(\mathbf{f}_\phi(\mathbf{x}_s))$
Soft Weight Features: $\mathbf{f}_\phi^\mathsf{T}(\mathbf{x}_s)\mathbf{w}_\phi(\mathbf{f}_\phi(\mathbf{x}_q)) + b_\phi(\mathbf{f}_\phi(\mathbf{x}_q))$
Same network run with support and query samples
Soft Weight Features: $\mathbf{f}_\phi^\mathsf{T}(\mathbf{x}_q)\mathbf{w}_\phi(\mathbf{f}_\phi(\mathbf{x}_s)) + b_\phi(\mathbf{f}_\phi(\mathbf{x}_s))$
Score
$\mathbf{x}_q$
Feature Embedding Layers
$\mathbf{f}_\phi(\mathbf{x}_q)$
Soft Weight Layer
$\mathbf{w}_\phi(\mathbf{f}_\phi(\mathbf{x}_q))$
$b_\phi(\mathbf{f}_\phi(\mathbf{x}_q))$

Fig. 1. Overview of the Soft Weight Network (SWN) configuration. The network learns both feature embeddings and soft weights that are used to weight the feature embeddings. That is, the features of the support set ($\mathbf{f}_\phi(\mathbf{x}_s)$) are weighted with the weights of the query set ($\mathbf{w}_\phi(\mathbf{f}_\phi(\mathbf{x}_q))$;$b_\phi(\mathbf{f}_\phi(\mathbf{x}_q))$) and the features of the query set ($\mathbf{f}_\phi(\mathbf{x}_q)$) are weighted with the weights of the support set ($\mathbf{w}_\phi(\mathbf{f}_\phi(\mathbf{x}_s))$;$b_\phi(\mathbf{f}_\phi(\mathbf{x}_s))$). The linear combination of the two metrics is the similarity score for a given query sample.

Finally, Mishra et al. proposed a Simple Neural Attentive Meta-Learner (SNAIL) that relaxes the constraint of using an SGD based algorithm for the base learner and showed incremental empirical improvements to performance when compared to previous methods [8]. Note that for most meta-learners, the interactions between meta-learner and base learner, as well as the addition of meta-learning parameters (with the exception of MAML) inherently, adds to the complexity of the network.

In contrast to parametric approaches, nonparametric methods often learn a pairwise similarity metric with episodic training to mimic the desired testing task during training. Koch et al. applied Siamese networks, which were originally designed for signature verification, to the one-shot learning task [9], [10]. Essentially, the Siamese network uses a set of convolutional neural networks (CNN) with tied weights to embed the features of the support sample and query sample. The output of the CNNs is passed through a distance layer that calculates the weighted distance between the two sets of embedded features. The final output is a sigmoid on the weighted distance which is trained to be one of the images matches and zero otherwise. Vinyals et al. expanded Siamese networks significantly by using a neural network to create a nonlinear feature embedding of all of the support samples as well a separate nonlinear embedding for the query sample [11]. An attention mechanism, which learns to match the query sample to the appropriate support sample, is selected via the softmax over the cosine distances between the query feature embedding and each of the support sample embeddings. Vinyals et al. also trained a Matching Network in episodes that were critical to the performance – a training method that continues to be used throughout the literature. During episodic training, a small set of samples are repeatedly and randomly selected from the training set to establish support and query batches for comparison during training iterations that more closely match the goal of the testing task (i.e., match query samples to support samples). The result is a network training scheme that cleverly uses repeated sub-sampling to learn similarities/differences between samples rather than using a conventional network approach of training on a much more extensive database to learn classification without support set on testing.

Snell et al. proposed a similar approach (i.e., Prototypical Networks) to Matching Networks with episodic training except that the support sample and query sample are passed through the same network, and a Euclidean distance between the feature embeddings are used as the network loss function [12]. The output of the network for the support samples defines the "prototypes," and the training scheme is used to learn good prototypes from support samples by minimizing the distance between the query samples and the prototype. Siamese Networks also compute a distance (i.e., L_1) between the embedded features of the support and query, but is limited in its scope to one-shot and does not include the softmax over the classes like Prototypical Networks and Matching Networks.

The SWN is also a nonparametric network that uses a loss function and episodic training to cross compare the nonlinear embeddings of the support and query features. This technique is similar to Siamese, Matching, and Prototypical networks; however, the SWN differs by expanding our network to include a set of output feature weights that come from the network features. The output of both features and feature weights allows the network to cross weight features for every support-query pairing to get a combined similarity metric.

III. Soft Weight Networks for Few Shot Learning

A. Model

Soft Weight Networks use episodic training to learn a similarity metric for each support-query pair. On each episode the network is given a support set, S that has N D-dimensional samples $\mathbf{x}_i \in \mathbb{R}^D$ with corresponding labels $y_i \in \mathbb{R}^1$. The support set is formally given by $S = \{(\mathbf{x}_{s1}, y_{s1}), ..., (\mathbf{x}_{sN}, y_{sN})\}$

and is the union of the sets from all C classes ($S := S_1 \cup \cdots \cup S_C$), where $S_c = \{(\mathbf{x}_{si}, y_{si}) \forall i : y_{si} = c\}$). Additionally, there is a query set, Q that has M D-dimensional samples with corresponding labels: $Q = \{(\mathbf{x}_{q1}, y_{q1}), \ldots, (\mathbf{x}_{qM}, y_{qM})\}$.

The novelty of the SWN is a network structure that allows it to cross compare a set of soft features and soft weights for every support-query pair. That is, with learnable parameters (ϕ), the SWN learns a non-linear embedding function $\mathbf{f}_\phi : \mathbb{R}^D \rightarrow \mathbb{R}^K$ that transforms a D-dimensional sample into a K-dimensional feature. Additionally, the network learns another set of functions $\mathbf{w}_\phi : \mathbb{R}^K \rightarrow \mathbb{R}^K$ and $b_\phi : \mathbb{R}^K \rightarrow \mathbb{R}^1$ that are used to weight and bias the feature embedding $\mathbf{f}_\phi$, respectively.

The features of each of the the query samples are weighted by the soft weights of the support samples, which is similar to how the classification layer of a conventional neural network node uses hard weights on the feature layer. More formally this is given by

$$\alpha_{qm}(c) = \textstyle\sum_{(\mathbf{x}_{si}, y_{si}) \in S_c} \left[b_\phi(\mathbf{f}_\phi(\mathbf{x}_{si})) + \mathbf{f}_\phi^\intercal(\mathbf{x}_{qm}) \mathbf{w}_\phi(\mathbf{f}_\phi(\mathbf{x}_{si})) \right] \quad (1)$$

where c is the respective support class and $a_{qm}(c)$ is the similarity metric.

Conversely, the weights of the query samples are used to weight to the features of the support samples to obtain the other similarity metric,

$$\beta_{qm}(c) = \textstyle\sum_{(\mathbf{x}_{si}, y_{si}) \in S_c} \left[b_\phi(\mathbf{f}_\phi(\mathbf{x}_{qm})) + \mathbf{f}_\phi^\intercal(\mathbf{x}_{si}) \mathbf{w}_\phi(\mathbf{f}_\phi(\mathbf{x}_{qm})) \right] \quad (2)$$

A distribution is established over the classes via a softmax function,

$$p_\phi(y = c | \mathbf{x}_{qm}) = \frac{e^{(\alpha_{qm}(c) + \beta_{qm}(c))}}{\sum_{c'=1}^{C} e^{(\alpha_{qm}(c') + \beta_{qm}(c'))}} \quad (3)$$

During training, the soft weight network randomly selects a subset of classes from training data and learns the network parameters, ϕ, via stochastic gradient descent by minimizing the negative log probability average

$$J(\phi) = -\frac{1}{CM} \sum_{c'=1}^{C} \sum_{q=1}^{M} \log(p_\phi(y = c' | \mathbf{x}_{qm}))$$

At the time of testing, the network predicts the query sample label, $\hat{y}_{q_m}$, as the class that has the largest probability for the given query sample

$$\widehat{y}_{q_m} = \underset{c \in C}{\arg\max} [p_\phi(y = c | \mathbf{x}_{q_m})]$$

Pseudo-code of the episodic training process, loss update, and class estimation are provided in Algorithm 1.

B. Additional Design Considerations

a) Combined Score: There are a total of four similarity scores that are measured by the SWN: (a) support weights against query features ($\alpha_{qm}(c)$) (b) query weights against support features ($\beta_{qm}(c)$) (c) support weights against support features and (d) query weights against query features. Scores (c) and (d) were found to have a slightly detrimental impact on performance. We hypothesize that these scores ((c) and (d)) would steer the network to learn a unity function for the weights so that they would directly match the features. A similar unregularized effect has been observed in other works regarding autoencoders which had lead to more regularized approaches such as the denoising autoencoders [13].

b) Similarity Metric: Removing the bias term (b_ϕ) in (1) and (2) effectively makes the scoring function a dot product between the features and the weights. The performance on the Omniglot dataset without the bias was found to be nearly equivalent (i.e., the bias had little to no effect). However, the training and testing schemes for this data set have an equal probability of each class, and the bias will be nearly uniform across the classes. If this was not the case, learning a bias could be beneficial, so it was left in as a design choice. Additionally, the similarity metrics in (1) and (2) could have also been structured with a distance metric such as Euclidean distance. Empirically, however, the metrics used in (1) and (2) obtained better performance results. This observation is addressed further in Section IV-C.

IV. Experiments

The SWN was compared to the current state-of-the-art few shot learning algorithms on the two canonical benchmark data sets, namely the omniglot [14] and miniImageNet [15]. Additionally, the training, validation, and testing splits are consistent with those proposed by Ravi and Larochelle [5], and used by Snell et al. [12]. The source will be made available on Github if accepted for publication.

A. Omniglot Classification Results

The Omniglot data set is a character database that is composed of 50 different alphabets and a total of 1,623 character sets [14]. For each of the 1,623 characters there are only 20 human drawn samples which results in a database size of 32,460 samples. The data are originally 105x105 binary images and are resized to 28x28 to be consistent with Vinyals et al. and Snell et al [11], [12]. Additionally, the same 1,200 characters are rotated in multiples of 90 degrees to produce the set of 4,800 classes (96,000 samples) for training/validation and the remaining 1,692 classes (423 with 90° rotations) for testing (33,840 samples).

The SWN was set up to be consistent with Vinyals et al. and Snell et al. [11], [12]. The feature embedding architecture has four CNN layers each with 64-filter 3x3 convolution layers. Each layer also performs bath normalization, uses rectified linear units and 2x2 max pooling. The resulting 64-dimensional feature output is passed through an additional (single layer) linear neural network to produce 64 weights and one bias that are used to weight and bias the feature layer, which is given by (1) and (2). The SWN was trained with PyTorch using the ADAM optimizer [16]. The learning rate schedule was set identical to the schedule presented in Snell et al. [12]: initial rate of 10^{-3} and halved every 2,000 training episodes.

Training was performed with both 1-shot and 5-shot experiments. Training episodes were set up as 60-way (i.e., classes), 5 query samples per class, and training shot was matched to testing shot to compare to Prototypical Networks. 1-shot training was found to perform almost equivalently to

Algorithm 1 Soft Weight Network pseudo-code

```
Input: Entire Training Set $Tr = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_{N_{Tr}}, y_{N_{Tr}})\}$, number of support samples per episode $N$, number of query samples per episode $M$, number of classes per episode $C$, initial model $\phi$, and model learning rate $\eta$
Output: updated model $\phi$, network loss $J(\phi)$, and class estimates $\widehat{y}_{q_m}$
1: for each episode in training data do
2:     Create a set, $E = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_{N_E}, y_{N_{TE}})\}$, of $C$ different classes randomly chosen from $Tr$ without replacement
3:     Create disjoint sets, $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ and $Q = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_M, y_M)\}$, of N support examples and M query samples from $E$
4:     for $q = 1, \ldots, M$ do                                    // Loop through query samples
5:         for $c = 1, \ldots, C$ do                                // Loop through class samples
6:             // Compute the classification scores for every query sample
7:             $\alpha_{qm}(c) = \sum_{(\mathbf{x}_{si}, y_{si}) \in S_c} \left[ b_\phi(\mathbf{f}_\phi(\mathbf{x}_{si})) + \mathbf{f}_\phi^\mathsf{T}(\mathbf{x}_{qm}) \mathbf{w}_\phi(\mathbf{f}_\phi(\mathbf{x}_{si})) \right]$
8:             $\beta_{qm}(c) = \sum_{(\mathbf{x}_{si}, y_{si}) \in S_c} \left[ b_\phi(\mathbf{f}_\phi(\mathbf{x}_{qm})) + \mathbf{f}_\phi^\mathsf{T}(\mathbf{x}_{si}) \mathbf{w}_\phi(\mathbf{f}_\phi(\mathbf{x}_{qm})) \right]$
9:         end for
10:        for $c = 1, \ldots, C$ do
11:            $p_\phi(y = c | \mathbf{x}_{qm}) = \frac{e^{(\alpha_{qm}(c) + \beta_{qm}(c))}}{\sum_{c' \in C} e^{(\alpha_{qm}(c') + \beta_{qm}(c'))}}$     // Estimate posterior via softmax
12:        end for
13:        $\widehat{y}_{q_m} = \arg\max_{c \in C} [p_\phi(y = c | \mathbf{x}_{q_m})]$       // Estimate class of query sample
14:    end for
15:    $J(\phi) = -\frac{1}{CM} \sum_{c'=1}^{C} \sum_{q=1}^{M} \log(p_\phi(y = c' | \mathbf{x}_{qm}))$     // Compute average loss of episode
16:    $\phi \leftarrow \phi - \eta \nabla_\phi J(\phi)$                 // Update model
17: end for
```

TABLE I
OMNIGLOT PERFORMANCE

Model	Classification Performance			
	1-Shot		5-Shot	
	5-Way	20-Way	5-Way	20-Way
Siamese* [9]	98.8%	95.5%	–	–
Matching Networks* [11]	98.1%	93.8%	98.9%	98.5%
Prototypical Networks* [12]	98.8%	96.0%	99.7%	98.9%
MAML* [7]	98.7%	95.8%	**99.9%**	98.9%
ConvNet w/Memory* [17]	98.4%	95.0%	99.6%	98.6%
mAP-SSVM* [18]	98.6%	95.2%	99.6%	98.6%
mAP-DLM* [18]	98.8%	95.4%	99.6%	98.6%
Soft Weight Networks	**99.7%**	**98.3%**	**99.9%**	**99.6%**

* Results reported by the authors.

5-shot training in both 1-shot and 5-shot testing so the 1-shot training model was used exclusively to report results. The SWN was compared to Siamese Networks, Matching Networks, Prototypical Networks [12], MAML [7], ConvNet with Memory [17], mAP-SSVM [18], and mAP-DLM [18]. Table I shows the classification accuracies for the Omniglot experiments. The SWN offer the best performance uniformly across all experiments. To our knowledge, they represent the state-of-the-art on this data set.

B. *miniImageNet Results*

The miniImageNet is a low-resolution image database that is a subset of ILSVRC-12 database [15]. Each sample is a 3x84x84 RGB image and the entire data set is made up of 100 classes with 600 samples per class (i.e., 60,000 samples total). The splits used for training, validation, and testing were the same splits used in Ravi and Larochelle [5]. These splits were used for direct comparison to the previous literature.

The network architecture is similar to the architecture described for the Omniglot data set. First, due to the increased image size from a 1x28x28 image to a 3x84x84 image, the set of output features produced by the CNN resulted in 1600 features. Additionally, to accommodate the increase in features, the linear neural network was also increased to 1600 weights with one bias to satisfy equations (1) and (2). Training was performed using 30-way for 1-shot and 20-way for 5-shot with 15 query samples per episode. This setup was used to directly compare against the results presented in Prototypical Networks. As in the Omniglot dataset, only one model (the 1-shot model) was trained and this model was used for testing both the 1-shot and 5-shot test scenarios. During training, the validation data was used to monitor training performance. When validation performance did not improve over 200 consecutive instances the training would stop. Retraining was done with both the training *and* validation data for the same number of instances determined in the validation process.

Table II shows the results for the miniImageNet. Similar to the Omniglot data set, we find that the SWN performs well compared to other network designs. To our knowledge, SWNs achieve state of the art performance with these results. Furthermore, it should be noted that Soft Weight Networks achieved these performance results using the same testing parameters used in Prototypical Networks to provide a direct comparison between models. However, the training setup and parameters for these results were tuned specifically for Prototypical Networks and not for Soft Weight Networks. Additionally, the performance of Soft Weight Networks for both 1-Shot and 5-Shot categories was achieved with a single model (the 1-Shot training model), whereas other results compare a model trained separately for both tasks.

C. *Comparison with Prototypical Networks*

The Prototypical Network is currently one of the best performing networks for few-shot learning on the commonly used test sets from the literature [12] (i.e., omniglot and miniImageNet). The network is a metric-based method that uses a soft set of features such that, given a set of support and query samples, the *support features* (i.e. "prototypes") most closely match the *query features*. However, the converse of this statement is implicitly true and therefore does not provide any additional information for learning. That is, it is equivalent to say that the Prototypical Network attempts to develop a soft

TABLE II
MINIIMAGENET PERFORMANCE

Model	**Classification Performance**	
	1-Shot; 5-Way	**5-Shot; 5-Way**
Siamese* [9]	48.42 ± 0.79%	–
Matching Nets * [11]	43.40 ± 0.78%	51.09 ± 0.71%
Matching Nets FCE* [11]	43.56 ± 0.84%	55.31 ± 0.73%
Prototypical Nets * [12]	49.42 ± 0.78%	68.20 ± 0.66%
MAML* [7]	48.70 ± 1.84%	63.11 ± 0.92%
Meta-learner LSTM* [5]	43.44 ± 0.77%	60.60 ± 0.71%
mAP-SSVM* [18]	50.32 ± 0.80%	63.94 ± 0.72%
mAP-DLM* [18]	50.28 ± 0.80%	63.70 ± 0.70%
Soft Weight Nets	**53.67 ± 0.73%**	**69.83** ± **0.63%**

* Results reported by the authors.

set of features such that the *query features* most closely match the *support features*.

The significant novelty of the SWN is that the network (also metric based) aims to use a soft set of features as well as a soft set of weights for those features such that the *support weights* are matched to the *query features*. The converse of this statement that the *query weights* are matched to the *support features* is not implicitly true and allows additional information for the network to learn. Learning to match in both ways allows the soft weight network the training advantage of not only validating the query set against the support set but also validating the support set against the query set.

A set of experiments were performed with 1-shot Omniglot performance using different network configurations for both SWN and Prototypical Networks to demonstrate the training advantage of the SWN. The SWN has more learnable parameters in the network since it includes a single linear layer at the output of the feature layer. To account for this expansion we experimented with two configurations were tested: (a) adding an equivalent linear layer to the output of a Prototypical Network; and (b) reducing the total number of nodes in the SWN by reducing all the convolutional filter sizes from 64 to 50. The results are presented in Table III and they show that the difference in network size has a negligible effect on the algorithms. The additional linear layer slightly reduces the performance of the Prototypical Network, which is an effect that can likely be attributed to overfitting.

In addition to the network configurations, the SWNs use a dot product similarity metric (with added bias) instead of a Euclidean distance to compute the classification score. However, it is straightforward to modify (1) and (2) to use Euclidean distance as follows:

$$\alpha_{qm:\text{Euclid}}(c) = \frac{1}{|S_c|}\sum_{(\mathbf{x}_{si},y_{si})\in S_c}[d(\mathbf{f}_\phi(\mathbf{x}_{q_m}), \mathbf{w}_\phi(\mathbf{f}_\phi(\mathbf{x}_{si})))] \quad (4)$$

$$\beta_{qm:\text{Euclid}}(c) = \frac{1}{|S_c|}\sum_{(\mathbf{x}_{si},y_{si})\in S_c}[d(\mathbf{f}_\phi(\mathbf{x}_{si}), \mathbf{w}_\phi(\mathbf{f}_\phi(\mathbf{x}_{qm})))] \quad (5)$$

where d is a distance function such as Euclidean distance.

SWNs implemented with the Euclidean distance are shown to perform better than Prototypical Networks (see Table III) which affirms the benefit of the network's design structure to cross compare samples. However, empirically Euclidean distance performs slightly worse than the metrics presented in (1) and (2).

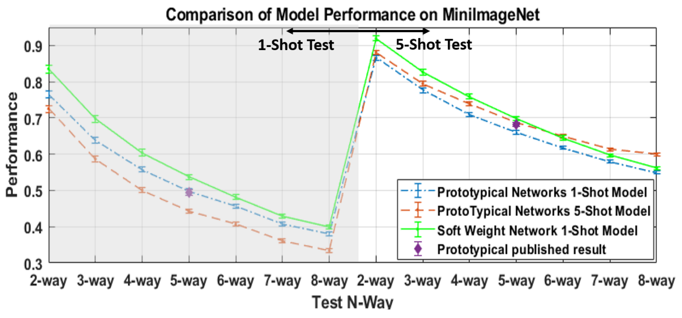

Fig. 2. Performance of SWN and Prototypical Networks on MiniImageNet. The overall best performing model is 1-shot SWN model

TABLE III
COMPARISON OF PROTOTYPICAL AND SOFT WEIGHT NETWORK CONFIGURATIONS ON OMNIGLOT DATA SET

Model	**Classification Performance**	
	1-Shot	
	5-Way	20-Way
Soft Weight Networks	**99.7%**	**98.3%**
Prototypical Networks* [12]	98.8%	96.0%
Prototypical Networks with expansion	98.2%	94.5%
Soft Weight Networks with reduction	99.7%	98.1%
Soft Weight Networks Euclidean distance	99.4%	96.7%

* Results reported by the authors.

Finally, it is important to emphasize that Prototypical Networks use *two* trained models to obtain its performance: 1-shot trained for 1-shot test and 5-shot trained for the 5-shot test. Additionally, Prototypical Networks used the validation data to fine-tune extensively. To obtain the clearest unambiguous comparison, SWN utilized the exact training setup used in Prototypical Networks and did not extensively fine-tune to the validation data. In Figure 2 we present a more complete picture of various N-way miniImageNet performance for the three models. It is clear from the plot that the 1-shot SWN (shown in green) is a significantly better performing network overall. For example, the 5-shot Prototypical Network model (orange) conveniently performs well on the 5-shot data when N-way$\geq$5 but performs significantly worse elsewhere. The SWN 1-shot model performs consistently better without changing models.

D. Conceptual Understanding of Soft Weight Networks: Weighting Image Features

It is often difficult to get a conceptual understanding of what many neural network designs actually learn. The loss function and the training scheme provide some guidance; however, the nonlinear feature embeddings of neural networks are often enigmatic. For SWNs it is possible to gain insight into what is learned by weighting the original image pixels (i.e. unembedded features rather than the nonlinear embedded features) with the network's embedded weights. A SWN was trained in this way and examples of the resultant support and query samples, weights, and weighted features are shown in Figure 3. The six example characters used are given the label 1-6 for reference.

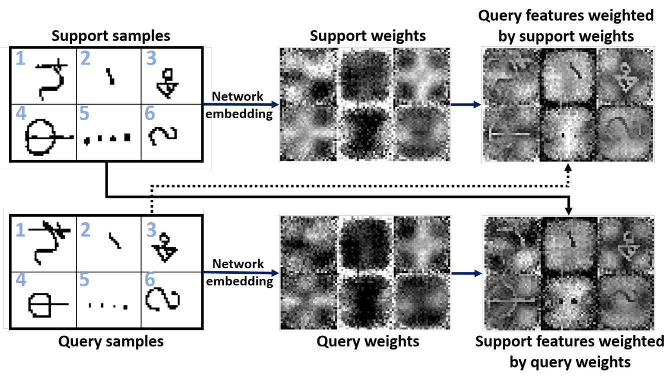
Support samples
Support weights
Query features weighted by support weights
Network embedding
Query samples
Query weights
Support features weighted by query weights
1
2
3
4
5
6

Fig. 3. Demonstration of network, soft weights, and features weighted by soft weights when the Soft Weight Network is configured to use the pixels of the original sample images for the features (i.e. unembedded). Characters are labeled 1-6 and are from the Omniglot dataset (with rotations).

We also observe from the figure that SWNs do not learn traditional prototypes, instead they learn generic character differences. As seen in the weighted features of characters labeled 1, 2, 4 and 5 in Figure 3, the background where the character is not expected to be is often weighted heavily. The characters that are simple (e.g., characters labeled 2 and 5 in Figure 3) have strong background weights and negative character weights. More complex characters (e.g., characters labeled 1 and 4) weight background regions more selectively and positively weight some character features (e.g., the horizontal line in character 4). Additionally, some character features are further unique and weighted higher (such as the external dots in character 5) whereas other character features cannot be heavily weighted since their spatial location is not particularly unique (e.g., the internal dots in character 5).

V. Conclusions

The Soft Weight Network (SWN) proposed in this work is a novel neural network design that is comparable in simplicity to other recent models for few-shot learning. Performance on the Omniglot and miniImageNet data sets were shown to improve the state-of-the-art when using the SWN. For each of the data sets, only one model (1-shot trained model) was needed for reported performance on both 1-shot and 5-shot testing. It was demonstrated that the performance gain was due to the novel network configuration because it uses more of the available information during episodic training. We also demonstrated there was a slight gain in SWNs from using the dot product similarity metric between feature-weight pairings instead of a Euclidean distance metric.

Additionally, SWNs – when configured to use the embedded weighting on the original image pixels – provided insight into how the classification was performed. Rather than focusing on weighting the distinct pixel features of a character from Omniglot, the network often focused on the amount of background or small portions of a character that was significantly different than other characters. It is common for neural networks to learn a task that performs generically well over a data set but is likely not the intended learned task [19]. Soft Weight Networks – in addition to obtaining promising performance on few-shot learning tasks – provide a window into how neural networks trained with a similarity metric operate.

Acknowledgment

This work was supported by an NVIDIA GPU Grant.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE CVPR*, pp. 770–778, 2016.
[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[3] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv:1609.03499*, 2016.
[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
[5] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," *International Conference on Learning Representations*, 2017.
[6] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-sgd: Learning to learn quickly for few shot learning," *arXiv preprint arXiv:1707.09835*, 2017.
[7] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv preprint arXiv:1703.03400*, 2017.
[8] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "Meta-learning with temporal convolutions," *arXiv preprint arXiv:1707.03141*, 2017.
[9] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learn. Wrksp*, 2015.
[10] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in Neural Information Processing Systems*, pp. 737–744, 1994.
[11] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems*, pp. 3630–3638, 2016.
[12] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems*, pp. 4080–4090, 2017.
[13] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.
[14] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum, "One shot learning of simple visual concepts," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 33, 2011.
[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[17] Ł. Kaiser and O. Nachum, "Aurko roy, and samy bengio," *Learning to remember rare events. arXiv preprint*, 2017.
[18] E. Triantafillou, R. Zemel, and R. Urtasun, "Few-shot learning through an information retrieval lens," in *Advances in Neural Information Processing Systems*, pp. 2252–2262, 2017.
[19] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and Brain Sciences*, vol. 40, 2017.