Meta Learner with Linear Nulling

Sung Whan Yoon shyoon8@kaist.ac.kr
Jun Seo tjwns0630@kaist.ac.kr
Jaekyun Moon jmoon@kaist.edu

School of Electrical Engineering,
Korea Advanced Institute of Science and Technology (KAIST)

Abstract

We propose a meta learning algorithm utilizing a linear transformer that carries out null-space projection of neural network outputs. The main idea is to construct a classification space such that the error signals during few-shot training are zero-forced on that space. The final decision on a test sample is obtained utilizing a null-space-projected distance measure between the network output and label-dependent weights that have been trained in the initial meta learning phase. Our meta learner achieves the best or near-best accuracies among known methods in few-shot image classification tasks with Omniglot and *mini*ImageNet. In particular, our method shows stronger relative performance by significant margins as the classification task becomes more complicated.

1 Introduction

Achieving human-like adaptability remains a daunting challenge for developers of artificial intelligence. In a related effort, researchers in the machine learning literature have tried to understand ways to learn quickly on a small number of training examples. Learning the ability to quickly learn a new task using only a few training samples is known as meta learning, and typically two-level learning is employed: initial learning on large data sets representing widely varying sets of tasks and few-shot learning on small amounts of data to conduct a new task. Significant advances have been made recently along this direction; in particular, the prior work on a combined system of a neural network and an external memory, known as the memory-augmented neural network (MANN) [9], showed a notable progress in meta learning. During initial meta-learning, learnable parameters include not only the neural network itself but also a separate set of label-specific linear weights that act on the memory outputs. While few-shot learning, external memories in MANN store key vectors extracted from a neural network so that the stored information in conjunction with the label-specific weights substantially enhances the classification accuracy. Several effective meta learning algorithms have since been suggested with varying flavors in the system structure, often resulting in notable performance improvements. The matching network of [11] yields decisions based on comparing the output of a network driven by an unlabeled sample to the output of another network fed by the labeled samples. In the initial training phase, two networks are learned so that the distance between the outputs corresponding to a same class are reduced. In yet another approach, a long short-term memory (LSTM) is trained to optimize another learner, which performs actual classification [7]. During the few-shot learning, the parameters of the learner are first set to the memory state of the LSTM, and then quickly learned by the memory state update rule of the LSTM, effectively capturing the knowledge from the few shots. The memory state update rule of the LSTM offers faster adaptation than the general gradient descent parameter update. An approach called the prototypical network has also been suggested which constructs a prototype reference for each class with a given support set [10]. For a given class, corresponding shots in the support set are processed by a convolutional neural network (CNN) and the mean-output is taken as the prototype for each class. For classifying the unlabeled input, the nearest prototype is selected in the embedding space. A method dubbed the simple neural attentive meta-learner (SNAIL) combines an embedding neural network with temporal convolution and soft attention to quickly draw from past experience while enabling precision access at the same time [6]. For one-shot learning of rare events in life-long learning, a long-term memory module has been devised in [3] to enable trace-back of rare training examples.

In this paper, we propose a highly improved meta learning algorithm utilizing a linear transformation to create a null-space for projecting network output vectors. More specifically, the linear transformation is composed of a learnable part that reduces dimensionality of the network output and a fixed transformer that acts as null-space projection. Decomposition in this fashion between the learnable "compressor" and the fixed transformation controls the number of learnable parameters and seems to allow fast adaptation without compromising accuracy. Another key aspect is that there exist label-specific linear weights which are also learned during initial training. These weights more or less settle in the initial meta-learning phase and provide class-specific references in the actual classification of unlabeled samples followed by few-shot learning. These weights also play a key role in constructing the null space. We note that linear learnable weights are also employed in the MANN when collecting external memory outputs. During few-shot training, the classification space or the null-space is quickly learned to zero-force the error between the network output average and the label-specific weight for each class. The network output average vector for each class is obtained at the output of the compressor function. Once the null space is created, classification of an unlabeled sample is done by comparing the null-space-projected network output with the label-specific weight vectors.

For classification accuracy, our meta learner generally shows the best or near-best performance among the existing meta learners. For complicated tasks, however, our learner consistently shows the best performance by significant margins. Specifically, our method achieves the best accuracy in 20-way Omniglot classification for both 1 and 5-shot tests. Moreover, in 5-way *mini*ImageNet classification, our method achieves the best performance for both 1 and 5-shot tests by considerable margins. The learning speed of the proposed meta learner during the initial episode training phase is also noteworthy, which may be an important issue in certain applications where the data set is limited. When compared with the prototypical network of [10] on 5-way *mini*ImageNet classification with 5 shots, not only the eventual classification accuracy of the proposed learner is considerably better but also the number of initial training episodes required in achieving 90% of the final accuracy is reduced by almost a factor of 5.

2 Meta Learner with Null-Space Projection

In this section, the details of meta learning with null-space projection are presented to show how the few-shot learning capability is improved. The null-space formulation is for cutting out unnecessary spaces so that the learner can more efficiently specify a classification space appropriate for the given task. Our meta learner employs both a neural network and a linear transformer with a better classification space easily and quickly learned through linear nulling of the error signal between the averaged network output and the label-specific weight vector for every class.

2.1 Basic Model and Notations

In Figure 1, a system diagram of our meta learner and a visual illustration of the proposed null-space construction are depicted with actual image shots from *mini*ImageNet. It is assumed that all leanable parameters in the system have already been stabilized at this point after being trained on a large initial set of images. The system is now ready to take on a support set for quick training before a test sample gets to be classified. A support set $S = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\}$ of N shots with their correct labels is given. For a shot, $\mathbf{x}_n \in \mathbb{R}^D$ is the feature vector with D dimensions and $y_n \in \{0, \cdots, N_c - 1\}$ is the class label. Let S_k be the set of shots in the support set with label k. By considering shots in the support set as inputs, our meta learner obtains the outputs of the embedding function (through network) $f_\theta : \mathbb{R}^D \to \mathbb{R}^M$ with learnable parameters θ. The M-dimensional output $f_\theta(\mathbf{x}_n)$ is consequently multiplied by a learnable matrix $\mathbf{W}$ with dimension $M' \times M$. Define an M'-dimensional vector $\mathbf{g} \triangleq f_\theta(\mathbf{x}_n)\mathbf{W}^T$, where $(\cdot)^T$ means matrix transpose. Here, M' is set smaller than M so that the vector $\mathbf{g}$ can be interpreted as a compressed version of the network output $f_\theta(\mathbf{x}_n)$. Note that MANN also employs a dimension-reducing filter in generating keys for memory access.

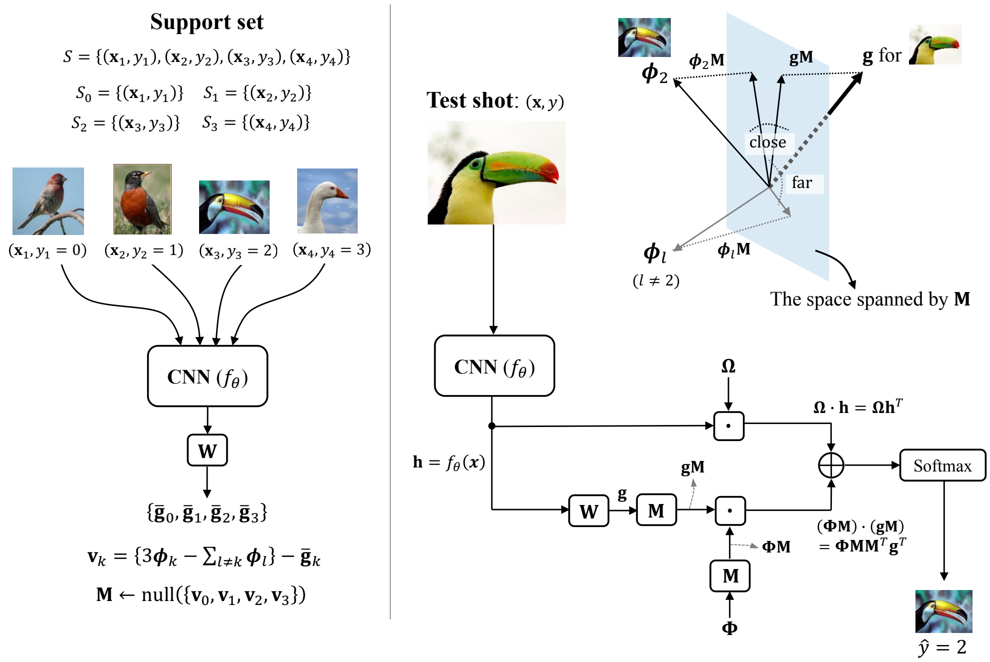

Figure 1: The proposed meta learner with the linear nulling

Our meta learner also relies on label-specific linear weight vectors acting on the network output for classification purposes. These label-specific weight vectors are collected as rows in matrix $\boldsymbol{\Phi}$. Note that the nonlinear function f_θ, the linear compressor $\mathbf{W}$ and the weights $\boldsymbol{\Phi}$ are all learned during the initial meta-learning phase and remain more or less stable during few-shot learning. The label-specific weights $\boldsymbol{\Phi}$ also play a key role in designing an effective projection space on which classification decisions are to be made. We note that in MANN there also are label-dependent weights acting on the output of the external memory.

In Figure 1, the given support set consists of four example image shots with four classes. On the left side of the figure, example images from the support set are processed by a CNN followed by a compression matrix $\mathbf{W}$ to give the averaged network (compressed) output vectors $\bar{\mathbf{g}}_k$ for each of N_c labels. After obtaining the error signals $\mathbf{v}_k$, the linear transformer $\mathbf{M}$ is determined by nulling them as will be described shortly. Unlike the learnable matrices $\mathbf{W}$ and $\boldsymbol{\Phi}$, the matrix $\mathbf{M}$ is not trained directly; it is determined by the averaged network output vectors $\bar{\mathbf{g}}_k$ and the weight matrix $\boldsymbol{\Phi}$.

2.2 Construction of Null Space

Suppose for the time being that a training sample labeled k is taken as the input of the meta learner. Then it would make sense to choose $\mathbf{M}$ such that the corresponding network output $\mathbf{g}_k$ and weight vector $\boldsymbol{\phi}_k$ should be aligned when projected on itself. This is to say that the inner product should be maximized in the sense of

$$\big(\boldsymbol{\phi}_k\mathbf{M}\big) \cdot \big(\mathbf{g}_k\mathbf{M}\big) > \big(\boldsymbol{\phi}_l\mathbf{M}\big) \cdot \big(\mathbf{g}_k\mathbf{M}\big) \text{ for all } l \neq k \text{ where } 0 \leq k, l \leq N_c - 1.$$

Finding the solution of these inequalities is not easy. Let us take an alternate route in finding $\mathbf{M}$. We proceed by writing a difference term:

$$\Delta_{k,l} = \big(\boldsymbol{\phi}_k\mathbf{M}\big) \cdot \big(\mathbf{g}_k\mathbf{M}\big) - \big(\boldsymbol{\phi}_l\mathbf{M}\big) \cdot \big(\mathbf{g}_k\mathbf{M}\big) = (\boldsymbol{\phi}_k - \boldsymbol{\phi}_l)\mathbf{M}\mathbf{M}^T\mathbf{g}_k^T. \tag{1}$$

By considering all cases of $l \neq k$, we further write:

$$\Delta_k \triangleq \sum_{l \neq k} \Delta_{k,l} = \big\{(N_c - 1)\boldsymbol{\phi}_k - \sum_{l \neq k} \boldsymbol{\phi}_l\big\}\mathbf{M}\mathbf{M}^T\mathbf{g}_k^T. \tag{2}$$

We wish to choose $\mathbf{M}$ to make Δ_k as large as possible for all k. At this point, all labeled samples from the support set should be utilized. The average vector is given by

$$\bar{\mathbf{g}}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_n, y_n) \in S_k} f_\theta(\mathbf{x}_n)\mathbf{W}^T \tag{3}$$

for each class. We replace the compressed output vector $\mathbf{g}_k$ in (2) with the average vector $\bar{\mathbf{g}}_k$:

$$\Delta_k = \big\{(N_c - 1)\boldsymbol{\phi}_k - \sum_{l \neq k} \boldsymbol{\phi}_l\big\}\mathbf{M}\mathbf{M}^T\bar{\mathbf{g}}_k^T. \tag{4}$$

It is easy to see that to make Δ_k large, $\mathbf{M}$ should be such that the two vectors $\{(N_c-1)\boldsymbol{\phi}_k - \sum_{l\neq k} \boldsymbol{\phi}_l\}$ and $\bar{\mathbf{g}}_k$, when projected on it, yields a large inner product. Defining the difference or error vector $\mathbf{v}_k = \{(N_c - 1)\boldsymbol{\phi}_k - \sum_{l\neq k} \boldsymbol{\phi}_l\} - \bar{\mathbf{g}}_k$, this objective is achieved by taking $\mathbf{M}$ to be a subspace of the null-space of $\mathbf{v}_k$: $\mathbf{M} \subseteq \text{null}(\mathbf{v}_k)$. In this way, the error vector is always forced to zero when projected on $\mathbf{M}$. The operator null$(\mathbf{v}_k)$ returns a set of orthonormal basis which are on a null-space of $\mathbf{v}_k$. By focusing only on the null-space, we are essentially cutting out unnecessary subspaces for classification that contain any non-orthogonal basis to the error vector. To null error vectors for all classes, $\mathbf{M}$ is obtained as $\mathbf{M} \subseteq \text{null}\big(\{\mathbf{v}_0, \cdots, \mathbf{v}_{N_c-1}\}\big)$ The operator null$\big(\{\mathbf{v}_0, \cdots, \mathbf{v}_{N_c-1}\}\big)$ returns a set of orthonormal basis which are all orthogonal to each of N_c vectors $\{\mathbf{v}_0, \cdots, \mathbf{v}_{N_c-1}\}$. With such an $\mathbf{M}$, all error vectors are forced to be zero on the space spanned by it. When M', the dimension of $\mathbf{v}_k$, is larger than N_c, matrix $\mathbf{M}$ always exists. Moreover, the rank of $\mathbf{M}$ is less than equal to $M' - N_c$.

2.3 Details of Learning and Classification Procedures

In Algorithm 1, the specific training procedures of our meta learner are provided. For each training episode, the number of labeled samples per class is increased from 1 to $N_s - 1$ (in line 3). In other words, the size of the support set is increased from N_c to $N_c(N_s - 1)$ depending on the number of shots allowed (in line 5). As the system runs through the support set, the class-dependent average network output vector, the corresponding error vector and the linear transformer $\mathbf{M}$ are updated (in lines 6, 7 and 9, respectively). The training loss is updated in a cumulative manner (from line 10 to 15). After exhausting all the labeled samples in the given training episode, the learnable parameters θ are updated with the training loss (in line 17). So are all three learnable matrices $\mathbf{W}$, $\boldsymbol{\Phi}$ and $\boldsymbol{\Omega}$.

With the stable CNN and linear mapping $\mathbf{M}$, the test shot can now be classified as illustrated on the right side of Figure 1. Recall that the weight matrix $\boldsymbol{\Phi}$ contains N_c label-specific vectors as its rows. The compressed network output vector is projected on $\mathbf{M}$ and compared with $\boldsymbol{\Phi}$ also projected on $\mathbf{M}$. Specifically the inner product $\boldsymbol{\Phi}\mathbf{M}\mathbf{M}^T\mathbf{g}^T$ provides a measure of the cosine distance between the compressed network output with each of the label-specific weight on the projected space.

As illustrated in the example shown in Figure 1, when the compressed network output $\mathbf{g}$ is from class 2, it is closer to the weight vector $\boldsymbol{\phi}_2$ than any other label-specific weight vectors when they are projected on the space spanned by $\mathbf{M}$.

We have another set of label-dependent weight vectors contained in a matrix $\boldsymbol{\Omega}$ with dimension $N_c \times M$. These weight vectors apply to the direct network output $\mathbf{h}$. In the same manner, the inner product between the direct network output $\mathbf{h}$ and each of the label-dependent weight vectors in $\boldsymbol{\Omega}$ are expressed collectively as $\boldsymbol{\Omega}\mathbf{h}^T$. The final decision is made by the softmax-normalized summation of these two inner product matrices $\boldsymbol{\Phi}\mathbf{M}\mathbf{M}^T\mathbf{g}^T$ and $\boldsymbol{\Omega}\mathbf{h}^T$. The softmax combination of these two branches of the direct network output and the transformed network output is motivated by the structure of the MANN [9].

3 Simulation Results

We use Omniglot [5] as well *mini*ImageNet generated from ILSVRC-2012 ImageNet [8] with the splits by Ravi and Larochelle [7].

3.1 Settings for Omniglot Few-shot Classification

Omniglot is a set of images of handwritten characters from many alphabets. There are 1623 characters from 50 alphabets and 20 examples per each class in the Omniglot dataset. For performance evaluation,

Algorithm 1 Training is done by N_E training episodes. Each episode E_i consists of N (shot, label) pairs. These N shots are composed of N_c classes and each class contains N_s shots. L_{train} is the loss to train learnable parameters and $L(\cdot)$ is the softmax cross-entropy loss function.

Input: Training set $E^T = \{E_1, ..., E_{N_E}\}$ where $E_i = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ is episode of length $N = N_c \times N_s$, and $y_t \in \{0, ..., N_c - 1\}$. $E_i^{(k)} = \left\{(\mathbf{x}_1^{(k)}, y_1^{(k)}), ..., (\mathbf{x}_{N_s}^{(k)}, y_{N_s}^{(k)})\right\}$ is the subset of E_i consisting of all $(\mathbf{x}_n, y_n)$ such that $y_n = k$.

for i in $\{1, ..., N_E\}$ **do**
  $L_{train} \leftarrow 0$
  for j in $\{1, ..., N_s - 1\}$ **do**
    for k in $\{0, ..., N_c - 1\}$ **do**
      $S_k \leftarrow \left\{(\mathbf{x}_n^{(k)}, y_n^{(k)})\right\}$ with $(\mathbf{x}_n^{(k)}, y_n^{(k)}) \in E_i^{(k)}, n \leq j$
      $\bar{\mathbf{g}}_k \leftarrow \frac{1}{j} \sum_{(\mathbf{x}_n^{(k)}, y_n^{(k)}) \in S_k} f_\theta(\mathbf{x}_n) \mathbf{W}^T$
      $\mathbf{v}_k \leftarrow \left\{(N_c - 1)\boldsymbol{\phi}_k - \sum_{l \neq k} \boldsymbol{\phi}_l\right\} - \bar{\mathbf{g}}_k$
    end for
    $\mathbf{M} \leftarrow \text{null}\left(\{\mathbf{v}_k\}_{k \in \{0, ..., N_c - 1\}}\right)$
    for k in $\{0, ..., N_c - 1\}$ **do**
      $(\mathbf{x}_q, y_q) \leftarrow (\mathbf{x}_n^{(k)}, y_n^{(k)}) \in E_i^{(k)}, n = j + 1$
      $\mathbf{h}_q \leftarrow f_\theta(\mathbf{x}_q)$
      $\mathbf{g}_q \leftarrow \mathbf{h}_q \mathbf{W}^T$
      $L_{train} \leftarrow L_{train} + L(\mathbf{\Phi} \mathbf{M} \mathbf{M}^T \mathbf{g}_q^T + \mathbf{\Omega} \mathbf{h}_q^T; y_q)$
    end for
  end for
  Update $\theta, \mathbf{W}, \mathbf{\Phi}, \mathbf{\Sigma}$ minimizing L_{train} via Adam optimizer
end for

we have used 28×28 downsized grayscale images and introduced angle rotation of images in multiples of 90 degrees, as done in prior works [9, 10, 11]. 1200 characters are used for training and test is done with the remaining untrained characters. For fair comparison, we use the same embedding network which is widely used in many prior works. The embedding network is composed of four convolutional blocks with a few variations. Each block consists of a 3×3 2D convolution layer with 64 filters, stride 1 and padding, a batch normalization layer [2], a ReLU nonlinear activation layer and a 2×2 maxpooling layer, as done in [6, 10, 11]. The convolutional blocks returns 256-dimensional neural network outputs, which are compressed to 160-dimensional vectors $\mathbf{g}$ by $\mathbf{W}$ with dimension 160×256. The same encoder is used for the support set and the current test input. The learner is trained via stochastic gradient descent (SGD) with the Adam optimizer [4]. We use a learning rate of 10^{-3} without learning-rate decay. No regularization is used except for batch normalization in convolutional blocks.

3.2 Settings for *mini*ImageNet Few-shot Classification

The *mini*ImageNet is a dataset suggested by Vinyals et al. for few-shot classification of colored images [11]. It is a subset of the ILSVRC-12 ImageNet dataset, and it consists of 100 classes and 600 images per each class. Since the details of splits in [11] is not open, we have used the splits introduced by Ravi and Larochelle [7]. For simulation, we have used 84×84 downsized color images, with rotation in multiples of 90 degrees as we did in Omniglot simulation. As in [7], we used 64 training classes and 20 test classes. We did not use validation classes. We use the same embedding network as one we have used in Omniglot simulation. Due to the increased size of the input, the output of the embedding network is of 2304 dimensions. However, the network output is compressed to a 160-dimensional vector $\mathbf{g}$ as we did in the Ominiglot case. Like the Omniglot experiment, the learner is trained with the Adam optimizer with a learning rate of 10^{-3} and without learning-rate decaying. Additional regularization is not considered.

Table 1: Few-shot classification accuracies for 5- & 20-way Omniglot

	5-way Omniglot		20-way Omniglot	
Methods	1-shot	5-shot	1-shot	5-shot
MANN [9]	82.8%	94.9%	-	-
Matching Nets [11]	98.1%	98.9%	88.2%	97.0%
MAML [1]	98.7%	**99.9%**	95.8%	98.9%
Prototypical Nets [10]	98.8%	99.7%	96.0%	98.9%
SNAIL [6]	**99.07%**	99.78%	97.64%	99.36%
MLN (proposed)	98.75%	99.81%	**98.64%**	**99.70%**

Table 2: Few-shot classification accuracies for 5-way *mini*ImageNet

	5-way *mini*ImageNet	
Methods	1-shot	5-shot
Matching Nets [11]	43.6%	55.3%
Meta-Learner LSTM [7]	43.44%	60.60%
MAML [1]	48.7%	63.1%
Prototypical Nets [10]	49.42%	68.20%
SNAIL [6]	55.71%	68.88%
MLN (proposed)	**57.20%**	**72.85%**

3.3 Comments on Classification Performance

In Table 1 and 2, few-shot classification accuracies of our meta learner with nulling (MLN) are presented. Prior meta learners including MANN [9], matching networks [11], meta learner LSTM [7], model-agnostic meta learners (MAML) [1], prototypical networks [10], SNAIL [6] are compared to our method.

For 5-way Omniglot classification, our learner is at third and second place in 1-shot and 5-shot classification accuracy, respectively. Although the performance is not the best, our learner achieves sufficiently good accuracies higher than 98.5% and 99.8% in 1-shot and 5-shot, respectively. Notably, for the more complex case with 20-way classification, our learner achieves the best performance for both 1 and 5-shot cases. Let us also see 5-way *mini*ImageNet classification which is much more complex than the Omniglot test. Our meta leaner with nulling again shows the best performance, this time by significant margins. It appears that the proposed meta learner aided by a linear nulling processor achieves the stronger relative performance for more complicated classification tasks.

3.4 Special Comments on Learning Speed

Although meta learning speed is not widely analyzed in prior works, it should be considered carefully for developing practical learning algorithms. Evaluations of prior work are done after initial training on sufficiently large datasets as in the cases of Omniglot and *mini*ImageNet. However, this would not be feasible in many realistic meta learning scenarios, and reducing the amount of data used in meta learning remains a key issue in developing practical meta learners.

Let us now evaluate the learning speed of our meta learner by measuring the number of initial training episodes used. There is unfortunately a general lack of available numerical results on the required number of initial training episodes for prior works. Here, we simulate only the prototypical network [10] for comparison. The prototypical network is an efficient meta learner with excellent classification performance for both Omniglot and *mini*ImageNet data, while retaining a very simple structure.

In Figure 2, during the initial training phase of the prototypical network and our meta learner, classification accuracies are plotted up to 1.0×10^5 training episodes. The vertical value is the classification accuracy associated with 5-shot learning assuming the system has been initialized using the number of training episodes given by the corresponding horizontal value.

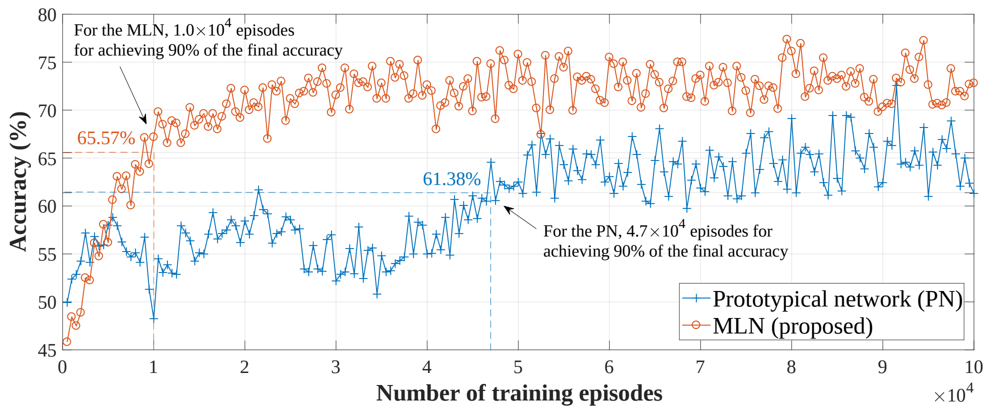

Figure 2: Initial training speed: MLN versus the prototypical network of [10]

The target task is 5-way *mini*ImageNet classification and 5-shot test. For the prototypical network, about 4.7×10^4 training episodes are required for reaching 90% of its final classification accuracy. On the other hand, our meta learner with nulling shows much more rapid and steady improvement with initial training. In particular, only 1.0×10^4 training episodes are required for exceeding the 65.57% accuracy, which is 90% of the scheme's final performance level. In addition, our meta learner only requires around 2.0×10^4 initial training episodes for surpassing a 68.20% classification accuracy which is the best performance of the fully-trained prototypical network with a sufficient number of training episodes.

We can easily see that MLN, our meta learner aided by nulling, shows a much faster meta-learning speed than the prototypical network. In the prototypical network, the network output is directly used for classification based on the boundaries that maximally separate the prototype references derived form the support set. On the other hand, our meta learner utilizes a separate linear processor that provides an improved classification space via nulling. Also, our meta learner employs significant linear processing, which is based on both learnable and fixed null-space transformations. We conjecture that this unique combination allows rapid adaptation during both initial episode learning and few-shot learning.

4 Relation to Prior Work

In this section, we take a close look at how our meta learner could be viewed in relation to two prior methods: the MANN and the prototypical network. For the MANN of [9], an external memory module is attached to a recurrent network to enhance the few-shot classification performance. The core process of the MANN is in utilizing the network output as a key for content-based accessing of the external memory. After that, the read output of the memory is used for classifying the given unlabeled sample. Although the read access there is highly non-linear, an approximation as a linear transformation provides telling insights.

We recall that in [9], the read output vector $\mathbf{r}$ of the memory is formulated as

$$\mathbf{r} \leftarrow \sum_i w_r(i)\mathbf{M}_e(i). \tag{5}$$

A column vector in $\mathbf{M}_e(i)$ represents the content in slot i of the external memory array. Also, the value $w_r(i)$ is the read weight for slot i. The read weight $w_r(i)$ is given as

$$w_r(i) \leftarrow \frac{\exp\left(K(\mathbf{k}_r, \mathbf{M}_e(i))\right)}{\sum_j \exp\left(K(\mathbf{k}_r, \mathbf{M}_e(j))\right)} \tag{6}$$

where $K(\mathbf{k}_r, \mathbf{M}_e(i))$ is cosine correlation of the key vector $\mathbf{k}_r$ and the external memory content $\mathbf{M}_e(i)$. Let us approximate the exponentiation by a linear function: $\exp(K(\mathbf{k}_r, \mathbf{M}_e(i))) \simeq 1 + K(\mathbf{k}_r, \mathbf{M}_e(i))$. Then the non-linear formulation of the read weight can be linearly approximated as

$$w^r(i) \simeq \frac{1}{L+P}\left(1 + \frac{\mathbf{k}_r \cdot \mathbf{M}_e(i)}{\|\mathbf{k}_r\|\|\mathbf{M}_e(i)\|}\right) \tag{7}$$

where L is the number of slots in the memory array and $P = \sum_j K(\mathbf{k}_r, \mathbf{M}_e(j))$. The read weights written in a vector form $\mathbf{w}_r = \{w_r(0), \cdots, w_r(L-1)\}^T$ becomes

$$\mathbf{w}_r \simeq \frac{1}{L+P} \cdot \mathbb{1} + \frac{1}{(L+P)\,\|\mathbf{k}_r\|\,P_M}\mathbf{M}_e^T(\mathbf{k}_r)^T \tag{8}$$

where the norm $\|\mathbf{M}_e(i)\|$ has been approximated by the average norm P_M over all memory slots, and $\mathbb{1}$ denotes a column vector of length of L filled with 1s. In this formulation, the left term is a constant bias which is class-independent. Let us just take the right term and define it as an approximated read weight vector $\hat{\mathbf{w}}_r$

$$\hat{\mathbf{w}}_r \triangleq \frac{1}{\mathcal{P}}\mathbf{M}_e^T(\mathbf{k}_r)^T \tag{9}$$

where $\mathcal{P} = (L+P)\,\|\mathbf{k}_r\|\,P_M$. Eventually, we can approximate the read output $\hat{\mathbf{r}}$ as

$$\hat{\mathbf{r}} = \mathbf{M}_e\hat{\mathbf{w}}_r = \frac{1}{\mathcal{P}}\mathbf{M}_e\mathbf{M}_e^T(\mathbf{k}_r)^T. \tag{10}$$

In the MANN, this memory read output is multiplied by learnable label-dependent weights. The idea is that the inner product between the read output vector and each of the label-dependent weight vector is taken as the measure of how likely the corresponding label is for the given input sample. By collecting the label-dependent weights in a single matrix $\mathbf{\Phi}$, The inner product values can be formulated as $\mathbf{\Phi}\mathbf{r}$. By using the linearly approximated read output $\hat{\mathbf{r}}$, the inner product evaluation becomes

$$\frac{1}{\mathcal{P}}\mathbf{\Phi}\mathbf{M}_e\mathbf{M}_e^T(\mathbf{k}_r)^T \tag{11}$$

which indicates that the classification process of the MANN can be approximated as the inner product evaluation between the key vector $\mathbf{k}_r$ and the label-dependent weight vector projected on $\mathbf{M}_e$.

Now recall that the inner product evaluation of the vectors projected on $\mathbf{M}$ in our meta learner is given by $\mathbf{\Phi}\mathbf{M}\mathbf{M}^T\mathbf{g}^T$, which is essentially the same as (11). Very interestingly, this is to say that the external memory $\mathbf{M}_e$ of MANN and the linear projection on $\mathbf{M}$ in our meta learner effectively serve the same purpose. This is rather surprising because the external memory array $\mathbf{M}_e$ of MANN is derived for content-based access while $\mathbf{M}$ of our scheme is derived as a projection space where the error signals in the few-shot learning stage are forced to zero.

For the prototypical network of [10], a prototype is obtained by averaging the network outputs of the labeled samples for the given class. The network output of an unlabeled sample is classified by measuring distance to the prototypes. The class-specific weight vector in our approach bears a certain resemblance to the prototype vector for each class in the prototypical network in the sense that it provides class reference for classification purposes. However, there are significant differences as well. First, the prototypical network has no additional processing of the network output. In prototypical networks, classification is solely based on the distance between the network output and the prototype for each class. In contrast, our meta learner employs additional linear processing, consisting of both learnable and fixed components. Utilizing a projection space where the compressed network output is compared with label-dependent references is also a unique feature of our meta learner.

5 Conclusion

In this work, we proposed a meta learning algorithm aided by a linear signal transformer that performs null-space projection. Our algorithm uses linear nulling in the few-shot learning phase to quickly shape the classification space where network outputs could be better classified. In the process, subspaces that are deemed unnecessary for classification are essentially cut out. An interpretation is also given that our projection method can substitute for the content-based external memory access of MANN. Our meta learner achieves the best or near-best performance in various meta learning tasks, with tendency to show stronger relative performance as the task becomes more complicated as in *mini*ImageNet. In addition, our algorithm can be trained with a fewer number of initial training episodes.

Acknowledgments

This work is supported by by the ICT R&D program of Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) [2016-0-005630021001, Research on Adaptive Machine Learning Technology Development for Intelligent Autonomous Digital Companion].

References

[1] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, 2017.

[2] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[3] Lukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to remember rare events. *CoRR*, abs/1703.03129, 2017.

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[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] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *NIPS 2017 Workshop on Meta-Learning*, 2017.

[7] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[8] 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.

[9] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

[10] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4080–4090, 2017.

[11] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.