Efficient K-Shot Learning with Regularized Deep Networks

Donghyun Yoo[1], Haoqi Fan[2], Vishnu Naresh Boddeti[3], Kris M. Kitani[1]

[1]The Robotics Institute, School of Computer Science, Carnegie Mellon University
[2]Facebook
[3]Michigan State University

{donghyuy,kkitani}@cs.cmu.edu, haoqifan@fb.com, vishnu@msu.edu

Abstract

Feature representations from pre-trained deep neural networks have been known to exhibit excellent generalization and utility across a variety of related tasks. Fine-tuning is by far the simplest and most widely used approach that seeks to exploit and adapt these feature representations to novel tasks with limited data. Despite the effectiveness of fine-tuning, it is often sub-optimal and requires very careful optimization to prevent severe over-fitting to small datasets. The problem of sub-optimality and over-fitting, is due in part to the large number of parameters ($\approx 10^6$) used in a typical deep convolutional neural network. To address these problems, we propose a simple yet effective regularization method for fine-tuning pre-trained deep networks for the task of k-shot learning. To prevent overfitting, our key strategy is to cluster the model parameters while ensuring intra-cluster similarity and inter-cluster diversity of the parameters, effectively regularizing the dimensionality of the parameter search space. In particular, we identify groups of neurons within each layer of a deep network that share similar activation patterns. When the network is to be fine-tuned for a classification task using only k examples, we propagate a single gradient to all of the neuron parameters that belong to the same group. The grouping of neurons is non-trivial as neuron activations depend on the distribution of the input data. To efficiently search for optimal groupings conditioned on the input data, we propose a reinforcement learning search strategy using recurrent networks to learn the optimal group assignments for each network layer. Experimental results show that our method can be easily applied to several popular convolutional neural networks and improve upon other state-of-the-art fine-tuning based k-shot learning strategies by more than 10% of accuracy.

Introduction

While deep neural networks continue to exhibit excellent performance on large scale data, they suffer from severe over-fitting when learning with a small number of samples. The growing complexity and size of these networks, the main factor that contributes to their effectiveness in learning from large scale data, is also the reason for their failure to generalize from limited data. Learning from very few training samples (*i.e.* k-shot learning), is an important learning paradigm that is widely believed to be how humans learn new concepts
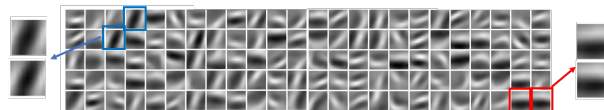
Figure 1: Convolutional Filters are often highly correlated. First layer visualization of LeNet. Correlated filters highlighted in red and blue.

as discussed in (Thorpe, Fize, and Marlot 1996) and (Li et al. 2002). However, k-shot learning still remains a key-challenge in machine learning.

Fine-tuning methods seek to overcome this limitation by leveraging networks that have been pre-trained on large scale data. Starting from such networks and carefully adapting their parameters have enabled deep neural networks to still be effective for learning from few samples. This procedure affords a few advantages: (1) enables us to exploit good feature representations learned from large scale data, (2) it is a very efficient process, often involving only a few quick iterations over the small scale, (3) scales linearly to a large number of k-shot learning tasks, and (4) is applicable to any existing pre-trained networks without the need for searching for optimal architectures or training from scratch.

Unfortunately, fine-tuning can be unstable, especially when the amount of training data is small. Large deep neural networks typically are comprised of many redundant parameters, with the parameters within each layer being highly correlated with each other. For instance consider the filters, shown in Fig.1, in the first layer of LeNet (LeCun et al. 1998) that was learned on the MNIST dataset. A number of filters are similar to other filters, *i.e.*, these filters functionally play the same role and tend to produce similar activations. The presence of a large number of correlated filters can potentially lead to over-fitting, especially when learning under a small sample regime.

To stabilize the learning process, we propose a simple yet effective procedure to regularize fine-tuning based k-shot learning approaches. The key idea of our approach is to identify the redundancies in the parameters and constrain their updates during fine-tuning. This is achieved by clustering the parameters in each layer of the network into multiple groups based on the similarity of their activations on a specific k-shot learning task. The parameters in each group share a common

update while ensuring intra-group similarity and inter-group diversity of activations. By grouping the model parameters and guiding the fine-tuning process with more supervisory signals, our approach is able to reduce the capacity of the network, to mitigate over-fitting and improve the effectiveness of pre-trained networks for k-shot learning.

We make the following contributions in this paper, (1) **grouping neurons by their activations for layer-wise clustering of parameters** while enforcing intra-group similarity and inter-group orthogonality of group activations, (2) **a hybrid loss function** for k-shot learning consisting of cross-entropy loss as well as triplet loss among the k-shot data, the latter providing more supervision for optimizing the model, and (3) **a reinforcement learning based mechanism to efficiently search for the optimal clustering of the parameters** across all the layers of the model. Our proposed k-shot learning approach affords the following advantages: (1) a task agnostic approach to k-shot learning that does not rely on any task-specific prior knowledge, (2) is applicable to any network without having to change the original network structure, and (3) a general purpose technique for decomposing the parameter space of high capacity deep neural networks.

To demonstrate the effectiveness of our approach, we experimentally evaluate it across two tasks: an one-shot domain-adaption task for matching images across three different domains and a k-shot transfer learning task. Our experimental results show that the proposed approach yields significant performance improvements over task agnostic fine-tuning approaches for small sample learning without the need for any task specific prior knowledge.

Related Work

k**-shot Learning:** One of the earliest work on one-shot learning for object categories was proposed by Fei-Fei *et al.* (Fei-Fei, Fergus, and Perona 2006). The authors developed a Bayesian learning framework with the premise that previously learned classes can inform a prior on the model parameters for a new class. Among recent work, powerful generative models have been developed that compose characters from a dictionary of parts (Wong and Yuille 2015) or strokes (Lake, Salakhutdinov, and Tenenbaum 2013). Such generative models have shown great promise on datasets with limited intra-class variation. Siamese networks (Koch, Zemel, and Salakhutdinov 2015) has been used to automatically learn feature representations where objects of the same class are closer together. Santoro *et al.* (Santoro et al. 2016) proposed the memory-augmented neural networks with an external content based memory. Wang and Hebert (Wang and Hebert 2016b; 2016a) propose a regression approach from classifiers trained on small datasets to classifiers trained on large datasets. Vinyals *et al.* (Vinyals et al. 2016) proposed matching networks that learns a non-parameteric k-nearest neighbor classifier through end-to-end learning, with the weights for the nearest neighbors are provided by an LSTM. Ravi and Larochelle (Ravi and Larochelle 2017) proposed LSTM-based meta-learner that uses its state to represent the learning updates of the parameters of a classifier for k-shot learning. Hariharan and Girshick (Hariharan and Girshick 2016) suggest a novel squared gradient magnitude regularization technique

and techniques to hallucinate additional training examples for small data classes. While these approaches have state-of-the-art performance on k-shot learning problems, they often utilize specific architectures designed for these problems. In contrast, we explore a more general method that can reuse existing networks, by fine-tuning them for k-shot learning.

Domain Adaptation: These methods seek to adapt a pre-trained model trained on one domain (source domain) to another domain (target domain). (Daume III 2009) proposed an adaptation method through feature augmentation, creating feature vectors with a source component, a target component, and a shared component. A Support Vector Machine (SVM) is then trained on this augmented feature vector. (Hoffman et al. 2013) used the feature representation of a pre-trained network like AlexNet that was trained on the 2012 ImageNet 1000-way classification dataset (Krizhevsky, Sutskever, and Hinton 2012). The authors replace the source domain classification layer with a domain-adaptive classification layer that takes the activations of one of the existing networks layers as input features. We are also interested in adapting a model learned on large scale data from the source domain to a model for the target domain with few examples. However, unlike these approaches, we propose a *task adaptive regularization* approach that improves the adaptability of exiting pre-trained networks to new target domains with limited training samples.

Proposed Approach

Our focus in this paper is the task of k-shot learning by fine-tuning an existing pre-trained network. We consider the setting where the pre-trained network was learned on a source domain with large amounts of data and the k-shot target domain consists of very few samples. To avoid the pitfalls of overfitting when training with only a few examples, we propose the following strategy. (1) We first search for similar activations to identify redundant filters, and then group them in a source domain. (2) After identifying the redundant parameters, the pre-trained network is fine-tuned with group-wise backpropagation in a target domain to regularize the network. The proposed layer-wise grouping method and model fine-tune by group-wise backpropagation effectively make the fine-tuning on k-shot samples more stable. However, our proposed grouping method has an important hyper-parameter, the number of groups. Deciding the number of groups for each layer is a non-trivial task as the optimal number of groups may be different in each layer. We suggest a hyper-parameter search method based on reinforcement learning to explore the optimal group numbers.

We now describe the three sub-components involved in our approach: (1) **grouping neurons by activations**, (2) **model fine-tuning for** k**-shot learning** and (3) **a reinforcement learning based policy for searching over the optimal grouping of the parameters**.

Grouping Neurons by Activations (GNA)

To identify redundant parameters to be grouped together for more stable fine-tuning, we define correlated filters as filters which have similar activations conditioned on a set of training
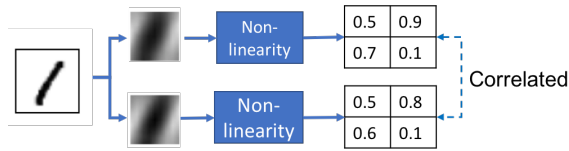
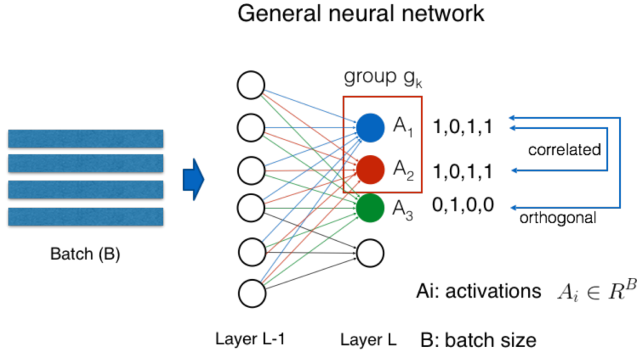Figure 2: Correlated filters in activation point of view

Figure 3: Correlation of parameters. Circles are neuron of a network. A_i is activation of neuron i. A_1 and A_2 have similar actions, so they are correlated. Thus, we can group A_1 and A_2 to one group g_k.

images. We would like to group these correlated filters as a means of regularizing the network. Fig. 2 illustrates a toy example of two convolutional filters with very correlated activations (heatmaps). Since the two filters have similar patterns, their outputs are very similar.

Now consider the fully connect layer of a neural network illustrated in Fig 3. Given a batch of data B as input, we can pass each data element (image) through the network to compute the activations at layer L. A_i is the output of the non-linear activation function of the i-th neuron in layer L. If we compare activation A_i to another activation A_j over the input data, we can measure the correlation between neurons. In our example, A_1 and A_2 have similar output patterns over the batch image data whereas, A_1 and A_3 have different output patterns. This implies that A_1 and A_2 are good candidates for grouping.

In our proposed approach, we use a clustering algorithm to group similar neurons based on their activations over the k-shot training data (*e.g.*, one image for each category). In particular, we use k-means clustering to group the neurons and the number of clusters k for each layer is learned via a reinforcement learning procedure described later.

Backpropagation with Groups

Once the redundant parameter groups in each layer are identified, an effective regularization method is required during fine-tuning to prevent over-fitting. To restrain the redundant parameters from overfitting, we can consider updating the parameters in a group with the same gradient because the gradients of the redundant weights in the same group would be expected to be very similar to each other. From this insight,

we update the parameters of each group g_k by a shared gradient W_a during learning to regularize the network. The shared update is computed as the average gradient of all the filters in the group *i.e.*, $\nabla W_a = \frac{1}{|g_k|} \sum_{i \in g_k} \nabla W_i$, where ∇W_i is the gradient of weight W_i. We demonstrate the feasibility of this backpropagation by an average gradient with domain adaptation and transfer learning experiments described later.

Loss Functions

The low sample complexity of typical k-shot learning results in extremely noisy gradient updates for each k-shot entity. To provide more supervisory signals to the learning process, we introduce a triplet loss to the network optimization objective. The triplet loss is similar to the one introduced by Schroff *et al* (Schroff, Kalenichenko, and Philbin 2015). The triplet loss serves the twin purposes of providing more supervisory signals to increase the separation between the k-shot entities as well as to reduce the noise in the gradient signal by averaging over larger number of loss terms

We define the triplet loss for the k-shot learning problem as:

$$L_{triplet} = \sum_{i,j,n} \left[d\left(f(x_i), f(x_j)\right) - d\left(f(x_i), f(x_n)\right) + \alpha \right]_+ \tag{1}$$

where $f(x)$ is the output of the network for input x, i,j are indices of samples belonging to the same class and n is the index of sample belonging to a different class, $d\left(f(x_i), f(x_j)\right)$ is the distance between $f(x_i)$ and $f(x_j)$ and $[\cdot]_+$ denotes the margin maximizing loss. The distance can be the Euclidean distance $\|\cdot\|_2^2$ and total variation $\|\cdot\|_1$ for regression and classification tasks respectively. We note that the triplet loss reduces to a margin loss for one-shot learning. The margin loss is defined as:

$$L_{margin} = \sum_{i,n} \left[-d\left(f(x_i), f(x_n)\right) + \alpha \right]_+. \tag{2}$$

In addition to the classification loss described above, it is important to ensure that the intra-group activations are similar to each other, while the inter-group activations are orthogonal to each other. We augment the k-shot learning loss function with these two criterion during training. Let the activation of a filter i in the l-th layer be A_i^l. The intra-group similarity loss is defined as:

$$L_{intra} = \sum_l \sum_n \sum_{i,j \in g_n} \left\| A_i^l - A_j^l \right\|_2. \tag{3}$$

The inter-group orthogonality loss is defined as:

$$L_{inter} = \sum_l \sum_{i,j} \| M_i^{l\top} M_j^l \|_F^2 \tag{4}$$

where M_i^l and M_j^l are matrices with the activations of all the weights in group g_i and g_j respectively at the l-th layer and $\|\cdot\|_F^2$ is the squared Frobenius norm.

Our k-shot learning task is trained to optimize a combination of the loss functions described above. The total loss is described as the following expression:

$$L = L_{class} + \alpha L_{intra} + \beta L_{inter} + \gamma L_{triplet} \tag{5}$$

where α, β and γ are hyper-parameters that control the importance of each of the loss terms.

Hyper-Parameter Search Through Reinforcement Learning

The performance of the proposed approach is critically dependent on the number of clusters that the weights in each layer are grouped into. Manually selecting the number of clusters can lead to sub-optimal performance while an exhaustive search is prohibitively expensive. This problem is exacerbated as the number of layers in the network increases. Common methods for determining hyper parameters are brute force search, grid search, or random search. While brute force search is guaranteed to find the optimal solution, it is very time consuming and is usually intractable. Grid search is the most used method for hyper-parameter selection, but is still limited by the granularity of the search and can potentially end up being computationally expensive. On the other hand, surprisingly, Bergstra and Bengio (Bergstra and Bengio 2012) suggest that random search is more effective than grid search. Recently, Hansen (Hansen 2016) proposed a reinforcement learning approach for determining hyper-parameters. Building upon this, Zoph and Le (Zoph and Le 2016) proposed a neural network architecture to find the optimal hyper-parameter of a neural architecture through reinforcement learning. In this work, we adopt a similar approach to determine the optimal number of clusters in each layer of the network for k-shot learning.

We pose the hyper-parameter search problem as a reinforcement learning problem to find a locally optimal layer-wise group size for the entire network. Figure 4(a) shows our reinforcement learning problem, where the environment is a pre-trained network that we wish to fine-tune for k-shot learning. Intuitively the policy network implicitly learns the relation between different groupings of the layer weights and the performance of the network. We model the problem as a fixed horizon episodic reinforcement learning problem where all actions (layer-wise prediction of number of clusters) have an equal affect on the final outcome. We represent the sequence of actions as $a_{1:L}$, where a_l is the action at the l-th layer, predicting the number of clusters in the l-th layer. We define the state as a vector that has the number of groups of each layer.

$$S = \{n_1, n_2, ..., n_i\} \qquad (6)$$

where n_i is the number of groups in layer i.

Our agent's policy network is a Long Short-Term Memory (LSTM) by Hochreier and Schmidhuber (Hochreiter and Schmidhuber 1997) as shown in Fig. 4(b) and is learned through a policy gradient method. The time horizon of the LSTM is equal to the number of layers in the pre-trained network. The output of the LSTM consists of a fully connected layer followed by a softmax layer to predict the probabilities of the action a_l at the l-th layer. The input of the policy network at the l-th layer, $I \in \mathbb{R}^{N_a+1}$ is a vector created by the concatenation of the number of filters (a single integer) in the l-th layer and the output action at the previous layer (one hot encoding), where N_a is the number of actions.

We adopt the policy gradient method (Zoph and Le 2016) to learn our agent's policy that maximizes the expected

(a) Hyper-Parameter Search Framework

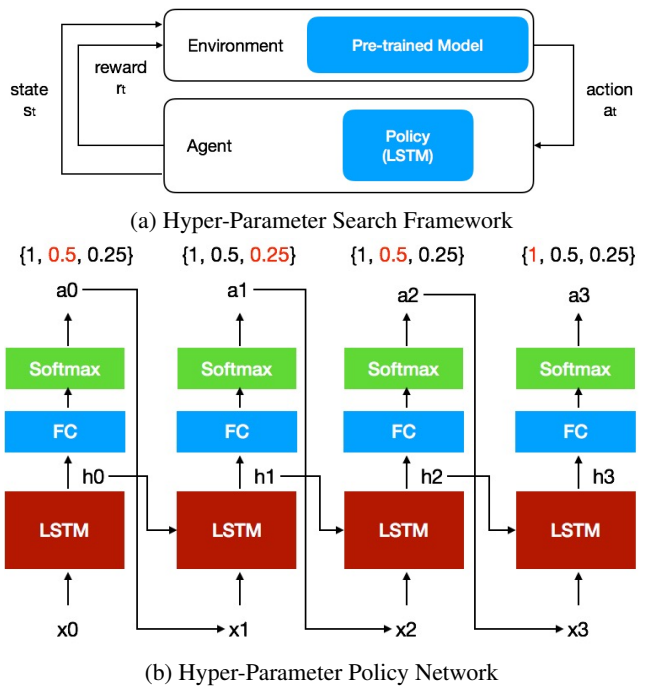(b) Hyper-Parameter Policy Network

Figure 4: We adopt a deep reinforcement learning framework to search over the clustering hyper-parameters, number of groups. Furthermore, we adopt an LSTM as our policy network to determine the numbers of clusters in each layer.

accuracy of the proposed fine-tuning process through parameter clustering since the cumulative reward R is non-differentiable. We define agent's reward returned by the environment as the accuracy A_{ft} of the fine-tuned model on the validation set for a valid action and -1 for an invalid action (impossible group size).

$$R = \begin{cases} A_{ft} & \text{if an action is valid} \\ -1 & \text{otherwise} \end{cases}$$

A_{ft} is the accuracy of the fine-tuned network of which parameters are clustered and calculated on the validation set. In each episode the agent predicts a list of actions $a_{1:L}$ corresponding to the number of groups in the L layers of the network. The parameters in each layer of the pre-network are clustered into the number of groups as determined by the action. The pre-trained network is then fine-tuned on the k-shot data until convergence, after which the validation accuracy of the network is recorded to use as a reward R for the agent. The agent's policy network is then updated by backpropagating the gradients computed from the loss Eq. 9. As the episodes are repeated, the policy network's predictions inch closer to the optimal number of parameter clusters in each layer, in turn resulting in a gradual increase in the accuracy of the fine-tuning process.

To estimate the optimal clustering of the network parameters, the policy network's parameters θ_c are optimized to maximize the expected reward J (θ_c), computed over all fu-

---

Algorithm 1: Grouping and average gradient update algorithm

Given a pre-trained network M, source domain samples and k samples of target domain
Step 1) Grouping and fine-tuning in a source domain
for *each iteration* **do**
> Generate actions that change the numbers of groups for each layer by a recurrent policy network
> Set the numbers of groups for each layers
> Cluster the parameters of each layer in network M to K groups by k-mean clustering
> Fine-tune the network M with source domain training samples
> $R_k \leftarrow$ the validation accuracy of fine-tuned network
> Update recurrent policy network by policy gradient update equation (9)

end
Step 2) Fine-tuning in a target domain
for *each iteration* **do**
> **for** *each group g_i, $i \in [1, K]$* **do**
>> compute
>> $L_{intra} = \sum_l \sum_n \sum_{i,j \in g_n} \left\| A_i^l - A_j^l \right\|_2$
>> compute $L_{inter} = \sum_l \sum_{i,j} \| M^{l_i^\top} M_j^l \|_F^2$
>> compute $L_{triplet}$ by equation (1)
>
> **end**
> $L = L_{class} + \alpha L_{intra} + \beta L_{inter} + \gamma L_{triplet}$
> Update by average gradient
> $\nabla W_a = \frac{1}{|g_k|} \sum_{i \in g_k} \nabla W_i$

end

---

ture episodes from current state.

$$J(\theta_c) = E_{P(a_{1:T}; \theta_c)}[R] \quad (7)$$

Since the reward signal is non-differentiable, we use an approximate policy gradient method to iteratively update the policy network. In this work, we use the REINFORCE rule from (J.Williams 1992)

$$\nabla_{\theta_c} J(\theta_c) = \sum_{t=1}^{T} E_{P(a_{1:T}; \theta_c)}[\nabla_{\theta_c} log P(a_t | a_{(t-1):1}; \theta_c) R] \quad (8)$$

The above quantity can be empirically approximated as:

$$\frac{1}{m} \sum_{k=1}^{m} \sum_{t=1}^{T} \nabla_{\theta_c} log P(a_t | a_{(t-1):1}; \theta_c) R_k \quad (9)$$

where R_k is a reward of k episode, and m is the number of episodes. $P(a_t | a_{(t-1):1}; \theta_c)$ denotes the probability of a history of actions $a_{1:T}$ given policy-defining weights θ_c. Our complete algorithm is presented in Algorithm 1.

Experiments

The usefulness of the proposed method is verified through experiments on two tasks, domain adaptation and transfer learning. In both tasks, we show how our approach can be

Table 1: Experimental Evaluation: k-shot domain adaptation on Office dataset

| Method	Feature type	Accuracy(%)
Late fusion (Hoffman et al. 2013)	DeCAF-7	64.29
Late fusion (Hoffman et al. 2013)	ResNet-18	71.08
Daume (Daume III 2009)	DeCAF-7	72.09
Daume (Daume III 2009)	ResNet-18	76.25
Fine-Tuning	ResNet-18	70.07
Fine-Tuning + margin loss	ResNet-18	70.34
Fine-Tuning + dropout	ResNet-18	76.66
GNA	**ResNet-18**	**79.94**
GNA + margin loss	**ResNet-18**	**82.16**
GNA + margin loss+Greedy	**ResNet-18**	**83.16**
GNA + margin loss+RL	**ResNet-18**	**85.04**
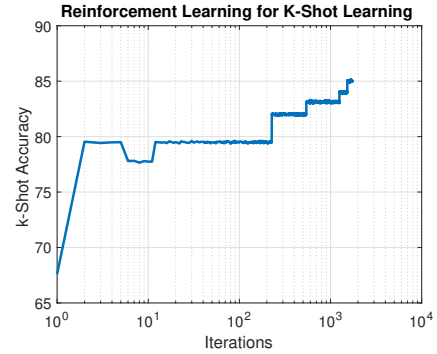
Figure 5: Progression of hyper-parameter search through reinforcement learning through the iterations for K-shot domain adaptation on Office dataset

used to learn a new model from only a few number of examples. We present the results of multiple baseline variants of our proposed approach, (1) **Fine-Tuning:** the standard approach of updating a pre-trained network on k-shot data with cross-entropy loss, (2) **Fine-Tuning+Triplet Loss:** updating a pre-trained network on k-shot data with cross-entropy loss and the triplet loss, (3) **GNA:** proposed grouping neurons by activation with cross-entropy loss and manual hyperparameter search, (4) **GNA+Triplet Loss:** proposed grouping neurons by activation with cross-entropy and triplet loss and manual hyperparameter search, (5) **GNA+Triplet Loss+Greedy:** proposed grouping neurons by activation with cross-entropy and triplet loss and greedy hyperparameter selection, and (6) **GNA+Triplet Loss+RL:** proposed grouping neurons by activation with cross-entropy and triplet loss and RL based hyperparameter search.

Domain Adaptation

For this task, we consider the Office dataset introduced by (Saenko et al. 2010) consisting of a collection of images from three distinct domains: Amazon, DSLR and Webcam. The dataset consists of 31 objects that are commonly encountered in office settings, such as keyboards, file cabinets, laptops *etc*. We follow the experimental protocol used in (Hoffman et al. 2013), and consider domain adaptation between the Amazon (source) and the Webcam (target) images.

Table 2: Experimental Evaluation: (Top) one-shot transfer learning from CIFAR-100 to CIFAR-10 and (Bottom) 10-shot transfer learning from CIFAR-100 to CIFAR-10.

Method	Accuracy(%)
Fine-Tuning	29.58
Fine-Tuning + margin loss	33.44
GNA	**32.70**
GNA+margin loss	**34.43**
GNA+margin loss+greedy	**33.50**
GNA+margin loss+RL	**35.95**

(a) 1-shot learning

Method	Accuracy(%)
Fine-Tuning	56.00
Fine-Tuning + margin loss	57.32
Fine-Tuning + triplet loss	58.17
GNA	**57.96**
GNA+margin loss	**59.05**
GNA+triplet loss	**58.56**
GNA+triplet loss+greedy	**58.56**
GNA+triplet loss+RL	**60.30**

(b) 10-shot learning

(a) one-shot

(b) 10-shot

Figure 6: Progression of hyper-parameter search through reinforcement learning through the iterations for K-shot transfer learning from CIFAR-100 to CIFAR-10

The experiment is conducted on 16 out of the 31 objects that are also present in the ImageNet dataset. Our pre-trained network is the ResNet-18 architecture by (He et al. 2016) trained on the ImageNet dataset. Our action space for this experiment is the number of possible clusters in each layer $a = \{2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8\}$, or equivalently the action space is the number of possible groups per layer. We set the action space to $\{1, 2, 4, ..., N_f\}$, where N_f is the number of filters. The minimum number of groups is one. The maximum number of groups is the same as the number of weights. In this work, we define the actions (number of possible clusters) as $\{2^0, 2^1, ..., N_f\}$ to reduce the size of the action space and speed up the search process. However, in general, the action space can also be densely discrete like $\{1, 2, 3, ..., N_f\}$.

We use 20 source images per each class for clustering the parameters and fine-tune the model with 16 one-shot examples, one image per class. The performance of our proposed approach is compared to the baselines in Table 1, and Figure 5 shows the progression of the reinforcement learning based hyper-parameter search on the k-shot learning accuracy. *Late fusion* (Hoffman et al. 2013) and *Daume* (Daume III 2009) are compared as the baselines. The *Late fusion* and *Daume* use DeCAF-7 features in their works, but we also apply their method with ResNet-18 features for fair comparison with our method. For fine-tuning, the learning-rate is 0.01, and it is changed to 0.001 after 1000 iteration. We tried 10 random runs with randomly selected different dataset to get average performance.

We also compare our proposed method with well-known regularization methods. We performed a comparison experiment with dropout on the Office dataset experiment. We add dropout layers to the pre-trained network and fine-tune it in source domain and one-shot target domain. The one-shot accuracy of normal fine-tuning with dropout regularization in target domain is 76.66%. Without dropout, the fine-tuning accuracy is 70.07% shown in Table 1. The accuracy of our pro-
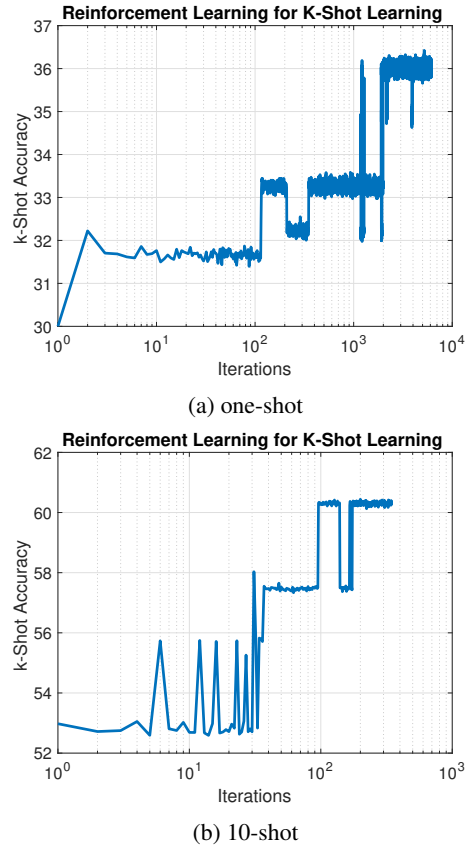
posed group-level regularization in the same setup is 79.94%. Notice that our group-level regularization method indeed outperforms the dropout regularization method by 3% in k-shot domain adaptation experiment.

We note that the clustering hyper-parameter search through the reinforcement learning is able to efficiently search the hyper-parameter space and find better parameter groupings compared to both manual and greedy search. For the *manual baseline*, we initialize the number of groups in all the layers to two and compute the accuracy of the network. We then compute the accuracy of the of the network by doubling and halving the number of groups in a layer. The action (doubling or halving) that results in higher accuracy is selected. We repeat this process and update the number of groups iteratively through the process described above.

For the *greedy baseline(Greedy)*, we set the number of groups in the first layer to two and compute the accuracy of the original network. If the accuracy is greater than before, then the number of groups is doubled, otherwise we set the number of groups to the previous number and move to the next layer. We repeat this procedure until the last layer.

Table 3: k-shot classification performance as a function of number of samples per category

The number	Accuracy (%)	
of clustering	w/o clustering	w/ clustering
25 shot	80.41	84.48
20 shot	76.90	82.09
15 shot	81.63	84.01
10 shot	70.64	72.88
5 shot	68.84	68.93
1 shot	52.25	53.77

(a) Accuracy

The number	Standard deviation	
of clustering	w/o clustering	w/ clustering
25 shot	5.68	0.76
20 shot	2.66	1.44
15 shot	5.16	0.79
10 shot	5.31	3.79
5 shot	6.95	2.72
1 shot	10.02	10.66

(b) Standard deviation

Table 4: K-shot classification performance as we vary layers where filters are clustered.

	Accuracy(%)	
the number of layers	w/o clustering	w/ clustering
1 layer	80.41	82.87
3 layers	80.41	81.68
5 layers	80.41	82.98
7 layers	80.41	83.03
all	80.41	84.08

(a) Accuracy

	Accuracy(%)	
the number of layers	w/o clustering	w/ clustering
1 layer	5.68	4.47
3 layers	5.68	4.01
5 layers	5.68	3.65
7 layers	5.68	3.11
all	5.68	0.76

(b) Standard deviation

Transfer Learning

In the domain adaptation experiment that has same classes, we showed our proposed method outperforms the baseline approaches. We can apply the grouping method to a task that the source classes are different from the target classes. We consider the task of transfer learning, where the k-shot learning task (target) is different from the pre-trained network task (source). Our pre-trained network is the ResNet-18 architecture trained on the CIFAR-100 dataset while the k-shot learning task is classification on the CIFAR-10 dataset. For the transfer learning setting, we select the classes that are different from ten target classes as our source classes. Our action space for this experiment is the number of possible clusters in each layer $a = \{2^0, 2^1, 2^2, 2^3, 2^4\}$. We consider two different k-shot settings, one with $k = 1$ and another with $k = 10$. The k-shot data are chosen randomly from the target training set for fine-tuning and we evaluate on the entire target test set. The performance of our proposed approach is compared to the baselines in Table 2 both for one-shot learning as well as for 10-shot learning. Our proposed margin loss improves the accuracies of the grouping method as well as fine-tuning. The accuracies of our grouping methods are higher than the fine-tuning result. Thus, the proposed method with RL search outperforms the baseline fine-tuning approach by 6.37% in 1-shot learning and 4.3% in 10-shot learning.

Effect of Sample Size K

In this experiment we are interested in comparing the performance of our proposed approach as we vary the difficulty of the k-shot learning problem. We consider different ranges of k, the number of samples per category. Table 3 presents the results of GNA with clustering and standard fine-tuning with-

out clustering as we vary k. Unsurprisingly, the performance decreases and there is greater uncertainty as k is lowered to one-shot learning. But we observe a consistent improvement in performance with our clustering approach in comparison to the standard fine-tuning procedure.

Effect of Clustering Across Layers

It is commonly believed that most deep convolutional neural networks have highly redundant filters at the initial layers only. If this is indeed the case, applying our clustering method to layers other than the initial few layers should not be helpful. To test this hypothesis, we perform clustering to increasing number of layers, starting at the initial layers of the network. For this experiment we considered a pre-trained ResNet-18 network trained on a few categories in the CIFAR-10 dataset and used the other categories as the k-shot learning task. The results of GNA in Table 4 surprisingly does not confirm our hypothesis. We found that all layers of the deep network did consist of redundant filters for the k-shot learning task. In fact, applying our method to all the layers of the network resulted in the best performance. This experiment suggests that large convolutional neural networks could potentially consist of redundant parameters even in the higher layers, necessitating search over the entire hyper-parameter space of parameter groupings. This motivates the need for efficient techniques to search the hyper-parameter space, like the one we proposed in this paper.

Conclusion

In this paper we proposed a new regularization method for fine-tuning a pre-trained network for k-shot learning. The key idea of our approach was to effectively reduce the dimensionality of the network parameter space, by clustering the weights in each layer while ensuring intra-group similarity

and inter-group orthogonality. To provide additional supervision to the k-shot learning problem we introduce a triplet loss to maximize the separation between the k-shot samples. Lastly, we introduced a reinforcement learning based approach to efficiently search over the hyper-parameters of our clustering approach. The experimental results demonstrate that our proposed regularization technique can significantly improve the performance of fine-tuning based k-shot learning approaches.

Acknowledgments

References

Bergstra, J., and Bengio, Y. 2012. Random search for hyper-parameter optimization. *Journal Machine Learning Research* 13:281–305.

Daume III, H. 2009. Frustratingly easy domain adaptation. *arXiv*.

Fei-Fei, L.; Fergus, R.; and Perona, P. 2006. One-shot learning of object categories.

Hansen, S. 2016. Using Deep Q-learning to Control Optimization Hyperparameters. *Optimization and Control*.

Hariharan, B., and Girshick, R. 2016. Low-shot visual recognition by shrinking and hallucinating features. *arXiv preprint arXiv:1606.02819*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 630–645. Springer.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Hoffman, J.; Tzeng, E.; Donahue, J.; Jia, Y.; Saenko, K.; and Darrell, T. 2013. One-shot adaptation of supervised deep convolutional models. *arXiv preprint* 1312.6204.

J.Williams, R. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *In Machine Learning*.

Koch, G.; Zemel, R.; and Salakhutdinov, R. 2015. Siamese neural networks for one-shot image recognition. *32nd International Conference on Machine Learning* 2252–2259.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. *In Advances in neural information processing systems* 1097–1105.

Lake, B. M.; Salakhutdinov, R.; and Tenenbaum, J. 2013. One- shot learning by inverting a compositional causal process. *NIPS*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-Based Learning Applied to Document Recognition. *Procedding of IEEE*.

Li, F. F.; VanRullen, R.; Koch, C.; and Perona, P. 2002. Rapid natural scene categorization in the near absence of attention. *Proceedings of the National Academy of Sciences* 99(14):9596–9601.

Ravi, S., and Larochelle, H. 2017. Optimization as a model for few-shot learning. *International Conference on Learning Representations (ICLR)*.

Saenko, K.; Kulis, B.; Fritz, M.; and Darrell, T. 2010. Adapting visual category models to new domains. *Computer Vision–ECCV 2010* 213–226.

Santoro, A.; Bartunov, S.; Botvinick, M.; Wierstra, D.; and Lillicrap, T. 2016. One-shot Learning with Memory-Augmented Neural Networks. *arXiv preprint*.

Schroff, F.; Kalenichenko, D.; and Philbin, J. 2015. Facenet: A unified embedding for face recognition and clustering. *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.

Thorpe, S.; Fize, D.; and Marlot, C. 1996. Speed of processing in the human visual system. *nature* 381(6582):520.

Vinyals, O.; Blundell, C.; Lillicrap, T.; Kavukcuoglu, K.; and Wierstra, D. 2016. Matching Networks for One Shot Learning. *arXiv preprint*.

Wang, Y.-X., and Hebert, M. 2016a. Learning from small sample sets by combining unsupervised meta-training with cnns. In *Advances in Neural Information Processing Systems*, 244–252.

Wang, Y., and Hebert, M. 2016b. Learning to learn: Model regression networks for easy small sample learning. *ECCV*.

Wong, A., and Yuille, A. L. 2015. One Shot Learning via Compositions of Meaningful Patches. *In Proceedings of the IEEE International Conference on Computer Vision* 1197–1205.

Zoph, B., and Le, Q. V. 2016. Neural architecture search with reinforcement learning. *arXiv*.