Extended Abstract: Semi-Supervised Meta Learning

Recently, unsupervised meta learning methods have been introduced [1], [2], where meta training tasks for few-shot image classification are automatically constructed from an unlabeled data set, and a meta learner is then trained on these tasks. We extend the CACTUs method [1] to a semi-supervised setting by training a meta learner on a combination of tasks that were constructed using CACTUs and a small number of given pre-defined tasks. We show that in certain settings where only a small number of labeled data and a larger amount of unlabeled data are available, semi-supervised CACTUs can significantly outperform both supervised meta learning (on the small number of tasks) and unsupervised CACTUs.

Furthermore, we propose a way of improving the performance of CACTUs in the semi-supervised setting by using Constrained DeepCluster, a new variant of DeepCluster [3] that uses constrained k-means [4] instead of k-means. Our experiments suggest that CACTUs with Constrained DeepCluster seems to perform better than CACTUs with unsupervised DeepCluster, but more experiments would be necessary in order to confirm this, because the difference is small enough that it could be explained by random chance.

Code

Code is available here:

- `https://github.com/cmacho/Semi-Supervised-Meta-Learning`
- `https://github.com/cmacho/deepcluster` (includes Constrained DeepCluster)

The readme of the first repository contains detailed instructions on how to run all the experiments that we are reporting on.

Semi-Supervised Meta Learning

Clemens Macho
Department of Computer Science
Stanford University
cmacho@stanford.edu

1 Introduction

One of the challenges of meta learning is that it requires a large amount of meta training data in order to perform well on previously unseen downstream tasks. Recently, unsupervised meta learning methods for few-shot image classification have been introduced which alleviate this need for labeled data [1], [2]. In these methods, tasks are constructed automatically from a data set of unlabeled images and meta learning is performed on these constructed tasks. In this work, we consider the situation where a small number of (labeled) meta training tasks is available in addition to a larger unlabeled set of images. In other words, we want to do semi-supervised meta learning. We train a meta learner on a combination of automatically constructed tasks and a small number of human-specified tasks and show that it significantly outperforms a meta learner that is trained on only one of the two kinds of data. We also suggest a method of using the given meta training tasks in order to guide the process of automatically generating further tasks and we show experimentally that this can improve performance slightly.

Our work is based on the CACTUs method [1] for unsupervised meta learning. In the next subsection we will briefly recap this method.

1.1 CACTUs

CACTUs creates tasks for meta learning through the following procedure. A data set of unlabeled images is used to learn an embedding from pixel space into a lower dimensional vector space, using an unsupervised learning algorithm such as DeepCluster [3] or ACAI [5]. These algorithms are known to produce features that are useful for image classification. Throughout this project, we use DeepCluster.

As a next step, the embeddings of all images from the data set are clustered using k-means. A label is assigned to each cluster. In order to create a meta training task for n-way k-shot classification with q query images per class, we pick n labels at random and then pick $k + q$ images (to be used as support and query images) from the corresponding clusters.

This is akin to how meta tasks are usually created when using a labeled image data set. In that situation, a task is also created by first picking n labels at random and then picking $k + q$ images with the corresponding label.

The constructed meta training tasks are then used to train a meta learning model such as MAML [6] or ProtoNets [7].

1.2 DeepCluster

We briefly describe the DeepCluster algorithm, which is used as a part of CACTUs. In DeepCluster, a convolutional neural network (ConvNet) is trained to learn an embedding from pixel space into a feature space. The ConvNet is initialized randomly and then the following three steps are repeated:

- Embeddings are computed for all images in the data set, using the ConvNet.

- The embedded images are clustered using k-means. Pseudo labels corresponding to the clusters are assigned to the images.

- An extra fully connected layer with randomly initialized weights is added at the end of the convolutional network. The network is then trained to predict the pseudo labels from the previous step. Afterwards, the extra fully connected layer is removed again.

2 Datasets and Problem Setup

We use the mini-imagenet data set, which consists of 600 images for each of 100 classes. It is customary in meta learning for few shot image classification to split this data set along image classes in order to be able to demonstrate that the model can perform well, even on tasks that it was not trained on [6], [8]. Namely, 20 of the image classes are used to sample meta test tasks from, 16 are used for meta validation and the other 64 are used for meta training. For semi supervised learning, we took this split a step further. We split the training set into a set of 52 classes which are used as an unlabeled data set and 12 classes from which we sampled a small number of meta training tasks.

By splitting the data like this, we simulate a real world scenario where we have a small number of labeled tasks and want to augment meta training with a larger number of unlabeled images that may not be from the exact same distribution. For instance, the labeled tasks could be from medical images of chest X-rays and the unlabeled images could be photos from a social media website (although we do not test this exact scenario in this project).

From the 12 image classes that were designated to be used for labeled meta training tasks, I sampled 100 tasks, each consisting of one support image and five query images for each of five classes. Thus, each meta training task contains a total of 30 images. Only these 100 tasks, as well as the $52 \cdot 600$ unlabeled images, are available during meta training. For the images in the labeled tasks, the meta learner also has no access to the underlying labels from the mini imagenet data set. Instead, it only gets to see the task specific labels of each task, which range from 0 to $n-1=4$ and are not consistent across tasks.

This last restriction is not necessarily realistic for real world applications. It is hard to imagine a real world scenario where we would not have access to the underlying labels for the images in the labeled

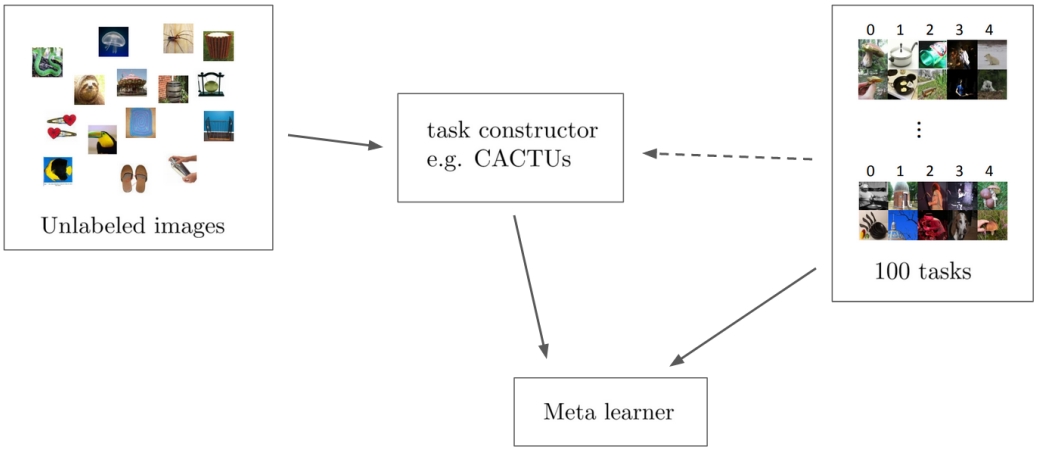


Figure 1: A meta learner is trained on a combination of a small number of labeled tasks and tasks that are constructed from unlabeled images. Note that, while in this diagram every task has one support image and one query image for each class, we actually used tasks with 1 support image and 5 query images in our experiments. The dashed arrow is relevant for Constrained DeepCluster, discussed in section 4.

tasks. However, this restriction makes sure that standard semi-supervised methods are not as easily applicable and that we instead have to develop methods specific to the meta learning setting.[1]

During meta training, we augment the 100 tasks by

- switching support images with query images at random (within the same class)
- permuting the labels
- flipping images horizontally.

We train on a mix of the 100 labeled tasks and tasks that were constructed from the unlabeled images with a method such as CACTUs, see figure 1.

3 Unsupervised Learning Results

Since CACTUs is a core component of our semi-supervised learning pipeline, we first aimed to reproduce the results from the paper. Unfortunately, the hyper parameters that were used for DeepCluster in [1] are not specified in the paper (and DeepCluster is also not included in the code base that came with the paper). Therefore, we tried several different hyperparameters and compared the resulting accuracy for (unsupervised) CACTUs-ProtoNets.

Table 1 shows results for five choices of hyperparameters, as well as (in the last row) the accuracy as reported by the paper. For most of these experiments we trained DeepCluster on the 52 classes that were designated as unlabeled data instead of on the full training set of 64 classes, because we wanted to be able to reuse the results for semi-supervised learning. Therefore it is expected that the resulting accuracy should be slightly worse than the one reported in [1].

Experiment number	number of classes from mini imagenet	modified first layer of AlexNet	number of clusters	epochs for training	reassign labels after every X epochs	Validation accuracy	Test accuracy
1	52	96 10×10 filters, stride 1	500	200	1	31.57%	35.23%
2	64	96 10×10 filters, stride 1	700	200	1	33.51%	35.18%
3	52	96 10×10 filters, stride 1	500	300	1	32.35%	-
4	52	96 10×10 filters, stride 1	600	200	1	32.25%	-
5	52	96 10×10 filters, stride 1	500	1500	3	33.75%	**36.52%**
Hsu et. al	64	?	?	?	?	-	**39.18%**

Table 1: Accuracy for CACTUs-ProtoNets with various choices of hyperparameters for DeepCluster.

In experiment 1 in table 1, we used the default hyperparameters from the official DeepCluster implementation by the authors of [3]. However, these hyperparameters are optimized for ImageNet, which is much larger than the mini-imagenet data set that we are using. When training on mini-imagenet, 200 epochs may not provide enough SGD updates in order for the convolutional network to produce useful features. This may explain the low accuracy that we see in experiment 1. Experiment 2, where we used the full training set of 64 mini-imagenet classes, also failed to reach much higher accuracy, showing that the slightly smaller size of our data set does not fully explain the difference in accuracy between our experiment and [1]. Experiments 3 and 5 showed that increasing the number of epochs can improve accuracy. Experiment 4 shows that increasing the number of clusters could potentially also be a promising direction, because experiment 4 achieved higher accuracy than experiment 1.

We decided to use the settings from experiment 5 (i.e. training for 1500 epochs) for all further experiments (including the ones with Constrained DeepCluster, discussed in section 4), because the accuracy seemed close enough to the one in [1], although it could probably be improved further given more time and computational resources.

Table 1 also shows that the test accuracy is usually much higher than the validation accuracy in these experiments. This is probably because the validation set contains more difficult image classes that are less similar to the ones in the training set. Therefore, validation accuracy is not a very good predictor of test accuracy. Reshuffling the image classes and creating a new train/val/test split could be useful, however, we decided to stick with the standard split that came with the dataset.

[1]For instance, if we had access to the underlying labels for the images of labeled tasks, we could train directly on these labels, in addition to pseudo labels, throughout the training of DeepCluster. Since supervised learning is known to work well for image classification, this would likely improve the performance of DeepCluster.

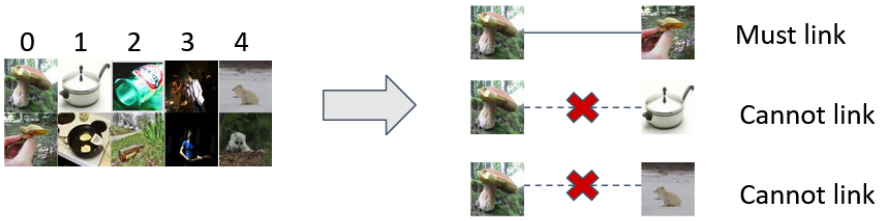


Figure 2: From each of the given meta training tasks, we can extract constraints for constrained k-means.

4 Semi Supervised Methods

Our default method is to simply train ProtoNets or MAML on a mix of tasks generated with CACTUs (using the unlabeled image data) and tasks sampled from the 100 labeled tasks. We try mixing these two kinds of data at several different ratios and compare the resulting validation accuracy for ProtoNets in order to determine the best ratio.

In addition, we try several experiments using constrained k-means (also known as COP-k-means) [4]. COP-k-means is a variant of k-means that can make use of *must-link* and *cannot-link* constraints, i.e. constraints that assert that a pair of points must be assigned to the same cluster or cannot be assigned to the same cluster. Such constraints can easily be extracted from each of the 100 labeled tasks, see figure 2. We can then use these constraints and run COP-k-means on the union of the unlabeled images and the images contained in the 100 labeled tasks. Since the unlabeled images contain different classes than the labeled tasks (see section 2), we should not expect that this will necessarily improve the quality of how the unlabeled images are clustered. However, when we use COP-k-means instead of k-means within the DeepCluster algorithm and train the ConvNet on both the images from the unlabeled data set and the images from the 100 tasks, this may provide a weak signal for the ConvNet from which it can learn that certain images are related while others are unrelated. It seems reasonable to assume that this could indeed improve the quality of the features learned by the ConvNet.

When we use COP-k-means instead of k-means within the DeepCluster algorithm, we call the resulting algorithm Constrained DeepCluster.

To be more precise, we use a slight variation of COP-k-means. Given a group of points with must-link constraints between every pair of points, the standard version of COP-k-means would assign all points from the group to the cluster center which is closest to the first one of the points (where the order in which the points are considered is random). Instead, we assign the points to to the cluster center that is closest to their mean. Furthermore, we initialize the cluster centers as follows. We first choose one labeled task at random and use the means of its five classes as our first five cluster centers. Then we choose five more cluster centers from the labeled tasks[2] following the k-means++ procedure [9]. Then we choose the remaining 490 cluster centers from among the unlabelebed images using k-means++.

In addition to using COP-k-means like this in DeepCluster, we also experiment with replacing the occurence of k-means in CACTUs with COP-k-means. As remarked above, this approach is less promising because of how the unlabeled images contain different classes of images than the labeled tasks. With our two choices for an embedding algorithm (DeepCluster and Constrained DeepCluster) and two choices for a clustering algorithm (k-means or COP-k-means), we have a total of four semi-supervised variants of CACTUs that we evaluate in the next section.

[2]Thus we choose a total of 10 cluster centers from labeled tasks. This number was chosen to be close to 12, which is the number of underlying classes from which the labeled tasks were sampled. So in this sense, we do make use of information about the underlying labels in mini-imagenet in order to choose hyperparameters, even though we do not use the labels directly for training.

Algorithm (way, shot)	(5,1)	(5,5)	(5,20)	(5,50)
Unsupervised CACTUs	36.52%	48.24%	55.50%	57.86%
Supervised Learning on the 100 tasks only	35.14%	49.69%	57.28%	60.14%
DeepCluster + k-means	40.01%	54.72%	62.91%	65.14%
Constrained DeepCluster + k-means	**41.72%**	**56.96%**	**64.77%**	**66.85%**
Constrained DeepCluster + constrained k-means	40.57%	55.64%	64.13%	66.29%
DeepCluster + constrained k-means	38.56%	53.20%	61.42%	64.19%

Table 2: Resulting accuracies on the meta test set for semi-supervised learning with ProtoNets. For comparison, the table also shows the resulting accuracies for purely unsupervised and purely supervised learning.

5 Semi Supervised Learning Results

The code for all our experiments is available at `https://github.com/cmacho/Semi-Supervised-Meta-Learning`.

5.1 Experiments with ProtoNets

In all our experiments with ProtoNets (including the ones described in section 3), we use the same hyperparameters as specified in [1], except we only use 5 query images per class during training instead of 15. Just like in [1], each model is trained on 5-way 1-shot classification but is tested on 5-shot, 20-shot and 50-shot classification in addition to 1-shot classification.

For each of the methods described in section 4, we train five instances of ProtoNets, mixing the data from our two sources (constructed tasks and labeled tasks) at a different ratio for each of the five runs. We then compare the validation accuracy over the five runs, and evaluate, for each method, only the one with the highest validation accuracy on the test set. The resulting test accuracies are shown in table 2. The validation accuracies are shown in figures 3.

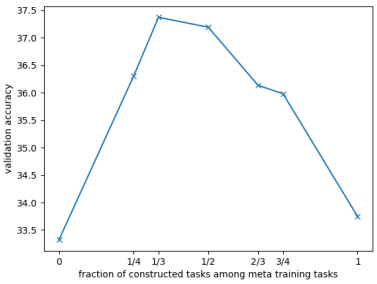
Table 2 shows that each of the semi supervised methods performs better than purely unsupervised training or purely supervised training by a significant margin. It also shows that using Constrained DeepCluster seems to improve performance further, although this could partly be attributed to the fact that we trained unsupervised DeepCluster on $52 \cdot 600 = 31200$ images, whereas Constrained DeepCluster was trained on a total of $31200 + 100 \cdot 30 = 34200$ images, because it was also trained on the images from the 100 labeled tasks[3]. Furthermore, we would ideally like to run each experiment multiple times and average the results, in order to account for the stochasticity in (constrained) k-means and in the Adam optimization procedure [10], which we use to train ProtoNets. We ran some of the experiments multiple times, sometimes with slightly different settings, and it seemed that the accuracy can vary by a few percent from time to time.[4] Here, we are only reporting on the final experiments where we ran each experiment exactly once in order to avoid introducing a bias towards those experiments that were run more often than others.

5.2 Experiments with MAML

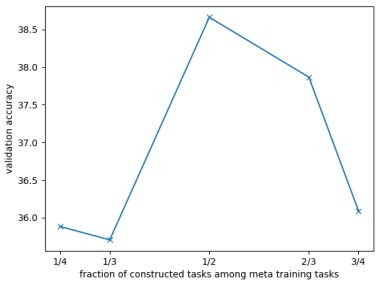
MAML takes much longer to train compared to ProtoNets, therefore we trained each of the four methods only once (instead of five times with different ratios of the two types of data). Based on figure 3 it seems that using an equal amount of constructed tasks and labeled tasks is close to optimal (at least for ProtoNets), therefore we train each of the MAML models in this way.

[3] To avoid this issue, we could have set these experiments up in such a way that we train unsupervised DeepCluster on the same 34200 images that Constrained DeepCluster is trained on (but without any labels).

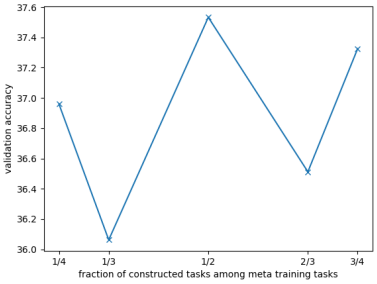
[4] For instance, in one experiment semi-supervised CACTUs with DeepCluster and k-means achieved 41.25% accuracy. In another it achieved only 39.08%. This is despite computing the accuracy from 5000 meta test examples, each having 25 query images (suggesting that the difference arises from stochasticity at meta train time and not at meta test time).



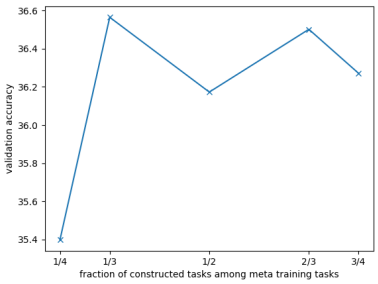
(a) DeepCluster + k-means



(b) Constrained DeepCluster + k-means



(c) Constrained DeepCluster + constrained k-means



(d) DeepCluster + constrained k-means

Figure 3: Validation accuracies for ProtoNets, trained, for each of the four semi-supervised methods, with various mixes of constructed tasks and tasks that were sampled from the 100 labeled tasks.

For MAML, we use the same hyper parameters that were used in [1], except we use a batch size of 4 instead of 8 in order to save computational resources.[5]

Table 3 shows the test accuracies for each of the four semi-supervised methods, as well as for unsupervised CACTUs and for supervised learning on the given labeled meta training tasks. Similar to the results for ProtoNets, we see that semi-supervised learning performs better than either training only on constructed tasks or training only on labeled tasks. We also see that Constrained DeepCluster slightly outperforms unsupervised DeepCluster. However, some of the methods that performed worse with ProtoNets seem to perform better with MAML. This suggests that further experiments would be necessary to determine, which algorithms really work best (or whether they all work equally well).

Note that the accuracies for MAML are much lower than for ProtoNets, overall. It is not entirely clear why this is the case, because in [1], they seemed to perform similarly well. Perhaps reducing the batch size to 4 had a large impact on the final accuracies.

6 Conclusion

We explored a semi-supervised version of CACTUs that performs semi-supervised meta learning by training on a mix of pre-defined tasks and tasks that are constructed from unlabeled data. We showed that in certain situations, where a small amount of labeled data and a larger amount of unlabeled data are available, semi-supervised meta learning can perform very well, significantly outperforming both supervised and unsupervised learning. Finally, we proposed Constrained DeepCluster as a method of improving the performance of CACTUs in the semi-supervised setting. Our results suggest that Constrained DeepCluster can indeed lead to better performance, although more experiments would be necessary to confirm this.

[5]Furthermore, we adjusted the learning rates to account for the fact that in our implementation the cross entropy loss is normalized by the number of (prediction, label) pairs that the loss is computed from whereas in the implementation that was used in [1], the loss is only normalized by the number of support images and the number of query images, but not by the number of classes in a task. However, this implementational detail should have no effect on the results.

Algorithm (way, shot)	(5,1)	(5,5)	(5,20)	(5,50)
Unsupervised CACTUs	34.48%	45.50%	53.36%	58.24%
Supervised Learning on the 100 tasks only	30.75%	37.83%	44.48%	48.76%
DeepCluster + k-means	35.49%	46.32%	52.80%	56.04%
Constrained DeepCluster + k-means	35.52%	45.70%	53.25%	57.70%
Constrained DeepCluster + constrained k-means	**37.79%**	**49.02%**	**57.32%**	**61.50%**
DeepCluster + constrained k-means	35.78%	47.30%	55.19%	59.03%

Table 3: Resulting accuracies on the meta test set for semi-supervised learning with MAML. For comparison, the table also shows the resulting accuracies for purely unsupervised and purely supervised learning.

References

[1] Kyle Hsu, Sergey Levine, and Chelsea Finn. Unsupervised learning via meta-learning, 2019.

[2] Siavash Khodadadeh, Sharare Zehtabian, Saeed Vahidian, Weijia Wang, Bill Lin, and Ladislau Bölöni. Unsupervised meta-learning through latent-space interpolation in generative models, 2020.

[3] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features, 2019.

[4] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. pages 577–584, 01 2001.

[5] David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer, 2018.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017.

[7] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4077–4087. Curran Associates, Inc., 2017.

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

[9] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, 2006.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.