Unsupervised Meta-Learning through Latent-Space Interpolation in Generative Models

Siavash Khodadadeh[1*] **Sharare Zehtabian**[1*] **Saeed Vahidian**[2] **Weijia Wang**[2]

Bill Lin[2] **Ladislau Bölöni**[1]

[1] Dept. of Computer Science
University of Central Florida
Orlando, FL 32816

[2] Dept. of Electrical Engineering and Computer Science
University of California San Diego
San Diego, CA 92161

{siavash.khodadadeh, sharare.zehtabian}@knights.ucf.edu

Abstract

Unsupervised meta-learning approaches rely on synthetic meta-tasks that are created using techniques such as random selection, clustering and/or augmentation. Unfortunately, clustering and augmentation are domain-dependent, and thus they require either manual tweaking or expensive learning. In this work, we describe an approach that generates meta-tasks using generative models. A critical component is a novel approach of sampling from the latent space that generates objects grouped into synthetic classes forming the training and validation data of a meta-task. We find that the proposed approach, LAtent Space Interpolation Unsupervised Meta-learning (LASIUM), outperforms or is competitive with current unsupervised learning baselines on few-shot classification tasks on the most widely used benchmark datasets. In addition, the approach promises to be applicable without manual tweaking over a wider range of domains than previous approaches.

1 Introduction

Meta-learning algorithms for neural networks (1; 2; 3) prepare networks to quickly adapt to unseen tasks. This is done in a meta-training phase that typically involves a large number of supervised learning tasks. Very recently, several approaches had been proposed that perform the meta-training by generating synthetic training tasks from an *unsupervised* dataset. This requires us to generate samples with specific pairwise information: *in-class* pairs of samples that are with high likelihood in the same class, and *out-of-class* pairs that are with high likelihood *not* in the same class. For instance, UMTRA (4) and AAL (5) achieve this through random selection from a domain with many classes for out-of-class pairs and by augmentation for in-class pairs. CACTUs (6) creates synthetic labels through unsupervised clustering of the domain. Unfortunately, these algorithms depend on domain specific expertise for the appropriate clustering and augmentation techniques.

In this paper, we rely on recent advances in the field of generative models, such as the variants of generative adversarial networks (GANs) and variational autoencoders (VAEs), to generate the in-class and out-of-class pairs of meta-training data. The fundamental idea of our approach is that in-class pairs are close while out-of-class pairs are far away in the latent space representation of the generative model. Thus, we can generate in-class pairs by interpolating between two out-of-class samples

[*] These authors contributed equally.

Preprint. Under review.

in the latent space and choosing interpolation ratios that put the new sample close to one of the objects. From this latent sample, the generative model creates the new in-class object. Our approach requires minimal domain-specific tweaking, and the necessary tweaks are human-comprehensible. For instance, we need to choose thresholds for latent space distance that ensure that classes are in different domains, as well as interpolation ratio thresholds that ensure that the sample is in the same class as the nearest edge. Another advantage of the approach is that we can take advantage of off-the-shelf, pre-trained generative models.

The main contributions of this paper can be summarized as follows:

- We describe an algorithm, LAtent Space Interpolation Unsupervised Meta-learning (LASIUM), that creates training data for a downstream meta-learning algorithm starting from an unlabeled dataset by taking advantage of interpolation in the latent space of a generative model.
- We show that on the most widely used few-shot learning datasets, LASIUM outperforms or performs competitively with other unsupervised meta-learning algorithms, significantly outperforms transfer learning in all cases, and in a number of cases approaches the performance of supervised meta-learning algorithms.

2 Related Work

Meta-learning or "learning to learn" in the field of neural networks is an umbrella term that covers a variety of techniques that involve training a neural network over the course of a meta-training phase, such that when presented with the target task, the network is able to learn it much more efficiently than an unprepared network would. Such techniques had been proposed since the 1980s (7; 8; 9; 10). In recent years, meta-learning has gained a resurgence, through approaches that either "learn to optimize" (2; 11; 12; 13; 14; 15) or learn embedding functions in a non-parametric setting (3; 16; 17; 18). Hybrids between these two approaches had also been proposed (19; 20).

Most approaches use labeled data during the meta-learning phase. While in some domains there is an abundance of labeled datasets, in many domains such labeled data is difficult to acquire. Unsupervised meta-learning approaches aim to learn from an unsupervised dataset from a domain similar from that of the target task. Typically these approaches generate synthetic few-shot learning tasks for the meta-learning phase through a variety of techniques. CACTUs (6) uses a progressive clustering method. UMTRA (4) utilizes the statistical diversity properties and domain-specific augmentations to generate synthetic training and validation data. AAL (5) uses augmentation of the unlabeled training set to generate the validation data. The accuracy of these approaches was shown to be comparable with but lower than supervised meta-learning approaches, but with the advantage of requiring orders of magnitude less labeled training data. A common weakness of these approaches is that the techniques used to generate the synthetic tasks (clustering, augmentation, random sampling) are highly domain dependent.

Our proposed approach, LASIUM, takes advantage of generative models trained on the specific domain to create the in-class and out-of-class pairs of meta-training data. The most successful neural-network based generative models in recent years are variational autoencoders (VAE) (21) and generative adversarial networks (GANs) (22). The implementation variants of the LASIUM algorithm described in this paper rely on the original VAE model and on two specific variations of the GAN concept, respectively. MSGAN (aka Miss-GAN) (23) aims to solve the missing mode problem of conditional GANs through a regularization term that maximizes the distance between the generated images with respect to the distance between their corresponding input latent codes. Progressive GANs (24) are growing both the generator and discriminator progressively, and approach resembling the layer-wise training of autoencoders.

3 Method

3.1 Preliminaries

We define an N-way, $K^{(tr)}$-shot supervised classification task, $\mathcal{T}$, as a set $\mathcal{D}_{\mathcal{T}}^{(tr)}$ composed of $i \in \{1, \ldots, N \times K^{(tr)}\}$ data points (x_i, y_i) such that there are exactly $K^{(tr)}$ samples for each

categorical label $y_i \in \{1, \ldots, N\}$. During meta-learning, an additional set ,$\mathcal{D}_{\mathcal{T}}^{(val)}$, is attached to each task that contains another $N \times K^{(val)}$ data points separate from the ones in $\mathcal{D}_{\mathcal{T}}^{(tr)}$. We have exactly $K^{(val)}$ samples for each class in $\mathcal{D}_{\mathcal{T}}^{(val)}$ as well.

It is straightforward to package N-way, $K^{(tr)}$-shot tasks with $\mathcal{D}_{\mathcal{T}}^{(tr)}$ and $\mathcal{D}_{\mathcal{T}}^{(val)}$ from a labeled dataset. However, in unsupervised meta-learning setting, a key challenge is how to automatically construct tasks from the unlabeled dataset $\mathcal{U} = \{\ldots x_i \ldots\}$.

3.2 Generating meta-tasks using generative models

We have seen that in order to generate the training data for the meta-learning phase, we need to generate N-way training tasks with $K^{(tr)}$ training and $K^{(val)}$ validation samples. The label associated with the classes in these tasks is not relevant, as it will be discarded after the meta-learning phase. Our objective is simply to generate samples of the type $x_{i,j}$ with $i \in \{1 \ldots N\}$ and $j \in \{1 \ldots K^{(tr)} + K^{(val)}\}$ with the following properties: (a) all the samples $x_{i,j}$ are different (b) any two samples with the same i index are in-class samples and (c) any two samples with different i index are out-of-class samples. In the absence of human provided labels, the class structure of the domain is defined only implicitly by the sample selection procedure. Previous approaches to unsupervised meta-learning chose samples directly from the training data $x_{i,j} \in \mathcal{U}$, or created new samples through augmentation. For instance, we can define the class structure of the domain by assuming that certain types of augmentations keep the samples in-class with the original sample. One challenge of such approaches is that the choice of the augmentation is domain dependent, and the augmentation itself can be a complex mathematical operation.

In this paper we approach the sample selection problem differently. Instead of sampling $x_{i,j}$ from $\mathcal{U}$, we use the unsupervised dataset to train a generative model $p(x)$. Generative models represent the full probability distribution of a model, and allow us to sample new instances from the distribution. For many models, this sampling process can be computationally expensive iterative process. Many successful neural network based generative models use the *reparametrization trick* for the training and sampling which concentrate the random component of the model in a latent representation z. By choosing the latent representation z from a simple (uniform or normal) distribution, we can obtain a sample from the complex distribution $p(x)$ by passing z through a deterministic generator $\mathcal{G}(z) \rightarrow x$. Two of the most popular generative models, variational autoencoders (VAEs) and generative adversarial networks (GANs) follow this model.

The idea of the LASIUM algorithm is that given a generator component $\mathcal{G}(.)$, *nearby* latent space values z_1 and z_2 map to *in-class* samples x_1 and x_2. Conversely, z_1 and z_2 values that are *far away* from each other, map to *out of class* samples. Naturally, we still need to define what we mean by "near" and "far" in the latent space and how to choose the corresponding z values. However, this is a significantly simpler task than, for instance, defining the set of complex augmentations that might retain class membership.

[ht]

Training a generative model Our method for generating meta-tasks is agnostic to the choice of training algorithm for the generative model and can use either a VAE or a GAN with minimal adjustments. In our VAE experiments, we used a network trained with the standard VAE training algorithm (21). For the experiments with GANs we used two different methods mode seeking GANs (MSGAN) (23) and progressive growing of GANs (proGAN) (24).

Algorithm 1 describes the steps of our method. We will delve into each step in the following parts of this section.

Sampling out of class instances from the latent space representation: Our sampling techniques differ slightly whether we are using a GAN or VAE. For GAN, we use rejection sampling to find N latent space vectors that are at a pairwise distance of at least threshold ϵ - see Figure 1(a). When using a VAE, we also have an encoder network that allows us to map from the domain to the latent space. Taking advantage of this, we can additionally sample data points from our unlabeled dataset $\mathcal{U}$ and embed them into a latent space. If the latent space representation of these N images are too close to each other, we re-sample, otherwise we can use the N images and their representations and

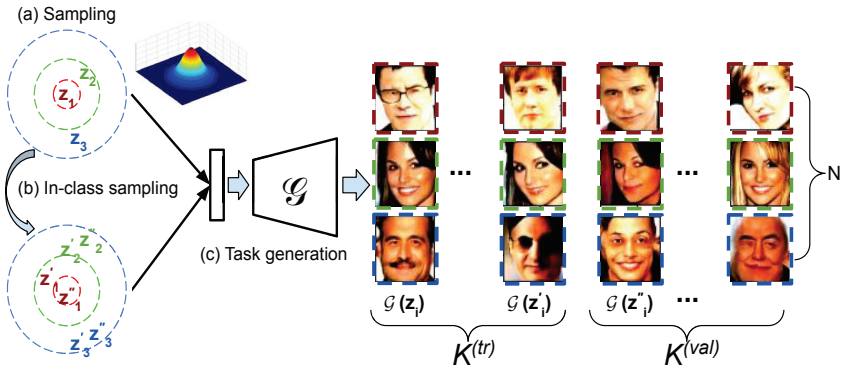

Figure 1: 3-way, $K^{(tr)}$-shot task generation with $K^{(val)}$ images for validation by a pre-trained GAN generator $\mathcal{G}$. **a)** Sample 3 random vectors. **b)** Generate new vectors by one of the proposed in-class sampling strategies. **c)** Generate images from all of the latent vectors and put them into train and validation set to construct a task. The images in this figure have been generated by our algorithm. The colored edge of each image indicates that it was generated from its corresponding latent vector.

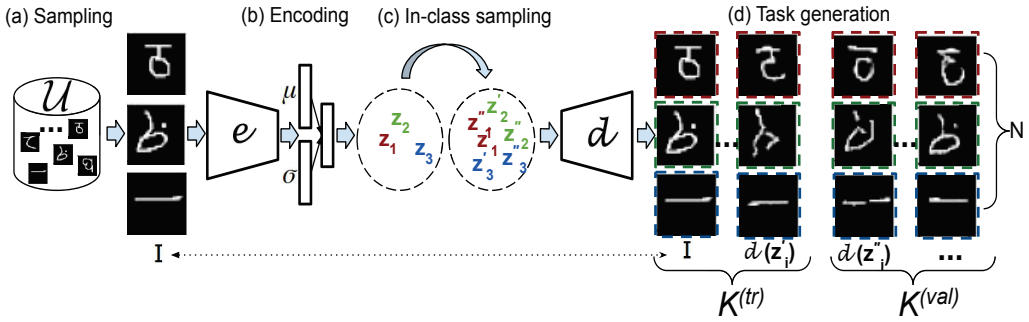

Figure 2: 3-way, $K^{(tr)}$-shot task generation by VAE on Omniglot dataset with $K^{(val)}$ images for validation set of each task. **a)** Sample 3 images from dataset. **b)** Encode the images into latent space and check if they are distanced. **c)** Use proposed in-class sampling techniques to generate new latent vectors. **d)** Generate images from the latent vectors and put them alongside with sampled images from step **a** into train and validation set to construct a task.

continue the following steps exactly the same as GANs - see Figure 2(a) and (b). We will refer to the vectors selected here as *anchor vectors*.

Generating in-class latent space vectors Next, having N sampled anchor vectors $\{z_1, \ldots, z_N\}$ from the latent space representation, we aim to generate N new vectors $\{z'_1, \ldots, z'_N\}$ from the latent space representation such that the generated image $\mathcal{G}(z_i)$ belongs to the same class as the one of $\mathcal{G}(z'_i)$ for $i \in 1, \ldots, N$. This process needs to be repeated $\mathcal{P}$ for $K^{(tr)} + K^{(val)} - 1$ times.

The sampling strategy takes as input the sampled vectors and a number $\omega \in \{1 \ldots K^{(tr)} + K^{(val)} - 1\}$ and returns N new vectors such that z_i and z'_i are an in-class pair for $i \in \{1 \ldots N\}$. This ensures that no two z'_i belong to the same class and creates N groups of $(K^{(tr)} + K^{(val)})$ vectors in our latent space. We feed these vectors to our generator to get N groups of $(K^{(tr)} + K^{(val)})$ images. From each group we pick the first $K^{(tr)}$ for $\mathcal{D}_{\mathcal{T}}^{(tr)}$ and the last $K^{(val)}$ for $\mathcal{D}_{\mathcal{T}}^{(val)}$.

What remains is to define the strategy to sample the individual in-class vectors. We propose three different sampling strategies, all of which can be seen as variations of the idea of latent space interpolation sampling. This motivates the name of the algorithm LAtent Space Interpolation Unsupervised Meta-learning (LASIUM).

LASIUM-N (adding Noise): This technique generates in-class samples by adding Gaussian noise to the anchor vector $z'_i = z_i + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ (see Figure 3-Left). In the context of LASIUM, we

Algorithm 1: LASIUM for unsupervised meta-learning task generation

require : Unlabeled dataset $\mathcal{U} = \{\dots x_i \dots\}$, Pre-trained generator $\mathcal{G}$, Policy $\mathcal{P}$
require : $K^{(tr)}$, $K^{(val)}$: number of samples for train and validation during meta-learning
require : N: class-count, N_{MB}: meta-batch size

$B = \{\}$; `// meta-batch of tasks`
for *i in* $1, \dots, N_{MB}$ **do**
  Sample N class-vectors in latent space of $\mathcal{G}$ and add them to task-vectors
  for ω *in* $1, \dots, K^{(tr)} + K^{(val)} - 1$ **do**
    generate new-vectors = $\mathcal{P}$(class-vectors, ω) and add them to task-vectors
  end
  Generate $N \times (K^{(tr)} + K^{(val)})$ images by feeding task-vectors to generator $\mathcal{G}$
  Construct task $\mathcal{T}_i$ by putting the first $N \times K^{(tr)}$ images in task train set and the last $N \times K^{(val)}$ images in task validation set
  $B \leftarrow B \cup \mathcal{T}_i$
end
return B

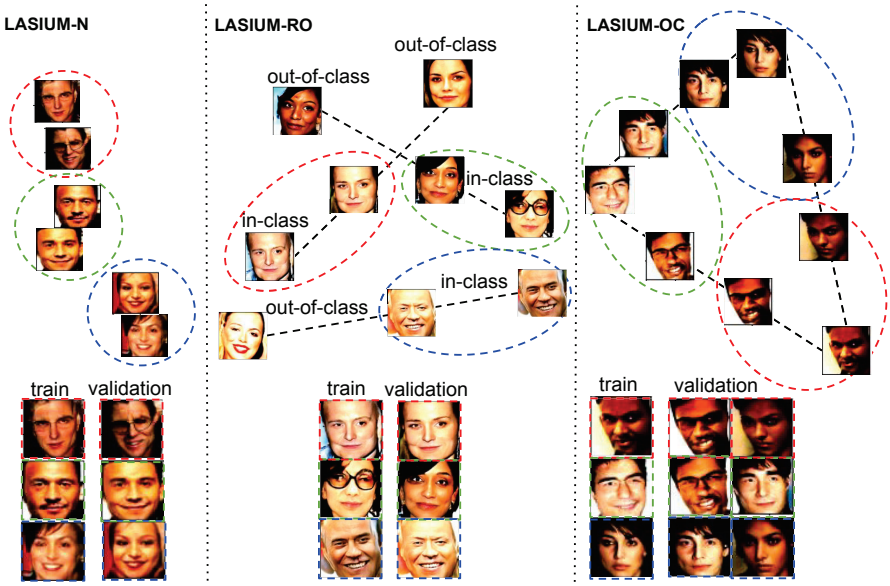

Figure 3: Latent space representation visualization of proposed strategies for generating in-class candidates. **Left**: LASIUM-N, adding random noise to the sample vector. **Middle**: LASIUM-RO, interpolate with random out-of-class samples. **Right**: LASIUM-OC, interpolate with other classes' samples.

can see this as an interpolation between the anchor vector and a noise vector, with the interpolation factor determined by σ. For the impact of different choices of σ see the ablation study in section 4.6.

LASIUM-RO (with Random Out-of-class samples) To generate a new in-class sample to anchor vector z_i we first find a random out-of-class sample v_i, and choose an interpolated version closer to the anchor: $z_i' = z_i + \alpha \times (v_i - z_i)$ (see Figure 3-Middle). Here, α is a hyperparameter, which can be tuned to define the size of the class. As we are in a comparatively high-dimensional latent space (in our case, 512 dimensions), we need relatively large values of α, such as $\alpha = 0.4$ to define classes of reasonable size. This model effectively allows us to define complex augmentations (such as a person seen without glasses, or in a changed lighting) with only one scalar hyperparameter to tune. By interpolating towards another sample we ensure that we are staying on the manifold that defines the dataset (in the case of Figure 3, this being human faces).

LASIUM-OC (with Other Classes' samples) This technique is similar to LASIUM-RO, but instead of using a randomly generated out-of-class vector, we are interpolating towards vectors already chosen

from the other classes in the same task (see Figure 3-Right). This limits the selection of the samples to be confined to the convex hull defined by the initial anchor points. The intuition behind this approach is that choosing the samples this way focuses the attention of the meta-learner towards the hard to distinguish samples that are *between* the classes in the few shot learning class (eg. they share certain attributes).

4 Experiments

We tested the proposed algorithms on three few-shot learning benchmarks: (a) the 5-way Omniglot (25), a benchmark for few-shot handwritten character recognition, (b) the 5-way CelebA few-shot identity recognition, and (c) the CelebA attributes dataset (26) proposed as a few-shot learning benchmark by (2) that comprises binary classification (2-way) tasks in which each task is defined by selecting 3 different attributes and 3 boolean values corresponding to each attribute. Every image in a certain task-specific class has the same attributes with each other while does not share any of these attributes with images in the other class. Last but not least we evaluate our results on (d) the mini-ImageNet (27) few-shot learning benchmark.

We partition each dataset into meta-training, meta-validation, and meta-testing splits between classes. To evaluate our method, we use the classes in the test set to generate 1000 tasks as described in section 3.2. We set $K^{(val)}$ to be 15. We average the accuracy on all tasks and report a 95% confidence interval. To ensure that comparisons are fair, we use the same random seed in the whole task generation process. For the Omniglot dataset, we report the results for $K^{(tr)} \in \{1, 5\}$, and $K^{(val)} = 15$. For CelebA identity recognition, we report our results for $K^{(tr)} \in \{1, 5, 15\}$ and $K^{(val)} = 15$. For CelebA attributes, we follow the $K^{(tr)} = 5$ and $K^{(val)} = 5$ tasks as proposed by (6).

4.1 Baselines

As baseline algorithms for our approach we follow the practice of recent papers in the unsupervised meta-learning literature. The simplest baseline is to train the same network architecture from scratch with $N \times K^{(tr)}$ images. More advanced baselines can be obtained by learning an unsupervised embedding on $\mathcal{U}$ and use it for downstream task training. We used the ACAI (28), BiGAN (29; 30), and DeepCluster (31) as representative of the unsupervised learning literature. On top of these embeddings, we report accuracy for K_{nn}-nearest neighbors, linear classifier, multi layer perceptron (MLP) with dropout, and cluster matching.

The direct competition for our approach are the current state-of-the-art algorithms in unsupervised meta-learning. We compare our results with CACTUs-MAML (6), CACTUs-ProtoNets (6) and UMTRA (4). Finally, it is useful to compare our approach with algorithms that require supervised data. We include results for supervised standard transfer learning from VGG19 pre-trained on ImageNet (32) and two supervised meta-learning algorithms, MAML (6), and ProtoNets (6).

4.2 Neural network architectures

Since excessive tuning of hyperparameters can lead to the overestimation of the performance of a model (33), we keep the hyperparameters of the unsupervised meta-learning as constant as possible (including the MAML, and ProtoNets model architectures) in all experiments. Our model architecture consists of four stacked convolutional blocks. Each block comprises 64 filters that carry out 3×3 convolutions, followed by batch normalization, a ReLU non-linearity, and 2×2 max-pooling. For the MAML experiments, classification is performed by a fully connected layer, whereas for the ProtoNets model we compute distances based on the feature vectors produced by the last convolution module without any dense layers. The input size to our model is $84 \times 84 \times 3$ for CelebA and $28 \times 28 \times 1$ for Omniglot.

For Omniglot, our VAE model is constructed symmetrically. The encoder is composed of four convolutional blocks, with batch normalization and ReLU activation following each of them. A dense layer is connected to the end such that given an input image of shape 28×28, the encoder produces a latent vector of length 20. On the other side, the decoder starts from a dense layer whose output has length $7 \times 7 \times 64 = 3136$. It is then fed into four modules each of which consists of

a transposed convolutional layer, batch normalization and the ReLU non-linearity. We use 3×3 kernels, 64 channels and a stride of 2 for all the convolutional and transposed convolutional layers. Hence, the generated image has the size of 28×28 that is identical to the input images. This VAE model is trained for 1000 epochs with a learning rate of 0.001.

Our GAN generator gets an input of size l which is the dimensionality of the latent space and feeds it into a dense layer of size $7 \times 7 \times 128$. After applying a Leaky ReLU with $\alpha = 0.2$, we reshape the output of dense layer to 128 channels of shape 7×7. Then we feed it into two upsampling blocks, where each block has a transposed convolution with 128 channels, 4×4 kernels and 2×2 strides. Finally, we feed the outcome of the upsampling blocks into a convolution layer with 1 channel and a 7×7 kernel with sigmoid activaiton. The discriminator takes a $28 \times 28 \times 1$ input and feeds it into three 3×3 convolution layers with 64, 128 and 128 channels and 2×2 strides. We apply leaky ReLU activation after each convolution layer with $\alpha = 0.2$. Finally we apply a global 2D max pooling layer and feed it into a dense layer with 1 neuron to classify the output as real or fake. We use the same loss function for training as described in (23).

For the CelebA GAN experiments, we use the pre-trained network architecture described in (24). For VAE, we use the same architecture as we described for Omniglot VAE with one more convolution block and more channels to handle the larger input size of $84 \times 84 \times 3$. The exact architecture is described in section 4.6.

4.3 Results on Omniglot

Table 1 shows the results on the Omniglot dataset. We find that the LASIUM-RO-GAN-MAML configuration outperforms all the unsupervised approaches, including the meta-learning based ones like CACTUs (6) and UMTRA (4). Beyond the increase in performance, we must note that the competing approaches use more domain specific knowledge (in case of UMTRA augmentations, in case of CACTUs, learned clustering). We also find that on this benchmark, LASIUM outperforms transfer learning using the much larger VGG-19 network.

As expected even the best LASIUM result is worse than the supervised meta-learning models. However, we need to consider that the unsupervised meta-learning approaches use several orders of magnitude less labels. For instance, the 95.29% accuracy of LASIUM-RO-GAN-MAML was obtained with only 25 labels, while the supervised approaches used 25,000.

4.4 Results on CelebA

Table 2 shows our results on the CelebA identity recognition tasks where the objective is to recognize N different people given $K^{(tr)}$ images for each. We find that on this benchmark as well, the LASIUM-RO-GAN-MAML configuration performs better than other unsupervised meta-learning models as well as transfer learning with VGG-19 - it only falls slightly behind LASIUM-RO-GAN-ProtoNets on the one-shot case. As we have discussed in the case of Omniglot results, the performance remains lower then the supervised meta-learning approaches which use several orders of magnitude more labeled data.

Finally, Table 3 shows our results for CelebA attributes benchmark introduced in (6). A peculiarity of this dataset is that the way in which classes are defined based on the attributes, the classes are unbalanced in the dataset, making the job of synthetic task selection more difficult. We find that LASIUM-N-GAN-MAML obtains the second best on this test with a performance of $\mathbf{74.79 \pm 1.01}$, within the confidence interval of the winner, CACTUs MAML with BiGAN $\mathbf{74.98 \pm 1.02}$. In this benchmark, transfer learning with the VGG-19 network performed better than all unsupervised meta-learning approaches, possibly due to existing representations of the discriminating attributes in that much more complex network.

4.5 Results on mini-ImageNet

In this section, we evaluate our algorithm on mini-ImageNet benchmark. Its complexity is high due to the use of ImageNet images. In total, there are 100 classes with 600 samples of 84×84 color images per class. These 100 classes are divided into 64, 16, and 20 classes respectively for sampling tasks for meta-training, meta-validation, and meta-test. A big difference between mini-ImageNet and CelebA is that we have to classify a group of concepts instead of just the identity of a subject. This

Table 1: Accuracy results on the Omniglot dataset averaged over 1000, 5-way, $K^{(tr)}$-shot downstream tasks with $K^{(val)} = 15$ for each task. $\pm$ indicates the 95% confidence interval. The top three unsupervised results are reported in **bold**.

Algorithm	Feature Extractor	$K^{(tr)}$ = 1	$K^{(tr)}$ = 5
Training from scratch	N/A	51.64 ± 0.65	71.44 ± 0.53
K-nearest neighbors	ACAI	57.46 ± 1.35	81.16 ± 0.57
Linear Classifier	ACAI	61.08 ± 1.32	81.82 ± 0.58
MLP with dropout	ACAI	51.95 ± 0.82	77.20 ± 0.65
Cluster matching	ACAI	54.94 ± 0.85	71.09 ± 0.77
K-nearest neighbors	BiGAN	49.55 ± 1.27	68.06 ± 0.71
Linear Classifier	BiGAN	48.28 ± 1.25	68.72 ± 0.66
MLP with dropout	BiGAN	40.54 ± 0.79	62.56 ± 0.79
Cluster matching	BiGAN	43.96 ± 0.80	58.62 ± 0.78
CACTUs-MAML	BiGAN	58.18 ± 0.81	78.66 ± 0.65
CACTUs-MAML	ACAI	68.84 ± 0.80	87.78 ± 0.50
UMTRA-MAML	N/A	$\mathbf{81.91 \pm 0.58}$	$\mathbf{94.58 \pm 0.25}$
LASIUM-RO-GAN-MAML	N/A	$\mathbf{83.26 \pm 0.55}$	$\mathbf{95.29 \pm 0.22}$
LASIUM-N-VAE-MAML	N/A	76.11 ± 0.64	$\mathbf{94.42 \pm 0.26}$
CACTUs-ProtoNets	BiGAN	54.74 ± 0.82	71.69 ± 0.73
CACTUs-ProtoNets	ACAI	68.12 ± 0.84	83.58 ± 0.61
LASIUM-RO-GAN-ProtoNets	N/A	$\mathbf{80.15 \pm 0.64}$	91.10 ± 0.35
LASIUM-OC-VAE-ProtoNets	N/A	73.22 ± 0.73	85.05 ± 0.46
Transfer Learning (VGG-19)	N/A	54.49 ± 0.90	89.57 ± 0.44
Supervised MAML	N/A	94.46 ± 0.35	98.83 ± 0.12
Supervised ProtoNets	N/A	98.35 ± 0.22	99.58 ± 0.09

Table 2: Accuracy results of unsupervised learning on CelebA for different unsupervised methods. The results are averaged over 1000, 5-way, $K^{(tr)}$-shot downstream tasks with $K^{(val)} = 15$ for each task. $\pm$ indicates the 95% confidence interval. The top three unsupervised results are reported in **bold**.

Algorithm	$K^{(tr)}$ = 1	$K^{(tr)}$ = 5	$K^{(tr)}$ = 15
Training from scratch	34.69 ± 0.50	56.50 ± 0.55	70.56 ± 0.49
CACTUs	41.42 ± 0.64	$\mathbf{62.71 \pm 0.57}$	$\mathbf{74.18 \pm 0.68}$
UMTRA	39.30 ± 0.59	60.44 ± 0.56	$\mathbf{72.41 \pm 0.48}$
LASIUM-RO-GAN-MAML	$\mathbf{43.88 \pm 0.57}$	$\mathbf{66.98 \pm 0.53}$	$\mathbf{78.13 \pm 0.44}$
LASIUM-RO-VAE-MAML	41.25 ± 0.57	58.22 ± 0.54	71.05 ± 0.49
LASIUM-RO-GAN-ProtoNets	$\mathbf{44.39 \pm 0.61}$	60.83 ± 0.58	66.66 ± 0.53
LASIUM-RO-VAE-ProtoNets	$\mathbf{43.22 \pm 0.58}$	$\mathbf{61.12 \pm 0.54}$	68.51 ± 0.51
Transfer Learning (VGG-19)	33.28 ± 0.57	58.74 ± 0.62	74.04 ± 0.49
Supervised MAML	85.46 ± 0.55	94.98 ± 0.25	96.18 ± 0.19
Supervised ProtoNets	84.17 ± 0.61	90.84 ± 0.38	90.85 ± 0.36

Table 3: Results on CelebA attributes benchmark 2-way, 5-shot tasks with $K^{(val)} = 5$. The results are averaged over 1000 downstream tasks and $\pm$ indicates 95% confidence interval. The top three unsupervised results are reported in **bold**.

Algorithm	Feature Extractor	Accuracy
Training from scratch	N/A	63.19 ± 1.06
K-nearest neighbors	BiGAN	56.15 ± 0.89
Linear Classifier	BiGAN	58.44 ± 0.90
MLP with dropout	BiGAN	56.26 ± 0.94
Cluster matching	BiGAN	56.20 ± 1.00
K-nearest neighbors	DeepCluster	61.47 ± 0.99
Linear Classifier	DeepCluster	59.57 ± 0.98
MLP with dropout	DeepCluster	60.65 ± 0.9
Cluster matching	DeepCluster	51.51 ± 0.89
CACTUs MAML	BiGAN	$\mathbf{74.98 \pm 1.02}$
CACTUs MAML	DeepCluster	$\mathbf{73.79 \pm 1.01}$
LASIUM-N-GAN-MAML	N/A	$\mathbf{74.79 \pm 1.01}$
CACTUs ProtoNets	BiGAN	65.58 ± 1.04
CACTUs ProtoNets	DeepCluster	74.15 ± 1.02
LASIUM-N-GAN-ProtoNets	N/A	73.41 ± 1.10
Transfer Learning (VGG-19)	N/A	79.76 ± 1.03
Supervised MAML	N/A	87.10 ± 0.85
Supervised ProtoNets	N/A	85.13 ± 0.92

makes interpreting the latent space a bit trickier. For example, it is not rational to interpolate between a bird and a piano. However, the assumption that nearby latent vectors belong to nearby instances is still valid. Thereby, we could be confident by not getting too far from the current latent vector, we generate something which belongs to the same class (identity).

For mini-ImageNet we use a pre-trained network BigBiGAN[2]. Our experiments show that our method is very effective and can outperform state-of-the-art algorithms. See Table 4 for the results on mini-ImageNet benchmark. Figure 4 demonstrates tasks constructed for mini-ImageNet by LASIUM-N with $\sigma^2 = 1.0$.

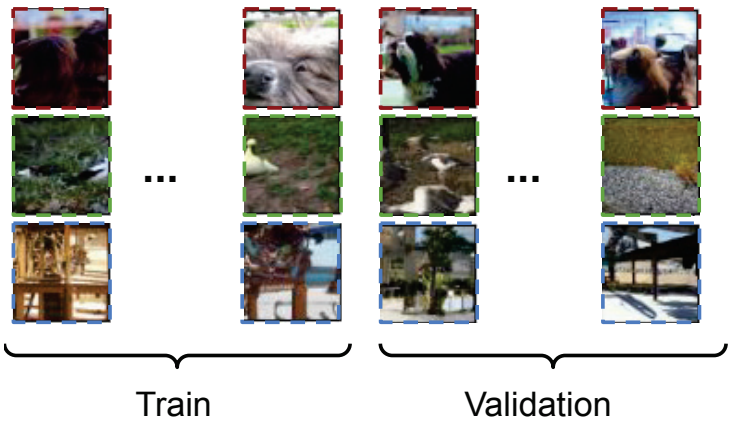

Figure 4: Train and validation tasks for mini-ImageNet constructed by LASIUM-N with $\sigma^2 = 1.0$

4.6 Hyperparameters and ablation studies

In this section, we report the hyperparameters of LASIUM-MAML in Table 5 and LASIUM-ProtoNets in Table 6 for Omniglot, CelebA, CelebA attributes and mini-ImageNet datasets.

[2]https://tfhub.dev/deepmind/bigbigan-resnet50/1

Table 4: Results on mini-ImageNet benchmark for 5-way, $K^{(tr)}$-shot tasks with $K^{(val)} = 15$. The results are averaged over 1000 downstream tasks and $\pm$ indicates 95% confidence interval. The top three unsupervised results are reported in **bold**.

Algorithm	Embedding	$K^{(tr)}$ = 1	$K^{(tr)}$ = 5	$K^{(tr)}$ = 20	$K^{(tr)}$ = 50
Training from scratch	N/A	27.59 ± 0.59	38.48 ± 0.66	51.53 ± 0.72	59.63 ± 0.74
K-nearest neighbors	BiGAN	25.56 ± 1.08	31.10 ± 0.63	37.31 ± 0.40	43.60 ± 0.37
Linear Classifier	BiGAN	27.08 ± 1.24	33.91 ± 0.64	44.00 ± 0.45	50.41 ± 0.37
MLP with dropout	BiGAN	22.91 ± 0.54	29.06 ± 0.63	40.06 ± 0.72	48.36 ± 0.71
Cluster matching	BiGAN	24.63 ± 0.56	29.49 ± 0.58	33.89 ± 0.63	36.13 ± 0.64
K-nearest neighbors	DeepCluster	28.90 ± 1.25	42.25 ± 0.67	56.44 ± 0.43	63.90 ± 0.38
Linear Classifier	DeepCluster	29.44 ± 1.22	39.79 ± 0.64	56.19 ± 0.43	65.28 ± 0.34
MLP with dropout	DeepCluster	29.03 ± 0.61	39.67 ± 0.69	52.71 ± 0.62	60.95 ± 0.63
Cluster matching	DeepCluster	22.20 ± 0.50	23.50 ± 0.52	24.97 ± 0.54	26.87 ± 0.55
CACTUs MAML	BiGAN	36.24 ± 0.74	51.28 ± 0.68	61.33 ± 0.67	66.91 ± 0.68
CACTUs MAML	DeepCluster	39.90 ± 0.74	$\mathbf{53.97 \pm 0.70}$	$\mathbf{63.84 \pm 0.70}$	$\mathbf{69.64 \pm 0.63}$
UMTRA MAML	N/A	**39.93**	50.73	61.11	**67.15**
LASIUM-N-GAN-MAML	N/A	$\mathbf{40.19 \pm 0.58}$	$\mathbf{54.56 \pm 0.55}$	$\mathbf{65.17 \pm 0.49}$	$\mathbf{69.13 \pm 0.49}$
CACTUs ProtoNets	BiGAN	36.62 ± 0.70	50.16 ± 0.73	59.56 ± 0.68	63.27 ± 0.67
CACTUs ProtoNets	DeepCluster	39.18 ± 0.71	$\mathbf{53.36 \pm 0.70}$	$\mathbf{61.54 \pm 0.68}$	63.55 ± 0.64
LASIUM-N-GAN-ProtoNets	N/A	$\mathbf{40.05 \pm 0.60}$	52.53 ± 0.51	59.45 ± 0.48	61.43 ± 0.45
Supervised MAML	N/A	46.81 ± 0.77	62.13 ± 0.72	71.03 ± 0.69	75.54 ± 0.62
Supervised ProtoNets	N/A	46.56 ± 0.76	62.29 ± 0.71	70.05 ± 0.65	72.04 ± 0.60

We also report the ablation studies on different strategies for task construction in Table 7. We run all the algorithm for just 1000 iterations and compared between them. We also apply a small shift to Omniglot images.

Table 5: LASIUM-MAML hyperparameters summary

Hyperparameter	Omniglot	CelebA	CelebA attributes	mini-ImageNet
Number of classes	5	5	2	5
Input size	$28 \times 28 \times 1$	$84 \times 84 \times 3$	$84 \times 84 \times 3$	$84 \times 84 \times 3$
Inner learning rate	0.4	0.05	0.05	0.05
Meta learning rate	0.001	0.001	0.001	0.001
Meta-batch size	4	4	4	4
$K^{(tr)}$ meta-learning	1	1	5	1
$K^{(val)}$ meta-learning	5	5	5	5
$K^{(val)}$ evaluation	15	15	5	15
Meta-adaptation steps	5	5	5	5
Evaluation adaptation steps	50	50	50	50

Table 6: LASIUM-ProtoNets hyperparameters summary

Hyperparameter	Omniglot	CelebA	CelebA attributes	mini-ImageNet
Number of classes	5	5	2	5
Input size	$28 \times 28 \times 1$	$84 \times 84 \times 3$	$84 \times 84 \times 3$	$84 \times 84 \times 3$
Meta learning rate	0.001	0.001	0.001	0.001
Meta-batch size	4	4	4	4
$K^{(tr)}$ meta-learning	1	1	5	1
$K^{(val)}$ meta-learning	5	5	5	5
$K^{(val)}$ evaluation	15	15	5	15

Table 7: Accuracy of different proposed strategies on Omniglot. For the sake of comparison, we stop meta-learning after 1000 iterations. Results are reported on 1000 tasks with a 95% confidence interval.

Sampling Strategy	Hyperparameters	GAN-MAML	VAE-MAML	GAN-Proto	VAE-Proto
LASIUM-N	σ^2=0.5	$\mathbf{77.16 \pm 0.65}$	70.41 ± 0.71	62.16 ± 0.79	61.57 ± 0.80
LASIUM-N	σ^2=1.0	71.10 ± 0.70	68.26 ± 0.71	60.95 ± 0.78	62.17 ± 0.80
LASIUM-N	σ^2=2.0	63.18 ± 0.71	65.18 ± 0.71	59.81 ± 0.78	$\mathbf{64.88 \pm 0.78}$
LASIUM-RO	α=0.2	$\mathbf{77.62 \pm 0.64}$	$\mathbf{75.02 \pm 0.66}$	$\mathbf{62.24 \pm 0.79}$	62.17 ± 0.80
LASIUM-RO	α=0.4	$\mathbf{75.79 \pm 0.65}$	$\mathbf{71.31 \pm 0.70}$	$\mathbf{64.19 \pm 0.76}$	$\mathbf{62.20 \pm 0.80}$
LASIUM-OC	α=0.2	74.70 ± 0.68	$\mathbf{74.98 \pm 0.67}$	61.79 ± 0.79	62.16 ± 0.78
LASIUM-OC	α=0.4	73.40 ± 0.68	68.79 ± 0.73	$\mathbf{64.59 \pm 0.76}$	$\mathbf{63.08 \pm 0.79}$

5 Conclusion

We described LASIUM, an unsupervised meta-learning algorithm for few-shot classification. The algorithm is based on interpolation in the latent space of a generative model to create synthetic meta-tasks. In contrast to other approaches, LASIUM requires minimal domain specific knowledge. We found that LASIUM outperforms state-of-the-art unsupervised algorithms on the Omniglot and CelebA identity recognition benchmarks and competes very closely with CACTUs on the CelebA attributes learning benchmark.

6 Acknowledgements

This work had been in part supported by the National Science Foundation under Grant Number IIS-1409823.

References

[1] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A Simple Neural Attentive Meta-Learner. In *Int'l Conf. on Learning Representations (ICLR)*, 2018.

[2] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. of Int'l Conf. on Machine Learning (ICML)*, pages 1126–1135, 2017.

[3] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4077–4087, 2017.

[4] Siavash Khodadadeh, Ladislau Bölöni, and Mubarak Shah. Unsupervised meta-learning for few-shot image classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10132–10142, 2019.

[5] Antreas Antoniou and Amos Storkey. Assume, augment and learn: Unsupervised few-shot meta-learning via random labels and data augmentation. *arXiv preprint arXiv:1902.09884*, 2019.

[6] Kyle Hsu, Sergey Levine, and Chelsea Finn. Unsupervised learning via meta-learning. In *Int'l Conf. on Learning Representations (ICLR)*, 2019.

[7] Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

[8] Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. *Learning a synaptic learning rule*. Université de Montréal, Département d'Informatique et de Recherche Opérationelle, 1990.

[9] Devang K Naik and Richard J Mammone. Meta-neural networks that learn by learning. In *[Proc. 1992] IJCNN Int'l Joint Conf. on Neural Networks*, volume 1, pages 437–442, 1992.

[10] Sebastian Thrun and Lorien Pratt. Learning to learn. *Kluwer Academic Publishers*, 1998.

[11] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *Int'l Conf. on Learning Representations (ICLR)*, 2016.

[12] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. Meta-learning with temporal convolutions. *arXiv preprint arXiv:1707.03141*, 2017.

[13] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[14] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-Learning with Latent Embedding Optimization. In *Int'l Conf. on Learning Representations (ICLR)*, 2019.

[15] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 113–124, 2019.

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

[17] Mengye Ren, Sachin Ravi, Eleni Triantafillou, Jake Snell, Kevin Swersky, Josh B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-Learning for Semi-Supervised Few-Shot Classification. In *Int'l Conf. on Learning Representations (ICLR)*, 2018.

[18] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sungju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. In *Int'l Conf. on Learning Representations (ICLR)*, 2019.

[19] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples. In *Int'l Conf. on Learning Representations (ICLR)*, 2020.

[20] Duo Wang, Yu Cheng, Mo Yu, Xiaoxiao Guo, and Tao Zhang. A hybrid approach with optimization-based and metric-based meta-learner for few-shot learning. *Neurocomputing*, 349:202–211, 2019.

[21] P Kingma Diederik and Max Welling. Auto-encoding variational bayes. In *Proc. of the Int'l Conf. on Learning Representations (ICLR)*, volume 1, 2014.

[22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2672–2680, 2014.

[23] Qi Mao, Hsin-Ying Lee, Hung-Yu Tseng, Siwei Ma, and Ming-Hsuan Yang. Mode seeking generative adversarial networks for diverse image synthesis. *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1429–1437, 2019.

[24] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *Proc. of the Int'l Conf. on Learning Representations (ICLR)*, 2018.

[25] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proc. of the Annual Meeting of the Cognitive Science Society*, volume 33, 2011.

[26] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proc. of Int'l Conf. on Computer Vision (ICCV)*, December 2015.

[27] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *Proc. of Int'l Conf. on Learning Representations (ICLR)*, 2016.

[28] David Berthelot*, Colin Raffel*, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *Int'l Conf. on Learning Representations (ICLR)*, 2019.

[29] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *Int'l Conf. on Learning Representations (ICLR)*, 2017.

[30] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. In *Int'l Conf. on Learning Representations (ICLR)*, 2017.

[31] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 132–149, 2018.

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Int'l Conf. on Learning Representations (ICLR)*, 2015.

[33] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3235–3246, 2018.