7 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. Our source code is also available on Github [2].

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

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 translation to Omniglot images.

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	$\mathbf{64.59 \pm 0.76}$	$\mathbf{63.08 \pm 0.79}$

Besides, we perform a hyperparameter search on CelebA attributes benchmark. Table 8 demonstrates the results for our experiments. We see that searching for hyperparameters for CelebA is almost as easy as doing the same thing for Omniglot. LASIUM-N with $\sigma^2 = 0.25$ outperforms state-of-the-art in this benchmark. We also see a bad performance in the case of LASIUM-OC, which we expected as the number of classes in this benchmark's tasks is $N = 2$. Thus samples generated during meta-learning are limited to only instances on the line connecting two anchor latent vectors. It is not the case for LASIUM-N and LASIUM-RO since we can sample latent codes in the neighborhood or any direction from anchor points in the latent space.

---

[2] https://github.com/siavash-khodadadeh/MetaLearning-TF2.0

Table 8: Accuracy of different proposed strategies on CelebA attributes task for GAN with 2-way, 5-shot tasks with $K^{(val)} = 5$. The results are averaged over 1000 downstream tasks and ± indicates 95% confidence interval.

Sampling Strategy	Hyperparameters	GAN-MAML	GAN-Proto
LASIUM-N	σ^2=0.1	71.83 ± 1.08	62.99 ± 1.14
LASIUM-N	σ^2=0.25	75.07 ± 1.08	70.49 ± 1.14
LASIUM-N	σ^2=0.5	71.41 ± 1.13	69.96 ± 1.15
LASIUM-N	σ^2=1.0	60.37 ± 1.01	69.98 ± 1.16
LASIUM-N	σ^2=2.0	50.00 ± 0.00	70.33 ± 1.14
LASIUM-RO	α=0.2	62.06 ± 1.06	62.73 ± 1.18
LASIUM-RO	α=0.4	67.57 ± 1.11	68.19 ± 1.12
LASIUM-RO	α=0.5	71.04 ± 1.03	68.94 ± 1.12
LASIUM-OC	α=0.25	59.69 ± 1.11	53.67 ± 1.02
LASIUM-OC	α=0.5	60.25 ± 1.08	56.05 ± 1.08

7.1 NEURAL NETWORK ARCHITECTURES

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 MSGAN (aka Miss-GAN) described in Mao et al. (2019).

For the CelebA GAN experiments, we use the pre-trained network architecture, progressive growing of GANs (ProGAN), described in Karras et al. (2018). 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 the supplemental material.

7.2 IMPACT OF GAN TRAINING ON LASIUM

Do we need a generative model that generates very high-quality images from data or can a premature trained GAN also work? We performed an ablation study to evaluate the impact of GAN training on LASIUM. First, we trained a generative network on Omniglot dataset with adversarial training for 500 epochs and saved the corresponding weights at every epoch. Then we trained LASIUM with various generative networks at different epochs. For the sake of comparison, we stopped LASIUM after 1000 iterations.

Figure 5 demonstrates the impact of GAN training on LASIUM. We visualize an image generated with the same exact latent code after different epochs. We can see that eventually, this latent code result in generating character "R" (after epoch 400 and 500). We see that GAN stabilizes after 400 epochs while LASIUM stabilizes sooner around epoch 200. Nevertheless, the impact of training GAN for at least 50 epochs is correlated with LASIUM performance.

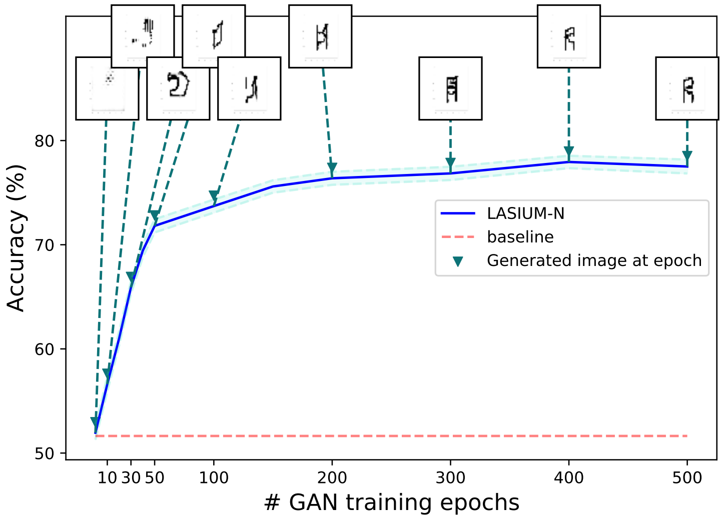

Figure 5: The impact of GAN training on LASIUM accuracy. The blue line shows test accuracy after 1000 iterations of LASIUM-N training with a 95% confidence interval on 1000 one-shot tasks with $K^{(val)} = 15$. The images generated by GAN are shown at epochs 0, 10, 30, 50, 100, 200, 300, 400, and 500. All of the images are generated from the same latent vector. The red line shows the training from scratch baseline.

7.3 ABLATION STUDY ON IMPACT OF ϵ ON LASIUM

In this section, we evaluate the accuracy of LASIUM with respect to the value of ϵ. For the sake of comparison, we consider LASIUM-N with $\sigma^2 = 0.5$ and stop the training after 1000 iterations on Omniglot. The results are reported on the same 1000 one-shot tasks with $K^{(val)} = 15$ in Table 9. Furthermore, the last column shows the number of times resampling occurred since at least two of the initial sampled latent codes were in a distance smaller than ϵ from each other. We found that (within reasonable bounds) the choice of this hyperparameter has a small but not negligible impact on the performance of the algorithm.

7.4 TRAINING LASIUM ON FUNGI

We also tried LASIUM on Fungi dataset. We report LASIUM-N-GAN-MAML accuracy over 1000 downstream tasks generated randomly from test dataset following Meta-dataset evaluation protocol proposed by Triantafillou et al. (2020). For the choice of generative network, we used state-of-the-art StyleGAN-v2 by Karras et al. (2020), and we trained it on Fungi images. Figure 6 shows some of the examples generated by StyleGAN-v2. Table 10 shows the results on LASIUM and some other relevant algorithms.

Table 9: Accuracy of LASIUM-N with $\sigma^2 = 0.5$ on Omniglot dataset with respect to different values of ϵ. ± indicates 95% confidence interval.

ϵ	Accuracy (%)	# Resampling Task
0.0	77.27 ± 0.62	0
0.1	77.34 ± 0.62	0
1	77.21 ± 0.62	0
10.0	77.54 ± 0.62	0
100.0	77.08 ± 0.63	0
125.0	79.51 ± 0.61	0
187.5	78.87 ± 0.60	395
218.75	77.95 ± 0.62	6012
234.375	77.15 ± 0.63	27432
242.1875	77.15 ± 0.64	61118
250.0	78.49 ± 0.61	155499
253.15625	78.48 ± 0.62	238714
256.3125	78.05 ± 0.62	378742
265.625	78.56 ± 0.61	1472860
281.25	77.73 ± 0.63	25936957
312.5	–	All
375	–	All
500.0	–	All
1000.0	–	All

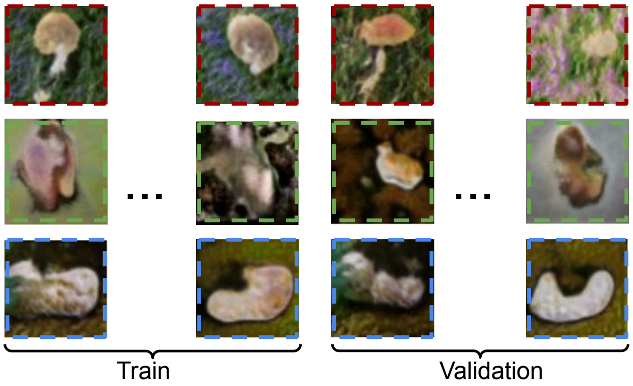

Figure 6: Meta-training tasks for Fungi constructed by LASIUM-N with $\sigma^2 = 0.25$

Table 10: Accuracy of 5-way 1-shot learning on the Fungi dataset (part of the proposed Meta-dataset by Triantafillou et al. (2020)). For each system we indicate the dataset on which the meta-training phase was performed. The results for supervised first-order MAML are from Triantafillou et al. (2020). LASIUM-N was run with $\sigma^2 = 0.25$ and used the StyleGAN-v2 trained on the unsupervised version of the Fungi dataset, as discussed in the text.

Method	Dataset	Accuracy (%)
Training from scratch	-	26.10 ± 0.42
fo-UMTRA	Unsupervised Fungi	28.27 ± 0.46
LASIUM-N-GAN-fo-MAML	Unsupervised Fungi	29.43 ± 0.49
LASIUM-N-GAN-MAML	Unsupervised Fungi	$\mathbf{31.29 \pm 0.52}$
fo-MAML	Supervised Imagenet	32.10 ± 1.10
fo-MAML	Supervised Meta-dataset	33.54 ± 1.11