A bridge between hyperparameter optimization and learning-to-learn

Luca Franceschi[1,2,*], **Paolo Frasconi**[3], **Michele Donini**[1], **Massimiliano Pontil**[1,2]
[1]Istituto Italiano di Tecnologia, [2]University College London, [3]Università degli Studi di Firenze
[*]luca.franeschi@iit.it

Abstract

We consider a class of a nested optimization problems involving inner and outer objectives. We observe that by taking into explicit account the optimization dynamics for the inner objective it is possible to derive a general framework that unifies gradient-based hyperparameter optimization and meta-learning (or learning-to-learn). Depending on the specific setting, the variables of the outer objective take either the meaning of hyperparameters in a supervised learning problem or parameters of a meta-learner. We show that some recently proposed methods in the latter setting can be instantiated in our framework and tackled with the same gradient-based algorithms. Finally, we discuss possible design patterns for learning-to-learn and present encouraging preliminary experiments for few-shot learning.

1 Introduction and framework

Hyperparameter optimization (see, e.g., Moore et al., 2011; Bergstra et al., 2011; Bergstra and Bengio, 2012; Maclaurin et al., 2015; Bergstra et al., 2013; Hutter et al., 2015; Franceschi et al., 2017) is the problem of tuning the value of certain parameters that control the behavior of a learning algorithm. This is typically obtained by minimizing the expected error w.r.t. the hyperparameters, using the empirical loss on a validation set as a proxy. Meta-learning (see, e.g., Thrun and Pratt, 1998; Baxter, 1998; Maurer, 2005; Maurer et al., 2016; Vinyals et al., 2016; Santoro et al., 2016; Ravi and Larochelle, 2017; Mishra et al., 2017; Finn et al., 2017) is the problem of inferring a learning algorithm from a collection of datasets in order to obtain good performances on unseen datasets. Although hyperparameter optimization and meta-learning are different and apparently unrelated problems, they can be both formulated as special cases of a wider framework that we will introduce. This connection and our observations on learning-to-learn represent the main contribution of this work.

We start by considering bilevel optimization problems of the form

$$\min_{\lambda \in \Lambda} f(\lambda) \tag{1.1}$$

where $\Lambda \subseteq \mathbb{R}^m$ and

$$f(\lambda) = \inf_w \{E(w, \lambda) : w \in \operatorname{argmin}_w L_\lambda(w)\}. \tag{1.2}$$

We will call the function $f : \Lambda \to \mathbb{R}$ the *outer objective* (or outer loss), and, for every $\lambda \in \Lambda$, $L_\lambda : \mathbb{R}^d \to \mathbb{R}$ is called the *inner objective* (or inner loss). Note that $\{L_\lambda \ : \ \lambda \in \Lambda\}$ is a class of objectives parameterized by λ. As prototypical example of (1.2) consider the case that L_λ is a regularized empirical error for supervised learning, E is an (unregularized) validation error, λ a regularization parameter and w the parameters of the model.

In general problem (1.1) may not have a solution (Colson et al., 2007) but we assume this is in not an issue here. Following (Domke, 2012; Maclaurin et al., 2015; Franceschi et al., 2017) we approximate

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

the solutions of problem (1.1) by replacing the "argmin" in problem (1.2) by the T-th iterate obtained by an iterative system of the form

$$w_0 = \Phi_0(\lambda); \quad w_t = \Phi_t(w_{t-1}, \lambda) \;\; t = 1, \dots, T \tag{1.3}$$

where T is the number of iterations, $\Phi_0 : \mathbb{R}^m \to \mathbb{R}^d$ is a smooth initialization mapping and, for every $t \in \{1, \dots, T\}$, $\Phi_t : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}^d$ is a smooth mapping that represents the operation performed by the t-th step of an optimization algorithm. Since the algorithm might involve auxiliary variables v, e.g. velocities when using stochastic gradient descent with momentum (SGDM), we replace w with a state vector $s = (w, v)$. Using this notation, we approximate problem (1.1) by the constrained optimization problem

$$\begin{aligned} \min_{\lambda, s_1, \dots, s_T} \quad & f(\lambda) = E(s_T, \lambda) \\ \text{subject to} \quad & s_0 = \Phi_0(\lambda) \\ & s_t = \Phi_t(s_{t-1}, \lambda), \; t \in \{1, \dots, T\}. \end{aligned} \tag{1.4}$$

This reformulation of the original problem allows for an efficient computation of the gradient of f, either in time or in memory (Maclaurin et al., 2015; Franceschi et al., 2017), by making use of Reverse or Forward mode algorithmic differentiation (Griewank and Walther, 2008). Moreover, by considering explicitly the learning dynamics, it is possible to compute the hyper-gradient with respect to the hyper-parameters (e.g. step size or momentum factor if Φ is SGDM), as opposed to other methods that compute the hyper-gradient at the minimizer of the inner objective (Pedregosa, 2016). This key fact allows for the inclusion of learning-to-learn, more specifically learning-to-optimize, into the framework. In the next two sections, we show that gradient-based hyperparameter optimization and learning-to-learn share this same underlying mathematical formulation.

2 Gradient-based hyperparameter optimization

In the context of hyperparameter optimization, we are interested in minimizing the generalization error of a model $g : \mathcal{X} \to \mathcal{Y}$, parametrized by a vector w, with respect to λ. The outer optimization variables are in this context called hyperparameters and the outer objective is generally an empirical validation loss. Specifically, a set of labeled examples $D = \{z_i\}_{i=1}^n$, where $z_i = (x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, is spit into training and validation sets D_{tr}, D_{val}. The inner objective is computed on (mini-batches of) examples from D_{tr} while the outer objective, that represents a proxy for the generalization error of g, is computed on D_{val}. Assuming, for simplicity, that the optimization dynamics is given by stochastic gradient descent, and thus that the state $s = w$, problem (1.4) becomes

$$\begin{aligned} \min_{\lambda, w_1, \dots, w_T} \quad & f(\lambda) = \sum_{z \in D_{\text{val}}} E(w_T, z) \\ \text{subject to} \quad & w_0 = \Phi_0(\lambda) \\ & w_t = w_{t-1} - \eta \sum_{z \in B_t} \nabla L_\lambda(w_{t-1}, z), \quad t \in \{1, \dots, T\}, \end{aligned} \tag{2.1}$$

where $B_t \subset D_{\text{tr}}$ is a mini-batch of samples at the t-th iteration, η is a learning rate (a component of λ) and where we made explicit the dependence of the loss functions on the examples. In this setting, the outer loss E does not depend explicitly on the hyperparameters λ. The above formulation allows for the computation of the hyper-gradient of any real valued hyperparameter, so that hyperparameters can be optimized with a gradient descent procedure. Having access to hyper-gradients makes it feasible to optimize a number of hyperparameters of the same order of that of parameters, a situation which arise in the setting of learning-to-learn.

Since in this context the total number of iterations might be often high due to large datasets or complex models, to speed up the optimization and to reduce memory requirements, it is possible to compute partial hyper-gradients at intermediate iterations, either in reverse or forward mode, and update λ online several times before reaching the final iteration T (Franceschi et al., 2017).

3 Learning-to-learn

The aim of meta-learning is to learn an algorithm capable of solving ground learning problems originated by a (unknown) distribution $\mathscr{P}$. A meta-dataset $\mathcal{D} = \{D^j\}_{j=1}^N$ is thus a collection of datasets, or *episodes*, sampled from $\mathscr{P}$, where each dataset $D^j = \{z_i^j\}_{i=1}^{n_j}$ with $z_i^j = (x_i^j, y_i^j) \in$

$\mathcal{X}^j \times \mathcal{Y}^j$ is linked to a specific task. We are interested in learning an algorithm capable of "producing" ground models $g^j : \mathcal{X}^j \to \mathcal{Y}^j$, which we assume identified by parameter vectors w^j. The algorithm itself can be thought of as a meta-model q, or *meta-learner*, parametrized by a vector λ, so that $w^j = q(D^j, \lambda)$. The meta-learner $q : \mathcal{D} \to \mathcal{W}$ is viewed as a function which maps datasets to models (or weights). As a learning dynamics, in general, the meta-learner can act in an iterative way, so that $q = q_T \circ q_{T-1} \circ \cdots \circ q_0$. Moreover, like the case of a standard optimization algorithm, the meta-learner can make use of auxiliary variables v^j, forming state vectors $s^j = (w^j, v^j)$. Since the ground models should exhibit good generalization performances on their specific task, each dataset D^j can be split into training and validation[1] sets $D^j_{\text{tr}}, D^j_{\text{val}}$, and q can be trained to minimize the average validation error over tasks, which constitutes a natural outer objective in this setting. For each task, the meta-learner produces a sequence of states $s^j_0 = q_0(D^j_{\text{tr}}, \lambda), \ldots, s^j_T = q_T(D^j_{\text{tr}}, s^j_{T-1}, \lambda) = q(D^j_{\text{tr}}, \lambda)$.

We can thus formulate problem (1.4) for learning-to-learn as follows:

$$\begin{aligned} \min_{\lambda, s^1_0, \ldots, s^N_T} \quad & f(\lambda) = \sum_{j=1}^{N} \frac{1}{|D^j_{\text{val}}|} \sum_{z \in D^j_{\text{val}}} E^j(s^j_T, \lambda, z) \\ \text{subject to} \quad & s^j_0 = q_0(D^j_{\text{tr}}, \lambda) \\ & s^j_t = q_t(D^j_{\text{tr}}, s^j_{t-1}, \lambda) \quad j \in \{1, \ldots, N\},\ t \in \{1, \ldots, T\}, \end{aligned} \tag{3.1}$$

where the functions E^j are task specific losses. The meta-model plays the role of the mapping Φ in (1.3), thus reducing the problem of learning-to-learn to that of *learning a training dynamics*, or its associated parameters λ. The meta-learner parameters mirror the hyperparameters in the context of hyperparameter optimization in Section 2 and can be optimized with a gradient descent procedure on the outer objective. The inner objective does not appear explicitly in problem (3.1), but we assume that the meta-learner has access to task specific inner objectives L^j.

While in principle q could be implemented by any parametrized mapping, the design of meta-learning models can follow three non-exclusive natural directions:

• *Learning-to-optimize*: q can replace a gradient-based optimization algorithm (Andrychowicz et al., 2016; Wichrowska et al., 2017), acting on the weights of ground models as $w^j_{t+1} = w^j_t - q_t(B^j_t, s^j_{t-1}, L^j, \nabla_w L^j)$, where $B^j_t \subseteq D^j_{\text{tr}}$ is a mini-batch of examples. The meta-model is often interpreted (Ravi and Larochelle, 2017) as a recurrent neural network, whose hidden states v^j are the analog of auxiliary variables in Section 1. Alongside the update rule, it is possible to learn an initialization for the ground models weights, described by the mapping q_0. For instance, (Finn et al., 2017) set $q_0(D^j_{\text{tr}}, \lambda) = \lambda = w^j_0$ assuming that all the input and output spaces of the tasks in $\mathcal{D}$ have the same dimensionality, and use gradient descent for the following steps;

• *Learning meta-representations*: the meta-learner is composed by a gradient descent procedure and a mapping from ground task instances x to intermediate representations $h(x, \lambda) \in \mathcal{Z}$. In this case the ground models are mappings $g^j : \mathcal{Z} \to \mathcal{Y}^j$ and an update on ground model weights is of the form $w^j_{t+1} = w^j_t - \eta \sum_{(x,y) \in D^j_{\text{tr}}} \nabla L^j(w^j_{t-1}, h(x, \lambda), y)$. This approach can prove particularly useful in cases where the instance spaces are structurally different among tasks. It differs from standard representation learning in deep learning (Bengio et al., 2009; Goodfellow et al., 2016) since the meta-training loss is specifically designed for promoting generalization across tasks;

• *Learning ground loss functions*: the meta-learner can be a gradient descent algorithm that optimize a learned inner objective. For example, we may directly parametrize the training error L (which in a standard supervised learning setting is usually a mean squared error for regression or a cross-entropy loss for classification), or to learn a multitask regularizer which provide a coupling among the different learning tasks in $\mathscr{P}$.

In the next section we presents experiments that explore the second design pattern. For experiments on gradient-based hyperparameter optimization we refer to (Franceschi et al., 2017).

4 Experiments

We report preliminary results on the problem of few-shots learning, using MiniImagenet (Vinyals et al., 2016), a subset of ImageNet (Deng et al., 2009), that contains 60000 downsampled images

[1]Note that some authors (e.g. Ravi and Larochelle, 2017) refer to this latter set as the test set.

from 100 different classes. As in (Ravi and Larochelle, 2017), we build meta-datasets by sampling ground classification problems with 5 classes, where each episode $D = (D_{\text{tr}}, D_{\text{val}})$ is constructed so that D_{tr} contains 1 (one-shot learning) or 5 (5-shots learning) examples per class and D_{val} contains 15 examples per class. Out of 100 classes, 64 classes are included in a training meta-dataset $\mathcal{D}_{\text{tr}}$ from which we sample datasets for solving problem (3.1); 16 classes form a validation meta-dataset $\mathcal{D}_{\text{val}}$ which is used to tune meta-learning hyperparameters while a third meta-dataset $\mathcal{D}_{\text{test}}$ with the remaining 20 classes is held out for testing. We use the same split and images proposed by (Ravi and Larochelle, 2017).

Our meta-model design involves the learning of a cross-episode intermediate representation. We design a meta-representation h as a four layers convolutional neural network, where each layer is composed by a convolution with 32 filters, a batch normalization followed by a ReLU activation and a 2x2 max-pooling. The ground models g^j are logistic regressors that take as input the output of h. Ground models parameters w^j are initialized to 0 and optimized by few gradient descent steps on the cross-entropy loss computed on D^j_{tr} (note that, fixing λ, the inner loss is convex with respect to w^j). The step-size η is also learned. For each task the final classification model is thus given by the composition of the meta-learner with the ground learner so that the prediction for an input sample x is equal to $g^j(h(x, \lambda), w^j_T)$. We highlight that, unlike in (Finn et al., 2017), the weight of the representation mapping λ are kept constant for each episode, and learned across datasets by minimizing the outer objective $f(\lambda)$ in (3.1). We compute a stochastic gradient of $f(\lambda)$ by sampling mini-batches of 4 episodes and use Adam with decaying learning rate as optimization method for the meta-model variables λ. Finally we perform early stopping during meta-training and optimize the number of gradient descent updates (see Figure 1) based on the mean accuracy on the test sets of episodes in $\mathcal{D}_{\text{val}}$. We report results in Table 1. The proposed method, called *Hyper-Representation*, achieves a competitive result despite its simplicity, highlighting the relative importance of learning a good representation independent from specific tasks, on the top of which simple logistic classifiers can perform and generalize well. Figure 2 provides a visual example of the goodness of the learned representation, showing that examples from similar classes (different dog breeds) are mapped near by h and, conversely, samples from dissimilar classes are mapped afar. In Appendix A we empirically show the importance of learning h with the proposed framework.

5-classes accuracy %	1-shot	5-shots
Fine-tuning	28.86 ± 0.54	49.76 ± 0.79
Nearest-neighbor	41.08 ± 0.70	51.04 ± 0.65
Matching nets	43.44 ± 0.77	55.31 ± 0.73
Meta-learner LSTM	43.56 ± 0.84	60.60 ± 0.71
MAML	48.70 ± 1.75	63.11 ± 0.92
Hyper-Repr. (ours)	47.01 ± 1.35	61.97 ± 0.76

Table 1: Mean accuracy scores with 95% confidence intervals, computed on episodes from $\mathcal{D}_{\text{test}}$, of various methods on 1-shot and 5-shot classification problems on MiniImagenet.

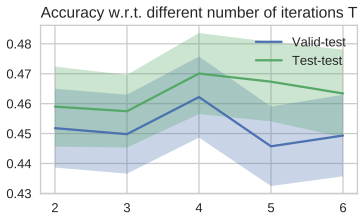

Figure 1: Meta-validation of the number of gradient descent steps on ground models parameters T for one-shot learning.

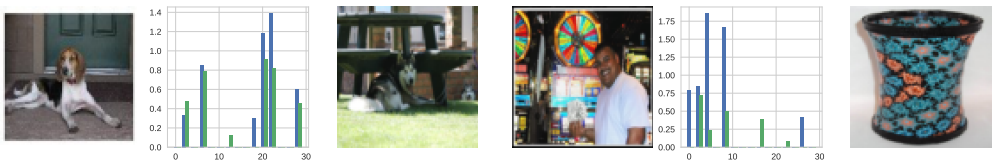

Figure 2: After sampling two datasets $D \in \mathcal{D}_{\text{tr}}$ and $D' \in \mathcal{D}_{\text{test}}$, we show on the left the two images $x \in D$, $x' \in D$ that minimize $||h(x, \lambda) - h(x', \lambda)||$ and on the right the ones that maximize it. In between each of the two couples we compare a random subset of components of $h(x, \lambda)$ (blue) and $h(x', \lambda)$ (green).

Ongoing experiments aim at combining the first and the second design patterns outlined in Section 4 both in depth (lower layers weights are hyperparameters and higher layers weights initial points) and in width (a portion of filters constitutes the meta-representation, while the weights relative to the rest of filters are considered initialization), and at experimenting with the third pattern. Moreover we plan to explore settings in which different datasets come form various domains (e.g. visual, natural language, speech, etc.), are linked to diverse tasks (e.g. classification, localization, segmentation, generation and others) and have structurally different instance spaces.

References

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989.

Baxter, J. (1995). Learning internal representations. In *Proceedings of the eighth annual conference on Computational learning theory*, pages 311–320. ACM.

Baxter, J. (1998). Theoretical models of learning to learn. *Learning to learn*, pages 71–94.

Bengio, Y. et al. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

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

Bergstra, J., Yamins, D., and Cox, D. D. (2013). Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. *ICML (1)*, 28:115–123.

Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554.

Colson, B., Marcotte, P., and Savard, G. (2007). An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

Domke, J. (2012). Generic Methods for Optimization-Based Modeling. In *AISTATS*, volume 22, pages 318–326.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, pages 1126–1135.

Franceschi, L., Donini, M., Frasconi, P., and Pontil, M. (2017). Forward and reverse gradient-based hyperparameter optimization. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, pages 1165–1173.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Griewank, A. and Walther, A. (2008). *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM.

Hutter, F., Lücke, J., and Schmidt-Thieme, L. (2015). Beyond Manual Tuning of Hyperparameters. *KI - Künstliche Intelligenz*, 29(4):329–337.

Maclaurin, D., Duvenaud, D., and Adams, R. P. (2015). Gradient-based hyperparameter optimization through reversible learning. In *Proceedings of the 32nd International Conference on Machine Learning*.

Maurer, A. (2005). Algorithmic stability and meta-learning. *Journal of Machine Learning Research*, 6:967–994.

Maurer, A., Pontil, M., and Romera-Paredes, B. (2016). The benefit of multitask representation learning. *Journal of Machine Learning Research*, 17(81):1–32.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. (2017). Meta-Learning with Temporal Convolutions. *arXiv:1707.03141 [cs, stat]*.

Moore, G., Bergeron, C., and Bennett, K. P. (2011). Model selection for primal SVM. *Machine Learning*, 85(1-2):175–208.

Pedregosa, F. (2016). Hyperparameter optimization with approximate gradient. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 737–746. PMLR.

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

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016). Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850.

Thrun, S. and Pratt, L. (1998). *Learning to learn*. Springer.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. (2016). Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, pages 3630–3638.

Wichrowska, O., Maheswaranathan, N., Hoffman, M. W., Colmenarejo, S. G., Denil, M., de Freitas, N., and Sohl-Dickstein, J. (2017). Learned optimizers that scale and generalize. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 3751–3760.

A On variants of representation learning methods

We report in Table 2 additional results on a series of experiments for one-shot learning on Mini-Imagenet with the aim of comparing the method for learning a meta-representation outlined in Sections 3 and 4 with other methods that involve the factorization of a classifier as $g^j \circ h$. The representation mapping h is either pretrained on the classification problem with all the images in the training meta-dataset or learned with different meta-learning algorithms. In all the experiments, for each episode g^j is a multinomial logistic regressor learned with few iterations of gradient descent as described in Section 4.

Method	Accuracy 1-shot	Method	Accuracy 1-shot
NN-conv	39.97	*Bilevel-train 1x5*	27.36
NN-linear	41.50	*Bilevel-train 16x5*	29.63
NN-softmax	41.36	*Approx-train 1x5*	24.74
Multiclass-conv	36.57	*Approx-train 16x5*	38.80
Multiclass-linear	43.02	*Classic-train 1x5*	24.70
Multiclass-softmax	37.60	*Classic-train 16x5*	40.46

Table 2: Performance of various methods where the representation is either transfered from models trained in a standard multiclass supervised learning setting (left column) or learned in a meta-learning setting (left column).

In the experiments in the left column we use as representation mapping h the outputs of different layers of two distinct neural networks (denominated *NN* and *Multiclass* in the Table) trained with a standard multiclass supervised learning approach on the totality of examples contained in the training meta-dataset (600 examples for each of the 64 classes [2]). The first network *NN*, which has 64 filters per layer, achieves a test accuracy of 43.41%. It is the same network used to reproduce the *Nearest-neighbor* baseline in Table 1 and it has been trained with an early stopping procedure on the nearest-neighbor classification accuracy computed on episodes sampled from the validation meta-dataset. The network *Multiclass*, which has 32 filters per layer, has instead been trained with an early stopping procedure on the accuracy on a small held-out validation set. Achieving a test accuracy of 46.33, this second model is superior on the (standard) multiclass classification problem. For each of the network we report experiment using as representation different layers. Specifically:

- *conv*: we use the output of the last convolutional layer as representation, that is $h(x) \in \mathbb{R}_+^{2304}$ for *NN* and $h(x) \in \mathbb{R}_+^{1152}$ for *Multiclass*;
- *linear*: we use as representation the linear output layer (before applying the softmax operation), so that $h(x) \in \mathbb{R}^{64}$.

[2] We hold-out 3840 uniformly drawn samples to form a small test set.

- *softmax*: the representation is given by the probability distribution output of the network; in this case $h(x) \in (0,1)^{64}$

The *linear* representation yields the best result for both of the networks and in the case of *Multiclass* achieves comparable results with previously proposed meta-learning methods.

The experiments in the right column, where h is learned with meta-learning techniques, span in two directions: the first is that of verifying the impact of various approximations on the computation of the hyper-gradient, and the second is to empirically assess the importance of the training/validation splitting of each training episode. In the experiments denoted *Bilevel-train*, we use a bilevel approach but, unlike in section 4, we optimize the parameter vector λ of the representation mapping by minimizing the loss on the training sets. The outer objective is thus given by

$$f(\lambda) = \sum_{j=1}^{N} \frac{1}{|D^j_{\text{tr}}|} \sum_{z \in D^j_{\text{tr}}} E^j(w^j_T, \lambda, z).$$

We consider episodes with training set composed by 1 and 16 examples per class, denoted *(1x5)* and *(16x5)* respectively. In these cases $f(\lambda)$ goes quickly to 0 and the learning ceases after few hundred iterations. In *Approx* experiments we consider an approximation of the hyper-gradient $\nabla f(\lambda)$ by disregarding the optimization dynamics of the inner objectives (i.e. we set $\partial_\lambda w^j_T = 0$). We also run this experiment considering the training/validation splitting obtaining a final test accuracy of 41.12%. In the experiments denoted as *Classic* we jointly minimize

$$f(\lambda, w^1, \dots, w^N) = \sum_{j=1}^{N} \frac{1}{|D^j_{\text{tr}}|} \sum_{z \in D^j_{\text{tr}}} E^j(w^j, \lambda, z).$$

and treat the problem as a standard multitask learning problem as suggested in (Baxter, 1995) (with the exception that at each iteration of gradient descent we evaluate f on a mini-batch of 4 episodes).

This series of experiments suggest that both the training/validation splitting and the full computation of the hyper-gradient constitute key factors for learning a good meta-representation. Nevertheless, provided that the training sets contain a sufficient number of examples, also the joint optimization method achieves decent results, while learning the representation using only the training sets of one-shot episodes (experiments *train 1x5*) proves unsuccessful in every tested setting, a result [3] in line with the theoretical analysis in (Baxter, 1995). On the other side, using pretrained representations, specially in a low-dimensional space, turns out to be a rather effective baseline. One possible explanation is that, in this context, some classes in the training and testing meta-datasets are rather similar (e.g. various dog breeds) and thus ground classifiers can leverage on very specific representations.

[3] It remains interesting to explore both theoretically and empirically how does the size of validation sets of meta-training episodes impacts on the generalization performances of meta-learning algorithms.