# Meta-Learning in Neural Networks: A Survey

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey

**Abstract**—The field of meta-learning, or learning-to-learn, has seen a dramatic rise in interest in recent years. Contrary to conventional approaches to AI where tasks are solved from scratch using a fixed learning algorithm, meta-learning aims to improve the learning algorithm itself, given the experience of multiple learning episodes. This paradigm provides an opportunity to tackle many conventional challenges of deep learning, including data and computation bottlenecks, as well as generalization. This survey describes the contemporary meta-learning landscape. We first discuss definitions of meta-learning and position it with respect to related fields, such as transfer learning and hyperparameter optimization. We then propose a new taxonomy that provides a more comprehensive breakdown of the space of meta-learning methods today. We survey promising applications and successes of meta-learning such as few-shot learning and reinforcement learning. Finally, we discuss outstanding challenges and promising areas for future research.

**Index Terms**—Meta-learning, learning-to-learn, few-shot learning, transfer learning, neural architecture search

## 1 Introduction

Contemporary machine learning models are typically trained from scratch for a specific task using a fixed learning algorithm designed by hand. Deep learning-based approaches specifically have seen great successes in a variety of fields [1], [2], [3]. However there are clear limitations [4]. For example, successes have largely been in areas where vast quantities of data can be collected or simulated, and where huge compute resources are available. This excludes many applications where data is intrinsically rare or expensive [5], or compute resources are unavailable [6].

Meta-learning is the process of distilling the experience of multiple learning episodes – often covering a distribution of related tasks – and using this experience to improve future learning performance. This 'learning-to-learn' [7] can lead to a variety of benefits such as improved data and compute efficiency, and it is better aligned with human and animal learning [8], where learning strategies improve both on a lifetime and evolutionary timescales [8], [9], [10].

Historically, the success of machine learning was driven by advancing hand-engineered features [11], [12]. Deep learning realised the promise of feature representation learning [13], providing a huge improvement in performance for many tasks [1], [3] compared to prior hand-designed features. Meta-learning in neural networks aims to provide the next step of integrating joint feature, model, and *algorithm* learning; that is, it targets replacing prior hand-designed learners with learned learning algorithms [7], [14], [15], [16].

- *Timothy Hospedales is with the Samsung AI Centre, CB1 2JD Cambridge, U.K., and also with the University of Edinburgh, EH8 9AB Edinburgh, U.K. E-mail: t.hospedales@ed.ac.uk.*
- *Antreas Antoniou, Paul Micaelli, and Amos Storkey are with the University of Edinburgh, EH8 9AB Edinburgh, U.K. E-mail: {a.antoniou, paul.micaelli, a.storkey}@ed.ac.uk.*

*Manuscript received 9 Oct. 2020; revised 11 Mar. 2021; accepted 27 Apr. 2021. Date of publication 11 May 2021; date of current version 4 Aug. 2022.*
*(Corresponding author: Timothy Hospedales.)*
*Recommended for acceptance by L. Wang.*
*Digital Object Identifier no. 10.1109/TPAMI.2021.3079209*

Neural network meta-learning has a long history [7], [17], [18]. However, its potential as a driver to advance the frontier of the contemporary deep learning industry has led to an explosion of recent research. In particular meta-learning has the potential to alleviate many of the main criticisms of contemporary deep learning [4], for instance by improving data efficiency, knowledge transfer and unsupervised learning. Meta-learning has proven useful both in multi-task scenarios where task-agnostic knowledge is extracted from a family of tasks and used to improve learning of new tasks from that family [7], [19]; and single-task scenarios where a single problem is solved repeatedly and improved over multiple *episodes* [15], [20], [21], [22]. Successful applications have been demonstrated in areas spanning few-shot image recognition [19], [23], unsupervised learning [16], data efficient [24], [25] and self-directed [26] reinforcement learning (RL), hyperparameter optimization [20], and neural architecture search (NAS) [21], [27], [28].

Many perspectives on meta-learning can be found in the literature, in part because different communities use the term differently. Thrun [7] operationally defines learning-to-learn as occurring when a learner's performance at solving tasks drawn from a given task family improves with respect to the number of tasks seen. (*cf.*, conventional machine learning performance improves as more data from a single task is seen). This perspective [29], [30], [31] views meta-learning as a tool to manage the 'no free lunch' theorem [32] and improve generalization by searching for the algorithm (inductive bias) that is best suited to a given problem, or problem family. However, this definition can include transfer, multi-task, feature-selection, and model-ensemble learning, which are not typically considered as meta-learning today. Another usage of meta-learning [33] deals with algorithm selection based on dataset features, and becomes hard to distinguish from automated machine learning (AutoML) [34].

In this paper, we focus on contemporary *neural-network* meta-learning. We take this to mean algorithm learning as per [29], [30], but focus specifically on where this is achieved by *end-to-end* learning of an *explicitly defined objective function* (such as cross-entropy loss). Additionally we consider single-task

meta-learning, and discuss a wider variety of (meta) objectives such as robustness and compute efficiency.

This paper provides a unique, timely, and up-to-date survey of the rapidly growing area of neural network meta-learning. In contrast, previous surveys are rather out of date and/or focus on algorithm selection for data mining [29], [33], [35], [36], AutoML [34], or particular applications of meta-learning such as few-shot learning [37] or NAS [38].

We address both meta-learning methods and applications (Fig. 1, Table 1). We first introduce meta-learning through a high-level problem formalization that can be used to understand and position work in this area. We then provide a new taxonomy in terms of meta-representation, meta-objective and meta-optimizer. This framework provides a design-space for developing new meta learning methods and customizing them for different applications. We survey several popular and emerging application areas including few-shot, reinforcement learning, and architecture search; and position meta-learning with respect to related topics such as transfer and multi-task learning. We conclude by discussing outstanding challenges and areas for future research.

## 2 Background

Meta-learning is difficult to define, having been used in various inconsistent ways, even within contemporary neural-network literature. In this section, we introduce our definition and key terminology, and then position meta-learning with respect to related topics.

Meta-learning is most commonly understood as *learning to learn*; the process of improving a learning algorithm over multiple learning episodes. In contrast, conventional ML improves model predictions over multiple data instances. During *base learning*, an *inner* (or *lower/base*) learning algorithm solves a *task* such as image classification [13], defined by a dataset and objective. During *meta-learning*, an *outer* (or *upper/meta*) algorithm updates the inner learning algorithm such that the model it learns improves an outer objective. For instance this objective could be generalization performance or learning speed of the inner algorithm.

As defined above, many conventional algorithms such as random search of hyper-parameters by cross-validation could fall within the definition of meta-learning. The salient characteristic of contemporary neural-network meta-learning is an explicitly defined *meta-level objective*, and *end-to-end* optimization of the inner algorithm with respect to this objective. Often, meta-learning is conducted on learning episodes sampled from a task family, leading to a base learning algorithm that performs well on new tasks sampled from this family. However, in a limiting case all training episodes can be sampled from a single task. In the following section, we introduce these notions more formally.

### 2.1 Formalizing Meta-Learning

*Conventional Machine Learning*. In conventional supervised machine learning, we are given a training dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, such as (input image, output label) pairs. We can train a predictive model $\hat{y} = f_\theta(x)$ parameterized by $\theta$, by solving:

$$\theta^* = \arg\min_\theta \ \mathcal{L}(\mathcal{D}; \theta, \omega), \tag{1}$$

where $\mathcal{L}$ is a loss function that measures the error between true labels and those predicted by $f_\theta(\cdot)$. Here, $\omega$ specifies assumptions about 'how to learn', such as the choice of optimizer for $\theta$ or function class for $f$. Generalization is then measured by evaluating $\mathcal{L}$ on test points with known labels.

The conventional assumption is that this optimization is performed *from scratch* for every problem $\mathcal{D}$; and that $\omega$ is pre-specified. However, the specification of $\omega$ can drastically affect performance measures like accuracy or data efficiency. Meta-learning seeks to improve these measures by learning the learning algorithm itself, rather than assuming it is pre-specified and fixed. This is often achieved by revisiting the first assumption above, and learning from a distribution of tasks rather than from scratch. Note that while the following formalization of meta-learning takes a supervised perspective for simplicity, all the ideas generalize directly to a reinforcement learning setting as discussed in Section 5.2.

*Meta-Learning: Task-Distribution View*. A common view of meta-learning is to learn a general purpose learning algorithm that generalizes across tasks, and ideally enables each new task to be learned better than the last. We can evaluate the performance of $\omega$ over a task distribution $p(\mathcal{T})$, where a task specifies a dataset and loss function $\mathcal{T} = \{\mathcal{D}, \mathcal{L}\}$. At a high level, learning how to learn thus becomes

$$\min_\omega \mathbb{E}_{\mathcal{T} \sim p(\mathcal{T})} \mathcal{L}(\mathcal{D}\emptyset\, \omega), \tag{2}$$

where $\mathcal{L}$ is a task specific loss. In typical machine-learning settings, we can split $\mathcal{D} = (\mathcal{D}^{train}, \mathcal{D}^{val})$. We then define the task specific loss to be $\mathcal{L}(\mathcal{D}; \omega) = \mathcal{L}(\mathcal{D}^{val}; \theta^*(\mathcal{D}^{train}, \omega), \omega)$; where $\theta^*$ is the parameters of the model trained using the 'how to learn' *meta-knowledge* $\omega$ on dataset $\mathcal{D}^{train}$.

More concretely, to solve this problem in practice, we often assume a *set* of $M$ source tasks is sampled from $p(\mathcal{T})$. Formally, we denote the set of $M$ source tasks used in the meta-training stage as $\mathscr{D}_{source} = \{(\mathcal{D}^{train}_{source}, \mathcal{D}^{val}_{source})^{(i)}\}_{i=1}^{M}$ where each task has both training and validation data. Often, the source train and validation datasets are respectively called *support* and *query* sets. The *meta-training* step of 'learning how to learn' can be written as:

$$\omega^* = \arg\min_\omega \sum_{i=1}^{M} \mathcal{L}(D^{(i)}_{source}; \omega). \tag{3}$$

Now we denote the set of $Q$ target tasks used in the meta-testing stage as $\mathscr{D}_{target} = \{(\mathcal{D}^{train}_{target}, \mathcal{D}^{test}_{target})^{(i)}\}_{i=1}^{Q}$ where each task has both training and test data. In the *meta-testing* stage we use the learned meta-knowledge $\omega^*$ to train the base model on each previously unseen target task $i$:

$$\theta^{*\,(i)} = \arg\min_\theta \ \mathcal{L}(\mathcal{D}^{train\,(i)}_{target}; \theta, \omega^*). \tag{4}$$

In contrast to conventional learning in Eq. (1), learning on the training set of a target task $i$ now benefits from meta-knowledge $\omega^*$ about the algorithm to use. This could be an estimate of the initial parameters [19], or an entire learning model [39] or optimization strategy [14]. We can evaluate the accuracy of our meta-learner by the performance of $\theta^{*\,(i)}$ on the test split of each target task $\mathcal{D}^{test\,(i)}_{target}$.

This setup leads to analogies of conventional underfitting and overfitting: *meta-underfitting* and *meta-overfitting*.

In particular, meta-overfitting is an issue whereby the meta-knowledge learned on the source tasks does not generalize to the target tasks. It is relatively common, especially in the case where only a small number of source tasks are available. It can be seen as learning an inductive bias $\omega$ that constrains the hypothesis space of $\theta$ too tightly around solutions to the source tasks.

*Meta-Learning: Bilevel Optimization View*. The previous discussion outlines the common flow of meta-learning in a multiple task scenario, but does not specify how to solve the meta-training step in Eq. (3). This is commonly done by casting the meta-training step as a bilevel optimization problem. While this picture is most accurate for the optimizer-based methods (see Section 3.1), it is helpful to visualize the mechanics of meta-learning more generally. Bilevel optimization [40] refers to a hierarchical optimization problem, where one optimization contains another optimization as a constraint [20], [41]. Using this notation, meta-training can be formalised as follows:

$$\omega^* = \arg\min_{\omega} \sum_{i=1}^{M} \mathcal{L}^{meta}(\mathcal{D}_{source}^{val\ (i)}; \theta^{*\ (i)}(\omega), \omega) \tag{5}$$

$$\text{s.t.} \qquad \theta^{*\ (i)}(\omega) = \arg\min_{\theta}\ \mathcal{L}^{task}(\mathcal{D}_{source}^{train\ (i)}; \theta, \omega), \tag{6}$$

where $\mathcal{L}^{meta}$ and $\mathcal{L}^{task}$ refer to the outer and inner objectives respectively, such as cross entropy in the case of classification; and we have dropped explicit dependence of $\omega$ on $\mathcal{D}$ for brevity. Note the leader-follower asymmetry between the outer and inner levels: the inner optimization Eq. (6) is conditional on the learning strategy $\omega$ defined by the outer level, but it cannot change $\omega$ during its training.

Here $\omega$ could indicate an initial condition in non-convex optimization [19] or a hyper-parameter such as regularization strength [20]. Section 4.1 discusses the space of choices for $\omega$ in detail. The outer level optimization trains $\omega$ to produce models $\theta^{*\ (i)}(\omega)$ that perform well on their validation sets after training. Section 4.2 discusses how to optimize $\omega$. Note that $\mathcal{L}^{meta}$ and $\mathcal{L}^{task}$ are distinctly denoted, because they can be different functions. For example, $\omega$ can *define* the inner task loss $\mathcal{L}_{\omega}^{task}$ (see Section 4.1, [25], [42]); and as explored in Section 4.3, $\mathcal{L}^{meta}$ can measure different quantities such as validation performance, learning speed or robustness.

Finally, we note that the above formalization of meta-training uses the notion of a distribution over tasks. While common in the meta-learning literature, it is not a necessary condition for meta-learning. More formally, if we are given a single train and test dataset ($M = Q = 1$), we can split the training set to get validation data such that $\mathscr{D}_{source} = (\mathcal{D}_{source}^{train}, \mathcal{D}_{source}^{val})$ for meta-training, and for meta-testing we can use $\mathscr{D}_{target} = (\mathcal{D}_{source}^{train} \cup \mathcal{D}_{source}^{val}, \mathcal{D}_{target}^{test})$. We still learn $\omega$ over several episodes, and different train-val splits are usually used during meta-training.

*Meta-Learning: Feed-Forward Model View*. As we will see, there are a number of meta-learning approaches that synthesize models in a feed-forward manner, rather than via an explicit iterative optimization as in Eqs. 5-6 above. While they vary in their degree of complexity, it can be instructive to understand this family of approaches by instantiating the abstract objective in Eq. (2) to define a toy example for meta-training linear regression [43].

$$\min_{\omega} \mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})(\mathcal{D}^{tr},\mathcal{D}^{val})\in\mathcal{T}} \sum_{(\mathbf{x},y)\in\mathcal{D}^{val}} \Big[(\mathbf{x}^T \mathbf{g}_{\omega}(\mathcal{D}^{tr}) - y)^2\Big]. \tag{7}$$

Here we meta-train by optimizing over a distribution of tasks. For each task a train and validation set is drawn. The train set $\mathcal{D}^{tr}$ is embedded [44] into a vector $\mathbf{g}_{\omega}$ which defines the linear regression weights to predict examples $\mathbf{x}$ from the validation set. Optimizing Eq. (7) 'learns to learn' by training the function $\mathbf{g}_{\omega}$ to map a training set to a weight vector. Thus $\mathbf{g}_{\omega}$ should provide a good solution for novel meta-test tasks $\mathcal{T}^{te}$ drawn from $p(\mathcal{T})$. Methods in this family vary in the complexity of the predictive model $\mathbf{g}$ used, and how the support set is embedded [44] (e.g., by pooling, CNN or RNN). These models are also known as *amortized* [45] because the cost of learning a new task is reduced to a feed-forward operation through $\mathbf{g}_{\omega}(\cdot)$, with iterative optimization already paid for during meta-training of $\omega$.

## 2.2 Historical Context of Meta-Learning

Meta-learning and learning-to-learn first appear in the literature in 1987[17]. J. Schmidhuber introduced a family of methods that can learn how to learn, using *self-referential* learning. Self-referential learning involves training neural networks that can receive as inputs their own weights and predict updates for said weights. Schmidhuber proposed to learn the model itself using evolutionary algorithms.

Meta-learning was subsequently extended to multiple areas. Bengio *et al*.[46], [47] proposed to meta-learn biologically plausible learning rules. Schmidhuber *et al*.continued to explore self-referential systems and meta-learning [48], [49]. S. Thrun *et al*. took care to more clearly define the term *learning to learn* in [7] and introduced initial theoretical justifications and practical implementations. Proposals for training meta-learning systems using gradient descent and backpropagation were first made in 1991 [50] followed by more extensions in 2001 [51], [52], with [29] giving an overview of the literature at that time. Meta-learning was used in the context of reinforcement learning in 1995 [53], followed by various extensions [54], [55].

## 2.3 Related Fields

Here we position meta-learning against related areas whose relation to meta-learning is often a source of confusion.

*Transfer Learning (TL)*. TL [35], [56] uses past experience from a source task to improve learning (speed, data efficiency, accuracy) on a target task. TL refers both to this problem area and family of solutions, most commonly parameter transfer plus optional fine tuning [57] (although there are numerous other approaches [35]).

In contrast, meta-learning refers to a paradigm that can be used to improve TL as well as other problems. In TL the prior is extracted by vanilla learning on the source task without the use of a meta-objective. In meta-learning, the corresponding prior would be defined by an outer optimization that evaluates the benefit of the prior when learn a new task, as illustrated by MAML [19]. More generally, meta-learning deals with a much wider range of meta-representations than solely model parameters (Section 4.1).

*Domain Adaptation (DA) and Domain Generalization (DG).* Domain-shift refers to the situation where source and target problems share the same objective, but the input distribution of the target task is shifted with respect to the source task [35], [58], reducing model performance. DA is a variant of transfer learning that attempts to alleviate this issue by adapting the source-trained model using sparse or unlabeled data from the target. DG refers to methods to train a source model to be robust to such domain-shift without further adaptation. Many knowledge transfer methods have been studied [35], [58] to boost target domain performance. However, as for TL, vanilla DA and DG don't use a meta-objective to optimize 'how to learn' across domains. Meanwhile, meta-learning methods can be used to perform both DA [59] and DG [42] (see Section 5.8).

*Continual Learning (CL).* Continual or lifelong learning [60], [61] refers to the ability to learn on a sequence of tasks drawn from a potentially non-stationary distribution, and in particular seek to do so while accelerating learning new tasks and without forgetting old tasks. Similarly to meta-learning, a task distribution is considered, and the goal is partly to accelerate learning of a target task. However most continual learning methodologies are not meta-learning methodologies since this meta objective is not solved for explicitly. Nevertheless, meta-learning provides a potential framework to advance continual learning, and a few recent studies have begun to do so by developing meta-objectives that encode continual learning performance [62], [63], [64].

*Multi-Task Learning (MTL)* Aims to jointly learn several related tasks, to benefit from regularization due to parameter sharing and the diversity of the resulting shared representation [65], [66], as well as compute/memory savings. Like TL, DA, and CL, conventional MTL is a single-level optimization without a meta-objective. Furthermore, the goal of MTL is to solve a fixed number of known tasks, whereas the point of meta-learning is often to solve unseen future tasks. Nonetheless, meta-learning can be brought in to benefit MTL, e.g., by learning the relatedness between tasks [67], or how to prioritise among multiple tasks [68].

*Hyperparameter Optimization (HO)* is within the remit of meta-learning, in that hyperparameters like learning rate or regularization strength describe 'how to learn'. Here we include HO tasks that define a meta objective that is trained end-to-end with neural networks, such as gradient-based hyperparameter learning [67], [69] and neural architecture search [21]. But we exclude other approaches like random search [70] and Bayesian Hyperparameter Optimization [71], which are rarely considered to be meta-learning.

*Hierarchical Bayesian Models (HBM)* involve Bayesian learning of parameters $\theta$ under a prior $p(\theta|\omega)$. The prior is written as a conditional density on some other variable $\omega$ which has its own prior $p(\omega)$. Hierarchical Bayesian models feature strongly as models for grouped data $\mathcal{D} = \{\mathcal{D}_i | i = 1, 2, \ldots, M\}$, where each group $i$ has its own $\theta_i$. The full model is $\left[\prod_{i=1}^{M} p(\mathcal{D}_i|\theta_i)p(\theta_i|\omega)\right]p(\omega)$. The levels of hierarchy can be increased further; in particular $\omega$ can itself be parameterized, and hence $p(\omega)$ can be learnt. Learning is usually full-pipeline, but using some form of Bayesian marginalisation to compute the posterior over $\omega$: $P(\omega|\mathcal{D}) \sim p(\omega) \prod_{i=1}^{M} \int d\theta_i p(\mathcal{D}_i|\theta_i)p(\theta_i|\omega)$. The ease of doing the marginalisation depends on the model: in some (e.g., Latent Dirichlet Allocation [72]) the marginalisation is exact due to the choice of conjugate exponential models, in others (see e.g., [73]), a stochastic variational approach is used to calculate an approximate posterior, from which a lower bound to the marginal likelihood is computed.

Bayesian hierarchical models provide a valuable viewpoint for meta-learning, by providing a modeling rather than an algorithmic framework for understanding the meta-learning process. In practice, prior work in HBMs has typically focused on learning simple tractable models $\theta$ while most meta-learning work considers complex inner-loop learning processes, involving many iterations. Nonetheless, some meta-learning methods like MAML [19] can be understood through the lens of HBMs [74].

*AutoML.* AutoML [33], [34] is a rather broad umbrella for approaches aiming to automate parts of the machine learning process that are typically manual, such as data preparation, algorithm selection, hyper-parameter tuning, and architecture search. AutoML often makes use of numerous heuristics outside the scope of meta-learning as defined here, and focuses on tasks such as data cleaning that are less central to meta-learning. However, AutoML sometimes makes use of end-to-end optimization of a meta-objective, so meta-learning can be seen as a specialization of AutoML.

## 3 Taxonomy

### 3.1 Previous Taxonomies

Previous [75], [76] categorizations of meta-learning methods have tended to produce a three-way taxonomy across optimization-based methods, model-based (or black box) methods, and metric-based (or non-parametric) methods.

*Optimization.* Optimization-based methods include those where the inner-level task (Eq. (6)) is literally solved as an optimization problem, and focuses on extracting meta-knowledge $\omega$ required to improve optimization performance. A famous example is MAML [19], which aims to learn the initialization $\omega = \theta_0$, such that a small number of inner steps produces a classifier that performs well on validation data. This is also performed by gradient descent, differentiating through the updates of the base model. More elaborate alternatives also learn step sizes [77], [78] or train recurrent networks to predict steps from gradients [14], [15], [79]. Meta-optimization by gradient over long inner optimizations leads to several compute and memory challenges which are discussed in Section 6. A unified view of gradient-based meta learning expressing many existing methods as special cases of a generalized inner loop meta-learning framework has been proposed [80].

*Black Box/Model-Based.* In model-based (or black-box) methods the inner learning step (Eq. (6), Eq. (4)) is wrapped up in the feed-forward pass of a single model, as illustrated in Eq. (7). The model embeds the current dataset $\mathcal{D}$ into activation state, with predictions for test data being made based on this state. Typical architectures include recurrent networks [14], [51], convolutional networks [39] or hypernetworks [81], [82] that embed training instances and labels of a given task to define a predictor for test samples. In this case all the inner-level learning is contained in the activation states of the model and is entirely feed-forward. Outer-level learning is performed with $\omega$ containing the CNN, RNN or hypernetwork parameters. The outer and inner-level optimizations are tightly coupled as $\omega$ and $\mathcal{D}$ directly specify $\theta$.

Memory-augmented neural networks [83] use an explicit storage buffer and can be seen as a model-based algorithm [84], [85]. Compared to optimization-based approaches, these enjoy simpler optimization without requiring second-order gradients. However, it has been observed that model-based approaches are usually less able to generalize to out-of-distribution tasks than optimization-based methods [86]. Furthermore, while they are often very good at data efficient few-shot learning, they have been criticised for being asymptotically weaker [86] as they struggle to embed a large training set into a rich base model.

*Metric-Learning*. Metric-learning or non-parametric algorithms are thus far largely restricted to the popular but specific few-shot application of meta-learning (Section 5.1.1). The idea is to perform non-parametric 'learning' at the inner (task) level by simply comparing validation points with training points and predicting the label of matching training points. In chronological order, this has been achieved with siamese [87], matching [88], prototypical [23], relation [89], and graph [90] neural networks. Here outer-level learning corresponds to metric learning (finding a feature extractor $\omega$ that represents the data suitably for comparison). As before $\omega$ is learned on source tasks, and used for target tasks.

*Discussion*. The common breakdown reviewed above does not expose all facets of interest and is insufficient to understand the connections between the wide variety of meta-learning frameworks available today. For this reason, we propose a new taxonomy in the following section.

### 3.2 Proposed Taxonomy

We introduce a new breakdown along three independent axes. For each axis we provide a taxonomy that reflects the current meta-learning landscape.

*Meta-Representation ("What?")*. The first axis is the choice of meta-knowledge $\omega$ to meta-learn. This could be anything from initial model parameters [19] to readable code in the case of program induction [91].

*Meta-Optimizer ("How?")*. The second axis is the choice of optimizer to use for the outer level during meta-training (see Eq. (5)). The outer-level optimizer for $\omega$ can take a variety of forms from gradient-descent [19], to reinforcement learning [91] and evolutionary search [25].

*Meta-Objective ("Why?")*. The third axis is the *goal* of meta-learning which is determined by choice of meta-objective $\mathcal{L}^{meta}$ (Eq. (5)), task distribution $p(\mathcal{T})$, and data-flow between the two levels. Together these can customize meta-learning for different purposes such as sample efficient few-shot learning [19], [39], fast many-shot optimization [91], [92], robustness to domain-shift [42], [93], label noise [94], and adversarial attack [95].

Together these axes provide a design-space for meta-learning methods that can orient the development of new algorithms and customization for particular applications. Note that the base model representation $\theta$ isn't included in this taxonomy, since it is determined and optimized in a way that is specific to the application at hand.

## 4 Survey: Methodologies

In this section we break down existing literature according to our proposed new methodological taxonomy.

### 4.1 Meta-Representation

Meta-learning methods make different choices about what meta-knowledge $\omega$ should be, i.e., which aspects of the learning strategy should be learned; and (by exclusion) which aspects should be considered fixed.

*Parameter Initialization*. Here $\omega$ corresponds to the initial parameters of a neural network to be used in the inner optimization, with MAML being the most popular example [19], [96], [97]. A good initialization is just a few gradient steps away from a solution to any task $\mathcal{T}$ drawn from $p(\mathcal{T})$, and can help to learn without overfitting in few-shot learning. A key challenge with this approach is that the outer optimization needs to solve for as many parameters as the inner optimization (potentially hundreds of millions in large CNNs). This leads to a line of work on isolating a subset of parameters to meta-learn, for example by subspace [76], [98], by layer [81], [98], [99], or by separating out scale and shift [100]. Another concern is whether a single initial condition is sufficient to provide fast learning for a wide range of potential tasks, or if one is limited to narrow distributions $p(\mathcal{T})$. This has led to variants that model mixtures over multiple initial conditions [98], [101], [102].

*Optimizer*. The above parameter-centric methods usually rely on existing optimizers such as SGD with momentum or Adam [103] to refine the initialization when given some new task. Instead, optimizer-centric approaches [14], [15], [79], [92] focus on learning the inner optimizer by training a function that takes as input optimization states such as $\theta$ and $\nabla_\theta \mathcal{L}^{task}$ and produces the optimization step for each base learning iteration. The trainable component $\omega$ can span simple hyper-parameters such as a fixed step size [77], [78] to more sophisticated pre-conditioning matrices [104], [105]. Ultimately $\omega$ can be used to define a full gradient-based optimizer through a complex non-linear transformation of the input gradient and other metadata [14], [15], [91], [92]. The parameters to learn here can be few if the optimizer is applied coordinate-wise across weights [15]. The initialization-centric and optimizer-centric methods can be merged by learning them jointly, namely having the former learn the initial condition for the latter [14], [77]. Optimizer learning methods have both been applied to for few-shot learning [14] and to accelerate and improve many-shot learning [15], [91], [92]. Finally, one can also meta-learn zeroth-order optimizers [106] that only require evaluations of $\mathcal{L}^{task}$ rather than optimizer states such as gradients. These have been shown [106] to be competitive with conventional Bayesian Optimization [71] alternatives.

*Feed-Forward Models (FFMs. aka, Black-Box, Amortized)*. Another family of models trains learners $\omega$ that provide a feed-forward mapping directly from the support set to the parameters required to classify test instances, i.e., $\theta = g_\omega(\mathcal{D}^{train})$ – rather than relying on a gradient-based iterative optimization of $\theta$. These correspond to black-box model-based learning in the conventional taxonomy (Section 3.1) and span from classic [107] to recent approaches such as CNAPs [108] that provide strong performance on challenging cross-domain few-shot benchmarks [109].

These methods have connections to Hypernetworks [110] which generate the weights of another neural network conditioned on some embedding – and are often used for compression or multi-task learning. Here $\omega$ is the hypernetwork

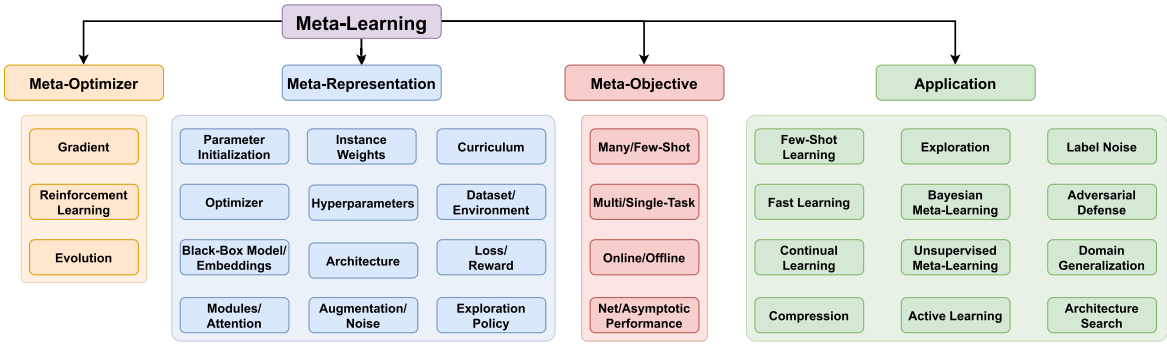


Fig. 1. Overview of the meta-learning landscape including algorithm design (meta-optimizer, meta-representation, meta-objective), and applications.

and it synthesises $\theta$ given the source dataset in a feed-forward pass [98], [111]. Embedding the support set is often achieved by recurrent networks [51], [112], [113] convolution [39], or set embeddings [45], [108]. Research here often studies architectures for paramaterizing the classifier by the task-embedding network: (i) Which parameters should be globally shared across all tasks, versus synthesized per task by the hypernetwork (e.g., share the feature extractor and synthesize the classifier [81], [114]), and (ii) How to parameterize the hypernetwork so as to limit the number of parameters required in $\omega$ (e.g., via synthesizing only lightweight adapter layers in the feature extractor [108], or class-wise classifier weight synthesis [45]).

Some FFMs can also be understood elegantly in terms of amortized inference in probabilistic models [45], [107], making predictions for test data $x$ as:

$$q_\omega(y|x,\mathcal{D}^{tr}) = \int p(y|x,\theta) q_\omega(\theta|\mathcal{D}^{tr}) d\theta, \tag{8}$$

where the meta-representation $\omega$ is a network $q_\omega(\cdot)$ that approximates the intractable Bayesian inference for parameters $\theta$ that solve the task with training data $\mathcal{D}^{tr}$, and the integral may be computed exactly [107], or approximated by sampling [45] or point estimate [108]. The model $\omega$ is then trained to minimise validation loss over a distribution of training tasks *cf.* Eq. (7).

Finally, memory-augmented neural networks, with the ability to remember old data and assimilate new data quickly, typically fall in the FFM category as well [84], [85].

*Embedding Functions (Metric Learning).* Here the meta-optimization process learns an embedding network $\omega$ that transforms raw inputs into a representation suitable for recognition by simple similarity comparison between query and support instances [23], [81], [88], [114] (e.g., with cosine similarity or euclidean distance). These methods are classified as metric learning in the conventional taxonomy (Section 3.1) but can also be seen as a special case of the feed-forward black-box models above. This can easily be seen for methods that produce logits based on the inner product of the embeddings of support and query images $x_s$ and $x_q$, namely $g_\omega^T(x_q)g_\omega(x_s)$ [81], [114]. Here the support image generates 'weights' to interpret the query example, making it a special case of a FFM where the 'hypernetwork' generates a linear classifier for the query set. Vanilla methods in this family have been further enhanced by making the embedding task-conditional [99], [115], learning a more elaborate comparison metric [89], [90], or combining with gradient-based meta-learning to train other hyper-parameters such as stochastic regularizers [116].

*Losses and Auxiliary Tasks.* These approaches learn the inner task-loss $\mathcal{L}_\omega^{task}$ for the base model (in contrast to $\mathcal{L}^{meta}$, which is fixed). Loss-learning approaches typically define a function that inputs relevant quantities (e.g., predictions, labels, or model parameters) and outputs a scalar to be treated as a loss by the inner (task) optimizer. This can lead to a learned loss that is *easier* to optimize (less local minima) than standard alternatives [22], [25], [117], provides faster learning with improved generalization [43], [118], [119], [120], robustness to label noise [121], or whose minima corresponds to a model more robust to domain shift [42]. Loss learning methods have also been used to learn from unlabeled instances [99], [122], or to learn $\mathcal{L}_\omega^{task}$ as a differentiable approximation to a true non-differentiable $\mathcal{L}^{meta}$ of interest such as area under precision recall curve [123].

Loss learning also arises in generalizations of self-supervised [124] or auxiliary task [125] learning. In these problems unsupervised predictive tasks (such as colourising pixels in vision [124], or simply changing pixels in RL [125]) are defined and optimized with the aim of improving the representation for the main task. In this case the best auxiliary task (loss) to use can be hard to predict in advance, so meta-learning can be used to select among several auxiliary losses according to their impact on improving main task learning. I.e., $\omega$ is a per-auxiliary task weight [68]. More generally, one can meta-learn an auxiliary task generator that annotates examples with auxiliary labels [126].

*Architectures.* Architecture discovery has always been an important area in neural networks [38], [127], and one that is not amenable to simple exhaustive search. Meta-Learning can be used to automate this very expensive process by learning architectures. Early attempts used evolutionary algorithms to learn the topology of LSTM cells [128], while later approaches leveraged RL to generate descriptions for good CNN architectures [28]. Evolutionary Algorithms [27] can learn blocks within architectures modelled as graphs which could mutate by editing their graph. Gradient-based architecture representations have also been visited in the form of DARTS [21] where the forward pass during training consists in a softmax across the outputs of all possible layers in a given block, which are weighted by coefficients to be meta learned (i.e., $\omega$). During meta-test, the architecture is discretized by only keeping the layers corresponding to the highest coefficients. Recent efforts to improve DARTS have focused on more efficient differentiable approximations [129], robustifying the discretization step

[130], learning easy to adapt initializations [131], or architecture priors [132]. See Section 5.3 for more details.

*Attention Modules*. have been used as comparators in metric-based meta-learners [133], to prevent catastrophic forgetting in few-shot continual learning [134] and to summarize the distribution of text classification tasks [135].

*Modules*. Modular meta-learning [136], [137] assumes that the task agnostic knowledge $\omega$ defines a set of modules, which are re-composed in a task specific manner defined by $\theta$ in order to solve each encountered task. These strategies can be seen as meta-learning generalizations of the typical structural approaches to knowledge sharing that are well studied in multi-task and transfer learning [66], [138], and may ultimately underpin compositional learning [139].

*Hyper-Parameters*. Here $\omega$ represents hyperparameters of the base learner such as regularization strength [20], [69], per-parameter regularization [93], task-relatedness in multi-task learning [67], or sparsity strength in data cleansing [67]. Hyperparameters such as step size [69], [77], [78] can be seen as part of the optimizer, leading to an overlap between hyperparameter and optimizer learning categories.

*Data Augmentation and Noise*. In supervised learning it is common to improve generalization by synthesizing more training data through label-preserving transformations on the existing data [13]. The data augmentation operation is wrapped up in the inner problem (Eq. (6)), and is conventionally hand-designed. However, when $\omega$ defines the data augmentation strategy, it can be learned by the outer optimization in Eq. (5) in order to maximize validation performance [140]. Since augmentation operations are often non-differentiable, this requires reinforcement learning [140], discrete gradient-estimators [141], or evolutionary [142] methods. Recent attempts use meta-gradient to learn mixing proportions in mixup-based augmentation [143]. For *stochastic* neural networks that exploit noise internally [116], $\omega$ can define a learnable noise distribution.

*Minibatch Selection, Instance Weights, and Curriculum Learning*. When the base algorithm is minibatch-based stochastic gradient descent, a design parameter of the learning strategy is the batch selection process. Various hand-designed methods [144] exist to improve on randomly-sampled minibatches. Meta-learning approaches can define $\omega$ as an instance selection probability [145] or neural network that picks instances [146] for inclusion in a minibatch. Related to mini-batch selection policies are methods that learn or infer *per-instance* loss weights for each sample in the training set [147], [148]. For example defining a base loss as $\mathcal{L} = \sum_i \omega_i \ell(f(x_i), y_i)$. This can be used to learn under label-noise by discounting noisy samples [147], [148], discount outliers [67], or correct for class imbalance [147].

More generally, the *curriculum* [149] refers to sequences of data or concepts to learn that produce better performance than learning items in a random order. For instance by focusing on instances of the right difficulty while rejecting too hard or too easy (already learned) instances. Instead of defining a curriculum by hand [150], meta-learning can automate the process and select examples of the right difficulty by defining a teaching policy as the meta-knowledge and training it to optimize the student's progress [146], [151].

*Datasets, Labels and Environments*. Another meta-representation is the support dataset itself. This departs from our initial formalization of meta-learning which considers the source datasets to be fixed (Section 2.1, Eqs. (2) and (3)). However, it can be easily understood in the bilevel view of Eqs. (5) and (6). If the validation set in the upper optimization is real and fixed, and a train set in the lower optimization is paramaterized by $\omega$, the training dataset can be tuned by meta-learning to optimize validation performance.

In dataset distillation [152], [153], [154], the support images themselves are learned such that a few steps on them allows for good generalization on real query images. This can be used to summarize large datasets into a handful of images, which is useful for replay in continual learning where streaming datasets cannot be stored.

Rather than learning input images $x$ for fixed labels $y$, one can also learn the input labels $y$ for fixed images $x$. This can be used in distilling core sets [155] as in dataset distillation; or semi-supervised learning, for example to directly learn the unlabeled set's labels to optimize validation set performance [156], [157].

In the case of sim2real learning [158] in computer vision or reinforcement learning, one uses an environment simulator to generate data for training. In this case, as detailed in Section 5.10, one can also train the graphics engine [159] or simulator [160] so as to optimize the real-data (validation) performance of the downstream model after training on data generated by that environment simulator.

*Discussion: Transductive Representations and Methods*. Most of the representations $\omega$ discussed above are parameter vectors of functions that process or generate data. However a few representations mentioned are transductive in the sense that the $\omega$ literally corresponds to data points [152], labels [156], or per-instance weights [67], [148]. Therefore the number of parameters in $\omega$ to meta-learn scales as the size of the dataset. While the success of these methods is a testament to the capabilities of contemporary meta-learning [154], this property may ultimately limit their scalability.

Distinct from a transductive representation are methods that are transductive in the sense that they operate on the query instances as well as support instances [99], [126].

*Discussion: Interpretable Symbolic Representations*. A cross-cutting distinction that can be made across many of the meta-representations discussed above is between uninterpretable (sub-symbolic) and human interpretable (symbolic) representations. Sub-symbolic representations, such as when $\omega$ parameterizes a neural network [15], are more common and make up the majority of studies cited above. However, meta-learning with symbolic representations is also possible, where $\omega$ represents human readable symbolic functions such as optimization program code [91]. Rather than neural loss functions [42], one can train symbolic losses $\omega$ that are defined by an expression analogous to cross-entropy [119], [121]. One can also meta-learn new symbolic activations [161] that outperform standards such as ReLU. As these meta-representations are non-smooth, the meta-objective is non-differentiable and is harder to optimize (see Section 4.2). So the upper optimization for $\omega$ typically uses RL [91] or evolutionary algorithms [119]. However, symbolic representations may have an advantage [91], [119], [161] in their ability to generalize across task families. I. e., to span wider distributions $p(\mathcal{T})$ with a single $\omega$ during meta-training, or to have the learned $\omega$ generalize to an out of distribution task during meta-testing (see Section 6).

*Discussion: Amortization*. One way to relate some of the representations discussed is in terms of the degree of learning *amortization* entailed [45]. That is, how much task-specific optimization is performed during meta-testing versus how much learning is amortized during meta-training. Training from scratch, or conventional fine-tuning [57] perform full task-specific optimization at meta-testing, with no amortization. MAML [19] provides limited amortization by fitting an initial condition, to enable learning a new task by *few-step* fine-tuning. Pure FFMs [23], [88], [108] are fully amortized, with no task-specific optimization, and thus enable the fastest learning of new tasks. Meanwhile some hybrid approaches [98], [99], [109], [162] implement semi-amortized learning by drawing on both feed-forward and optimization-based meta-learning in a single framework.

### 4.2 Meta-Optimizer

Given a choice of which facet of the learning strategy to optimize, the next axis of meta-learner design is actual outer (meta) optimization strategy to use for training $\omega$.

*Gradient*. A large family of methods use gradient descent on the meta parameters $\omega$ [14], [19], [42], [67]. This requires computing derivatives $d\mathcal{L}^{meta}/d\omega$ of the outer objective, which are typically connected via the chain rule to the model parameter $\theta$, $d\mathcal{L}^{meta}/d\omega = (d\mathcal{L}^{meta}/d\theta)(d\theta/d\omega)$. These methods are potentially the most efficient as they exploit analytical gradients of $\omega$. However key challenges include: (i) Efficiently differentiating through many steps of inner optimization, for example through careful design of differentiation algorithms [20], [69], [190] and implicit differentiation [154], [163], [191], and dealing tractably with the required second-order gradients [192]. (ii) Reducing the inevitable gradient degradation problems whose severity increases with the number of inner loop optimization steps. (iii) Calculating gradients when the base learner, $\omega$, or $\mathcal{L}^{task}$ include discrete or other non-differentiable operations.

*Reinforcement Learning*. When the base learner includes non-differentiable steps [140], or the meta-objective $\mathcal{L}^{meta}$ is itself non-differentiable [123], many methods [24] resort to RL to optimize the outer objective Eq. (5). This estimates the gradient $\nabla_{\omega}\mathcal{L}^{meta}$, typically using the policy gradient theorem. However, alleviating the requirement for differentiability in this way is typically extremely costly. High-variance policy-gradient estimates for $\nabla_{\omega}\mathcal{L}^{meta}$ mean that many outer-level optimization steps are required to converge, and each of these steps are themselves costly due to wrapping task-model optimization within them.

*Evolution*. Another approach for optimizing the meta-objective are evolutionary algorithms (EA) [17], [127], [193]. Many evolutionary algorithms have strong connections to reinforcement learning algorithms [194]. However, their performance does not depend on the length and reward sparsity of the inner optimization as for RL.

EAs are attractive for several reasons [193]: (i) They can optimize any base model and meta-objective with no differentiability constraint. (ii) Not relying on backpropagation avoids both gradient degradation issues and the cost of high-order gradient computation of conventional gradient-based methods. (iii) They are highly parallelizable for scalability. (iv) By maintaining a diverse population of solutions, they can avoid local minima that plague gradient-based methods [127]. However, they have a number of disadvantages: (i) The population size required increases rapidly with the number of parameters to learn. (ii) They can be sensitive to the mutation strategy and may require careful hyperparameter optimization. (iii) Their fitting ability is generally inferior to gradient-based methods, especially for large models such as CNNs.

EAs are relatively more commonly applied in RL applications [25], [169] (where models are typically smaller, and inner optimizations are long and non-differentiable). However they have also been applied to learn learning rules [195], optimizers [196], architectures [27], [127] and data augmentation strategies [142] in supervised learning. They are also particularly important in learning human interpretable symbolic meta-representations [119].

*Discussion*. These three optimizers are also all used in conventional learning. However meta-learning comparatively more often resorts to RL and Evolution, e.g., as $\mathcal{L}^{meta}$ is often non-differentiable with respect to representation $\omega$.

### 4.3 Meta-Objective and Episode Design

The final component is to define the meta-learning goal through choice of meta-objective $\mathcal{L}^{meta}$, and associated data flow between inner loop episodes and outer optimizations. Most methods define a meta-objective using a performance metric computed on a validation set, after updating the task model with $\omega$. This is in line with classic validation set approaches to hyperparameter and model selection. However, within this framework, there are several design options:

*Many versus Few-Shot Episode Design*. According to whether the goal is improving few- or many-shot performance, inner loop learning episodes may be defined with many [67], [91], [92] or few- [14], [19] examples per-task.

*Fast Adaptation versus Asymptotic Performance*. When validation loss is computed at the end of the inner learning episode, meta-training encourages better *final* performance of the base task. When it is computed as the sum of the validation loss after each inner optimization step, then meta-training also encourages *faster* learning in the base task [78], [91], [92]. Most RL applications also use this latter setting.

*Multi versus Single-Task* When the goal is to tune the learner to better solve any task drawn from a given family, then inner loop learning episodes correspond to a randomly drawn task from $p(\mathcal{T})$ [19], [23], [42]. When the goal is to tune the learner to simply solve one specific task better, then the inner loop learning episodes all draw data from the same underlying task [15], [67], [173], [180], [181], [197].

It is worth noting that these two meta-objectives tend to have different assumptions and value propositions. The multi-task objective obviously requires a task family $p(\mathcal{T})$ to work with, which single-task does not. Meanwhile for multi-task, the data and compute cost of meta-training can be amortized by potentially boosting the performance of multiple target tasks during meta-test; but single-task – without the new tasks for amortization – needs to improve the final solution or asymptotic performance of the current task, or meta-learn fast enough to be online.

*Online versus Offline*. While the classic meta-learning pipeline defines the meta-optimization as an outer-loop of the inner base learner [15], [19], some studies have attempted to

TABLE 1
Research Papers According to Our Taxonomy

| Meta-Representation | Meta-Optimizer | | |
|---|---|---|---|
| | Gradient | RL | Evolution |
| Initial Condition | MAML [19], [162]. MetaOptNet [163]. [76], [85], [99], [164] | MetaQ [165]. [166], [167] [19], [61], [62] | ES-MAML [168]. [169] |
| Optimizer | $GD^2$ [15]. MetaLSTM [14]. [91] [16], [76], [103], [104], [170] | PSD [78]. [90] | |
| Hyperparam | HyperRep [20], HyperOpt [66], LHML [68] | MetaTrace [171]. [172] | [169] [173] |
| Feed-Forward model | SNAIL [38], CNAP [107]. [44], [83], [174], [175] [176]–[178] | PEARL [110]. [23], [112] | |
| Metric | MatchingNets [87], ProtoNets [22], RelationNets [88]. [89] | | |
| Loss/Reward | MetaReg [92]. [41] [120] | MetaCritic [117]. [122] [179] [120] | EPG [24]. [119] [173] |
| Architecture | DARTS [21]. [130] | [27] | [26] |
| Exploration Policy | | MetaCuriosity [25]. [180]–[184] | |
| Dataset/Environment | Data Distillation [151]. [152] [155] | Learn to Sim [158] | [159] |
| Instance Weights | MetaWeightNet [146]. MentorNet [150]. [147] | | |
| Data Augmentation/Noise | MetaDropout [185]. [140] [115]. | AutoAugment [139]. | [141] |
| Modules | [135], [136] | | |
| Annotation Policy | [186], [187] | [188] | |

*We use color to indicate salient meta-objective or application goal. We focus on the main goal of each paper for simplicity. The color code is: sample efficiency (red), learning speed (green), asymptotic performance (purple), cross-domain (blue).*

preform meta-optimization *online* within a single base learning episode [42], [180], [197], [198]. In this case the base model $\theta$ and learner $\omega$ co-evolve during a single episode. Since there is now no set of source tasks to amortize over, meta-learning needs to be fast compared to base model learning in order to benefit sample or compute efficiency.

*Other Episode Design Factors*. Other operators can be inserted into the episode generation pipeline to customize meta-learning for particular applications. For example one can simulate domain-shift between training and validation to meta-optimize for good performance under domain-shift [42], [59], [93]; simulate network compression such as quantization [199] between training and validation to meta-optimize for network compressibility; provide noisy labels during meta-training to optimize for label-noise robustness [94], or generate an adversarial validation set to meta-optimize for adversarial defense [95]. These opportunities are explored in more detail in the following section.

## 5 Applications

In this section we briefly review the ways in which meta-learning has been exploited in computer vision, reinforcement learning, architecture search, and so on.

### 5.1 Computer Vision and Graphics

Computer vision is a major consumer domain of meta-learning techniques, notably due to its impact on few-shot learning, which holds promise to deal with the challenge posed by the long-tail of concepts to recognise in vision.

#### 5.1.1 Few-Shot Learning Methods

Few-shot learning (FSL) is extremely challenging, especially for large neural networks [1], [13], where data volume is often the dominant factor in performance [200], and training large models with small datasets leads to overfitting or non-convergence. Meta-learning-based approaches are increasingly able to train powerful CNNs on small datasets in many vision problems. We provide a non-exhaustive representative summary as follows.

*Classification*. The most common application of meta-learning is few-shot multi-class image recognition, where the inner and outer loss functions are typically the cross entropy over training and validation data respectively [14], [23], [75], [77], [78], [88], [90], [98], [99], [102], [105], [201], [202], [203], [204]. Optimizer-centric [19], black-box [39], [81] and metric learning [88], [89], [90] models have all been considered.

This line of work has led to a steady improvement in performance compared to early methods [19], [87], [88]. However, performance is still far behind that of fully supervised methods, so there is more work to be done. Current research issues include improving cross-domain generalization [116], recognition within the joint label space defined by meta-train and meta-test classes [82], and incremental addition of new few-shot classes [134], [175].

*Object Detection*. Building on progress in few-shot classification, few-shot object *detection* [175], [205] has been demonstrated, often using feed-forward hypernetwork-based approaches to embed support set images and synthesize final layer classification weights in the base model.

*Landmark Prediction*. aims to locate a skeleton of key points within an image, such as such as joints of a human or robot. This is typically formulated as an image-conditional regression. For example, a MAML-based model was shown to work for human pose estimation [206], modular-meta-learning was successfully applied to robotics [136], while a hypernetwork-based model was applied to few-shot clothes fitting for novel fashion items [175].

*Few-Shot Object Segmentation*. is important due to the cost of obtaining pixel-wise labeled images. Hypernetwork-based meta-learners have been applied in the one-shot regime [207], and performance was later improved by adapting prototypical networks [208]. Other models tackle cases where segmentation has low density [209].

*Actions*. Beyond still images, meta-learning has been applied to few-shot action recognition [210], [211] and prediction [212] using both FFM [210] and optimization-based [211], [212] approaches.

*Image and Video Generation*. In [45] an amortized probabilistic meta-learner is used to generate multiple views of an

object from just a single image, generative query networks [213] render scenes from novel views, and talking faces are generated from little data by learning the initialization of an adversarial model for quick adaptation [214]. In video domain, [215] meta-learns a weight generator that synthesizes videos given few example images as cues. Meta-learning has also accelerated style transfer by learning a FFM to solve the corresponding optimization problem [216].

*Generative Models and Density Estimation*. Density estimators capable of generating images typically require many parameters, and as such overfit in the few-shot regime. Gradient-based meta-learning of PixelCNN generators was shown to enable their few-shot learning [217].

#### 5.1.2 Few-Shot Learning Benchmarks

Progress in AI and machine learning is often measured, and spurred, by well designed benchmarks [218]. Conventional ML benchmarks define a task and dataset for which a model should generalize from seen to unseen *instances*. In meta-learning, benchmark design is more complex, since we are often dealing with a learner that should generalize from seen to unseen *tasks*. Benchmark design thus needs to define families of tasks from which meta-training and meta-testing tasks can be drawn. Established FSL benchmarks include miniImageNet [14], [88], Tiered-ImageNet [219], SlimageNet [220], Omniglot [88] and Meta-Dataset [109].

*Dataset Diversity, Bias and Generalization*. The standard benchmarks provide tasks for training and evaluation, but suffer from a lack of diversity (narrow $p(\mathcal{T})$); performance on these benchmarks are not reflective of performance on real-world few shot tasks. For example, switching between kinds of animal photos in miniImageNet is not a strong test of generalization. Ideally bechmarks would span more diverse categories and types of images (satellite, medical, agricultural, underwater, etc), and even test domain-shifts between meta-train and meta-test tasks.

There is work still to be done here as, even in the many-shot setting, fitting a deep model to a very wide distribution of data is itself non-trivial [221], as is generalizing to out-of-sample data [42], [93]. Similarly, the performance of meta-learners often drops drastically when introducing a domain shift between the source and target task distributions [114]. This motivates the recent Meta-Dataset [109] and CVPR cross-domain few-shot challenge [222]. Meta-Dataset aggregates a number of individual recognition benchmarks to provide a wider distribution of tasks $p(\mathcal{T})$ to evaluate the ability to fit a wide task distribution and generalize across domain-shift. Meanwhile, [222] challenges methods to generalize from the everyday ImageNet images to medical, satellite and agricultural images. Recent work has begun to try and address these issues by meta-training for domain-shift robustness as well as sample efficiency [116]. Generalization issues also arise in applying models to data from under-represented countries [223].

### 5.2 Meta Reinforcement Learning and Robotics

Reinforcement learning is typically concerned with learning control policies that enable an agent to obtain high reward after performing a sequential action task within an environment. RL typically suffers from extreme sample inefficiency due to sparse rewards, the need for exploration, and the high-variance [224] of optimization algorithms. However, applications often naturally entail task families which meta-learning can exploit – for example locomoting-to or reaching-to different positions [185], navigating within different environments [39], traversing different terrains [64], driving different cars [184], competing with different competitor agents [62], and dealing with different handicaps such as failures in individual robot limbs [64]. Thus RL provides a fertile application area in which meta-learning on task distributions has had significant successes in improving sample efficiency over standard RL algorithms. One can intuitively understand the efficacy of these methods. For instance meta-knowledge of a maze layout is transferable for all tasks that require navigating within the maze.

*Problem Setup*. Meta-RL uses essentially the same problem setup outlined in Section 2.1, but in the setting of sequential decision making within Markov decision processes (MDPs). A conventional RL problem is defined by an MDP $\mathcal{M}$, in which policy $\pi_\theta$ chooses actions $a_t$ based on states $s_t$ and obtains rewards $r_t$. The policy is trained to maximize the expected cumulative return $R_\tau = \sum_t^T \gamma^t r_t$ of an episode $\tau \sim \mathcal{M}, \tau = (s_0, a_0, \ldots, s_T, a_T, r_T)$ after $T$ time-steps, $\theta^* = \arg\max_\theta E_{\tau\sim\mathcal{M},\pi_\theta}[R_\tau]$. Meta-RL, usually assumes a distribution over MDPs $p(\mathcal{M})$, analogous to the distribution over tasks in Section 2.1. Some aspect of the learning process for policy $\theta$ is then paramaterized by meta-knowledge $\omega$, for example the initial condition [19], or surrogate reward/loss function [22], [25]. Then the goal of Meta-RL is to train the meta-representation $\omega$ so as to maximize the expected return of learning within the MDP distribution $p(\mathcal{M})$:

$$\omega^* = \arg\max_\omega \ \mathbb{E}_{\mathcal{M}\sim p(\mathcal{M})}\mathbb{E}_{\tau\sim\mathcal{M},\pi_{\theta^*}}[R_\tau]. \tag{9}$$

This is the RL version of the meta-training problem in Eq. (2), and solutions are similarly based on bi-level optimization (e.g., [19], cf: Eqs 5-6), or feed-forward approaches (e.g., [39], cf: Eq. (7)). Finally, once trained, the meta-learner $\omega^*$ can also be deployed to rapidly learn in a new MDP $\mathcal{M} \sim p(\mathcal{M})$ analogously to the meta-test step in Eq. (4).

#### 5.2.1 Methods

Several previously discussed meta-representations have been explored in RL including learning the initial conditions [19], [170], hyperparameters [170], [174], step directions [77] and step sizes [172]. These enable gradient-based learning of a neural policy with fewer environmental interactions and training fast convolutional [39] or recurrent [24], [113] black-box models to synthesize a policy by embedding the enviroment experience. Recent work has developed improved meta-optimization algorithms [167], [168], [169] for these tasks, and provided theoretical guarantees for meta-RL [225].

*Exploration*. A meta-representation rather unique to RL is the exploration policy. RL is complicated by the fact that the data distribution is not fixed, but varies according to the agent's actions. Furthermore, sparse rewards may mean that an agent must take many actions before achieving a reward that can be used to guide learning. As such, how to explore and acquire data for learning is a crucial factor in any RL algorithm. Traditionally exploration is based on

sampling random actions [226], or hand-crafted heuristics [227]. Several meta-RL studies have instead explicitly treated exploration strategy or curiosity function as meta-knowledge $\omega$; and modeled their acquisition as a meta-learning problem [26], [183], [184], [228] – leading to sample efficiency improvements by 'learning how to explore'.

*Optimization*. RL is a difficult optimization problem where the learned policy is usually far from optimal, even on 'training set' episodes. This means that, in contrast to meta-SL, meta-RL methods are more commonly deployed to increase asymptotic performance [25], [174], [180] as well as sample-efficiency, and can lead to significantly better solutions overall. The meta-objective of many meta-RL frameworks is the net return of the agent over a full episode, and thus both sample efficient and asymptotically performant learning are rewarded. Optimization difficulty also means that there has been relatively more work on learning losses (or rewards) [22], [120], [180], [229] which an RL agent should optimize instead of – or in addition to – the conventional sparse reward objective. Such learned losses may be easier to optimize (denser, smoother) compared to the true target [25], [229]. This also links to exploration as reward learning and can be considered to instantiate meta-learning of learning intrinsic motivation [181].

*Online Meta-RL* A significant fraction of meta-RL studies addressed the single-task setting, where the meta-knowledge such as loss [22], [180], reward [174], [181], hyperparameters [172], [173], or exploration strategy [182] are trained online together with the base policy while learning a single task. These methods thus do not require task families and provide a direct improvement to their respective base learners' performance.

*On- versus Off-Policy Meta-RL*. A major dichotomy in conventional RL is between on-policy and off-policy learning such as PPO [226] versus SAC [230]. Off-policy methods are usually significantly more sample efficient. However, off-policy methods have been harder to extend to meta-RL, leading to more meta-RL methods being built on on-policy RL methods, thus limiting the absolute performance of meta-RL. Early work in off-policy meta-RL methods has led to strong results [22], [111], [166], [229]. Off-policy learning also improves the efficiency of the meta-train stage [111], which can be expensive in meta-RL. It also provides new opportunities to accelerate meta-testing by replay buffer sample from meta-training [166].

*Other Trends and Challenges*. [64] is noteworthy in demonstrating successful meta-RL on a real-world physical robot. Knowledge transfer in robotics is often best studied *compositionally* [231]. E.g., walking, navigating and object pick/place may be subroutines for a room cleaning robot. However, developing meta-learners with effective compositional knowledge transfer is an open question, with modular meta-learning [137] being an option. Unsupervised meta-RL variants aim to perform meta-training without manually specified rewards [232], or adapt at meta-testing to a changed environment but without new rewards [233]. Continual adaptation provides an agent with the ability to adapt to a sequence of tasks within one meta-test episode [62], [63], [64], similar to continual learning. Finally, meta-learning has also been applied to imitation [112] and inverse RL [234].

#### 5.2.2 Benchmarks

Meta-learning benchmarks for RL typically define a family to solve in order to train and evaluate an agent that learns how to learn. These can be tasks (reward functions) to achieve, or domains (distinct environments or MDPs).

*Discrete Control RL*. An early meta-RL benchmark for vision-actuated control is the arcade learning environment (ALE) [235], which defines a set of classic Atari games split into meta-training and meta-testing. The protocol here is to evaluate return after a fixed number of timesteps in the meta-test environment. A challenge is the great diversity (wide $p(\mathcal{T})$) across games, which makes successful meta-training hard and leads to limited benefit from knowledge transfer [235]. Another benchmark [236] is based on splitting Sonic-hedgehog levels into meta-train/meta-test. The task distribution here is narrower and beneficial meta-learning is relatively easier to achieve. Cobbe *et al.*[237] proposed two purpose designed video games for benchmarking meta-RL. CoinRun game [237] provides $2^{32}$ procedurally generated levels of varying difficulty and visual appearance. They show that some 10,000 levels of meta-train experience are required to generalize reliably to new levels. CoinRun is primarily designed to test direct generalization rather than fast adaptation, and can be seen as providing a distribution over MDP environments to test generalization rather than over tasks to test adaptation. To better test fast learning in a wider task distribution, ProcGen [237] provides a set of 16 procedurally generated games including CoinRun.

*Continuous Control RL*. While common benchmarks such as gym [238] have greatly benefited RL research, there is less consensus on meta-RL benchmarks, making existing work hard to compare. Most continuous control meta-RL studies have proposed home-brewed benchmarks that are low dimensional parametric variants of particular tasks such as navigating to various locations or velocities [19], [111], or traversing different terrains [64]. Several multi-MDP benchmarks [239], [240] have recently been proposed but these primarily test generalization across different environmental perturbations rather than different tasks. The Meta-World benchmark [241] provides a suite of 50 continuous control tasks with state-based actuation, varying from simple parametric variants such as lever-pulling and door-opening. This benchmark should enable more comparable evaluation, and investigation of generalization within and across task distributions. The meta-world evaluation [241] suggests that existing meta-RL methods struggle to generalize over wide task distributions and meta-train/meta-test shifts. This may be due to our meta-RL models being too weak and/or benchmarks being too small, in terms of number and coverage tasks, for effective learning-to-learn. Another recent benchmark suitable for meta-RL is PHYRE [242] which provides a set of 50 vision-based physics task templates which can be solved with simple actions but are likely to require model-based reasoning to address efficiently. These also provide within and cross-template generalization tests.

*Discussion*. One complication of vision-actuated meta-RL is disentangling visual generalization (as in computer vision) with fast learning of control strategies more generally. For example CoinRun [237] evaluation showed large benefit from standard vision techniques such as batch norm suggesting that perception is a major bottleneck.

### 5.3 Neural Architecture Search (NAS)

Architecture search [21], [27], [28], [38], [127] can be seen as a kind of hyperparameter optimization where $\omega$ specifies the architecture of a neural network. The inner optimization trains networks with the specified architecture, and the outer optimization searches for architectures with good validation performance. NAS methods have been analysed [38] according to 'search space', 'search strategy', and 'performance estimation strategy'. These correspond to the hypothesis space for $\omega$, the meta-optimization strategy, and the meta-objective. NAS is particularly challenging because: (i) Fully evaluating the inner loop is expensive since it requires training a many-shot neural network to completion. This leads to approximations such as sub-sampling the train set, early termination of the inner loop, and interleaved descent on both $\omega$ and $\theta$ [21] as in online meta-learning. (ii.) The search space is hard to define, and optimize. This is because most search spaces are broad, and the space of architectures is not trivially differentiable. This leads to reliance on cell-level search [21], [28] constraining the search space, RL [28], discrete gradient estimators [129] and evolution [27], [127].

*Topical Issues*. While NAS itself can be seen as an instance of hyper-parameter or hypothesis-class meta-learning, it can also interact with meta-learning in other forms. Since NAS is costly, a topical issue is whether discovered architectures can generalize to new problems [243]. Meta-training across multiple datasets may lead to improved cross-task generalization of architectures [132]. Finally, one can also define NAS meta-objectives to train an architecture suitable for few-shot learning [244]. Similarly to fast-adapting initial condition meta-learning approaches such as MAML [19], one can train good initial architectures [131] or architecture priors [132] that are easy to adapt towards specific tasks.

*Benchmarks* NAS is often evaluated on CIFAR-10, but it is costly to perform and results are hard to reproduce due to confounding factors such as tuning of hyperparameters. To support reproducible and accessible research, the NAS-benches [245] provide pre-computed performance measures for a large number of network architectures.

### 5.4 Hyper-Parameter Optimization

Meta-learning address hyperparameter optimization when considering $\omega$ to specify hyperparameters, such as regularization strength or learning rate. There are two main settings: we can learn hyperparameters that improve training over a distribution of tasks, just a single task. The former case is usually relevant in few-shot applications, especially in optimization based methods. For instance, MAML can be improved by learning a learning rate per layer per step [78]. The case where we wish to learn hyperparameters for a single task is usually more relevant for many-shot applications [69], [154], where some validation data can be extracted from the training dataset, as discussed in Section 2.1. End-to-end gradient-based meta-learning has already demonstrated promising scalability to millions of parameters (as demonstrated by MAML [19] and Dataset Distillation [152], [154], for example) in contrast to the classic approaches (such cross-validation by grid or random [70] search, or Bayesian Optimization [71]) which are typically only successful with dozens of hyper-parameters.

### 5.5 Bayesian Meta-Learning

Bayesian meta-learning approaches formalize meta-learning via Bayesian hierarchical modelling, and use Bayesian inference for learning rather than direct optimization of parameters. In the meta-learning context, Bayesian learning is typically intractable, and so approximations such as stochastic variational inference or sampling are used.

Bayesian meta-learning importantly provides uncertainty measures for the $\omega$ parameters, and hence measures of prediction uncertainty which can be important for safety critical applications, exploration in RL, and active learning.

A number of authors have explored Bayesian approaches to meta-learning complex neural network models with competitive results. For example, extending variational autoencoders to model task variables explicitly [73]. Neural Processes [176] define a feed-forward Bayesian meta-learner inspired by Gaussian Processes but implemented with neural networks. Deep kernel learning is also an active research area that has been adapted to the meta-learning setting [246], and is often coupled with Gaussian Processes [247]. In [74] gradient based meta-learning is recast into a hierarchical empirical Bayes inference problem (i.e., prior learning), which models uncertainty in task-specific parameters $\theta$. Bayesian MAML [248] improves on this model by using a Bayesian ensemble approach that allows non-Gaussian posteriors over $\theta$, and later work removes the need for costly ensembles [45], [249]. In Probabilistic MAML [96], it is the uncertainty in the meta-knowledge $\omega$ that is modelled, while a MAP estimate is used for $\theta$. Increasingly, these Bayesian methods are shown to tackle ambiguous tasks, active learning and RL problems.

Separate from the above, meta-learning has also been proposed to aid the Bayesian inference process itself, as in [250] where the authors adapt a Bayesian sampler to provide efficient adaptive sampling methods.

### 5.6 Unsupervised and Semi-Supervised Meta-Learning

There are several distinct ways in which unsupervised learning can interact with meta-learning, depending on whether unsupervised learning in performed in the inner loop or outer loop, and during meta-train versus meta-test.

*Unsupervised Learning of a Supervised Learner*. The aim here is to learn a supervised learning algorithm (e.g., via MAML [19] style initial condition for supervised fine-tuning), but do so without the requirement of a large set of source tasks for meta-training [251], [252], [253]. To this end, synthetic source tasks are constructed without supervision via clustering or class-preserving data augmentation, and used to define the meta-objective for meta-training.

*Supervised Learning of an Unsupervised Learner*. This family of methods aims to meta-train an unsupervised learner. For example, by training the unsupervised algorithm such that it works well for downstream supervised learning tasks. One can train unsupervised learning rules [16] or losses [99], [122] such that downstream supervised learning performance is optimized – after re-using the unsupervised representation for a supervised task [16], or adapting based on unlabeled data [99], [122]. Alternatively, when unsupervised tasks such as clustering exist in a family, rather than in isolation, then learning-to-learn of 'how-to-cluster' on several source tasks can provide better performance on new clustering tasks in the family

[177], [178], [179], [254], [255]. The methods in this group that make use of feed-forward models are often known as *amortized clustering* [178], [179], because they amortize the typically iterative computation of clustering algorithms into the cost of training a single inference model, which subsequently performs clustering using a single feed-froward pass. Overall, these methods help to deal with the ill-definedness of the unsupervised learning problem by transforming it into a problem with a clear supervised (meta) objective.

*Semi-Supervised Learning (SSL)*. This family of methods aim to train a base-learner capable of performing well on validation data after learning from a mixture of labeled and unlabeled training examples. In the few-shot regime, approaches include SSL extensions of metric-learners [219], and meta-learning measures of confidence [256] or instance weights [157] to ensure that reliably labeled instances are used in self-training. In the many-shot regime, approaches include direct transductive training of labels [156] , or training a teacher network to generate labels for a student [257].

### 5.7 Continual, Online and Adaptive Learning

*Continual Learning*. Refers to the human-like capability of learning tasks presented in sequence. Ideally this is done while exploiting forward transfer so new tasks are learned better given past experience, without forgetting previously learned tasks, and without needing to store past data [61]. Deep Neural Networks struggle to meet these criteria, especially as they tend to forget information seen in earlier tasks – a phenomenon known as *catastrophic forgetting*. Meta-learning can include the requirements of continual learning into a meta-objective, for example by defining a sequence of learning episodes in which the support set contains one new task, but the query set contains examples drawn from all tasks seen until now [105], [171]. Various meta-representations can be learned to improve continual learning performance, such as weight priors [134], gradient descent preconditioning matrices [105], or RNN learned optimizers [171], or feature representations [258]. A related idea is meta-training representations to support local editing updates [259] for improvement without interference.

*Online and Adaptive Learning* Also consider tasks arriving in a stream, but are concerned with the ability to effectively adapt to the current task in the stream, more than remembering the old tasks. To this end an online extension of MAML was proposed [97] to perform MAML-style meta-training online during a task sequence. Meanwhile others [62], [63], [64] consider the setting where meta-training is performed in advance on source tasks, before meta-testing adaptation capabilities on a sequence of target tasks.

*Benchmarks*. A number of benchmarks for continual learning work quite well with standard deep learning methods. However, most cannot readily work with meta-learning approaches as their their sample generation routines do not provide a large number of explicit learning sets and an explicit evaluation sets. Some early steps were made towards defining meta-learning ready continual benchmarks in [97], [171], [258], mainly composed of Omniglot and perturbed versions of MNIST. However, most of those were simply tasks built to demonstrate a method. More explicit benchmark work can be found in [220], which is built for meta and non meta-learning approaches alike.

### 5.8 Domain Adaptation and Domain Generalization

Domain-shift refers to the statistics of data encountered in deployment being different from those used in training. Numerous domain adaptation and generalization algorithms have been studied to address this issue in supervised, unsupervised, and semi-supervised settings [58].

*Domain Generalization*. Domain *generalization* aims to train models with increased robustness to train-test domain shift [260], often by exploiting a distribution over training domains. Using a validation domain that is shifted with respect to the training domain [261], different kinds of meta-knowledge such as regularizers [93], losses [42], and noise augmentation [116] can be (meta) learned to maximize the robustness of the learned model to train-test domain-shift.

*Domain Adaptation*. To improve on conventional domain *adaptation*[58], meta-learning can be used to define a meta-objective that optimizes the performance of a base unsupervised DA algorithm [59].

*Benchmarks*. Popular benchmarks for DA and DG consider image recognition across multiple domains such as photo/sketch/cartoon. PACS [262] provides a good starter benchmark, with Visual Decathlon [42], [221] and Meta-Dataset [109] providing larger scale alternatives.

### 5.9 Language and Speech

*Language Modelling*. Few-shot language modelling increasingly showcases the versatility of meta-learners. Early matching networks showed impressive performances on one-shot tasks such as filling in missing words [88]. Many more tasks have since been tackled, including text classification [135], neural program induction [263] and synthesis [264], English to SQL program synthesis [265], text-based relationship graph extractor [266], machine translation [267], and quickly adapting to new personas in dialogue [268].

*Speech Recognition*. Deep learning is now the dominant paradigm for state of the art automatic speech recognition (ASR). Meta-learning is beginning to be applied to address the many few-shot adaptation problems that arise within ASR including learning how to train for low-resource languages [269], cross-accent adaptation [270] and optimizing models for individual speakers [271].

### 5.10 Emerging Topics

*Environment Learning and Sim2Real*. In Sim2Real we are interested in training a model in simulation that is able to generalize to the real-world. The classic domain randomization approach simulates a wide distribution over domains/MDPs, with the aim of training a sufficiently robust model to succeed in the real world – and has succeeded in both vision [272] and RL [158]. Nevertheless tuning the simulation distribution remains a challenge. This leads to a meta-learning setup where the inner-level optimization learns a model in simulation, the outer-level optimization $\mathcal{L}^{meta}$ evaluates the model's performance in the real-world, and the meta-representation $\omega$ corresponds to the parameters of the simulation environment. This paradigm has been used in RL [160] as well as vision [159], [273]. In this case the source tasks used for meta-train tasks are not a pre-provided data distribution, but paramaterized by omega,

$\mathscr{D}_{source}(\omega)$. However, challenges remain in terms of costly back-propagation through a long graph of inner task learning steps; as well as minimising the number of real-world $\mathcal{L}^{meta}$ evaluations in the case of Sim2Real.

*Meta-Learning for Social Good*. Meta-learning lends itself to challenges that arise in applications of AI for social good such as medical image classification and drug discovery, where data is often scarce. Progress in the medical domain is especially relevant given the global shortage of pathologists [274]. In [5] an LSTM is combined with a graph neural network to predict molecular behaviour (e.g., toxicity) in the one-shot data regime. In [275] MAML is adapted to weakly-supervised breast cancer detection tasks, and the order of tasks are selected according to a curriculum. MAML is also combined with denoising autoencoders to do medical visual question answering [276], while learning to weigh support samples [219] is adapted to pixel wise weighting for skin lesion segmentation tasks that have noisy labels [277].

*Non-Backprop and Biologically Plausible Learners*. Most meta-learning work that uses explicit (non feed-forward/black-box) optimization for the base model is based on gradient descent by backpropagation. Meta-learning can define the function class of $\omega$ so as to lead to the discovery of novel learning rules that are unsupervised [16] or biologically plausible [46], [278], [279], making use of ideas less commonly used in contemporary deep learning such as Hebbian updates [278] and neuromodulation [279].

*Network Compression*. Contemporary CNNs require large amounts of memory that may be prohibitive on embedded devices. Thus network compression in various forms such as quantization and pruning are topical research areas [280]. Meta-learning is beginning to be applied to this objective as well, such as training gradient generator meta-networks that allow quantized networks to be trained [199], and weight generator meta-networks that allow quantized networks to be trained with gradient [281].

*Communications*. systems are increasingly impacted by deep learning. For example by learning coding systems that surpass hand designed codes for realistic channels [282]. Few-shot meta-learning can be used to provide rapid adaptation of codes to changing channel characteristics [283].

*Active Learning (AL)*. Methods wrap supervised learning, and define a policy for selective data annotation – typically in the setting where annotation can be obtained sequentially. The goal of AL is to find the optimal subset of data to annotate so as to maximize performance of downstream supervised learning with the fewest annotations. AL is a well studied problem with numerous hand designed algorithms [284]. Meta-learning can map active learning algorithm design into a learning task by: (i) defining the inner-level optimization as conventional supervised learning on the annotated dataset so far, (ii) defining $\omega$ to be a query policy that selects the best unlabeled datapoints to annotate, (iii), defining the meta-objective as validation performance after iterative learning and annotation according to the query policy, (iv) performing outer-level optimization to train the optimal annotation query policy [187], [188], [189]. However, if labels are used to train AL algorithms, they need to generalize across tasks to amortize their training cost [189].

*Learning with Label Noise*. Commonly arises when large datasets are collected by web scraping or crowd-sourcing. While there are many algorithms hand-designed for this situation, recent meta-learning methods have addressed label noise. For example by transductively learning sample-wise weighs to down-weight noisy samples [147], or learning an initial condition robust to noisy label training [94].

*Adversarial Attacks and Defenses*. Deep Neural Networks can be fooled into misclassifying a data point that should be easily recognizable, by adding a carefully crafted human-invisible perturbation to the data [285]. Numerous attack and defense methods have been published in recent years, with defense strategies usually consisting in carefully hand-designed architectures or training algorithms. Analogous to the case in domain-shift, one can train the learning algorithm for robustness by defining a meta-loss in terms of performance under adversarial attack [95], [286].

*Recommendation Systems*. Are a mature consumer of machine learning in the commerce space. However, bootstrapping recommendations for new users with little historical interaction data, or new items for recommendation remains a challenge known as the *cold-start* problem. Meta-learning has applied black-box models to item cold-start [287] and gradient-based methods to user cold-start [288].

## 6 Challenges and Open Questions

*Diverse and Multi-Modal Task Distributions* The difficulty of fitting a meta-learner to a distribution of tasks $p(\mathcal{T})$ can depend on its *width*. Many big successes of meta-learning have been within narrow task families, while learning on diverse task distributions can challenge existing methods [109], [221], [241]. This may be partly due to conflicting gradients between tasks [289].

Many meta-learning frameworks [19] implicitly assume that the distribution over tasks $p(\mathcal{T})$ is *uni-modal*, and a single learning strategy $\omega$ provides a good solution for them all. However task distributions are often multi-modal; such as medical versus satellite versus everyday images in computer vision, or putting pegs in holes versus opening doors [241] in robotics. Different tasks within the distribution may require different learning strategies, which is hard to achieve with today's methods. In vanilla multi-task learning, this phenomenon is relatively well studied with, e.g., methods that group tasks into clusters [290] or subspaces [291]. However this is only just beginning to be explored in meta-learning [292].

*Meta-Generalization*. Meta-learning poses a new generalization challenge across tasks analogous to the challenge of generalizing across instances in conventional machine learning. There are two sub-challenges: (i) The first is generalizing from meta-train to novel meta-test tasks drawn from $p(\mathcal{T})$. This is exacerbated because the number of *tasks* available for meta-training is typically low (much less than the number of *instances* available in conventional supervised learning), making it difficult to generalize. One failure mode for generalization in few-shot learning has been well studied under the guise of *memorisation* [201], which occurs when each meta-training task can be solved directly without performing any task-specific adaptation based on the support set. In this case models fail to generalize in meta-testing, and specific regularizers [201] have been proposed to prevent this kind of meta-

overfitting. (ii) The second challenge is generalizing to meta-test tasks drawn from a different distribution than the training tasks. This is inevitable in many potential practical applications of meta-learning, for example generalizing few-shot visual learning from everyday training images of ImageNet to specialist domains such as medical images [222]. From the perspective of a learner, this is a meta-level generalization of the domain-shift problem, as observed in supervised learning. Addressing these issues through meta-generalizations of regularization, transfer learning, domain adaptation, and domain generalization are emerging directions [116]. Furthermore, we have yet to understand which kinds of meta-representations tend to generalize better under certain types of domain shifts.

*Task Families*. Many existing meta-learning frameworks, especially for few-shot learning, require task families for meta-training. While this indeed reflects lifelong human learning, in some applications data for such task families may not be available. Unsupervised meta-learning [251], [252], [253] and single-task meta-learning methods [42], [173], [180], [181], [197], could help to alleviate this requirement; as can improvements in meta-generalization discussed above.

*Computation Cost & Many-Shot*. A naive implementation of bilevel optimization as shown in Section 2.1 is expensive in both time (because each outer step requires several inner steps) and memory (because reverse-mode differentiation requires storing the intermediate inner states). For this reason, much of meta-learning has focused on the few-shot regime [19], where the number of inner gradient steps (aka the horizon) is small. However, there is an increasing focus on methods which seek to extend optimization-based meta-learning to the many-shot regime, where long horizons are inevitable. Popular solutions include implicit differentiation of $\omega$ [154], [163], [293], forward-mode differentiation of $\omega$ [67], [69], [294], gradient preconditioning [105], evolutionary strategies [295], solving for a greedy version of $\omega$ online by alternating inner and outer steps [21], [42], [198], truncation [296], shortcuts [297] or inversion [190] of the inner optimization. Long horizon meta-learning can also be achieved by learning an initialization that minimizes the gradient descent trajectory length over task manifolds [298]. Finally, another family of approaches accelerate meta-training via closed-form solvers in the inner loop [164], [165].

Each of these strategies provides different trade-offs on the following axes: the accuracy at meta-test time, the compute and memory cost as the size of $\omega$ increases, and the computate and memory cost as the horizon gets longer. Implicit gradients scale to high-dimensional $\omega$; but only provide approximate gradients for it, and require the inner task loss to be a function of $\omega$. Forward-mode differentiation is exact and doesn't have such constraints, but scales poorly with the dimension of $\omega$. Online methods are cheap in both memory and compute but suffer from a short-horizon bias [299] and thus have lower meta-test accuracy. The same could be said of truncation methods which are cheap but suffer from truncation bias [300]. Gradient degradation is also a challenge in the many-shot regime, and solutions include warp layers [105] or gradient averaging [69].

In terms of the cost of solving new tasks at the meta-test stage, FFMs have a significant advantage over optimization-based meta-learners, which makes them appealing for applications involving deployment of learning algorithms on mobile devices such as smartphones [6], for example to achieve personalisation. This is especially so because the embedded device versions of contemporary deep learning software frameworks typically lack support for backpropagation-based training, which FFMs do not require.

## 7 Conclusion

The field of meta-learning has seen a rapid growth in interest. This has come with some level of confusion, with regards to how it relates to neighbouring fields, what it can be applied to, and how it can be benchmarked. In this survey we have sought to clarify these issues by thoroughly surveying the area both from a methodological point of view – which we broke down into a taxonomy of meta-representation, meta-optimizer and meta-objective; and from an application point of view. We hope that this survey will help newcomers and practitioners to orient themselves to develop and exploit in this growing field, as well as highlight opportunities for future research.

## Acknowledgments

The work of Timothy Hospedales was supported in part by the Engineering and Physical Sciences Research Council of the U.K. (EPSRC) under Grant EP/S000631/1, in part by the U.K. MOD University Defence Research Collaboration (UDRC) in Signal Processing, and in part by the EPSRC under Grant EP/R026173/1.

## References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[2] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 2016.
[3] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Assoc. Comput. Linguistics*, 2019, pp. 4171–4186.
[4] G. Marcus, "Deep learning: A critical pppraisal," 2018, *arXiv: 1801.00631*.
[5] H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. Pande, "Low data drug discovery with one-shot learning," *ACS Central Sci.*, vol. 3, no. 4, pp. 283–293, 2017.
[6] A. Ignatov *et al.* "AI benchmark: All about deep learning on smartphones in 2019," 2019, *arXiv: 1910.06663*.
[7] S. Thrun and L. Pratt, "Learning to learn: Introduction and overview,"in, *Learning to Learn*. Boston, MA, USA: Springer, 1998.
[8] H. F. Harlow, "The formation of learning sets," *Psychol. Rev.*, vol. 56, pp. 51–65, 1949.
[9] J. B. Biggs, "The role of meta-learning in study processes," *Brit. J. Educ. Psychol.*, vol. 55, pp. 185–212, 1985.
[10] A. M. Schrier, "Learning how to learn: The significance and current status of learning set formation," *Primates*, vol. 25, pp. 95–102, 1984.
[11] P. Domingos, "A few useful things to know about machine learning," in *Proc. Commun. ACM*, 2012, pp. 78–87.
[12] D. G. Lowe, "Distinctive image features from scale-invariant," *Int. J. Comput. Vis.*, vol. 60, pp. 91–110, 2004.
[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Commun. ACM*, 2012, pp. 84–90.
[14] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proc. Int. Conf. Learn. Representations*, 2016.
[15] M. Andrychowicz *et al.*, "Learning to learn by gradient descent by gradient descent," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3988–3996.

[16] L. Metz, N. Maheswaranathan, B. Cheung, and J. Sohl-Dickstein, "Meta-learning update rules for unsupervised representation learning," in *Proc. Int. Conf. Learn. Representations*, 2019.

[17] J. Schmidhuber, "Evolutionary principles in self-referential learning. On learning now to learn: The meta-meta-meta...-hook," Diploma thesis, Technische Universitat Munchen, Munich, Germany, 1987.

[18] J. Schmidhuber, J. Zhao, and M. Wiering, "Shifting inductive bias with success-story algorithm, adaptive Levin search, and incremental self-improvement," *Mach. Learn.*, vol. 28, pp. 105–130, 1997.

[19] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[20] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, "Bilevel programming for hyperparameter optimization and meta-learning," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 1568–1577.

[21] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2019.

[22] W. Zhou, Y. Li, Y. Yang, H. Wang, and T. M. Hospedales, "Online meta-critic learning for off-policy actor-critic methods," in *Proc. 34th Conf. Neural Inf. Process. Syst.*, 2020.

[23] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical Networks for few shot learning," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, 2017.

[24] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "$RL^2$: Fast reinforcement learning via slow reinforcement learning," 2016, *arXiv:1611.02779*.

[25] R. Houthooft *et al.*, "Evolved policy gradients," in *Proc. 32nd Conf. Neural Inf. Process. Syst.*, 2018.

[26] F. Alet, M. F. Schneider, T. Lozano-Perez, and L. PackKaelbling, "Meta-learning curiosity algorithms," in *Proc. Int. Conf. Learn. Representations*, 2020.

[27] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. 33rd Assoc. Adv. Artif. Intell.*, 2019, pp. 4780–4789.

[28] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017.

[29] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artif. Intell. Rev.*, vol. 18, pp. 77–95, 2002.

[30] S. Thrun, "Lifelong learning algorithms," in *Learning to Learn*. Berlin, Germany: Springer, 1998, pp. 181–209.

[31] J. Baxter, "Theoretical models of learning to learn," in *Learning to Learn*. Berlin, Germany: Springer, 1998, pp. 71–94.

[32] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Comput.*, vol. 8, pp. 1341–1390, 1996.

[33] J. Vanschoren, "Meta-learning: A survey," 2018, *arXiv: 1810. 03548*.

[34] F. Hutter, L. Kotthoff, and J. Vanschoren, Eds.,*Automatic Machine Learning: Methods, Systems, Challenges*. Berlin, Germany: Springer, 2019.

[35] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[36] C. Lemke, M. Budka, and B. Gabrys, "Meta-learning: A survey of trends and technologies," *Artif. Intell. Rev.*, vol. 44, pp. 117–130, 2015, Art. no. 63.

[37] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, Jun. 2020, Art. no. 63.

[38] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, pp. 1–21, 2019.

[39] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," *Proc. Int. Conf. Learn. Representations*, 2018.

[40] H. Stackelberg, *The Theory of Market Economy*. London, U.K.: Oxford Univ. Press, 1952.

[41] A. Sinha, P. Malo, and K. Deb, "A review on Bilevel optimization: From classical to evolutionary approaches and applications," *IEEE Trans. Evol. Comput.*, vol. 22, no. 2, pp. 276–295, Apr. 2018.

[42] Y. Li, Y. Yang, W. Zhou, and T. M. Hospedales, "Feature-critic networks for heterogeneous domain generalization," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 3915–3924.

[43] G. Denevi, C. Ciliberto, D. Stamos, and M. Pontil, "Learning to learn around a common mean," in *Proc. 32nd Conf. Neural Inf. Process. Syst.*, 2018.

[44] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 3394–3404.

[45] J. Gordon, J. Bronskill, M. Bauer, S. Nowozin, and R. E. Turner, "Meta-learning probabilistic inference for prediction," *Proc. Int. Conf. Learn. Representations*, 2019, pp. 3915–3924.

[46] Y. Bengio, S. Bengio, and J. Cloutier, "Learning a synaptic learning rule," in *Proc. Int. Joint Conf. Neural Netw.*, 1990, p. 969.

[47] S. Bengio, Y. Bengio, and J. Cloutier, "On the search for new learning rules for ANNs," *Neural Process. Lett.*, vol. 2, pp. 26–30, 1995.

[48] J. Schmidhuber, J. Zhao, and M. Wiering, "Simple principles of meta-learning," Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Lugano, Switzerland, Tech. Rep. TR-IDSIA-69–96, 1996.

[49] J. Schmidhuber, "A neural network that embeds its own meta-levels," in *Proc. IEEE Int. Conf. Neural Netw.*, 1993, pp. 407–412.

[50] J.-A. Meyer and S. W. Wilson, "A possibility for implementing curiosity and boredom in model-building neural controllers," in *Proc. Animals Animats, 1st Int. Conf. Simul. Adaptive Behav.*, 1991, pp. 222–227.

[51] S. Hochreiter, A. S. Younger, and P. R. Conwell, "Learning to learn using gradient descent," in *Proc. Int. Conf. Artif. Neural Netw.*, 2001, pp. 87–94.

[52] A. S. Younger, S. Hochreiter, and P. R. Conwell, "Meta-learning with backpropagation," in *Proc. Int. Joint Conf. Neural Netw.*, 2001, pp. 2001–2006.

[53] J. Storck, S. Hochreiter, and J. Schmidhuber, "Reinforcement driven information acquisition in non-deterministic environments," in *Proc. Int. Conf. Artif. Neural Netw.*, 1995, pp. 159–164.

[54] M. Wiering and J. Schmidhuber, "Efficient model-based exploration," in *Proc. 5th Int. Conf. Simul. Adaptive Behav.*, 1998, pp. 223–228.

[55] N. Schweighofer and K. Doya, "Meta-learning in reinforcement learning," *Neural Netw.*, vol. 16, pp. 5–9, 2003.

[56] L. Y. Pratt, J. Mostow, C. A. Kamm, and A. A. Kamm, "Direct transfer of learned information among neural networks," in *Proc. Assoc. Adv. Artif. Intell.*, 1991, pp. 584–589.

[57] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 3320–3328.

[58] G. Csurka, *Domain Adaptation in Computer Vision Applications*. Berlin, Germany: Springer, 2017.

[59] D. Li and T. Hospedales, "Online meta-learning for multi-source and semi-supervised domain adaptation," in *Proc. Eur. Conf. Comput. Vis.*, 2020.

[60] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Netw.*, vol. 113, pp. 54–71, 2019.

[61] Z. Chen and B. Liu, "Lifelong machine learning," *Synth. Lectures Artif. Intell. Mach. Learn.*, vol. 12, no. 3, pp. 1–207, 2018.

[62] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel, "Continuous adaptation via meta-learning in nonstationary and competitive environments," in *Proc. Int. Conf. Learn. Representations*, 2018.

[63] S. Ritter *et al.*, "Been there, done that: Meta-learning with episodic recall," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 4354–4363.

[64] I. Clavera *et al.*, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2019.

[65] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, pp. 41–75, 1997.

[66] Y. Yang and T. M. Hospedales, "Deep multi-task representation learning: A tensor factorisation approach," in *Proc. Int. Conf. Learn. Representations*, 2017.

[67] L. Franceschi, M. Donini, P. Frasconi, and M. Pontil, "Forward and reverse gradient-based hyperparameter optimization," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1165–1173.

[68] X. Lin, H. Baweja, G. Kantor, and D. Held, "Adaptive auxiliary task weighting for reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019.

[69] P. Micaelli and A. Storkey, "Non-greedy gradient-based hyperparameter optimization over long horizons," 2020, *arXiv: 2007.07869*.

[70] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[71] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, Jan. 2016.

[72] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirchlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.

[73] H. Edwards and A. Storkey, "Towards a neural statistician," in *Proc. Int. Conf. Learn. Representations*, 2017.

[74] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths, "Recasting gradient-based meta-learning as hierarchical Bayes," in *Proc. Int. Conf. Learn. Representations*, 2018.

[75] H. Yao *et al.*, "Automated relational meta-learning," in *Proc. Int. Conf. Learn. Representations*, 2020.

[76] S. C. Yoonho Lee, "Gradient-based meta-learning with learned layerwise metric and subspace," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 2927–2936.

[77] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-SGD: Learning to learn quickly for few shot learning," 2017, *arXiv: 1707.09835*.

[78] A. Antoniou, H. Edwards, and A. J. Storkey, "How to train your MAML," in *Proc. Int. Conf. Learn. Representations*, 2018.

[79] K. Li and J. Malik, "Learning to optimize," in *Proc. Int. Conf. Learn. Representations*, 2017.

[80] E. Grefenstette *et al.* "Generalized inner loop meta-learning," 2019, *arXiv: 1910.01727*.

[81] S. Qiao, C. Liu, W. Shen, and A. L. Yuille, "Few-shot image recognition by predicting parameters from activations," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7229–7238.

[82] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4367–4375.

[83] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," 2014, *arXiv:1410.5401*.

[84] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta learning with memory-augmented neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2016.

[85] T. Munkhdalai and H. Yu, "Meta networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 2554–2563.

[86] C. Finn and S. Levine, "Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–12.

[87] G. Kosh, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015.

[88] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016.

[89] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1199–1208.

[90] V. Garcia and J. Bruna, "Few-shot learning with graph neural networks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–13.

[91] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le, "Neural optimizer search with reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 459–468.

[92] O. Wichrowska *et al.*, "Learned optimizers that scale and generalize," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3751–3760.

[93] Y. Balaji, S. Sankaranarayanan, and R. Chellappa, "MetaReg: Towards domain generalization using meta-regularization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 1–11.

[94] J. Li, Y. Wong, Q. Zhao, and M. S. Kankanhalli, "Learning to learn from noisy labeled data," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019.

[95] M. Goldblum, L. Fowl, and T. Goldstein, "Adversarially robust few-shot learning: A meta-learning approach," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020.

[96] C. Finn, K. Xu, and S. Levine, "Probabilistic model-agnostic meta-learning," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 9537–9548.

[97] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine, "Online meta-learning," *Proc. Int. Conf. Mach. Learn.*, 2019.

[98] A. A. Rusu *et al.*, "Meta-learning with latent embedding optimization," in *Proc. Int. Conf. Learn. Representations*, 2019.

[99] A. Antoniou and A. Storkey, "Learning to learn by self-critique," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019.

[100] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 403–412.

[101] R. Vuorio, S.-H. Sun, H. Hu, and J. J. Lim, "Multimodal model-agnostic meta-learning via task-aware modulation," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019.

[102] H. Yao, Y. Wei, J. Huang, and Z. Li, "Hierarchically structured meta-learning," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 7045–7054.

[103] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–15.

[104] E. Park and J. B. Oliva, "Meta-curvature," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019.

[105] S. Flennerhag, A. A. Rusu, R. Pascanu, F. Visin, H. Yin, and R. Hadsell, "Meta-learning with warped gradient descent," in *Proc. Int. Conf. Learn. Representations*, 2020.

[106] Y. Chen *et al.*, "Learning to learn without gradient descent by gradient descent," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 748–756.

[107] T. Heskes, "Empirical Bayes for learning to learn," in *Proc. Int. Conf. Mach. Learn.*, 2000.

[108] J. Requeima, J. Gordon, J. Bronskill, S. Nowozin, and R. E. Turner, "Fast and flexible multi-task classification using conditional neural adaptive processes," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019.

[109] E. Triantafillou *et al.*, "Meta-dataset: A dataset of datasets for learning to learn from few examples," in *Proc. Int. Conf. Learn. Representations*, 2020.

[110] D. Ha, A. Dai, and Q. V. Le, "HyperNetworks," *Proc. Int. Conf. Learn. Representations*, 2017.

[111] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 5331–5340.

[112] Y. Duan *et al.*, "One-shot imitation learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017.

[113] J. X. Wang *et al.* "Learning to reinforcement learn," 2016, *arXiv:1611.05763*.

[114] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. Wang, and J.-B. Huang, "A closer look at few-shot classification," in *Proc. Int. Conf. Learn. Representations*, 2019.

[115] B. Oreshkin, P. Rodr íguez López, and A. Lacoste, "TADAM: Task dependent adaptive metric for improved few-shot learning," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018.

[116] H.-Y. Tseng, H.-Y. Lee, J.-B. Huang, and M.-H. Yang, "Cross-domain few-shot classification via learned feature-wise transformation," *Proc. Int. Conf. Learn. Representations*, Jan. 2020, pp. 1–18.

[117] F. Sung, L. Zhang, T. Xiang, T. Hospedales, and Y. Yang, "Learning to learn: Meta-critic networks for sample efficient learning," 2017, *arXiv: 1706.09529*.

[118] G. Denevi, D. Stamos, C. Ciliberto, and M. Pontil, "Online-within-online meta-learning," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–11.

[119] S. Gonzalez and R. Miikkulainen, "Improved training speed, accuracy, and data utilization through loss function optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2020, pp. 1–8.

[120] S. Bechtle *et al.*, "Meta-learning via learned loss," in *Proc. Int. Conf. Pattern Recognit.*, 2020.

[121] B. Gao, H. Gouk, and T. M. Hospedales, "Searching for robustness: Loss learning for noisy classification tasks," 2021, *arXiv:2103.00243*.

[122] A. I. Rinu Boney, "Semi-supervised few-shot learning with MAML," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–4.

[123] C. Huang *et al.*, "Addressing the loss-metric mismatch with adaptive loss alignment," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2891–2900.

[124] C. Doersch and A. Zisserman, "Multi-task self-supervised visual learning," in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 2051–2060.

[125] M. Jaderberg *et al.*, "Reinforcement learning with unsupervised auxiliary tasks," in *Proc. Int. Conf. Learn. Representations*, 2017.

[126] S. Liu, A. Davison, and E. Johns, "Self-supervised generalisation with meta auxiliary learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1677–1687.

[127] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," *Nat. Mach. Intell.*, vol. 1, pp. 24–35, 2019.

[128] J. Bayer, D. Wierstra, J. Togelius, and J. Schmidhuber, "Evolving memory cell structures for sequence learning," in *Proc. Int. Conf. Artif. Neural Netw.*, 2009, pp. 755–764.

[129] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: Stochastic neural architecture search," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–17.

[130] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter, "Understanding and robustifying differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020.

[131] D. Lian *et al.*, "Towards fast adaptation of neural architectures with meta learning," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–16.

[132] A. Shaw, W. Wei, W. Liu, L. Song, and B. Dai, "Meta architecture search," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019.

[133] R. Hou, H. Chang, M. Bingpeng, S. Shan, and X. Chen, "Cross attention network for few-shot classification," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019.

[134] M. Ren, R. Liao, E. Fetaya, and R. Zemel, "Incremental few-shot learning with attention attractor networks," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–11.

[135] Y. Bao, M. Wu, S. Chang, and R. Barzilay, "Few-shot text classification with distributional signatures," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–24.

[136] F. Alet, T. Lozano-P érez, and L. P. Kaelbling, "Modular meta-learning," in *Proc. 2nd Conf. Robot Learn.*, 2018, pp. 856–868.

[137] F. Alet, E. Weng, T. Lozano-P érez, and L. P. Kaelbling, "Neural relational inference with fast modular meta-learning," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–12.

[138] C. Fernando *et al.*, "PathNet: Evolution channels gradient descent in super neural networks," 2017, *arXiv: 1701.08734*.

[139] B. M. Lake, "Compositional generalization through meta sequence-to-sequence learning," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–11.

[140] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le, "AutoAugment: Learning augmentation policies from data," *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 113–123.

[141] Y. Li, G. Hu, Y. Wang, T. Hospedales, N. M. Robertson, and Y. Yang, "DADA: Differentiable automatic data augmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 1–16.

[142] R. Volpi and V. Murino, "Model vulnerability to distributional shifts over image transformation sets," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 33–35.

[143] Z. Mai, G. Hu, D. Chen, F. Shen, and H. T. Shen, "MetaMixUp: Learning adaptive interpolation policy of mixup with meta-learning," 2019, *arXiv: 1908.10059*.

[144] C. Zhang, C. Öztireli, S. Mandt, and G. Salvi, "Active mini-batch sampling using repulsive point processes," in *Proc. 33rd Conf. Assoc. Adv. Artif. Intell.*, 2019, 5741–5748.

[145] I. Loshchilov and F. Hutter, "Online batch selection for faster training of neural networks," in *Proc. Int. Conf. Learn. Representations*, 2016, pp. 1–20.

[146] Y. Fan, F. Tian, T. Qin, X. Li, and T. Liu, "Learning to teach," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–16.

[147] J. Shu *et al.*, "Meta-weight-net: Learning an explicit mapping for sample weighting," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1917–1928.

[148] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 4334–4343.

[149] J. L. Elman, "Learning and development in neural networks: The importance of starting small," *Cognition*, vol. 48, pp. 71–99, 1993.

[150] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. 26th Int. Conf. Mach. Learn.*, 2009, pp. 41–48.

[151] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, "MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 2304–2313.

[152] T. Wang, J. Zhu, A. Torralba, and A. A. Efros, "Dataset distillation," 2018, *arXiv: 1811.10959*.

[153] B. Zhao, K. R. Mopuri, and H. Bilen, "Dataset condensation with gradient matching," in *Proc. Int. Conf. Learn. Representations*, 2021.

[154] J. Lorraine, P. Vicol, and D. Duvenaud, "Optimizing millions of hyperparameters by implicit differentiation," in *Proc. 23rd Int. Conf. Artif. Intell. Stat.*, 2020, pp. 1540–1552.

[155] O. Bohdal, Y. Yang, and T. Hospedales, "Flexible dataset distillation: Learn labels instead of images," 2020, *arXiv: 2006.08572*.

[156] W.-H. Li, C.-S. Foo, and H. Bilen, "Learning to impute: A general framework for semi-supervised learning," 2019, *arXiv: 1912.10364*.

[157] Q. Sun, X. Li, Y. Liu, S. Zheng, T.-S. Chua, and B. Schiele, "Learning to self-train for semi-supervised few-shot classification," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019.

[158] O. M. Andrychowicz *et al.* "Learning dexterous in-hand manipulation," *Int. J. Robot. Res.*, vol. 39, pp. 3–20, 2020.

[159] N. Ruiz, S. Schulter, and M. Chandraker, "Learning to simulate," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–12.

[160] Q. Vuong, S. Vikram, H. Su, S. Gao, and H. I. Christensen, "How To pick the domain randomization parameters for sim-to-real transfer of reinforcement learning policies?," 2019, *arXiv: 1903.11774*.

[161] Q. V. Le, P. Ramachandran, and B. Zoph, "Searching for activation functions," 2017, *arXiv: 1710.05941*.

[162] H. B. Lee *et al.*, "Learning to balance: Bayesian meta-learning for imbalanced and out-of-distribution tasks," in *Proc. Int. Conf. Learn. Representations*, 2020.

[163] A. Rajeswaran, C. Finn, S. Kakade, and S. Levine, "Meta-learning with implicit gradients," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–12.

[164] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10657–10665.

[165] L. Bertinetto, J. F. Henriques, P. H. Torr, and A. Vedaldi, "Meta-learning with differentiable closed-form solvers," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–15.

[166] R. Fakoor, P. Chaudhari, S. Soatto, and A. J. Smola, "Meta-Q-learning," in *Proc. Int. Conf. Learn. Representations*, 2020.

[167] H. Liu, R. Socher, and C. Xiong, "Taming MAML: Efficient unbiased meta-reinforcement learning," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 4061–4071.

[168] J. Rothfuss, D. Lee, I. Clavera, T. Asfour, and P. Abbeel, "ProMP: Proximal meta-policy search," in *Proc. Int. Conf. Learn. Representations*, 2019.

[169] X. Song, W. Gao, Y. Yang, K. Choromanski, A. Pacchiano, and Y. Tang, "ES-MAML: Simple Hessian-free meta learning," in *Proc. Int. Conf. Learn. Representations*, 2020.

[170] C. Fernando *et al.*, "Meta-learning by the Baldwin effect," in *Proc. Genet. Evol. Comput. Conf. Companion*, 2018, pp. 109–110.

[171] R. Vuorio, D.-Y. Cho, D. Kim, and J. Kim, "Meta continual learning," 2018, *arXiv: 1806.06928*.

[172] K. Young, B. Wang, and M. E. Taylor, "Metatrace actor-critic: Online step-size tuning by meta-gradient descent for reinforcement learning control," in *Proc. 28th Inte. Joint Conf. Artif. Intell.*, 2019, pp. 4185–4191.

[173] Z. Xu, H. van Hasselt, and D. Silver, "Meta-gradient reinforcement learning," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 1–12.

[174] M. Jaderberg *et al.* "Human-level performance in 3D multiplayer games with population-based reinforcement learning," *Science*, vol. 364, pp. 859–865, 2019.

[175] J.-M. Perez-Rua, X. Zhu, T. Hospedales, and T. Xiang, "Incremental few-shot object detection," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 13846–13855.

[176] M. Garnelo *et al.*, "Conditional neural processes," *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 1704–1713.

[177] A. Pakman, Y. Wang, C. Mitelut, J. Lee, and L. Paninski, "Neural clustering processes," in *Proc. 37th Int. Conf. Mach. Learn.*, 2019, pp. 7455–7465.

[178] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 3744–3753.

[179] J. Lee, Y. Lee, and Y. W. Teh, "Deep amortized clustering," in *Proc. Int. Conf. Learn. Representations*, 2019.

[180] V. Veeriah *et al.*, "Discovery of useful questions as auxiliary tasks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019.

[181] Z. Zheng, J. Oh, and S. Singh, "On learning intrinsic rewards for policy gradient methods," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 1–11.

[182] T. Xu, Q. Liu, L. Zhao, and J. Peng, "Learning to explore with meta-policy gradient," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5463–5472.

[183] B. C. Stadie *et al.*, "Some considerations on learning to explore via meta-reinforcement learning," in *Proc. 36th Int. Conf. Neural Inf. Process. Syst.*, 2018.
[184] F. Garcia and P. S. Thomas, "A meta-MDP approach to exploration for lifelong reinforcement learning," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–10.
[185] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine, "Meta-reinforcement learning of structured exploration strategies," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 5307–5316.
[186] H. B. Lee, T. Nam, E. Yang, and S. J. Hwang, "Meta dropout: Learning to perturb latent features for generalization," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–11.
[187] P. Bachman, A. Sordoni, and A. Trischler, "Learning algorithms for active learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 301–310.
[188] K. Konyushkova, R. Sznitman, and P. Fua, "Learning active learning from data," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1–11.
[189] K. Pang, M. Dong, Y. Wu, and T. M. Hospedales, "Meta-learning transferable active learning policies by deep reinforcement learning," 2018, *arXiv: 1806.04798*.
[190] D. Maclaurin, D. Duvenaud, and R. P. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2113–2122.
[191] C. Russell, M. Toso, and N. Campbell, "Fixing implicit derivatives: Trust-region based learning of continuous energy functions," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–11.
[192] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," 2018, *arXiv: 1803.02999*.
[193] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," 2017, *arXiv: 1703.03864*.
[194] F. Stulp and O. Sigaud, "Robot skill learning: From reinforcement learning to evolution strategies," *Paladyn, J. Behav. Robot.*, vol. 4, pp. 49–61, 2013.
[195] A. Soltoggio, K. O. Stanley, and S. Risi, "Born to learn: The inspiration, progress, and future of evolved plastic artificial neural networks," *Neural Netw.*, vol. 108, pp. 48–67, 2018.
[196] Y. Cao, T. Chen, Z. Wang, and Y. Shen, "Learning to optimize in swarms," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–11.
[197] F. Meier, D. Kappler, and S. Schaal, "Online learning of a memory for learning rates," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 2425–2432.
[198] A. G. Baydin, R. Cornish, D. Mart ínez-Rubio, M. Schmidt, and F. D. Wood, "Online learning rate adaptation with hypergradient descent," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–11.
[199] S. Chen, W. Wang, and S. J. Pan, "MetaQuant: Learning to quantize by learning to penetrate non-differentiable quantization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–11.
[200] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 843–852.
[201] M. Yin, G. Tucker, M. Zhou, S. Levine, and C. Finn, "Meta-learning without memorization," in *Proc. Int. Conf. Learn. Representations*, 2020.
[202] S. W. Yoon, J. Seo, and J. Moon, "TapNet: Neural network augmented with task-adaptive projection for few-shot learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7115–7123.
[203] J. W. Rae, S. Bartunov, and T. P. Lillicrap, "Meta-learning neural bloom filters," *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5271–5280.
[204] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, "Rapid learning or feature reuse? Towards understanding the effectiveness of MAML," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–21.
[205] B. Kang, Z. Liu, X. Wang, F. Yu, J. Feng, and T. Darrell, "Few-shot object detection via feature reweighting," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 8420–8429.
[206] L.-Y. Gui, Y.-X. Wang, D. Ramanan, and J. Moura, "Few-shot human motion prediction via meta-learning," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 441–459.
[207] A. Shaban, S. Bansal, Z. Liu, I. Essa, and B. Boots, "One-shot learning for semantic segmentation," in *Proc. Brit. Mach. Vis. Conf.*, 2017, pp. 167.1–167.13.
[208] N. Dong and E. P. Xing, "Few-shot semantic segmentation with prototype learning," in *Proc. Brit. Mach. Vis. Conf.*, 2018, pp. 1–13.
[209] K. Rakelly, E. Shelhamer, T. Darrell, A. A. Efros, and S. Levine, "Few-shot segmentation propagation with guided networks," in *Proc. Int. Conf. Mach. Learn.*, 2019.
[210] M. Bishay, G. Zoumpourlis, and I. Patras, "TARN: Temporal attentive relation network for few-shot and zero-shot action recognition," in *Proc. Brit. Mach. Vis. Conf.*, 2019, pp. 1–14.
[211] H. Coskun *et al.*, "Domain-specific priors and meta learning for few-shot first-person action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Feb. 10, 2021, doi: 10.1109/TPAMI.2021.3058606.
[212] L.-Y. Gui, Y.-X. Wang, D. Ramanan, and J. M. Moura, "Few-shot human motion prediction via meta-learning," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 1–19.
[213] S. A. Eslami *et al.*, "Neural scene representation and rendering," *Science*, vol. 360, no. 6394, pp. 1204–1210, 2018.
[214] E. Zakharov, A. Shysheya, E. Burkov, and V. S. Lempitsky, "Few-shot adversarial learning of realistic neural talking head models," in *Proc. Int. Conf. Comput. Vis.*, 2019.
[215] T.-C. Wang, M.-Y. Liu, A. Tao, G. Liu, J. Kautz, and B. Catanzaro, "Few-shot video-to-video synthesis," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019.
[216] F. Shen, S. Yan, and G. Zeng, "Neural style transfer via meta networks," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8061–8069.
[217] S. E. Reed *et al.*, "Few-shot autoregressive density estimation: towards learning to learn distributions," in *Proc. Int. Conf. Learn. Representations*, 2018.
[218] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, pp. 211–252, 2015.
[219] M. Ren *et al.*, "Meta-learning for semi-supervised few-shot classification," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–15.
[220] A. Antoniou, P. Massimiliano, O. Mateusz, and A. Storkey, "Defining benchmarks for continual few-shot learning," 2020, *arXiv: 2004.11967*.
[221] S.-A. Rebuffi, H. Bilen, and A. Vedaldi, "Learning multiple visual domains with residual adapters," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017.
[222] Y. Guo *et al.*, "A broader study of cross-domain few-shot learning," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 1–18.
[223] T. De Vries, I. Misra, C. Wang, and L. van der Maaten, "Does object recognition work for everyone?," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 52–59.
[224] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, pp. 229–256, 1992.
[225] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Provably convergent policy gradient methods for model-agnostic meta-reinforcement learning," 2020, *arXiv: 2002.05135*.
[226] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv: 1707.06347*.
[227] O. Sigaud and F. Stulp, "Policy search in continuous action domains: An overview," *Neural Netw.*, vol. 113, pp. 28–40, 2019.
[228] J. Schmidhuber, "What's interesting?," Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Lugano, Switzerland, Tech. Rep. TR-IDSIA-35–97, 1997.
[229] L. Kirsch, S. van Steenkiste, and J. Schmidhuber, "Improving generalization in meta reinforcement learning using learned objectives," in *Proc. Int. Conf. Learn. Representations*, 2020.
[230] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[231] O. Kroemer, S. Niekum, and G. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms," *J. Mach. Learn. Res.*, vol. 22, pp. 1–82, 2021.
[232] A. Jabri, K. Hsu, A. Gupta, B. Eysenbach, S. Levine, and C. Finn, "Unsupervised curricula for visual meta-reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019.
[233] Y. Yang, K. Caluwaerts, A. Iscen, J. Tan, and C. Finn, "NoRML: No-reward meta learning," in *Proc. Int. Conf. Autonom. Agents Multi-Agent Syst.*, 2019, pp. 323–331.
[234] S. K. S. Ghasemipour, S. S. Gu, and R. Zemel, "SMILe: Scalable meta inverse reinforcement learning through context-conditional policies," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–11.
[235] M. C. Machado, M. G. Bellemare, E. Talvitie, J. Veness, M. Hausknecht, and M. Bowling, "Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents," *J. Artif. Intell. Res.*, vol. 61, pp. 523–562, 2018.

[236] A. Nichol, V. Pfau, C. Hesse, O. Klimov, and J. Schulman, "Gotta learn fast: A new benchmark for generalization in RL," 2018, *arXiv: 1804.03720*.

[237] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1282–1289.

[238] G. Brockman *et al.* "OpenAI gym," 2016, *arXiv:1606.01540*.

[239] C. Packer, K. Gao, J. Kos, P. Krähenbühl, V. Koltun, and D. Song, "Assessing generalization in deep reinforcement learning," 2018, *arXiv: 1810.12282*.

[240] C. Zhao, O. Siguad, F. Stulp, and T. M. Hospedales, "Investigating generalisation in continuous deep reinforcement learning," 2019, *arXiv: 1902.07015*.

[241] T. Yu *et al.*, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," 2019, *arXiv: 1910.10897*.

[242] A. Bakhtin, L. van der Maaten, J. Johnson, L. Gustafson, and R. Girshick, "PHYRE: A new benchmark for physical reasoning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019.

[243] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8697–8708.

[244] T. Elsken, B. Staffler, J. H. Metzen, and F. Hutter, "Meta-learning of neural architectures for few-shot learning," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12365–12375.

[245] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-bench-101: Towards reproducible neural architecture search," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7105–7114.

[246] P. Tossou, B. Dura, F. Laviolette, M. Marchand, and A. Lacoste, "Adaptive deep kernel learning," 2019, *arXiv: 1905.12131*.

[247] M. Patacchiola, J. Turner, E. J. Crowley, M. O'Boyle, and A. Storkey, "Deep kernel transfer in Gaussian processes for few-shot learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020.

[248] T. Kim, J. Yoon, O. Dia, S. Kim, Y. Bengio, and S. Ahn, "Bayesian model-agnostic meta-learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018.

[249] S. Ravi and A. Beatson, "Amortized Bayesian meta-learning," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–14.

[250] Z. Wang, Y. Zhao, P. Yu, R. Zhang, and C. Chen, "Bayesian meta sampling for fast uncertainty adaptation," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–22.

[251] K. Hsu, S. Levine, and C. Finn, "Unsupervised learning via meta-learning," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–24.

[252] S. Khodadadeh, L. Boloni, and M. Shah, "Unsupervised meta-learning for few-shot image classification," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 10132–10142.

[253] A. Antoniou and A. Storkey, "Assume, augment and learn: Unsupervised few-shot meta-learning via random labels and data augmentation," 2019, *arXiv: 1902.09884*.

[254] Y. Jiang and N. Verma, "Meta-learning to cluster," 2019, *arXiv: 1910.14134*.

[255] V. Garg and A. T. Kalai, "Supervising unsupervised learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018.

[256] S. M. Kye, H. B. Lee, H. Kim, and S. J. Hwang, "Meta-learned confidence for few-shot learning," 2020, *arXiv: 2002.12017*.

[257] H. Pham, Q. Xie, Z. Dai, and Q. V. Le, "Meta pseudo labels," 2020, *arXiv: 2003.10580*.

[258] K. Javed and M. White, "Meta-learning representations for continual learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–11.

[259] A. Sinitsin, V. Plokhotnyuk, D. Pyrkin, S. Popov, and A. Babenko, "Editable neural networks," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–12.

[260] K. Muandet, D. Balduzzi, and B. Schölkopf, "Domain generalization via invariant feature representation," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. I-10–I-18.

[261] D. Li, Y. Yang, Y. Song, and T. Hospedales, "Learning to generalize: Meta-learning for domain generalization," in *Proc. 32nd Assoc. Adv. Artif. Intell.*, 2018.

[262] D. Li, Y. Yang, Y.-Z. Song, and T. Hospedales, "Deeper, broader and artier domain generalization," in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 5542–5550.

[263] J. Devlin, R. Bunel, R. Singh, M. J. Hausknecht, and P. Kohli, "Neural program meta-induction," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, 2017, pp. 1–9.

[264] X. Si, Y. Yang, H. Dai, M. Naik, and L. Song, "Learning a meta-solver for syntax-guided program synthesis," *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–11.

[265] P. Huang, C. Wang, R. Singh, W. Yih, and X. He, "Natural language to structured query generation via meta-learning," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2018, pp. 732–738.

[266] Y. Xie, H. Jiang, F. Liu, T. Zhao, and H. Zha, "Meta learning with relational information for short sequences," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019.

[267] J. Gu, Y. Wang, Y. Chen, V. O. K. Li, and K. Cho, "Meta-learning for low-resource neural machine translation," in *Proc. Conf. Empirical Methods Nat. Lang. Process.*, 2018, pp. 3622–3631.

[268] Z. Lin, A. Madotto, C. Wu, and P. Fung, "Personalizing dialogue agents via meta-learning," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 5454–5459.

[269] J.-Y. Hsu, Y.-J. Chen, and H. yi Lee, "Meta learning for end-to-end low-resource speech recognition," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, 2019.

[270] G. I. Winata *et al.*, "Learning fast adaptation on cross-accented speech recognition," in *Proc. Interspeech*, 2020, pp. 1276–1280.

[271] O. Klejch, J. Fainberg, and P. Bell, "Learning to adapt: A meta-learning approach for speaker adaptation," in *Proc. Interspeech*, 2018, pp. 867–871.

[272] J. Tremblay *et al.*, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 969–977.

[273] A. Kar *et al.*, "Meta-sim: Learning to generate synthetic datasets," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 4551–4560.

[274] D. M. Metter, T. J. Colgan, S. T. Leung, C. F. Timmons, and J. Y. Park, "Trends in the US and Canadian pathologist workforces from 2007 to 2017," *JAMA Network Open*, vol. 2, 2019, Art. no. e194337.

[275] G. Maicas, A. P. Bradley, J. C. Nascimento, I. D. Reid, and G. Carneiro, "Training medical image analysis systems like radiologists," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Interv.*, 2018.

[276] B. D. Nguyen, T.-T. Do, B. X. Nguyen, T. Do, E. Tjiputra, and Q. D. Tran, "Overcoming data limitation in medical visual question answering," in *Proc. Med. Image Comput. Comput.-Assist. Interv.*, 2019.

[277] Z. Mirikharaji, Y. Yan, and G. Hamarneh, "Learning to segment skin lesions from noisy annotations," 2019, *arXiv: 1906.03815*.

[278] T. Miconi, J. Clune, and K. O. Stanley, "Differentiable plasticity: Training plastic neural networks with backpropagation," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3559–3568.

[279] T. Miconi, A. Rawal, J. Clune, and K. O. Stanley, "Backpropamine: Training self-modifying neural networks with differentiable neuromodulated plasticity," in *Proc. Int. Conf. Learn. Representations*, 2019.

[280] B. Dai, C. Zhu, and D. Wipf, "Compressing neural networks using the variational information bottleneck," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018.

[281] Z. Liu *et al.*, "Metapruning: Meta learning for automatic neural network channel pruning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3295–3304.

[282] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, Dec. 2017.

[283] Y. Jiang, H. Kim, H. Asnani, and S. Kannan, "MIND: Model independent neural decoder," 2019, *arXiv: 1903.02268*.

[284] B. Settles, "Active learning," *Synth. Lectures Artif. Intel. Mach. Learn.*, vol. 6, pp. 1–114, 2012.

[285] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Representations*, 2015.

[286] C. Yin, J. Tang, Z. Xu, and Y. Wang, "Adversarial meta-learning," 2018, *arXiv: 1806.03316*.

[287] M. Vartak, A. Thiagarajan, C. Miranda, J. Bratman, and H. Larochelle, "A meta-learning perspective on cold-start recommendations for items," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, 2017, pp. 1–11.

[288] H. Bharadhwaj, "Meta-learning for user cold-start recommendation," in *Proc. Int. Joint Conf. Neural Netw.*, 2019, pp. 1–8.

[289] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020.

[290] Z. Kang, K. Grauman, and F. Sha, "Learning with whom to share in multi-task feature learning," in *Proc. 25th Int. Conf. Mach. Learn.*, 2011, pp. 521–528.
[291] Y. Yang and T. Hospedales, "A unified perspective on multi-domain and multi-task learning," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–9.
[292] K. Allen, E. Shelhamer, H. Shin, and J. Tenenbaum, "Infinite mixture prototypes for few-shot learning," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 232–241.
[293] F. Pedregosa, "Hyperparameter optimization with approximate gradient," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 737–746.
[294] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.
[295] L. Metz, N. Maheswaranathan, C. D. Freeman, B. Poole, and J. Sohl-Dickstein, "Tasks, stability, architecture, and compute: Training more effective learned optimizers, and using them to train themselves," 2020, *arXiv: 2009.11243*.
[296] A. Shaban, C.-A. Cheng, N. Hatch, and B. Boots, "Truncated back-propagation for bilevel optimization," in *Proc. 22nd Int. Conf. Artif. Intell. Stat.*, 2019.
[297] J. Fu, H. Luo, J. Feng, K. H. Low, and T.-S. Chua, "DrMAD: Distilling reverse-mode automatic differentiation for optimizing hyperparameters of deep neural networks," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 1469–1475.
[298] S. Flennerhag, P. G. Moreno, N. Lawrence, and A. Damianou, "Transferring knowledge across learning processes," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–23.
[299] Y. Wu, M. Ren, R. Liao, and R. Grosse, "Understanding short-horizon bias in stochastic meta-optimization," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–6.
[300] L. Metz, N. Maheswaranathan, J. Nixon, D. Freeman, and J. Sohl-Dickstein, "Understanding and correcting pathologies in the training of learned optimizers," in *Proc. Int. Conf. Mach. Learn.*, 2019.

**Timothy Hospedales** is currently a professor at the University of Edinburgh, and a principal researcher with Samsung AI Research. His research interests include data efficient and robust learning-to-learn with diverse applications in vision, language, reinforcement learning, and beyond.

**Antreas Antoniou** is currently working toward the PhD degree, supervised by Amos Storkey, from the University of Edinburgh. His research contributions in meta-learning and few-shot learning are commonly seen as key benchmarks in the field. His research interest includes meta-learning better priors such as losses, initializations, and neural network layers, to improve few-shot and life-long learning.

**Paul Micaelli** is currently working toward the PhD degree, supervised by Amos Storkey and Timothy Hospedales, from the University of Edinburgh. His research interests include zero-shot knowledge distillation and on meta-learning over long horizons for many-shot problems.

**Amos Storkey** is currently a professor of machine learning and AI at the School of Informatics, University of Edinburgh. He leads a research team focused on deep neural networks, Bayesian and probabilistic models, efficient inference and meta-learning.