Meta-Learning in Neural Networks: A Survey

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, 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 a given task is 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 of the conventional challenges of deep learning, including data and computation bottlenecks, as well as the fundamental issue of generalization. In this survey we describe the contemporary meta-learning landscape. We first discuss definitions of meta-learning and position it with respect to related fields, such as transfer learning, multi-task 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 including few-shot learning, reinforcement learning and architecture search. 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 have seen great successes in a variety of fields [1]–[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], [7].

Meta-learning provides an alternative paradigm where a machine learning model gains experience over multiple learning episodes – often covering a distribution of related tasks – and uses this experience to improve its future learning performance. This 'learning-to-learn' [8] can lead to a variety of benefits such as data and compute efficiency, and it is better aligned with human and animal learning [9], where learning strategies improve both on a lifetime and evolutionary timescale [9]–[11]. Machine learning historically built models upon hand-engineered features, and feature choice was often the determining factor in ultimate model performance [12]–[14]. Deep learning realised the promise of joint feature and model learning [15], [16], providing a huge improvement in performance for many tasks [1], [3]. Meta-learning in neural networks can be seen as aiming to provide the next step of integrating joint feature, model, and *algorithm* learning. Neural network meta-learning has a long history [8], [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 providing better data efficiency, exploitation of prior knowledge transfer, and enabling unsupervised and self-directed learning. Successful applications have been demonstrated in areas spanning few-shot image recognition [19], [20], unsupervised learning [21], data efficient [22], [23] and self-directed [24] reinforcement learning (RL), hyperparameter optimization [25], and neural architecture search (NAS) [26]–[28].

Many different perspectives on meta-learning can be found in the literature. Especially as different communities use the term somewhat differently, it can be difficult to define. A perspective related to ours [29] views meta-learning as a tool to manage the 'no free lunch' theorem [30] and improve generalization by searching for the algorithm (inductive bias) that is best suited to a given problem, or family of problems. However, taken broadly, this definition can include transfer, multi-task, feature-selection, and model-ensemble learning, which are not typically considered as meta-learning today. Another perspective on meta-learning [31] broadly covers algorithm selection and configuration techniques based on dataset features, and becomes hard to distinguish from automated machine learning (AutoML) [32]. In this paper, we focus on contemporary *neural-network* meta-learning. We take this to mean algorithm or inductive bias search as per [29], but focus on where this is achieved by *end-to-end* learning of an *explicitly defined objective function* (such as cross-entropy loss, accuracy or speed).

This paper thus 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 in this fast moving field, and/or focus on algorithm selection for data mining [29], [31], [33]–[37], AutoML [32], or particular applications of meta-learning such as few-shot learning [38] or neural architecture search [39].

We address both meta-learning methods and applications. In particular, we first provide a high-level problem formalization which can be used to understand and position recent work. We then provide a new taxonomy of methodologies, in terms of meta-representation, meta-objective and meta-optimizer. We survey several of the popular and emerging application areas including few-shot, reinforcement learning, and architecture search; and position meta-learning with respect to related topics such

T. Hospedales is with Samsung AI Centre, Cambridge and University of Edinburgh. A. Antoniou, P. Micaelli and Storkey are with University of Edinburgh. Email: {t.hospedales,a.antoniou,paul.micaelli,a.storkey}@ed.ac.uk.

as transfer learning, multi-task learning and AutoML. 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 the contemporary neural-network literature. In this section, we introduce our definition and key terminology, which aims to be useful for understanding a large body of literature. We then position meta-learning with respect to related topics such as transfer and multi-task learning, hierarchical models, hyper-parameter optimization, lifelong/continual learning, and AutoML.

Meta-learning is most commonly understood as *learning to learn*, which refers to the process of improving a learning algorithm over multiple learning episodes. In contrast, conventional ML considers the process of improving model predictions over multiple data instances. During **base learning**, an *inner* (or *lower*, *base*) learning algorithm solves a *task* such as image classification [15], defined by a dataset and objective. During **meta-learning**, an *outer* (or *upper*, *meta*) algorithm updates the inner learning algorithm, such that the model learned by the inner algorithm improves an outer objective. For instance this objective could be generalization performance or learning speed of the inner algorithm. Learning episodes of the base task, namely (base algorithm, trained model, performance) tuples, can be seen as providing the instances needed by the outer algorithm in order to learn the base learning algorithm.

As defined above, many conventional machine learning practices such as random hyper-parameter search 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 is tuned to perform well on new tasks sampled from this family. This can be a particularly powerful technique to improve data efficiency when learning new tasks. 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 θ, by solving:

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

where $\mathcal{L}$ is a loss function that measures the match between true labels and those predicted by $f_\theta(\cdot)$. We include condition ω to make explicit the dependence of this solution on factors such as choice of optimizer for θ or function class for f, which we denote by ω. Generalization is then measured by evaluating a number of test points with known labels.

The conventional assumption is that this optimization is performed *from scratch* for every problem $\mathcal{D}$; and furthermore that ω is pre-specified. However, the specification ω of '*how to learn*' θ can dramatically affect generalization, data-efficiency, computation cost, and so on. Meta-learning addresses improving performance by learning the learning algorithm itself, rather than assuming it is pre-specified and fixed. This is often (but not always) achieved by revisiting the first assumption above, and learning from a distribution of tasks rather than from scratch.

Meta-Learning: Task-Distribution View Meta-learning aims to improve performance by learning '*how to learn*' [8]. In particular, the vision is often to learn a general purpose learning algorithm, that can generalize across tasks and ideally enable each new task to be learned better than the last. As such ω specifies 'how to learn' and is often evaluated in terms of performance over a distribution of tasks $p(\mathcal{T})$. Here we loosely define a task to be a dataset and loss function $\mathcal{T} = \{\mathcal{D}, \mathcal{L}\}$. Learning how to learn thus becomes

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

where $\mathcal{L}(\mathcal{D}; \omega)$ measures the performance of a model trained using ω on dataset $\mathcal{D}$. The knowledge ω of 'how to learn' is often referred to as *across-task* knowledge or *meta-knowledge*.

To solve this problem in practice, we usually assume access to a *set* of source tasks sampled from $p(\mathcal{T})$, with which we learn ω. 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. Denoting the meta-knowledge as ω, the **meta-training** step of 'learning how to learn' is then:

$$\omega^* = \arg\max_\omega \log p(\omega | \mathscr{D}_{source}) \tag{3}$$

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 to train the base model on each previously unseen target task i:

$$\theta^{*\,(i)} = \arg\max_\theta \log p(\theta | \omega^*, \mathcal{D}^{train\,(i)}_{target}) \tag{4}$$

In contrast to conventional learning in Eq. 1, learning on the training set of a target task i now benefits from meta-knowledge ω about the algorithm to use. This could take the form of an estimate of the initial parameters [19], in which case ω and θ are the same sized objects referring to the same quantities. However, ω can more generally encode other objects such as an entire learning model [40] or optimization strategy [41]. Finally, 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. In terms of meta-learning as inductive-bias learning [29], meta-overfitting corresponds to learning an inductive bias ω that constrains the hypothesis space of θ 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 arguably only accurate for the optimizer-based methods (see section 3.1), it is helpful to visualize the mechanics of meta-learning more generally. Bilevel optimization [42] refers to a hierarchical optimization problem, where one optimization contains another optimization as a constraint [25], [43]. Using this notation, meta-training can be formalised as follows:

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

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

where $\mathcal{L}^{meta}$ and $\mathcal{L}^{task}$ refer to the outer and inner objectives respectively, such as cross entropy in the case of few-shot classification. A key characteristic of the bilevel paradigm is the leader-follower asymmetry between the outer and inner levels respectively: the inner level optimization Eq. 6 is conditional on the learning strategy ω defined by the outer level, but it cannot change ω during its training.

Here ω could indicate an initial condition in non-convex optimization [19], a hyper-parameter such as regularization strength [25], or even a parameterization of the loss function to optimize $\mathcal{L}^{task}$ [44]. Section 4.1 discusses the space of choices for ω in detail. The outer level optimization trains the learning strategy ω such that it produces models $\theta^{*\,(i)}(\omega)$ that perform well on their validation sets after training. Section 4.2 discusses how to optimize ω in detail. Note that while $\mathcal{L}^{meta}$ can measure simple validation performance, we shall see that it can also measure more subtle quantities such as learning speed and model robustness, as discussed in Section 4.3.

Finally, we note that the above formalization of meta-training uses the notion of a distribution over tasks, and using M samples from that distribution. While this is powerful, and widely used 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, we can split the training set to get validation data such that $\mathscr{D}_{source} = (\mathcal{D}^{train}_{source}, \mathcal{D}^{val}_{source})$ for meta-training, and for meta-testing we can use $\mathscr{D}_{target} = (\mathcal{D}^{train}_{source} \cup \mathcal{D}^{val}_{source}, \mathcal{D}^{test}_{target})$. We still learn ω over several episodes, and one could consider that $M = Q = 1$, although 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 [45].

$$\min_{\omega} \mathop{\mathbb{E}}_{\substack{\mathcal{T}\sim p(\mathcal{T}) \\ (\mathcal{D}^{tr}, \mathcal{D}^{val}) \in \mathcal{T}}} \sum_{(\mathbf{x}, y) \in \mathcal{D}^{val}} \left[(\mathbf{x}^T \mathbf{g}_{\omega}(\mathcal{D}^{tr}) - y)^2 \right] \quad (7)$$

Here we can see that we meta-train by optimising over a distribution of tasks. For each task a train and validation (aka query and support) set is drawn. The train set $\mathcal{D}^{tr}$ is embedded into a vector $\mathbf{g}_{\omega}$ which defines the linear regression weights to predict examples $\mathbf{x}$ drawn from the test set. Optimising the above objective thus 'learns how to learn' by training the function $\mathbf{g}_{\omega}$ to instantiate a learning algorithm that maps a training set to a weight vector. Thus if a novel meta-test task $\mathcal{T}^{te}$ is drawn from $p(\mathcal{T})$ we might also expect $\mathbf{g}_{\omega}$ to provide a good solution. Different methods in this family vary in the complexity of the predictive model used (parameters $\mathbf{g}$ that they instantiate), and how the support set is embedded (e.g., by simple pooling, CNN or RNN).

2.2 Historical Context of Meta-Learning

Meta-learning first appears in the literature in 1987 in two separate and independent pieces of work, by J. Schmidhuber and G. Hinton [17], [46]. Schmidhuber [17] set the theoretical framework for a new 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 further proposed that the model itself can be learned using evolutionary algorithms. Hinton *et al*. [46] proposed the usage of two weights per neural network connection instead of one. The first weight is the standard *slow-weight* which acquires knowledge slowly (called *slow-knowledge*) over optimizer updates, whereas the second weight or *fast-weight* acquires knowledge quickly (called *fast-knowledge*) during inference. The fast weight's responsibility is to be able to *deblur* or recover slow weights learned in the past, that have since been forgotten due to optimizer updates. Both of these papers introduce fundamental concepts that later on branch out and give rise to contemporary meta-learning.

After the introduction of meta-learning, one can see a rapid increase in the usage of the idea in multiple different areas. Bengio *et al*. [47], [48] proposed systems that attempt to meta-learn biologically plausible learning rules. Schmidhuber *et al*.continued to explore self-referential systems and meta-learning in subsequent work [49], [50]. S. Thrun *et al*.coined the term *learning to learn* in [8] as an alternative to meta-learning and proceeded to explore and dissect available literature in meta-learning in search for a general meta-learning definition. Proposals for training meta-learning systems using gradient descent and back-propagation were first made in 2001 [51], [52]. Additional overviews of the meta-learning literature shortly followed [29]. Meta-learning was first used in reinforcement learning in work by Schweighofer *et al*. [53] after which came the first usage of meta-learning in zero-shot learning by Larochelle *et al*. [54]. Finally in 2012 Thrun *et al*. [8] re-introduced meta-learning in the modern era of deep neural networks, which

marked the beginning of modern meta-learning of the type discussed in this survey.

Meta-Learning is also closely related to methods for hierarchical and multi-level models in statistics for grouped data. In such hierarchical models, grouped data elements are modelled with a *within-group* model and the differences between each group is modelled with an *between-group* model. Examples of such hierarchical models in the machine learning literature include topic models such as Latent Dirichlet Allocation [55] and its variants. In topic models, a model for a new document is learnt from the document's data; the learning of that model is guided by the set of topics already learnt from the whole corpus. Hierarchical models are discussed further in Section 2.3.

2.3 Related Fields

Here we position meta-learning against related areas, which is often the source of confusion in the literature.

Transfer Learning (TL) TL [34] uses past experience of a source task to improve learning (speed, data efficiency, accuracy) on a target task – by transferring a parameter prior, initial condition, or feature extractor [56] from the solution of a previous task. TL refers both to this endeavour to a problem area. In contemporary neural network context it often refers to a particular methodology of parameter transfer plus optional fine tuning (although there are numerous other approaches to this problem [34]).

While TL can refer to a problem area, meta-learning refers to a methodology which can be used to improve TL as well as other problems. TL as a methodology is differentiated to meta-learning as 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 how well the prior performs when helping to learn a new task, as illustrated, e.g., 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 tasks have the same classes but the input distribution of the target task is shifted with respect to the source task [34], [57], leading to reduced model performance upon transfer. 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 methods have been studied [34], [57], [58] to transfer knowledge and boost performance in the target domain. However, as for TL, vanilla DA and DG are differentiated in that there is no meta-objective that optimizes 'how to learn' across domains. Meanwhile, meta-learning methods can be used to perform both DA and DG, which we cover in section 5.9.

Continual learning (CL) Continual and lifelong learning [59], [60] refer 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. It is related insofar as working with a task distribution, and that 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 [61]–[63].

Multi-Task Learning (MTL) aims to jointly learn several related tasks, and benefits from the effect regularization due to parameter sharing and of the diversity of the resulting shared representation [64]–[66]. 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 such as learning rate or regularization strength can be included in the definition of 'how to learn'. Here we focus on HO tasks defining a meta objective that is trained end-to-end with neural networks. This includes some work in HO, such as gradient-based hyperparameter learning [67] and neural architecture search [26]. But we exclude other approaches like random search [69] and Bayesian Hyperparameter Optimization [70], which are rarely considered to be meta-learning.

Hierarchical Bayesian Models (HBM) involve Bayesian learning of parameters θ under a prior $p(\theta|\omega)$. The prior is written as a conditional density on some other variable ω 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 θ_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 ω 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 ω: $P(\omega|\mathcal{D}) \sim p(\omega) \prod_{i=1}^{M} \int d\theta_i p(\mathcal{D}_i|\theta_i)p(\theta_i|\omega)$. The ease of doing the marginalisation depends on the model: in some (e.g. Latent Dirichlet Allocation [55]) the marginalisation is exact due to the choice of conjugate exponential models, in others (see e.g. [71]), 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, in that they provide a modeling rather than an algorithmic framework for understanding the meta-learning process. In practice, prior work in Bayesian hierarchical models has typically focused on learning simple tractable models θ; most meta-learning work however 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 [72].

AutoML: AutoML [31], [32] is a rather broad umbrella

for approaches aiming to automate parts of the machine learning process that are typically manual, such as data preparation and cleaning, feature selection, algorithm selection, hyper-parameter tuning, architecture search, and so on. AutoML often makes use of numerous heuristics outside the scope of meta-learning as defined here, and addresses tasks such as data cleaning that are less central to meta-learning. However, AutoML sometimes makes use of meta-learning as we define it here in terms 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 [73], [74] 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 ω required to improve optimization performance. The most famous of these is perhaps MAML [19], where the meta-knowledge ω is the initialization of the model parameters in the inner optimization, namely θ_0. The goal is to learn θ_0 such that a small number of inner steps on a small number of train instances produces a classifier that performs well on validation data. This is also performed by gradient descent, differentiating through the updates to the base model. More elaborate alternatives also learn step sizes [75], [76] or train recurrent networks to predict steps from gradients [41], [77], [78]. Meta-optimization by gradient leads to the challenge of efficiently evaluating expensive second-order derivatives and differentiating through a graph of potentially thousands of inner optimization steps (see Section 6). For this reason it is often applied to few-shot learning where few inner-loop steps may be sufficient.

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 [41], [51], convolutional networks [40] or hypernetworks [79], [80] that embed training instances and labels of a given task to define a predictor that inputs testing example and predicts its label. 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 ω containing the CNN, RNN or hypernetwork parameters. The outer and inner-level optimizations are tightly coupled as ω directly specifies θ. Memory-augmented neural networks [81] use an explicit storage buffer and can also be used as a model-based algorithm [82], [83]. It has been observed that model-based approaches are usually less able to generalize to out-of-distribution tasks than optimization-based methods [84]. Furthermore, while they are often very good at data efficient few-shot learning, they have been criticised for being asymptotically weaker [84] as it isn't clear that black-box models can successfully 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 methods such as siamese networks [85], matching networks [86], prototypical networks [20], relation networks [87], and graph neural networks [88]. Here the outer-level learning corresponds to metric learning (finding a feature extractor ω that encodes the data to a representation suitable for comparison). As before ω 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. In the following subsections we therefore present a new cross-cutting breakdown of meta-learning methods.

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 representation of meta-knowledge ω. This could span an estimate of model parameters [19] used for optimizer initialization, to readable code in the case of program induction [89]. Note that the base model representation θ is usually application-specific, for example a convolutional neural network (CNN) [1] in the case of computer vision.

Meta-Optimizer ("How?") The second axis is the choice of optimizer to use for the outer level during meta-training (see Eq. 5)[1]. The outer-level optimizer for ω can take a variety of forms from gradient-descent [19], to reinforcement learning [89] and evolutionary search [23].

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], [40], fast many-shot optimization [89], [91], or robustness to domain-shift [44], [92], label noise [93], and adversarial attack [94].

4 Survey: Methodologies

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

1. In contrast, the inner level optimizer for θ (Eq. 6) may be specified by the application at hand (e.g., gradient-descent supervised learning of cross-entropy loss in the case of image recognition [1], or policy-gradient reinforcement learning in the case of continuous control [90]).

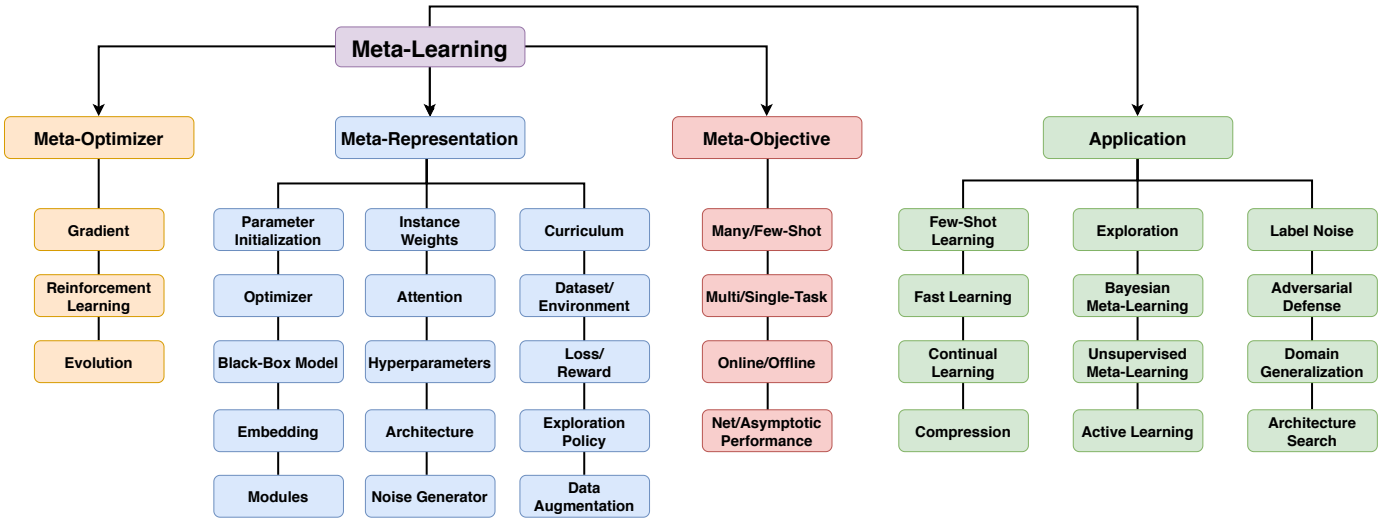

Fig. 1. Overview of the meta-learning landscape including algorithm design (meta-optimizer, meta-representation, meta-objective), and applications.

4.1 Meta-Representation

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

Parameter Initialization In this first family of methods ω corresponds to the initial parameters of a neural network. In MAML [19], [95], [96] these are interpreted as initial conditions of the inner optimization. A good initialization is just a few gradient steps away from a solution to any task $\mathcal{T}$ drawn from $p(\mathcal{T})$. These approaches are widely used for few-shot learning, where target problems can be learned without over-fitting using few examples, given such a carefully chosen initial condition. 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 [74], [97], by layer [79], [97], [98], or by separating out scale and shift [99]. While inner loop initialization is a popular and effective choice of meta-representation, a key debate here 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 fairly narrow distributions $p(\mathcal{T})$. This has led to variants that model mixtures over multiple initial conditions [97], [100], [101].

Optimizer The above parameter-centric methods usually rely on existing optimizers such as SGD with momentum or Adam [102] to refine the initialization when given some new task. Rather than relying on hand-designed optimizers, optimizer-centric approaches [41], [77], [78], [91] focus on learning the inner optimizer by training a function that takes as input optimization states such as θ and $\nabla_\theta \mathcal{L}^{task}$ and produces the optimization step to take at each base learning iteration. The trainable component ω can span simple hyper-parameters such as a fixed step size [75], [76] to more sophisticated pre-conditioning matrices [103]. Ultimately ω can be used to define a full gradient-based optimizer in the sense of defining a complex non-linear transformation of the input gradient and other metadata [41], [78], [89], [91]. The parameters to learn here can potentially be few if the optimizer is applied coordinate-wise across weights [78]. 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 [41], [75]. Optimizer learning methods have both been applied to for few-shot learning [41] and to accelerate and improve many-shot learning [78], [89], [91]. Finally, one can also meta-learn black-box zeroth-order optimizers [104] that only require evaluations of $\mathcal{L}^{task}$ rather than optimizer states such as gradients. These have been shown [104] to be competitive with conventional Bayesian Optimization [70] alternatives.

Black-Box Models (Recurrent, Convolutional, HyperNetwork) Another family of models trains learners ω 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 gradient [78] (or zero-order [104]) *iterative* optimization of θ. These correspond to the black-box model-based learning in the conventional taxonomy (Section 3.1). Embedding the support set is commonly achieved by recurrent networks [51], [105], [106] or convolution [40].

These methods have strong connections to Hypernetworks. Hypernetworks [107], [108] are networks that generate the weights of another neural network conditioned on some embedding – and are often used for compression or multi-task learning. Hypernetworks can also be used to synthesize predictive models by conditioning on an embedding of the source (aka. support) dataset [97], [109]. In this case ω is the weight synthesis hypernetwork that produces θ given a support set in a feed-forward pass. Finally, memory-augmented neural networks have the ability to remember old data and assimilate new data quickly, and typically fall in the black-box model category as well. In [82], the authors adapt Neural Turing Machines [81] to the meta-learning setting by changing their memory retrieval mechanism. Meta networks [83] then improve on this model by combining fast weights (predicted per task by a network)

and slow weights (trained with REINFORCE across tasks) to access memory. We note that some methods implement both model-based and initial-condition [97] or optimizer-centric [98] meta-learning in a single framework.

Embedding Functions (Metric Learning) This category of methods are inspired by metric-learning approaches in conventional machine learning, and are therefore categorised as such in the conventional taxonomy (Section 3.1). They are mainly applied to few-shot learning. Here the meta-optimization process learns an embedding network ω that transforms raw inputs into a representation suitable for recognition by simple similarity comparison between query and support instances [20], [79], [86], [110] (e.g., with cosine similarity or euclidean distance).

However, metric-learning methods can be seen to be a special case of the feed-forward black-box models above. This is clearly the case for methods that produce logits based on inner product between the embeddings of support and query images x_s and x_q as $g_\omega^T(x_q)g_\omega(x_s)$ [79], [110]. Here the support image generates weights to interpret the query example, making it a special case of a BBM 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 [98], [111] or learning a more elaborate comparison metric [87], [88].

Losses and Auxiliary Tasks Analogously to the meta-learning approach to optimizer design, these approaches aim to learn the inner task-loss $\mathcal{L}_\omega^{task}(\cdot)$ for the base model. Loss-learning approaches typically define a small neural network that inputs quantities that are typically inputs to losses (e.g., predictions, features, or model parameters) and outputs a scalar to be treated as a loss by the inner (task) optimizer. This has potential benefits such as leading to a learned loss that is *easier* to optimize (e.g., less local minima) than commonly used ones [23], [112], [113], leads to faster learning with improved generalization [45], [114], [115], or one whose minima correspond to a model more robust to domain shift [44]. Furthermore, loss learning methods have also been used to learn to learn from unlabeled instances [98], [116]. Other applications include learning $\mathcal{L}_\omega^{task}()$ as a differentiable approximation to a true non-differentiable task loss such as area under precision recall curve [117], [118].

Loss learning also arises in generalizations of self-supervised [119], [120] or auxiliary task [121] learning. In these problems unsupervised predictive tasks (such as colourising pixels in vision [119], or simply changing pixels in RL [121]) are defined and optimized in a multi-task manner with the main task, but with the aim of improving the representation in support of 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., ω is a per-auxiliary task weight [68]. More generally, one can meta-learn an auxiliary task generator that annotates examples with auxiliary labels for the main multi-task model to predict [122].

Architectures Architecture discovery has always been an important area in neural networks [39], [123], 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 use RL and LSTMs to learn to generate a description for a good architecture [28]. Evolutionary Algorithms [27] were also used in an attempt to 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 [26] 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. ω). During meta-test, the architecture is discretized by only keeping the layers corresponding to the highest coefficients. The coefficients are learned greedily by alternating a single inner step and a single outer step to update architecture coefficients and network weights. Since DARTS is still relatively slow and of limited accuracy, recent efforts have focused on making architecture learning more efficient through better differentiable approximations [124], learning easy to adapt initializations [125], or architecture priors [126]. More details on neural architecture search can be found in Section 5.4.

Attention Modules Attention mechanisms have been shown to improve generalization performance and interpretability. Such mechanisms have also formed part of the meta-representation of various meta-learning models. For example, they have been used as comparators of support and target set items for metric-based transductive meta-learners [127], as well as feature extractors to prevent catastrophic forgetting in few-shot continual learning [128]. More recently, attention was also used to summarize the distribution of an incoming text classification task [129].

Modules Modular meta-learning [130], [131] assumes that the task agnostic knowledge ω defines a set of modules, which are re-composed in a task specific manner defined by θ 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 [65], [66], [132].

Hyper-parameters In these methods ω includes hyperparameters of the base learning algorithm such as regularization strength [25], per-parameter regularization [92], task-relatedness in multi-task learning [67], or sparsity strength in data cleansing [67]. Note that hyperparameters such as step size and direction [75], [76] can be seen as part of definition of the optimizer, and as such leads to an overlap between hyper-parameter and optimizer learning categories.

Data Augmentation In supervised learning it is common to improve generalization by synthesizing more training data through label-preserving transformations on the existing data. The data augmentation operation is wrapped up in optimization steps of the inner problem Eq. 6, and is conventionally hand-designed. However, when ω defines the data augmentation strategy, it can be learned by the outer optimization in Eq. 5 in order to maximize validation performance [133]. Since augmentation operations are typically non-differentiable, this requires reinforcement learning [133], discrete gradient-estimators [134], or evolutionary [135] methods. An open question is whether powerful GAN-based data augmentation methods [136] can be used in

inner-level learning and optimized in outer-level learning.

Minibatch Selection, Sample 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 [137] exist to improve on classic randomly-sampled minibatches. Meta-learning approaches to mini-batch selection define ω as an instance selection probability [138] or small neural network that picks or excludes instances [139] for inclusion in the next minibatch, while the meta-loss can be defined as the learning progress of the base model given the defined mini-batch selector.

Such selection methods can also provide a way to automate the learning of a *curriculum*. In conventional machine learning, curricula are sequences of data or concepts to learn that are hand-designed to produce better performance than learning items in a random order [140], for instance by focusing on instances of the right difficulty while rejecting too hard or too easy (already learned) instances. Meta-learning has the potential to automate this process and select examples of the right difficulty by defining the teaching policy as the meta-knowledge and training it to optimize the progress of the student [139], [141].

Related to mini-batch selection policies are methods that learn *per-sample* loss weights ω for the training set [142], [143]. This can be used to learn under label-noise by discounting noisy samples [142], [143], discount outliers [67], or correct for class imbalance [142].

Datasets, Labels and Environments Perhaps the strangest choice of 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. 2-3). However, it can be easily understood in the bilevel view of Eqs. 5-6. If the validation set in the upper optimization is real and fixed, and a train set in the lower optimization is paramaterized by ω, the training dataset can be tuned by meta-learning to optimize validation performance.

In dataset distillation [144], [145], 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 the input images x for fixed labels y, one can also learn the input labels y for fixed images x. This can be used in semi-supervised learning, for example to directly learn the unlabeled set's labels to optimize validation set performance [146], or to train a label generation function [147].

In the case of sim2real learning [148] in computer vision or reinforcement learning, one uses an environment simulator to generate data for training. In this case one can also train the graphics engine [149] or simulator [150] 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 ω discussed above are parameter vectors of functions that process or generate data. However a few of the representations mentioned are transductive in the sense that the ω literally corresponds to data points [144], labels [146], or per-sample weights [142]. This means that the number of parameters in ω 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 [145], this property may ultimately limit their scalability.

Distinct from a transductive representation are methods that are transductive in the sense that they are designed to operate on the query instances as well as support instances [98], [122].

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 ω parameterizes a neural network [78], are more commonly studied and make up the majority of studies cited above. However, meta-learning with symbolic representations is also possible, where ω represents symbolic functions that are human readable as a piece of program code [89], comparable to Adam [102]. Rather than neural loss functions [44], one can train symbolic losses ω that are defined by an expression comparable to cross-entropy [115]. One can also meta-learn new symbolic activations [151] 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 ω typically uses RL [89] or evolutionary algorithms [115]. However, symbolic representations may have an advantage [89], [115], [151] in their ability to generalize across task families. I.e., to span wider distributions $p(\mathcal{T})$ with a single ω during meta-training, or to have the learned ω generalize to an out of distribution task during meta-testing (see Section 6).

4.2 Meta-Optimizer

Given a choice of which facet of the learning strategy to optimize (as summarised above), the next axis of meta-learner design is actual outer (meta) optimization strategy to use for tuning ω.

Gradient A large family of methods use gradient descent on the meta parameters ω [19], [41], [44], [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 θ, $d\mathcal{L}^{meta}/d\omega = (d\mathcal{L}^{meta}/d\theta)(d\theta/d\omega)$. These methods are potentially the most efficient as they exploit analytical gradients of ω. However key challenges include: (i) Efficiently differentiating through long computation graphs where the inner optimization uses many steps, for example through careful design of auto-differentiation algorithms [25], [178] and implicit differentiation [145], [153], [179], and dealing tractably with the required second-order gradients [180]. (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, ω, or $\mathcal{L}^{task}$ include discrete or other non-differentiable operations.

Meta-Representation	Meta-Optimizer		
	Gradient	RL	Evolution
Initial Condition	[19], [75], [84], [99], [152], [152]–[154]	[155]–[157] [19], [61], [62]	[158], [159]
Optimizer	[78], [91] [41], [75], [103], [160], [161]	[77], [89]	
Hyperparam	[25]	[162], [163]	[159] [164]
Black-box model	[40], [82], [165], [166]	[22], [106], [109]	
Metric	[20], [86], [87]		
Loss/Reward	[44], [92] [118]	[117] [113], [167]	[115] [23] [164]
Architecture	[26] [125]	[28]	[27]
Exploration Policy		[24], [168]–[172]	
Dataset/Environment	[144] [146]	[149]	[150]
Instance Weights	[142]		
Feature/Metric	[20], [86]–[88]		
Data Augmentation/Noise Generator	[134] [173] [174]	[133]	[135]
Modules	[130], [131]		
Annotation Policy	[175], [176]	[177]	

TABLE 1
Classification of research papers according to our taxonomy. We use color to indicate salient meta-objective or application goal focus. We focus on the main goal of each paper for simplicity. The color code is as follows: sample efficiency (red), learning speed (green), asymptotic performance (purple), cross-domain (blue).

Reinforcement Learning When the base learner includes non-differentiable steps [133], or the meta-objective $\mathcal{L}^{meta}$ is itself non-differentiable [117], many methods [22] 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], [123], [181]. Many evolutionary algorithms have strong connections to reinforcement learning algorithms [182]. 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 [181] : (i) They can optimize any type of base model and meta-objective with no requirement on differentiability. (ii) They do not rely on backpropagation, which rectifies both gradient degradation issues and avoids the cost of high-order gradient computation required by conventional gradient-based methods above. (iii) They are highly parallelizable, making meta-training more easily scalable. (iv) By maintaining a diverse population of solutions, they can avoid local minima that plague gradient-based methods [123]. However, they have a number of disadvantages: (i) The size of the population required to train a model increases rapidly with the number of learn-able parameters. (ii) They can be sensitive to the mutation strategy (e.g., magnitude and direction of noise) and thus 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 [23], [158] (where models are typically smaller, and inner optimizations are long and non-differentiable). However they have also been applied to learn learning rules [183], optimizers [184], architectures [27], [123] and data augmentation strategies [135] in supervised learning. They are also particularly important in learning human interpretable symbolic meta-representations [115].

4.3 Meta-Objective and Episode Design

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

Many vs 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], [89], [91] or few- [19], [41] examples per-task.

Fast Adaptation vs 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 [76], [89], [91]. Most RL applications also use this latter setting.

Multi vs 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], [20], [44]. 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 [67], [78], [162], [167], [168], [185].

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 vs Offline While the classic meta-learning pipeline defines the meta-optimization as an outer-loop of the inner base learner [19], [78], some studies have attempted to preform meta-optimization *online* within a single base learning episode [44], [167], [185], [186]. In this case the base model θ and learner ω co-evolve during a single episode. Since there is now no set of learning operations 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 [44], [92]; simulate network compression such as quantization [187] between training and validation to meta-optimize for good network compressibility; provide noisy labels during meta-training to optimize for label-noise robustness [93], or generate an adversarial validation set to meta-optimize for adversarial defense [94]. These opportunities are explored in more detail in the following applications section.

5 Applications

In this section we discuss the ways in which meta-learning has been exploited – in terms of application domains such as computer vision and reinforcement learning and cross-cutting problems such as architecture search, hyperparameter optimization, Bayesian and unsupervised meta-learning.

5.1 Computer Vision and Graphics

Computer vision is one of the major consumer domains of meta-learning techniques. This is particularly driven by the impact of meta-learning 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 network models [1], [15], where data volume is often the dominant factor in performance [188], and training large models with small datasets leads to overfitting or even non-convergence. Meta-learning-based few-shot learning methods train algorithms that enable powerful deep networks to successfully learn on small datasets. There are numerous vision problems where meta-learning helps in the few-shot setting, and we provide a non-exhaustive summary as follows.

Classification The most common application of meta-learning thus far is few-shot multi-class classification in image recognition, where the inner and outer loss functions are typically the cross entropy over training and validation data respectively [19], [20], [41], [73], [75], [76], [86], [88], [97], [98], [101], [161], [189]–[193]. Optimizer-centric [19], black-box [40], [79] and metric learning [86]–[88] models have all been considered. Relevant benchmarks are covered in Section 5.1.2.

This line of work has led to a steady improvement in performance compared to early methods [19], [85], [86]. However, performance is still far behind that of fully supervised methods, so there is more work to be done. Current research issues include few-shot models with better cross-domain generalization [173], recognition within the joint label space defined by meta-train and meta-test classes [80], and incremental addition of new few-shot classes [128], [165].

Object Detection Building on the rapid progress in few-shot classification, recent work has also generalized to few-shot object *detection* [165], [194], often using feed-forward hypernetwork-based approaches to embed support set images and synthesize final layer classification weights in the base model.

Landmark Prediction The goal of landmark estimation is to find the location of skeleton key points within an image, such as such as joints in human or robot images. This is typically formulated as an image-conditional regression problem. For example, a MAML-based model was shown to work for human pose estimation [195], modular-meta-learning was successfully applied to robotics [130], while a hypernetwork-based model was applied to few-shot clothes fitting for novel fashion items [165].

Object Segmentation Few-shot object segmentation is important due to the cost of obtaining pixel-wise labeled images in this domain. Meta-learning methods based on hypernetworks have been shown to work in the one-shot regime [196], and performance was later improved by adapting prototypical networks [197]. Other models tackle cases where segmentation has low density [198].

Image Generation In [199] an amortized probabilistic meta-learner is used to generate multiple views of an object from just a single image, and talking faces are generated from little data by learning the initialization of an adversarial model for quick adaptation [200].

Video Synthesis In [201], the authors propose meta-learning a weight generator that takes as input a few frames and generates a network that can achieve strong results in video synthesis for the given task.

Density Estimation Since autoregressive models typically require large depths to capture the distribution of the data, the few-shot regime lands itself to overfitting and is particularly challenging. Meta-learning coupled with an attention mechanism has shown to enable PixelCNNs to shine in such a regime [202].

5.1.2 Few-Shot Learning Benchmarks

Progress in AI and machine learning is often measured by, and spurred by, well designed benchmarks [203]. In machine learning, a benchmark is composed by a dataset and a task that a model should perform well on, while generalising from training to testing instances from within that dataset. In meta-learning, benchmark design is more complex, since we are often dealing with a learner that should be (meta) trained on a set of tasks, after which it should generalize to learning on previously unseen tasks. Benchmark design is thus more complex due to the need to define families of tasks from which meta-training and meta-testing tasks can be drawn. In this section we will outline the main few-shot benchmarks.

Benchmarks and Setup

Most FSL studies consider the *set-to-set* setting, where a model must learn to do well in a large number of small few-shot learning tasks. Each such task is composed of a small training set (referred to as a *support set*) consisting of a number of a few labelled examples from a number of classes and a small validation set (referred to as a *query set*) consisting of previously unseen instances of the same classes contained in the support set. A learner should be able to extract task-specific information from a support set, and then generate a model that can perform well on the query set. Across-task knowledge can be learned by learning the learner that can do this task well. We usually use the notation of N-way K-shot task, to indicate a task with N classes per task, and K samples per class.

There are a number of established FSL datasets that are used in this setting, such as miniImageNet [41], tieredImageNet [204], SlimageNet [205], CUB-200 [110] and Omniglot [86]. These benchmarks re-purpose prior datasets with rather large numbers of classes by breaking them into many smaller (lower 'way') recognition problems to define a task distribution for benchmarking meta-training and meta-testing.

Dataset Diversity, Bias and Generalization While the above approach is convenient to generate enough tasks for training and evaluation, it suffers from a lack of diversity (narrow $p(\mathcal{T})$) which makes it hard for performance on these benchmarks to reflect performance on real-world few shot task. For example, switching between different kinds of animals in miniImageNet or birds in CUB is a rather weak test of transferability. Ideally we would like to span more diverse categories and types of images (satellite, medical, agricultural, underwater, etc); and even be robust to domain-shifts between meta-train and meta-test tasks.

There is much 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 [206], as is generalising to out-of-sample data [44], [92]. In particular, the performance of meta-learners has been shown to drop drastically when introducing a domain shift between the source and target task distributions [110]. This motivates the recent Meta-Dataset [207] and CVPR cross-domain few-shot challenge [208]. 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, [208] challenges methods to generalize from the everyday images of ImageNet 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 [173]. Generalization issues also arise in applying models to data from under-represented countries [209]. Another recent dataset that could facilitate research in few-shot learner generalization is [210], which offers samples across environments from simulation, to high definition simulation and real-world.

Real-World Few-Shot Recognition The most common few-shot problem setting is N-way recognition among the classes in the support set [19], [20]. However, this may not be representative of practical application requirements where recognition among both the source and target is of interest at testing-time. This generalized few-shot setting is considered in an increasing number of studies [128], [165], [211]. In a generalized few-shot setting, other goals include efficient incremental enrolment of novel few-shot classes without forgetting the base classes or re-accessing the source data [128], [165]. Other real-world challenges include scaling up few-shot learning beyond the widely studied $N = 1 \ldots 20$-way recognition setting, at which point the popular and effective metric learner method family [20], [87] begin to struggle.

Few-Shot Object Detection The few studies [165] on few-shot detection have thus far re-purposed standard detection datasets such as COCO and Pascal VOC. However these only offer a few classes for meta-training/testing compared to classification benchmarks, and so more benchmarks are needed.

Regression Benchmarks Unfortunately there has been less work on establishing common benchmarks for few-shot regression than for classification. Toy problems such as 1d sinusoidal regressions have been proposed in [19], [212]. Image completion by regressing from pixel coordinate to RGB value have been considered [166], some work regresses to interest points in human pose and fashion [165], while [213] considers the task of face pose regression, with additional occlusion to introduce ambiguity. Overall, these tasks are all scattered and the meta-learning community has yet to reach consensus on regression benchmarks.

Non meta-learning few-shot methods Recently, a number of non meta-learning methods have obtained competitive performance on few-shot benchmarks, questioning the need for learning to learn in this setting. It was shown in [110] that training on all the base tasks at once and finetuning on the target tasks is a stronger baseline than initially reported, mainly because augmentation was unfairly omitted. Furthermore, using a deeper backbone may shrink the performance gap between common meta-learning methods, and the baseline can outperform these methods for larger domain shifts between source and target task distributions [207] – although more recent meta-learning methods obtained good performance in this setting [173]. On a similar theme, [214] show that simple feature transformations like L2-normalization can make a nearest neighbour classifier competitive without meta-learning. Thus the debate here is ongoing, but overall carefully implemented baselines and more diverse datasets are important, as well as maintaining

fair and consistent best practice for all methods.

5.2 Meta Reinforcement Learning and Robotics

Reinforcement learning is typically concerned with learning control policies that enable an agent to obtain high reward in achieving a sequential action task within an environment, in contrast to supervised learning's focus on accuracy on a given dataset. RL typically suffers from extreme sample inefficiency due to sparse rewards, the need for exploration, and high-variance [215] optimization algorithms. However, applications also often naturally entail task families which meta-learning can exploit – for example locomoting-to or reaching-to different positions [172], navigating within different maps/environments [40] or traversing different terrains [63], driving different cars [171], competing with different competitor agents [61], and dealing with different handicaps such as failures in individual robot limbs [63]. 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 'how to stand up' for a humanoid robot is a transferable skill for all tasks within a family that require locomotion, while meta-knowledge of a maze layout is transferable for all tasks that require navigating within the maze.

5.2.1 Methods

Several meta-representations that we have already seen have been explored in RL including learning the initial conditions [19], [159], hyperparameters [159], [164], step directions [75] and step sizes [163], which enables gradient-based learning to train a neural policy with fewer environmental interactions; and training fast convolutional [40] or recurrent [22], [106] black-box models to embed the experience of of a given environment thus far and use it to synthesize a feed-forward policy. Recent work has developed improved meta-optimization algorithms [155], [156], [158] for these tasks, and provided theoretical guarantees for Meta RL [216].

Exploration A meta-representation that is rather unique to RL is that of 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. Thus how to explore and acquire data for learning is a crucial factor in any RL algorithm. Traditionally exploration is based on sampling random actions [90], or hand-crafted exploration heuristics [217]. Several meta-RL studies have instead explicitly treated exploration strategy or curiosity function as meta-knowledge ω; and modeled their acquisition as meta-learning problem [24], [170], [171] – leading to significant sample efficiency improvements by 'learning how to explore'.

Optimization It is worth noting that, unlike SL where optimization often leads to good local minima with perfect accuracy on the train set; RL is usually a very difficult optimization problem where the learned policy is 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 training performance [23], [164], [167] as well as sample-efficiency, and can lead to significantly better solutions overall. Indeed, the meta-objective of most meta-RL frameworks is the net return of the agent over a full training episode, and thus both sample efficient and asymptotically performant learning are rewarded. Optimization difficulty also means that there has also been relatively more work on learning losses (or rewards) [113], [167], [218] which an RL agent should optimize instead of – or in addition to – the conventional sparse reward objective. Such meta-learned losses may be easier to optimize (denser, smoother) compared to the true target [23], [218]. This also links back to exploration as reward learning and can be considered to instantiate meta-learning approaches to learning intrinsic motivation [168].

Online MetaRL We note that a significant fraction of meta-RL studies addressed the online single-task setting, where the meta-knowledge such as loss [113], [167], reward [164], [168], hyperparameters [162], [163], or exploration strategy [169] 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.

On- vs Off-Policy Meta-RL A major dichotomy in conventional RL methodologies is between on-policy and off-policy learning such as PPO [90] vs SAC [219]. Off-policy methods are usually significantly more sample efficient for conventional RL. However, off-policy methods have been harder to extend to meta-RL, leading to the majority of Meta-RL methods being built on on-policy base algorithms, and thus limiting the absolute performance of Meta-RL. A few recent works have begun to design meta-RL generalizations of off-policy methods leading to strong results [109], [113], [157], [218]. Notably off-policy learning also improves the efficiency of the meta-train stage [109], which can be very expensive in Meta-RL. It also provides new opportunities to accelerate meta-testing by replay buffer sample from meta-training stage [157].

Other Trends and Challenges We finish this section by mentioning other recent trends in meta-RL. [63] is noteworthy in demonstrating successful meta-RL on a real-world physical robot. Knowledge transfer in robotics often makes sense to study *compositionally* [220]. E.g., walking, navigating and object pick/place may be subroutines of cleaning up the room for a robot. However, developing meta-learners that support a compositional knowledge that transfers well is an open question, with modular meta-learning [131] being an option. Unsupervised meta-RL variants aim to perform meta-training without manually specified rewards [221], or adapt at meta-testing to a changed environment but without new rewards [222]. Continual adaptation uses meta-learning to provide an agent with the ability to adapt to a sequence of tasks within one meta-test episode [61]–[63], which is connected to continual learning. Finally, meta-learning has also been applied to imitation [105] and inverse reinforcement learning [223].

5.2.2 Benchmarks

Meta-learning benchmarks for RL should define a family of problems for an agent to solve in order to learn how

to learn, and subsequently evaluate the learner. These can be tasks (reward functions) to achieve, or domains (distinct environments or MDPs). RL benchmarks can be divided according whether they test continuous or discrete control, and actuation from state or observations such as images.

Discrete Control RL An early meta-RL benchmark for vision-actuated control is the arcade learning environment (ALE) [224], which defines a set of classic Atari games which can be split into meta-training and meta-testing. The typical protocol here is to evaluate return after a fixed number of timesteps in the meta-test environment. One issue with Atari games is their determinism which means that an open-loop policy is potentially sufficient to solve them, leading to efforts to insert stochasticity [224]. Another challenge is that there is great diversity (wide $p(\mathcal{T})$) across games, which makes successful meta-training hard and leads to limited benefit from knowledge transfer [224]. Another benchmark [225] 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. Recently Cobbe *et al*. [226] proposed two purpose designed video games for benchmarking Meta-RL. CoinRun game [226] 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 [226] provides a set of 16 procedurally generated games including CoinRun.

Continuous Control RL While the use of common benchmarks such as gym [227] has greatly benefited RL research, there has not yet been a consensus on benchmarks for meta-RL, making existing work hard to compare. Most studies of continuous control meta-RL have proposed home-brewed benchmarks that are low dimensional parametric variants of particular tasks such as navigating to various locations or velocities [19], [109], or traversing different terrains [63]. Several multi-MDP benchmarks [228], [229] have recently been proposed but these primarily test generalization across different environmental perturbations rather than new task adaptation of interest in Meta-RL. This situation is set to improve with the release of Meta-World benchmark [230] which 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 both within and across task distributions of various widths. The meta-world evaluation [230] suggests that existing Meta-RL methods struggle to generalize over wide task distributions and meta-train/meta-test shifts, so more work is necessary. Another recent benchmark suitable for Meta-RL is PHYRE [231] 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 are organised into 2 difficulty tiers, and provide within and cross-template generalization tests.

Discussion One complication of vision-actuated meta-RL is unpicking visual generalization and adaptation (as in common with broader computer vision) with fast learning of control strategies more generally. For example CoinRun [226] evaluation showed large benefit from standard vision techniques such as batch norm suggesting that perception is a major bottleneck.

A topical issue in Meta-RL is that it is difficult to fit wide meta-train task distributions with multi-task or meta-learning models – before even getting to the performance of meta-testing on novel tasks. This may be due to our RL models being too weak and/or benchmarks being too small in terms of number of tasks. Even Meta-World, ProcGen and PHYRE have dozens rather than hundreds of tasks like vision benchmarks such as tieredImageNet. While these latest benchmarks are improving, the field would still benefit from still larger benchmarks with controllable generalization gaps. It would also be beneficial to have benchmarks with greater difficulty such as requiring memory and abstract reasoning, to provide opportunities for more abstract strategies to be meta-learned and exploited across tasks.

5.3 Environment Learning and Sim2Real

In Sim2Real we are interested in training a model in simulation that is able to generalize to the real-world, which is challenging since the simulation does not match the real world exactly. 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 [232] and RL [148]. Nevertheless how to tune the simulation distribution is a challenge. This naturally 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 ω corresponds to the parameters of the simulation environment. This paradigm has been used in RL [150] as well as computer vision [149], [233]. 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 back-propagating through a costly and long graph of learning steps of the inner task; as well as minimising the number of real-world $\mathcal{L}^{meta}$ evaluations in the case of Sim2Real meta-learning for RL.

5.4 Neural Architecture Search (NAS)

Architecture search [26]–[28], [39], [123] can be seen as corresponding to a kind hyperparameter optimization where ω 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 are commonly analysed [39] according to 'search space', 'search strategy', and 'performance estimation strategy. These correspond to the hypothesis space for ω, the meta-optimization strategy, and the meta-objective. NAS is particularly challenging because: (i) Fully evaluating the inner loop is generally very 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 ultimately approximations such as interleaved descent on both ω and θ [26] as in online meta-learning. (ii.) The search space is hard to define, and optimizing it is costly. This is because most search spaces are broad, and represent architectures that aren't differentiable. This leads to methods that perform cell-level search [26], [28] to constrain the search space; and then rely on RL [28], discrete gradient estimators that provide a differentiable approximation to the search space [26], [124], and evolution [27], [123].

Examples Some notable examples include: (i) NASNet [28], [234] where the search space is restricted to cell-level learning, and defined as a string generated by an RNN which indicates what operations should be at what parts of the cell-tree, optimized using RL. (ii) Requlalized Evolution [27] where the authors use NASNet's search space but optimize it using regularized evolution, i.e. standard tournament based evolution with removal of oldest individuals after every iteration. (iii.) *DARTS* [26] where the authors carefully cast the space of cell architectures as a sequence of softmax selections over a number of pre-selected operations, thus making the search space differentiable. Learning the architecture then corresponds to jointly learning the softmax weights with the network parameters. This allows architecture learning to be sped up by 2-3 levels of magnitude both in computational overheads and wall-clock time. (iv) T-NAS [125] where the authors utilize the DARTS search space, but train it using a data-flow that enforces the architecture to be learned using very few data-points and very few updates, while keeping the generalization performance high. As a result of learning such softmax weights, they achieve few-shot architecture search. Once trained, these weights can be adapted to new tasks within seconds rather than days.

An interesting special case of NAS is activation function search [151]. While hand-designed activation functions such as ReLU are dominant in NN literature, a successful example of NAS meta-learning is the discovery of the Swish activation function [151] with RL in the space of symbolic activation functions. Swish has gone on to contribute to several influential state-of-the-art and general purpose CNN architectures [235], [236].

Multi-Objective NAS Architectures to be deployed on mobile devices have additional constraints besides validation accuracy [7], and NAS can also be deployed to produce compact and efficient models [237]. This can be realised by defining a multi-objective meta-objective that contains terms related both to validation performance as well as latency or size of the model product by a given θ, and thus leading to good performance-cost tradeoffs.

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 are dataset specific, or general purpose with ability to generalize to new problems [234]. Recent results suggest that meta-training across multiple datasets can lead to improved cross-task generalization of architectures [126].

While few-shot meta-learning is typically addressed from a parameter learning perspective in the context of hand-crafted architectures [19], [20], [87], one can also define NAS meta-objectives to train an architecture suitable for few-shot learning [238], [239]. Furthermore, analogously to fast-adapting initial condition meta-learning approaches such as MAML [19], one can train good initial architectures [125] or architecture priors [126] that are easy to adapt towards specific tasks.

Benchmarks NAS is often evaluated on the CIFAR-10 dataset. However even on this small dataset, architecture search is costly to perform, making it inaccessible to many researchers; and furthermore results are hard to reproduce due to other confounding factors such as tuning of hyperparameters [240]. To support reproducible and accessible research, the recently released NASbenches [241], [242] provides pre-computed performance measures for a large number of network architectures.

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 different approximation methods can be used. Variational approaches, especially stochastic variational methods, are the most common, but sampling approaches can also be considered.

One by-product of Bayesian meta-learning is that it provides uncertainty measures for the θ parameters, and hence measures of prediction uncertainty. Knowing the uncertainty of learner's predictions can be vital in safety critical domains such as few-shot medical tasks, and can be used for exploration in Reinforcement Learning and for some active learning methods, where a model can seek information about datapoints with high uncertainty.

Recently a number of authors have explored Bayesian approaches to meta-learning complex models with competitive results. Many of these have utilized deep neural networks as components within the framework, for example extending variational autoencoders to model task variables explicitly [71]. Neural Processes [166] aim to combine the uncertainty quantification of Gaussian Processes with the versatility of neural networks, but they are not shown to work on modern few-shot benchmarks. Deep kernel learning is also an active research area that has been adapted to the meta-learning setting [243], and is often coupled with Gaussian Processes [213]. In [72] gradient based meta-learning is recast into a hierarchical empirical Bayes inference problem (i.e. prior learning), which models uncertainty in task-specific parameters θ. Bayesian MAML [212] improves on this model by using a Bayesian ensemble approach that allows non-Gaussian posteriors over θ, and later work removes the need for costly ensembles [199], [244]. In Probabilistic MAML [95], it is the uncertainty in the metaknowledge ω that is modelled, while a MAP estimate is used for θ. Increasingly, these Bayesian methods are shown to tackle ambiguous tasks, active learning and RL problems.

Separate from the above approaches, meta-learning has also been proposed to aid the Bayesian inference process itself. By way of example, in [245], the authors use a meta-learning framework to adapt a Bayesian sampler to provide efficient adaptive sampling methods.

Benchmarks In Bayesian meta-learning, the point is usually to model the uncertainty in the predictions of our meta-learner, and so performance on standard few-shot classification benchmarks doesn't necessarily capture what we care about. For this reason different tasks have been developed in the literature. Bayesian MAML [212] extends the sinusoidal regression task of MAML [19] to make it more challenging. Probabilistic MAML [95] provides a suite of 1D toy examples capable of showing model uncertainty and how this uncertainty can be used in an active learning scenario. It also creates a binary classification task from celebA [246], where the positive class is determined by the presence of two facial attributes, but training images show three attributes, thus introducing ambiguity in which two attributes should be classified on. It is observed that sampling ω can correctly reflect this ambiguity. Active learning toy experiments are also shown in [212] as well as reinforcement learning applications, and ambiguous one-shot image generation tasks are used in [199]. Finally, some researchers propose to look at the accuracy v.s. confidence of the meta-learners (i.e. their calibration) [244].

5.6 Unsupervised Meta-Learning and Meta-Learning Unsupervised Learning

In the meta-learning literature, there exist two main variants of meta-learning involving unsupervised learning. In the first one the meta-objective of the outer loop is unsupervised, and therefore the learner itself is learned without any labels available. We refer to this case as *Unsupervised Meta-Learning*. In the second variant, meta-learning is used as a means to learn an unsupervised inner loop task. The outer objective in this case can be anything from supervised, unsupervised or reinforcement based. We refer to this as *Meta-Learning Unsupervised Learning*.

Unsupervised Meta-Learning [247]–[249] aims to relax the conventional assumption of an annotated set of source tasks for meta-training, while still producing good downstream performance for supervised few-shot learning. Typically synthetic source tasks are constructed without supervision via clustering or class-preserving data augmentation.

Meta-Learning Unsupervised Learning aims to use meta-learning to train unsupervised learning algorithms that work well for downstream supervised learning tasks. One can train unsupervised clustering algorithms [21], [250], [251] or losses [98], [116] such that downstream supervised learning performance is optimized. This helps to deal with the ill-definedness of the unsupervised learning problem by transforming it into a problem with a clear (meta) supervised objective.

5.7 Active Learning

The meta-learning paradigm can also be used to train active learning, rather than supervised or reinforcement learners as discussed so far. 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 [252]. Meta-learning can transform active learning algorithm design into a learning task by: considering the inner-level optimization as a conventional supervised learning task on the annotated dataset so far, considering ω to be a query policy that selects the best unlabeled datapoints to annotate, or by letting the outer-level optimization train the query policy to optimize a meta-objective corresponding to downstream learning performance given the queried and annotated datapoints [175]–[177]. However, as for clustering, if labels are used to train AL algorithms, they need to generalize across tasks to amortise their training cost [177].

5.8 Continual, Online and Adaptive Learning

Continual Learning refers to the humanlike 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 all past data for rehearsal against forgetting [60]. Deep Neural Networks struggle at meeting these criteria, especially as they tend to forget information seen in earlier tasks – a phenomenon known as *catastrophic forgetting*. Meta-learning has been applied to improve continual learning in deep nets. The requirements of continual learning can be integrated 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 [160], [161]. With this meta-objective design, various meta-representations can be trained so as to improve continual learning performance. For example: weight priors [128], gradient descent preconditioning matrices [161], or RNN learned optimizers [160], or feature representations [253].

Although not directly applied to continual learning, another interesting idea is meta-training representations to support local editing [254] where the authors learn a model that can quickly improve itself on single samples, without forgetting any of the information it already has learned.

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 [96] to perform MAML-style meta-training online during a task sequence. Meanwhile others [61]–[63] 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 There exist a number of benchmarks for continual learning that work quite well with standard deep learning methods. However, most of those benchmarks cannot readily work with meta-learning approaches. Most of them would require adjustments to their sample generation routines to include 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 [96], [160], [253], 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 [205], where

Continual Few-Shot Learning is defined as a new type of task to be tackled, and the benchmark is built for meta and non meta-learning approaches alike. In this setting, a task is composed by a number of small training sets, each potentially made of different classes, after which the learned model should generalize well on previously unseen samples from all the tasks it learned from. The benchmark proposes the usage of Omniglot and SlimageNet as the datasets to be used.

5.9 Domain Adaptation and Domain Generalization

Domain-shift often hampers machine learning models in practice, when the statistics of data encountered in deployment differ 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 [57].

Domain Generalization Domain *generalization* approaches aim to train models with increased robustness to train-test domain shift by design [255], often by exploiting a distribution over training domains. Meta-learning can be an effective tool to support this goal by defining the outer-loop validation set to have a domain shift with respect to the inner loop train set [58]. In this way different kinds of meta-knowledge such as regularizers [92], losses [44], and noise augmentation [173] can be (meta) learned so as to maximize the typical robustness of the learned model to train-test domain-shift.

Domain Adaptation While the extensive prior work on domain *adaptation* has been conventional learning [57], recent work [256] has begun to consider meta-learning approaches to boosting domain adaptation as well.

Benchmarks Popular benchmarks for DA and DG are oriented around recognition of different image types such as photo/sketch/cartoon. Datasets with multiple domains are often used in order to provide a domain distribution for meta-learning. PACS [257] provides a good starter benchmark with Visual Decathlon [44], [206], DomainNet [258] and Meta-Dataset [207] providing larger scale alternatives.

5.10 Hyper-parameter Optimization

Meta-learning can address hyperparameter optimization by considering ω 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, or hyperparameters that improve learning for 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 [76]. The case where we wish to learn hyperparameters for a single task is usually more relevant for many-shot applications [145], where some validation data can be extracted from the training dataset, as discussed in Section 2.1. Meta-learning over long inner horizons comes with memory and compute scaling issues, which is an active research area as discussed in Section 6. However, it is noteworthy that end-to-end gradient-based meta-learning has already demonstrated promising scalability to millions of parameters (as demonstrated by MAML [19], [145] and Dataset Distillation [144], [145], for example) in contrast to the classic approaches (such cross-validation by grid or random [69] search, or Bayesian Optimization [70]) which are usually only successful with dozens of hyper-parameters.

5.11 Novel and Biologically Plausible Learners

Most meta-learning work that uses an explicit (non feed-forward/black-box) optimization for the base model is based on gradient descent by backpropagation (as is most conventional deep learning work). An intriguing possibility for meta-learning is to define the function class of learning rules ω so as to lead to the discovery of novel effective learning rules that are potentially unsupervised [21], biologically plausible [47], [259], [260] making use of ideas less commonly used in contemporary deep learning such as Hebbian updates [259] and neuromodulation [260].

5.12 Language and Speech

Language Modelling. Few-shot language modelling has been a popular way to showcase the versatility of meta-learners, with early methods like matching networks showing impressive performances on one-shot tasks such as filling in missing words [86]. Many more tasks have since been tackled, including neural program induction [261] and synthesis [262], English to SQL program synthesis [263], text-based relationship graph extractor [264], machine translation [265], and quickly adapting to new personas in dialogue tasks [266].

Speech Recognition Deep learning is now established as 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 [267], cross-accent adaptation [268] and optimising models for individual speakers [269].

5.13 Meta-learning for Social Good

Meta-learning lands itself to various challenging tasks 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 [270]. In [5] an LSTM is combined with a graph neural network to predict the behaviour of a molecule (e.g. its toxicity) in the one-shot data regime. In [271] MAML is adapted to weakly-supervised breast cancer detection tasks, and the order of tasks are selected according to a curriculum rather than randomly. MAML is also combined with denoising autoencoders to do medical visual question answering [272], while learning to weigh support samples as done in [204] is adapted to pixel wise weighting in order to tackle skin lesion segmentation tasks that have noisy labels [273].

5.14 Abstract and Compositional Reasoning

Abstract Reasoning A recent goal in deep learning research is to develop models that go beyond simple perception tasks to solving more abstract reasoning problems

such as IQ tests in the form of Raven's Progressive Matrices (RPMs) [274]. Solving RPMs can be seen as asking for few-shot generalization from the context panels to the answer panels. Recent meta-learning approaches to abstract reasoning with RPMs achieved significant improvement via meta-learning a teacher that defines the data generating distribution for the panels [275]. The teacher is trained jointly with the student, and rewarded by the student's progress, thus automatically defining an optimal curriculum.

Compositional Learning One of the traits that allows humans to be good at problem solving is the ability to learn how to compose concepts. For example being able to take a newly learned verb and use it with all potential adverbs. Recently meta-learning approaches have been shown to improve such generalization ability by requiring compositional generalization between query and support sets during meta-training [276]. Such meta-learning regimes can also benefit fundamental challenges such as enabling sequence models to generalize to test sequences longer than observed during training [276].

5.15 Systems

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 [277], [278]. 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 [187], and weight generator meta-networks that allow quantized networks to be trained with gradient [279].

Communications Deep learning has recently made waves in communications systems. For example by learning coding systems that exceed the best hand designed codes for realistic channels [280]. Insofar as optimal performance is achieved by learning a coding scheme tuned for the characteristics of a particular channel, few-shot meta-learning can be used to provide rapid online adaptation of coding to changing channel characteristics [281].

Learning with Label Noise is a challenge in contemporary deep learning when large datasets are collected by web scraping or crowd-sourcing. Again, while there are several algorithms hand-designed for this situation, recent meta-learning methods have addressed label noise by transductive learning sample-wise weighs to down-weight noisy samples [142], or learning an initial condition robust to noisy label training [93].

Adversarial Attacks and Defenses Deep Neural Networks can be easily fooled into misclassifying a data point that should be easily recognizable, by adding a carefully crafted human-invisible perturbation to the data [282]. Numerous methods have been published in recent years introducing stronger attack and defense methods. Typical defenses are carefully hand-designed architectures or training strategies. Analogous to the case in domain-shift, an under-studied potential application of meta-learning is to train the learning algorithm end-to-end for robustness by defining a meta-loss in terms of performance under adversarial attack [94], [283]. New benchmarks for adversarial defenses have recently been proposed [284] where defenses should generalize to unforseen attacks. It will be interesting to see whether future meta-learning approaches can make progress on this benchmark.

6 Challenges and Open Questions

Meta-generalization Meta-learning suffers from a generalization challenge across tasks analogous to the challenge of generalising across instances in conventional machine learning. There are three sub-challenges: (i) The first challenge is fitting a meta-learner to a wide distribution of tasks $p(\mathcal{T})$, which as we have seen is challenging for existing methods [206], [207], [230], and may be partly due to conflicting gradients between tasks [285]. (ii) The second challenge is generalising 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 fit complex task distributions. Thus meta-learners' biggest successes have thus far been within very similar task families. (iii) The third challenge is generalising 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 generalising few-shot visual learning from everyday training images of ImageNet to specialist domains such as medical images [208]. 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 [173]. Furthermore, we have yet to understand which kinds of meta-representations tend to generalize better under certain types of domain shifts.

Another interesting direction could be investigating how introducing yet another level of learning abstractions could affect generalization performance, that is, *meta-meta-learning*. By learning how to do meta-learning, perhaps we can find meta-optimizers that can generalize very strongly across a large variety of types and intensities of domain and even modality shifts. Of course computational costs would be exponentially larger.

Multi-modality of task distribution Many meta-learning frameworks [19] implicitly assume that the distribution over tasks $p(\mathcal{T})$ is *uni-modal*, and a single learning strategy ω provides a good solution for them all. However in reality task distributions can clearly be multi-modal. Consider for example, medical vs satellite vs everyday images in computer vision. Or the diversity of tasks that robots could be asked to perform from putting pegs in holes to opening doors [230]. Different tasks within the distribution may require different learning strategies, and this can degrade existing meta-learner performance. In vanilla multi-task learning, this phenomenon is relatively well studied with, e.g., methods that group tasks into clusters [286] or subspaces [287]. However this area is only just beginning to be explored in meta-learning [288].

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. How to relax this assumption is an ongoing challenge. Unsupervised meta-learning [247]–[249] and online meta-learning methods [44], [162], [167], [168], [185], could help to alleviate this; as can improvements in meta-generalization discussed above.

Computation Cost Naive implementation of bilevel optimization as shown in Section 2.1 leads to a quadratic number of learning steps, since each outer step requires multiple inner steps. Moreover, there are a large number of inner steps in the case of many-shot experiments, and these need to be stored in memory. For this reason most meta-learning frameworks are extremely expensive in both time and memory, and often limited to small architectures in the few-shot regime [19]. However there is an increasing focus on methods to tackle this problem. For instance, one can alternate inner and outer updates [44], or train surrogate models [108]. Another family of recent approaches accelerate meta-training via closed-form solvers in the inner loop [152], [154]. However, the cost is still quite large, and the significance of the former set heuristics for convergence is unclear. A recent method for computing the gradients for the outer loop using implicit gradients provides a cheaper alternative [153], but only focused on learning the initialization of a network for MAML. While implicit gradients were then shown to work for more general meta-learning tasks such as learning an augmentation network [145], they can only learn parameters involved in the loss function directly and make several assumptions (like zero training gradients at θ^*) often leading to inaccurate ω gradients.

Cross-modal transfer and heterogeneous tasks Most meta-learning methods studied so far have considered tasks all drawn from the same modality such as vision, text, proprioceptive state, or audio. Humans appear to be able to transfer knowledge across modalities (e.g., by visual imitation learning). How to do meta-learning that extracts abstract knowledge from a set of tasks that may each span a unique modality is an open question. Most studies have addressed transfer between tasks of the same type such as object recognition, but ideally we would like to be able to transfer between heterogeneous tasks such as those studied in Taskonomy [289].

7 Conclusion

The field of meta-learning has recently 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 in this growing field, as well as highlight opportunities for future research.

Acknowledgments

The authors would like to thank Massimiliano Patacchiola, Joseph Mellor, Bohdal Ondrej and Gabriella Pizzuto for their feedback on the survey. We would also like to thank Da Li, Yongxin Yang and Henry Gouk for helpful discussions. T. Hospedales was supported by the Engineering and Physical Sciences Research Council of the UK (EPSRC) Grant number EP/S000631/1 and the UK MOD University Defence Research Collaboration (UDRC) in Signal Processing, and EPSRC Grant EP/R026173/1.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning For Image Recognition," in *CVPR*, 2016.
[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering The Game Of Go With Deep Neural Networks And Tree Search," *Nature*, 2016.
[3] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training Of Deep Bidirectional Transformers For Language Understanding," in *ACL*, 2019.
[4] G. Marcus, "Deep Learning: A Critical Appraisal," *arXiv e-prints*, 2018.
[5] H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. S. Pande, "Low Data Drug Discovery With One-shot Learning," *CoRR*, 2016.
[6] X. Xu, Y. Ding, S. X. Hu, M. Niemier, J. Cong, Y. Hu, and Y. Shi, "Scaling For Edge Inference Of Deep Neural Networks," *Nature Electronics*, 2018.
[7] A. Ignatov, R. Timofte, A. Kulik, S. Yang, K. Wang, F. Baum, M. Wu, L. Xu, and L. Van Gool, "AI Benchmark: All About Deep Learning On Smartphones In 2019," *arXiv e-prints*, 2019.
[8] S. Thrun and L. Pratt, "Learning To Learn: Introduction And Overview," in *Learning To Learn*, 1998.
[9] H. F. Harlow, "The Formation Of Learning Sets." *Psychological Review*, 1949.
[10] J. B. Biggs, "The Role of Meta-Learning in Study Processes," *British Journal of Educational Psychology*, 1985.
[11] A. M. Schrier, "Learning How To Learn: The Significance And Current Status Of Learning Set Formation," *Primates*, 1984.
[12] P. Domingos, "A Few Useful Things To Know About Machine Learning," *Commun. ACM*, 2012.
[13] D. G. Lowe, "Distinctive Image Features From Scale-Invariant," *International Journal of Computer Vision*, 2004.
[14] N. Dalal and B. Triggs, "Histograms Of Oriented Gradients For Human Detection," in *CVPR*, 2005.
[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet Classification With Deep Convolutional Neural Networks," in *NeurIPS*, 2012.
[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
[17] J. Schmidhuber, "Evolutionary Principles In Self-referential Learning," *On learning how to learn: The meta-meta-... hook*, 1987.
[18] J. Schmidhuber, J. Zhao, and M. Wiering, "Shifting Inductive Bias With Success-Story Algorithm, Adaptive Levin Search, And Incremental Self-Improvement," *Machine Learning*, 1997.
[19] C. Finn, P. Abbeel, and S. Levine, "Model-Agnostic Meta-learning For Fast Adaptation Of Deep Networks," in *ICML*, 2017.
[20] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical Networks For Few Shot Learning," in *NeurIPS*, 2017.
[21] L. Metz, N. Maheswaranathan, B. Cheung, and J. Sohl-Dickstein, "Meta-learning Update Rules For Unsupervised Representation Learning," *ICLR*, 2019.
[22] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "RL^2: Fast Reinforcement Learning Via Slow Reinforcement Learning," in *ArXiv E-prints*, 2016.
[23] R. Houthooft, R. Y. Chen, P. Isola, B. C. Stadie, F. Wolski, J. Ho, and P. Abbeel, "Evolved Policy Gradients," *NeurIPS*, 2018.
[24] F. Alet, M. F. Schneider, T. Lozano-Perez, and L. Pack Kaelbling, "Meta-Learning Curiosity Algorithms," *ICLR*, 2020.
[25] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, "Bilevel Programming For Hyperparameter Optimization And Meta-learning," in *ICML*, 2018.

[26] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable Architecture Search," in *ICLR*, 2019.

[27] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized Evolution For Image Classifier Architecture Search," *AAAI*, 2019.

[28] B. Zoph and Q. V. Le, "Neural Architecture Search With Reinforcement Learning," *ICLR*, 2017.

[29] R. Vilalta and Y. Drissi, "A Perspective View And Survey Of Meta-learning," *Artificial intelligence review*, 2002.

[30] D. H. Wolpert, "The Lack Of A Priori Distinctions Between Learning Algorithms," *Neural Computation*, 1996.

[31] J. Vanschoren, "Meta-Learning: A Survey," *CoRR*, 2018.

[32] Q. Yao, M. Wang, H. J. Escalante, I. Guyon, Y. Hu, Y. Li, W. Tu, Q. Yang, and Y. Yu, "Taking Human Out Of Learning Applications: A Survey On Automated Machine Learning," *CoRR*, 2018.

[33] R. Vilalta, C. Giraud-Carrier, P. Brazdil, and C. Soares, "Using Meta-Learning To Support Data Mining," *International Journal of Computer Science & Applications*, 2004.

[34] S. J. Pan and Q. Yang, "A Survey On Transfer Learning," *IEEE TKDE*, 2010.

[35] N. Jankowski, D. Wlodzislaw, and K. Grabczewski, *Meta-Learning In Computational Intelligence*. Springer, 2011.

[36] C. Lemke, M. Budka, and B. Gabrys, "Meta-Learning: A Survey Of Trends And Technologies," *Artificial intelligence review*, 2015.

[37] I. Khan, X. Zhang, M. Rehman, and R. Ali, "A Literature Survey And Empirical Study Of Meta-Learning For Classifier Selection," *IEEE Access*, 2020.

[38] Y. Wang and Q. Yao, "Few-shot Learning: A Survey," *CoRR*, 2019.

[39] T. Elsken, J. H. Metzen, and F. Hutter, "Neural Architecture Search: A Survey," *Journal of Machine Learning Research*, 2019.

[40] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A Simple Neural Attentive Meta-learner," *ICLR*, 2018.

[41] S. Ravi and H. Larochelle, "Optimization As A Model For Few-Shot Learning," in *ICLR*, 2016.

[42] H. Stackelberg, *The Theory Of Market Economy*. Oxford University Press, 1952.

[43] A. Sinha, P. Malo, and K. Deb, "A Review On Bilevel Optimization: From Classical To Evolutionary Approaches And Applications," *IEEE Transactions on Evolutionary Computation*, 2018.

[44] Y. Li, Y. Yang, W. Zhou, and T. M. Hospedales, "Feature-Critic Networks For Heterogeneous Domain Generalization," in *ICML*, 2019.

[45] G. Denevi, C. Ciliberto, D. Stamos, and M. Pontil, "Learning To Learn Around A Common Mean," in *NeurIPS*, 2018.

[46] G. E. Hinton and D. C. Plaut, "Using Fast Weights To Deblur Old Memories," in *Conference Of The Cognitive Science Society*, 1987.

[47] Y. Bengio, S. Bengio, and J. Cloutier, *Learning A Synaptic Learning Rule*. IEEE, 1990.

[48] S. Bengio, Y. Bengio, and J. Cloutier, "On The Search For New Learning Rules For ANNs," *Neural Processing Letters*, 1995.

[49] J. Schmidhuber, J. Zhao, and M. Wiering, "Simple Principles Of Meta-Learning," *Technical report IDSIA*, 1996.

[50] J. Schmidhuber, "A Neural Network That Embeds Its Own Meta-levels," in *IEEE International Conference On Neural Networks*, 1993.

[51] S. Hochreiter, A. S. Younger, and P. R. Conwell, "Learning To Learn Using Gradient Descent," in *ICANN*, 2001.

[52] A. S. Younger, S. Hochreiter, and P. R. Conwell, "Meta-learning With Backpropagation," in *IJCNN*, 2001.

[53] N. Schweighofer and K. Doya, "Meta-learning In Reinforcement Learning," *Neural Networks*, 2003.

[54] H. Larochelle, D. Erhan, and Y. Bengio, "Zero-data Learning Of New Tasks." in *AAAI*, 2008.

[55] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirchlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[56] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How Transferable Are Features In Deep Neural Networks?" in *NeurIPS*, 2014.

[57] G. Csurka, *Domain Adaptation In Computer Vision Applications*. Springer, 2017.

[58] D. Li, Y. Yang, Y. Song, and T. M. Hospedales, "Learning To Generalize: Meta-Learning For Domain Generalization," in *AAAI*, 2018.

[59] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual Lifelong Learning With Neural Networks: A Review," *Neural Networks*, 2019.

[60] Z. Chen and B. Liu, "Lifelong Machine Learning, Second Edition," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2018.

[61] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel, "Continuous Adaptation Via Meta-Learning In Nonstationary And Competitive Environments," *ICLR*, 2018.

[62] S. Ritter, J. X. Wang, Z. Kurth-Nelson, S. M. Jayakumar, C. Blundell, R. Pascanu, and M. Botvinick, "Been There, Done That: Meta-learning With Episodic Recall," *ICML*, 2018.

[63] I. Clavera, A. Nagabandi, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning To Adapt In Dynamic, Real-World Environments Through Meta-Reinforcement Learning," in *ICLR*, 2019.

[64] R. Caruana, "Multitask Learning," *Machine Learning*, 1997.

[65] Y. Yang and T. M. Hospedales, "Deep Multi-Task Representation Learning: A Tensor Factorisation Approach," in *ICLR*, 2017.

[66] E. Meyerson and R. Miikkulainen, "Modular Universal Reparameterization: Deep Multi-task Learning Across Diverse Domains," in *NeurIPS*, 2019.

[67] L. Franceschi, M. Donini, P. Frasconi, and M. Pontil, "Forward And Reverse Gradient-Based Hyperparameter Optimization," in *ICML*, 2017.

[68] X. Lin, H. Baweja, G. Kantor, and D. Held, "Adaptive Auxiliary Task Weighting For Reinforcement Learning," in *NeurIPS*, 2019.

[69] J. Bergstra and Y. Bengio, "Random Search For Hyper-Parameter Optimization," in *Journal Of Machine Learning Research*, 2012.

[70] 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," *Proceedings of the IEEE*, 2016.

[71] H. Edwards and A. Storkey, "Towards A Neural Statistician," in *ICLR*, 2017.

[72] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths, "Recasting Gradient-Based Meta-Learning As Hierarchical Bayes," in *ICLR*, 2018.

[73] H. Yao, X. Wu, Z. Tao, Y. Li, B. Ding, R. Li, and Z. Li, "Automated Relational Meta-learning," in *ICLR*, 2020.

[74] S. C. Yoonho Lee, "Gradient-Based Meta-Learning With Learned Layerwise Metric And Subspace," in *ICML*, 2018.

[75] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-SGD: Learning To Learn Quickly For Few Shot Learning," *arXiv e-prints*, 2017.

[76] A. Antoniou, H. Edwards, and A. J. Storkey, "How To Train Your MAML," in *ICLR*, 2018.

[77] K. Li and J. Malik, "Learning To Optimize," in *ICLR*, 2017.

[78] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, "Learning To Learn By Gradient Descent By Gradient Descent," in *NeurIPS*, 2016.

[79] S. Qiao, C. Liu, W. Shen, and A. L. Yuille, "Few-Shot Image Recognition By Predicting Parameters From Activations," *CVPR*, 2018.

[80] S. Gidaris and N. Komodakis, "Dynamic Few-Shot Visual Learning Without Forgetting," in *CVPR*, 2018.

[81] A. Graves, G. Wayne, and I. Danihelka, "Neural Turing Machines," in *ArXiv E-prints*, 2014.

[82] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta Learning With Memory-Augmented Neural Networks," in *ICML*, 2016.

[83] T. Munkhdalai and H. Yu, "Meta Networks," in *ICML*, 2017.

[84] C. Finn and S. Levine, "Meta-Learning And Universality: Deep Representations And Gradient Descent Can Approximate Any Learning Algorithm," in *ICLR*, 2018.

[85] G. Kosh, R. Zemel, and R. Salakhutdinov, "Siamese Neural Networks For One-shot Image Recognition," in *ICML*, 2015.

[86] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching Networks For One Shot Learning," in *NeurIPS*, 2016.

[87] 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 *CVPR*, 2018.

[88] V. Garcia and J. Bruna, "Few-Shot Learning With Graph Neural Networks," in *ICLR*, 2018.

[89] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le, "Neural Optimizer Search With Reinforcement Learning," in *ICML*, 2017.

[90] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv e-prints*, 2017.

[91] O. Wichrowska, N. Maheswaranathan, M. W. Hoffman, S. G. Colmenarejo, M. Denil, N. de Freitas, and J. Sohl-Dickstein, "Learned Optimizers That Scale And Generalize," in *ICML*, 2017.

[92] Y. Balaji, S. Sankaranarayanan, and R. Chellappa, "MetaReg: Towards Domain Generalization Using Meta-Regularization," in *NeurIPS*, 2018.

[93] J. Li, Y. Wong, Q. Zhao, and M. S. Kankanhalli, "Learning To Learn From Noisy Labeled Data," in *CVPR*, 2019.

[94] M. Goldblum, L. Fowl, and T. Goldstein, "Adversarially Robust Few-shot Learning: A Meta-learning Approach," *arXiv e-prints*, 2019.

[95] C. Finn, K. Xu, and S. Levine, "Probabilistic Model-agnostic Meta-learning," in *NeurIPS*, 2018.

[96] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine, "Online Meta-learning," *ICML*, 2019.

[97] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-Learning With Latent Embedding Optimization," *ICLR*, 2019.

[98] A. Antoniou and A. Storkey, "Learning To Learn By Self-Critique," *NeurIPS*, 2019.

[99] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-Transfer Learning For Few-Shot Learning," in *CVPR*, 2018.

[100] R. Vuorio, S.-H. Sun, H. Hu, and J. J. Lim, "Multimodal Model-Agnostic Meta-Learning Via Task-Aware Modulation," in *NeurIPS*, 2019.

[101] H. Yao, Y. Wei, J. Huang, and Z. Li, "Hierarchically Structured Meta-learning," *ICML*, 2019.

[102] D. Kingma and J. Ba, "Adam: A Method For Stochastic Optimization," in *ICLR*, 2015.

[103] E. Park and J. B. Oliva, "Meta-Curvature," in *NeurIPS*, 2019.

[104] Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, M. Botvinick, and N. de Freitas, "Learning To Learn Without Gradient Descent By Gradient Descent," in *ICML*, 2017.

[105] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot Imitation Learning," in *NeurIPS*, 2017.

[106] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning To Reinforcement Learn," *CoRR*, 2016.

[107] D. Ha, A. Dai, and Q. V. Le, "HyperNetworks," *ICLR*, 2017.

[108] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "SMASH: One-Shot Model Architecture Search Through Hypernetworks," *ICLR*, 2018.

[109] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient Off-Policy Meta-Reinforcement Learning Via Probabilistic Context Variables," in *ICML*, 2019.

[110] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. Wang, and J.-B. Huang, "A Closer Look At Few-Shot Classification," in *ICLR*, 2019.

[111] B. Oreshkin, P. Rodríguez López, and A. Lacoste, "TADAM: Task Dependent Adaptive Metric For Improved Few-shot Learning," in *NeurIPS*, 2018.

[112] F. Sung, L. Zhang, T. Xiang, T. Hospedales, and Y. Yang, "Learning To Learn: Meta-critic Networks For Sample Efficient Learning," *arXiv e-prints*, 2017.

[113] W. Zhou, Y. Li, Y. Yang, H. Wang, and T. M. Hospedales, "Online Meta-Critic Learning For Off-Policy Actor-Critic Methods," 2020.

[114] G. Denevi, D. Stamos, C. Ciliberto, and M. Pontil, "Online-Within-Online Meta-Learning," in *NeurIPS*, 2019.

[115] S. Gonzalez and R. Miikkulainen, "Improved Training Speed, Accuracy, And Data Utilization Through Loss Function Optimization," *arXiv e-prints*, 2019.

[116] A. I. Rinu Boney, "Semi-Supervised Few-Shot Learning With MAML," *ICLR*, 2018.

[117] C. Huang, S. Zhai, W. Talbott, M. B. Martin, S.-Y. Sun, C. Guestrin, and J. Susskind, "Addressing The Loss-Metric Mismatch With Adaptive Loss Alignment," in *ICML*, 2019.

[118] J. Grabocka, R. Scholz, and L. Schmidt-Thieme, "Learning Surrogate Losses," *CoRR*, 2019.

[119] C. Doersch and A. Zisserman, "Multi-task Self-Supervised Visual Learning," in *ICCV*, 2017.

[120] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros, "Context Encoders: Feature Learning By Inpainting," in *CVPR*, 2016.

[121] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement Learning With Unsupervised Auxiliary Tasks," in *ICLR*, 2017.

[122] S. Liu, A. Davison, and E. Johns, "Self-supervised Generalisation With Meta Auxiliary Learning," in *NeurIPS*, 2019.

[123] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing Neural Networks Through Neuroevolution," *Nature Machine Intelligence*, 2019.

[124] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: Stochastic Neural Architecture Search," in *ICLR*, 2019.

[125] D. Lian, Y. Zheng, Y. Xu, Y. Lu, L. Lin, P. Zhao, J. Huang, and S. Gao, "Towards Fast Adaptation Of Neural Architectures With Meta Learning," in *ICLR*, 2020.

[126] A. Shaw, W. Wei, W. Liu, L. Song, and B. Dai, "Meta Architecture Search," in *NeurIPS*, 2019.

[127] R. Hou, H. Chang, M. Bingpeng, S. Shan, and X. Chen, "Cross Attention Network For Few-shot Classification," in *NeurIPS*, 2019.

[128] M. Ren, R. Liao, E. Fetaya, and R. Zemel, "Incremental Few-shot Learning With Attention Attractor Networks," in *NeurIPS*, 2019.

[129] Y. Bao, M. Wu, S. Chang, and R. Barzilay, "Few-shot Text Classification With Distributional Signatures," *arXiv e-prints*, 2019.

[130] F. Alet, T. Lozano-Pérez, and L. P. Kaelbling, "Modular Meta-learning," in *CORL*, 2018.

[131] F. Alet, E. Weng, T. Lozano-Pérez, and L. P. Kaelbling, "Neural Relational Inference With Fast Modular Meta-learning," in *NeurIPS*, 2019.

[132] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "PathNet: Evolution Channels Gradient Descent In Super Neural Networks," in *ArXiv E-prints*, 2017.

[133] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le, "AutoAugment: Learning Augmentation Policies From Data," *CVPR*, 2019.

[134] Y. Li, G. Hu, Y. Wang, T. Hospedales, N. M. Robertson, and Y. Yang, "DADA: Differentiable Automatic Data Augmentation," 2020.

[135] R. Volpi and V. Murino, "Model Vulnerability To Distributional Shifts Over Image Transformation Sets," in *ICCV*, 2019.

[136] A. Antoniou, A. Storkey, and H. Edwards, "Data Augmentation Generative Adversarial Networks," *arXiv e-prints*, 2017.

[137] C. Zhang, C. Öztireli, S. Mandt, and G. Salvi, "Active Mini-batch Sampling Using Repulsive Point Processes," in *AAAI*, 2019.

[138] I. Loshchilov and F. Hutter, "Online Batch Selection For Faster Training Of Neural Networks," in *ICLR*, 2016.

[139] Y. Fan, F. Tian, T. Qin, X. Li, and T. Liu, "Learning To Teach," in *ICLR*, 2018.

[140] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum Learning," in *ICML*, 2009.

[141] 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 *ICML*, 2018.

[142] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng, "Meta-Weight-Net: Learning An Explicit Mapping For Sample Weighting," in *NeurIPS*, 2019.

[143] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning To Reweight Examples For Robust Deep Learning," in *ICML*, 2018.

[144] T. Wang, J. Zhu, A. Torralba, and A. A. Efros, "Dataset Distillation," *CoRR*, 2018.

[145] J. Lorraine, P. Vicol, and D. Duvenaud, "Optimizing Millions Of Hyperparameters By Implicit Differentiation," *arXiv e-prints*, 2019.

[146] W.-H. Li, C.-S. Foo, and H. Bilen, "Learning To Impute: A General Framework For Semi-supervised Learning," *arXiv e-prints*, 2019.

[147] 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 *NeurIPS*, 2019.

[148] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning Dexterous In-Hand Manipulation," 2018.

[149] N. Ruiz, S. Schulter, and M. Chandraker, "Learning To Simulate," *ICLR*, 2018.

[150] 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?" *CoRR*, 2019.

[151] Q. V. L. Prajit Ramachandran, Barret Zoph, "Searching For Activation Functions," in *ArXiv E-prints*, 2017.

[152] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-Learning With Differentiable Convex Optimization," in *CVPR*, 2019.

[153] A. Rajeswaran, C. Finn, S. Kakade, and S. Levine, "Meta-Learning With Implicit Gradients," in *NeurIPS*, 2019.

[154] L. Bertinetto, J. F. Henriques, P. H. Torr, and A. Vedaldi, "Meta-learning With Differentiable Closed-form Solvers," in *ICLR*, 2019.

[155] H. Liu, R. Socher, and C. Xiong, "Taming MAML: Efficient Unbiased Meta-reinforcement Learning," in *ICML*, 2019.

[156] J. Rothfuss, D. Lee, I. Clavera, T. Asfour, and P. Abbeel, "ProMP: Proximal Meta-Policy Search," in *ICLR*, 2019.

[157] R. Fakoor, P. Chaudhari, S. Soatto, and A. J. Smola, "Meta-Q-Learning," in *ICLR*, 2020.

[158] X. Song, W. Gao, Y. Yang, K. Choromanski, A. Pacchiano, and Y. Tang, "ES-MAML: Simple Hessian-Free Meta Learning," in *ICLR*, 2020.

[159] C. Fernando, J. Sygnowski, S. Osindero, J. Wang, T. Schaul, D. Teplyashin, P. Sprechmann, A. Pritzel, and A. Rusu, "Meta-Learning By The Baldwin Effect," in *Proceedings Of The Genetic And Evolutionary Computation Conference Companion*, 2018.

[160] R. Vuorio, D.-Y. Cho, D. Kim, and J. Kim, "Meta Continual Learning," *arXiv e-prints*, 2018.

[161] S. Flennerhag, A. A. Rusu, R. Pascanu, F. Visin, H. Yin, and R. Hadsell, "Meta-Learning With Warped Gradient Descent," in *ICLR*, 2020.

[162] Z. Xu, H. van Hasselt, and D. Silver, "Meta-Gradient Reinforcement Learning," in *NeurIPS*, 2018.

[163] K. Young, B. Wang, and M. E. Taylor, "Metatrace Actor-Critic: Online Step-Size Tuning By Meta-gradient Descent For Reinforcement Learning Control," in *IJCAI*, 2019.

[164] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castañeda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, N. Sonnerat, T. Green, L. Deason, J. Z. Leibo, D. Silver, D. Hassabis, K. Kavukcuoglu, and T. Graepel, "Human-level Performance In 3D Multiplayer Games With Population-based Reinforcement Learning," *Science*, 2019.

[165] J.-M. Perez-Rua, X. Zhu, T. Hospedales, and T. Xiang, "Incremental Few-Shot Object Detection," in *CVPR*, 2020.

[166] M. Garnelo, D. Rosenbaum, C. J. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. J. Rezende, and S. M. A. Eslami, "Conditional Neural Processes," *ICML*, 2018.

[167] V. Veeriah, M. Hessel, Z. Xu, R. Lewis, J. Rajendran, J. Oh, H. van Hasselt, D. Silver, and S. Singh, "Discovery Of Useful Questions As Auxiliary Tasks," in *NeurIPS*, 2019.

[168] Z. Zheng, J. Oh, and S. Singh, "On Learning Intrinsic Rewards For Policy Gradient Methods," in *NeurIPS*, 2018.

[169] T. Xu, Q. Liu, L. Zhao, and J. Peng, "Learning To Explore With Meta-Policy Gradient," *ICML*, 2018.

[170] B. C. Stadie, G. Yang, R. Houthooft, X. Chen, Y. Duan, Y. Wu, P. Abbeel, and I. Sutskever, "Some Considerations On Learning To Explore Via Meta-Reinforcement Learning," in *NeurIPS*, 2018.

[171] F. Garcia and P. S. Thomas, "A Meta-MDP Approach To Exploration For Lifelong Reinforcement Learning," in *NeurIPS*, 2019.

[172] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine, "Meta-Reinforcement Learning Of Structured Exploration Strategies," in *NeurIPS*, 2018.

[173] H.-Y. Tseng, H.-Y. Lee, J.-B. Huang, and M.-H. Yang, ""Cross-Domain Few-Shot Classification Via Learned Feature-Wise Transformation"," *ICLR*, Jan. 2020.

[174] H. B. Lee, T. Nam, E. Yang, and S. J. Hwang, "Meta Dropout: Learning To Perturb Latent Features For Generalization," in *ICLR*, 2020.

[175] P. Bachman, A. Sordoni, and A. Trischler, "Learning Algorithms For Active Learning," in *ICML*, 2017.

[176] K. Konyushkova, R. Sznitman, and P. Fua, "Learning Active Learning From Data," in *NeurIPS*, 2017.

[177] K. Pang, M. Dong, Y. Wu, and T. M. Hospedales, "Meta-Learning Transferable Active Learning Policies By Deep Reinforcement Learning," *CoRR*, 2018.

[178] D. Maclaurin, D. Duvenaud, and R. P. Adams, "Gradient-based Hyperparameter Optimization Through Reversible Learning," in *ICML*, 2015.

[179] C. Russell, M. Toso, and N. Campbell, "Fixing Implicit Derivatives: Trust-Region Based Learning Of Continuous Energy Functions," in *NeurIPS*, 2019.

[180] A. Nichol, J. Achiam, and J. Schulman, "On First-Order Meta-Learning Algorithms," in *ArXiv E-prints*, 2018.

[181] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution Strategies As A Scalable Alternative To Reinforcement Learning," *arXiv e-prints*, 2017.

[182] F. Stulp and O. Sigaud, "Robot Skill Learning: From Reinforcement Learning To Evolution Strategies," *Paladyn, Journal of Behavioral Robotics*, 2013.

[183] A. Soltoggio, K. O. Stanley, and S. Risi, "Born To Learn: The Inspiration, Progress, And Future Of Evolved Plastic Artificial Neural Networks," *Neural Networks*, 2018.

[184] Y. Cao, T. Chen, Z. Wang, and Y. Shen, "Learning To Optimize In Swarms," in *NeurIPS*, 2019.

[185] F. Meier, D. Kappler, and S. Schaal, "Online Learning Of A Memory For Learning Rates," in *ICRA*, 2018.

[186] A. G. Baydin, R. Cornish, D. Martínez-Rubio, M. Schmidt, and F. D. Wood, "Online Learning Rate Adaptation With Hypergradient Descent," *CoRR*, 2017.

[187] S. Chen, W. Wang, and S. J. Pan, "MetaQuant: Learning To Quantize By Learning To Penetrate Non-differentiable Quantization," in *NeurIPS*, 2019.

[188] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting Unreasonable Effectiveness Of Data In Deep Learning Era," in *ICCV*, 2017.

[189] L. M. Zintgraf, K. Shiarlis, V. Kurin, K. Hofmann, and S. Whiteson, "Fast Context Adaptation Via Meta-learning," *ICML*, 2018.

[190] M. Yin, G. Tucker, M. Zhou, S. Levine, and C. Finn, "Meta-Learning Without Memorization," *ICLR*, 2020.

[191] S. W. Yoon, J. Seo, and J. Moon, "Tapnet: Neural Network Augmented With Task-adaptive Projection For Few-shot Learning," *ICML*, 2019.

[192] J. W. Rae, S. Bartunov, and T. P. Lillicrap, "Meta-learning Neural Bloom Filters," *ICML*, 2019.

[193] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, "Rapid Learning Or Feature Reuse? Towards Understanding The Effectiveness Of Maml," *arXiv e-prints*, 2019.

[194] B. Kang, Z. Liu, X. Wang, F. Yu, J. Feng, and T. Darrell, "Few-shot Object Detection Via Feature Reweighting," in *ICCV*, 2019.

[195] L.-Y. Gui, Y.-X. Wang, D. Ramanan, and J. Moura, *Few-Shot Human Motion Prediction Via Meta-learning*. Springer, 2018.

[196] A. Shaban, S. Bansal, Z. Liu, I. Essa, and B. Boots, "One-Shot Learning For Semantic Segmentation," *CoRR*, 2017.

[197] N. Dong and E. P. Xing, "Few-Shot Semantic Segmentation With Prototype Learning," in *BMVC*, 2018.

[198] K. Rakelly, E. Shelhamer, T. Darrell, A. A. Efros, and S. Levine, "Few-Shot Segmentation Propagation With Guided Networks," *ICML*, 2019.

[199] J. Gordon, J. Bronskill, M. Bauer, S. Nowozin, and R. E. Turner, "Meta-Learning Probabilistic Inference For Prediction," *ICLR*, 2019.

[200] E. Zakharov, A. Shysheya, E. Burkov, and V. S. Lempitsky, "Few-Shot Adversarial Learning Of Realistic Neural Talking Head Models," *CoRR*, 2019.

[201] T.-C. Wang, M.-Y. Liu, A. Tao, G. Liu, J. Kautz, and B. Catanzaro, "Few-shot Video-to-video Synthesis," *arXiv e-prints*, 2019.

[202] S. E. Reed, Y. Chen, T. Paine, A. van den Oord, S. M. A. Eslami, D. J. Rezende, O. Vinyals, and N. de Freitas, "Few-shot Autoregressive Density Estimation: Towards Learning To Learn Distributions," *CoRR*, 2017.

[203] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, 2015.

[204] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel, "Meta-Learning For Semi-Supervised Few-Shot Classification," *ICLR*, 2018.

[205] A. Antoniou, P. Massimiliano, O. Mateusz, and A. Storkey, "Defining Benchmarks For Continual Few-shot Learning," *arXiv e-prints*, 2020.

[206] S.-A. Rebuffi, H. Bilen, and A. Vedaldi, "Learning Multiple Visual Domains With Residual Adapters," in *NeurIPS*, 2017.

[207] E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P. Manzagol, and H. Larochelle, "Meta-Dataset: A Dataset Of Datasets For Learning To Learn From Few Examples," *ICLR*, 2020.

[208] Y. Guo, N. C. F. Codella, L. Karlinsky, J. R. Smith, T. Rosing, and R. Feris, "A New Benchmark For Evaluation Of Cross-Domain Few-Shot Learning," 2019.

[209] T. de Vries, I. Misra, C. Wang, and L. van der Maaten, "Does Object Recognition Work For Everyone?" in *CVPR*, 2019.

[210] M. W. Gondal, M. Wuthrich, D. Miladinovic, F. Locatello, M. Breidt, V. Volchkov, J. Akpo, O. Bachem, B. Schölkopf, and S. Bauer, "On The Transfer Of Inductive Bias From Simulation To The Real World: A New Disentanglement Dataset," in *NeurIPS*, 2019.

[211] B. Hariharan and R. Girshick, "Low-Shot Visual Recognition By Shrinking And Hallucinating Features," in *ICCV*, 2017.

[212] T. Kim, J. Yoon, O. Dia, S. Kim, Y. Bengio, and S. Ahn, "Bayesian Model-Agnostic Meta-Learning," *NeurIPS*, 2018.

[213] M. Patacchiola, J. Turner, E. J. Crowley, M. O'Boyle, and A. Storkey, "Deep Kernel Transfer In Gaussian Processes For Few-shot Learning," *arXiv e-prints*, 2019.

[214] Y. Wang, W.-L. Chao, K. Q. Weinberger, and L. van der Maaten, "SimpleShot: Revisiting Nearest-Neighbor Classification For Few-Shot Learning," 2019.

[215] R. J. Williams, "Simple Statistical Gradient-Following Algorithms For Connectionist Reinforcement Learning," *Machine learning*, 1992.

[216] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Provably Convergent Policy Gradient Methods For Model-Agnostic Meta-Reinforcement Learning," *arXiv e-prints*, 2020.

[217] O. Sigaud and F. Stulp, "Policy Search In Continuous Action Domains: An Overview," *Neural Networks*, 2019.

[218] L. Kirsch, S. van Steenkiste, and J. Schmidhuber, "Improving Generalization In Meta Reinforcement Learning Using Learned Objectives," in *ICLR*, 2020.

[219] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning With A Stochastic Actor," in *ICML*, 2018.

[220] O. Kroemer, S. Niekum, and G. D. Konidaris, "A Review Of Robot Learning For Manipulation: Challenges, Representations, And Algorithms," *CoRR*, 2019.

[221] A. Jabri, K. Hsu, A. Gupta, B. Eysenbach, S. Levine, and C. Finn, "Unsupervised Curricula For Visual Meta-Reinforcement Learning," in *NeurIPS*, 2019.

[222] Y. Yang, K. Caluwaerts, A. Iscen, J. Tan, and C. Finn, "Norml: No-reward Meta Learning," in *AAMAS*, 2019.

[223] S. K. Seyed Ghasemipour, S. S. Gu, and R. Zemel, "SMILe: Scalable Meta Inverse Reinforcement Learning Through Context-Conditional Policies," in *NeurIPS*, 2019.

[224] 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," *Journal of Artificial Intelligence Research*, 2018.

[225] A. Nichol, V. Pfau, C. Hesse, O. Klimov, and J. Schulman, "Gotta Learn Fast: A New Benchmark For Generalization In RL," *CoRR*, 2018.

[226] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying Generalization In Reinforcement Learning," *ICML*, 2019.

[227] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," 2016.

[228] C. Packer, K. Gao, J. Kos, P. Krähenbühl, V. Koltun, and D. Song, "Assessing Generalization In Deep Reinforcement Learning," *arXiv e-prints*, 2018.

[229] C. Zhao, O. Siguad, F. Stulp, and T. M. Hospedales, "Investigating Generalisation In Continuous Deep Reinforcement Learning," *arXiv e-prints*, 2019.

[230] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A Benchmark And Evaluation For Multi-task And Meta Reinforcement Learning," *CORL*, 2019.

[231] A. Bakhtin, L. van der Maaten, J. Johnson, L. Gustafson, and R. Girshick, "Phyre: A New Benchmark For Physical Reasoning," in *NeurIPS*, 2019.

[232] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training Deep Networks With Synthetic Data: Bridging The Reality Gap By Domain Randomization," in *CVPR*, 2018.

[233] A. Kar, A. Prakash, M. Liu, E. Cameracci, J. Yuan, M. Rusiniak, D. Acuna, A. Torralba, and S. Fidler, "Meta-Sim: Learning To Generate Synthetic Datasets," *CoRR*, 2019.

[234] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning Transferable Architectures For Scalable Image Recognition," in *CVPR*, 2018.

[235] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling For Convolutional Neural Networks," *ICML*, 2019.

[236] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching For Mobilenetv3," in *ICCV*, 2019.

[237] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware Neural Architecture Search For Mobile," in *CVPR*, 2019.

[238] J. Kim, Y. Choi, M. Cha, J. K. Lee, S. Lee, S. Kim, Y. Choi, and J. Kim, "Auto-Meta: Automated Gradient Based Meta Learner Search," *CoRR*, 2018.

[239] T. Elsken, B. Staffler, J. H. Metzen, and F. Hutter, "Meta-Learning Of Neural Architectures For Few-Shot Learning," 2019.

[240] L. Li and A. Talwalkar, "Random Search And Reproducibility For Neural Architecture Search," *arXiv e-prints*, 2019.

[241] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-Bench-101: Towards Reproducible Neural Architecture Search," in *ICML*, 2019.

[242] X. Dong and Y. Yang, "NAS-Bench-201: Extending The Scope Of Reproducible Neural Architecture Search," in *ICLR*, 2020.

[243] P. Tossou, B. Dura, F. Laviolette, M. Marchand, and A. Lacoste, "Adaptive Deep Kernel Learning," *CoRR*, 2019.

[244] S. Ravi and A. Beatson, "Amortized Bayesian Meta-Learning," in *ICLR*, 2019.

[245] Z. Wang, Y. Zhao, P. Yu, R. Zhang, and C. Chen, "Bayesian meta sampling for fast uncertainty adaptation," in *ICLR*, 2020.

[246] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep Learning Face Attributes In The Wild," in *ICCV*, 2015.

[247] K. Hsu, S. Levine, and C. Finn, "Unsupervised Learning Via Meta-learning," *ICLR*, 2019.

[248] S. Khodadadeh, L. Boloni, and M. Shah, "Unsupervised Meta-Learning For Few-Shot Image Classification," in *NeurIPS*, 2019.

[249] A. Antoniou and A. Storkey, "Assume, Augment And Learn: Unsupervised Few-shot Meta-learning Via Random Labels And Data Augmentation," *arXiv e-prints*, 2019.

[250] V. Garg and A. T. Kalai, "Supervising Unsupervised Learning," in *NeurIPS*, 2018.

[251] Y. Jiang and N. Verma, "Meta-Learning To Cluster," 2019.

[252] B. Settles, "Active Learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2012.

[253] K. Javed and M. White, "Meta-learning Representations For Continual Learning," in *NeurIPS*, 2019.

[254] A. Sinitsin, V. Plokhotnyuk, D. Pyrkin, S. Popov, and A. Babenko, "Editable Neural Networks," in *ICLR*, 2020.

[255] K. Muandet, D. Balduzzi, and B. Schölkopf, "Domain Generalization Via Invariant Feature Representation," in *ICML*, 2013.

[256] D. Li and T. Hospedales, "Online Meta-Learning For Multi-Source And Semi-Supervised Domain Adaptation," *arXiv e-prints*, 2020.

[257] D. Li, Y. Yang, Y.-Z. Song, and T. Hospedales, "Deeper, Broader And Artier Domain Generalization," in *ICCV*, 2017.

[258] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *ICCV*, 2019.

[259] T. Miconi, J. Clune, and K. O. Stanley, "Differentiable Plasticity: Training Plastic Neural Networks With Backpropagation," in *ICML*, 2018.

[260] T. Miconi, A. Rawal, J. Clune, and K. O. Stanley, "Backpropamine: Training Self-modifying Neural Networks With Differentiable Neuromodulated Plasticity," in *ICLR*, 2019.

[261] J. Devlin, R. Bunel, R. Singh, M. J. Hausknecht, and P. Kohli, "Neural Program Meta-Induction," in *NIPS*, 2017.

[262] X. Si, Y. Yang, H. Dai, M. Naik, and L. Song, "Learning A Meta-solver For Syntax-guided Program Synthesis," *ICLR*, 2018.

[263] P. Huang, C. Wang, R. Singh, W. Yih, and X. He, "Natural Language To Structured Query Generation Via Meta-Learning," *CoRR*, 2018.

[264] Y. Xie, H. Jiang, F. Liu, T. Zhao, and H. Zha, "Meta Learning With Relational Information For Short Sequences," in *NeurIPS*, 2019.

[265] J. Gu, Y. Wang, Y. Chen, V. O. K. Li, and K. Cho, "Meta-Learning For Low-Resource Neural Machine Translation," in *EMNLP*, 2018.

[266] Z. Lin, A. Madotto, C. Wu, and P. Fung, "Personalizing Dialogue Agents Via Meta-Learning," *CoRR*, 2019.

[267] J.-Y. Hsu, Y.-J. Chen, and H. yi Lee, "Meta Learning For End-to-End Low-Resource Speech Recognition," in *ICASSP*, 2019.

[268] G. I. Winata, S. Cahyawijaya, Z. Liu, Z. Lin, A. Madotto, P. Xu, and P. Fung, "Learning Fast Adaptation On Cross-Accented Speech Recognition," *arXiv e-prints*, 2020.

[269] O. Klejch, J. Fainberg, and P. Bell, "Learning To Adapt: A Meta-learning Approach For Speaker Adaptation," *Interspeech*, 2018.

[270] 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*, 2019.

[271] G. Maicas, A. P. Bradley, J. C. Nascimento, I. D. Reid, and G. Carneiro, "Training Medical Image Analysis Systems Like Radiologists," *CoRR*, 2018.

[272] 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," *arXiv e-prints*, 2019.
[273] Z. Mirikharaji, Y. Yan, and G. Hamarneh, "Learning To Segment Skin Lesions From Noisy Annotations," *CoRR*, 2019.
[274] D. Barrett, F. Hill, A. Santoro, A. Morcos, and T. Lillicrap, "Measuring Abstract Reasoning In Neural Networks," in *ICML*, 2018.
[275] K. Zheng, Z.-J. Zha, and W. Wei, "Abstract Reasoning With Distracting Features," in *NeurIPS*, 2019.
[276] B. M. Lake, "Compositional Generalization Through Meta Sequence-to-sequence Learning," in *NeurIPS*, 2019.
[277] B. Dai, C. Zhu, and D. Wipf, "Compressing Neural Networks Using The Variational Information Bottleneck," *ICML*, 2018.
[278] J. Kossaifi, A. Bulat, G. Tzimiropoulos, and M. Pantic, "T-net: Parametrizing Fully Convolutional Nets With A Single High-order Tensor," in *CVPR*, 2019.
[279] Z. Liu, H. Mu, X. Zhang, Z. Guo, X. Yang, K.-T. Cheng, and J. Sun, "Metapruning: Meta Learning For Automatic Neural Network Channel Pruning," in *ICCV*, 2019.
[280] T. OShea and J. Hoydis, "An Introduction To Deep Learning For The Physical Layer," *IEEE Transactions on Cognitive Communications and Networking*, 2017.
[281] Y. Jiang, H. Kim, H. Asnani, and S. Kannan, "MIND: Model Independent Neural Decoder," *arXiv e-prints*, 2019.
[282] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining And Harnessing Adversarial Examples," in *ICLR*, 2015.
[283] C. Yin, J. Tang, Z. Xu, and Y. Wang, "Adversarial Meta-Learning," *CoRR*, 2018.
[284] D. Kang, Y. Sun, D. Hendrycks, T. Brown, and J. Steinhardt, "Testing Robustness Against Unforeseen Adversaries," *arXiv e-prints*, 2019.
[285] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient Surgery For Multi-Task Learning," 2020.
[286] Z. Kang, K. Grauman, and F. Sha, "Learning With Whom To Share In Multi-task Feature Learning," in *ICML*, 2011.
[287] Y. Yang and T. Hospedales, "A Unified Perspective On Multi-Domain And Multi-Task Learning," in *ICLR*, 2015.
[288] K. Allen, E. Shelhamer, H. Shin, and J. Tenenbaum, "Infinite Mixture Prototypes For Few-shot Learning," in *ICML*, 2019.
[289] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling Task Transfer Learning," in *CVPR*, 2018.