Meta-Learning and Meta-Reinforcement Learning - Tracing the Path towards DeepMind's Adaptive Agent

Björn Hoppmann, Christoph Scholz

May 7, 2026

Abstract

Humans are highly effective at utilizing prior knowledge to adapt to novel tasks, a capability that standard machine learning models struggle to replicate due to their reliance on task-specific training. Meta-learning overcomes this limitation by allowing models to acquire transferable knowledge from various tasks, enabling rapid adaptation to new challenges with minimal data. This survey provides a rigorous, task-based formalization of meta-learning and meta-reinforcement learning and uses that paradigm to chronicle the landmark algorithms that paved the way for DeepMind's Adaptive Agent, consolidating the essential concepts needed to understand the Adaptive Agent and other generalist approaches.

1 Introduction

Humans excel at reusing prior knowledge to adapt rapidly to novel tasks. In contrast, standard machine learning models are typically trained for a single task on a large amount of task-specific data. Optimized for task-specific peak performance, they excel in trained domains but struggle to generalize to new tasks. Meta-learning (or learning how to learn) seeks to overcome this issue by extracting higher-level knowledge and strategies from a distribution of distinct yet related tasks, enabling models to adapt quickly to new challenges with minimal additional data.

Historically, the conceptual groundwork for self-adapting artificial intelligence was laid by [134]. Numerous studies in the subsequent decades established the theoretical foundations of meta-learning [148], e.g., by incorporating inductive bias shifts into the framework [135] (see [167], [72], [148] for a detailed historic review). The field experienced another resurgence, concurrent with the rise of deep learning, when substantially larger datasets and computational resources became available. By introducing Model-Agnostic Meta-Learning (MAML), [42] proposed the first landmark algorithm in the class of gradient-based meta-learners, the abstraction of gradient-based learning to the meta-level (see Section 3.1). Around the same time, [34] introduced RL^2, the first landmark of the class of memory-based learners (see Section 3.2). Since then, meta-learning techniques have diversified across few-shot image classification [64], [51], [91], neural architecture search [36], [68], [129], [118], natural language processing [184], [83], [84], reinforcement learning (see Section 2.2), and application fields like

- robotics, where meta-learners learn novel strategies from only a few demonstrations [45], or experiences [75].
- healthcare [122], where patient- or disease-specific data is often sparse [154], [99].
- adaptive control [102], [35], where system parameterizations change over time.

Even for the preparation of space missions [50], researchers use meta-learning to pre-train models that can rapidly adapt in real time to changing environmental conditions. Consequently, several works survey meta-learning by categorizing different sub-classes of meta-learning [9], [169], differentiating between the numerous related topics [169], [7], [166], addressing open problems [9], or reviewing overlaps between meta-learning and its most related fields [166].

However, the literature lacks a compact, rigorous mathematical treatment tying meta-learning theory to practical implementations: Performance measures are frequently illustrated informally (e.g., in figures or captions) but are rarely defined mathematically, which complicates fair comparisons between methods. Additionally, practical concepts such as validation or meta-validation are seldom formalized. This issue is particularly acute in meta-RL, where agents' actions influence data collection; accordingly, cumulative reward must be explicitly incorporated into the meta-objective. But a careful understanding of meta-learning and meta-RL paradigms is highly relevant - perhaps more than ever. As large black-box foundation models and generalist agents scale, they increasingly exhibit emergent capabilities that reproduce behaviours previously engineered via bespoke training schemes (see Section 4). Without precise formalisms and metrics, it remains difficult to assess whether and to what extent such capabilities constitute genuine meta-learning - a problem that is particularly acute for newcomers to the field.

Contribution and Paper Outline

The primary goal of this work is to close the gap in formalism by providing a rigorous formalization of meta-learning and meta-RL within the task-based paradigm. This survey thus offers newcomers an accessible entry point into the field. Meta-learning is distinguished from its closest neighbors in Appendix B; however, related areas such as continual learning, self-supervised learning, and active learning remain outside the scope. For these topics, readers are referred to [169] or to dedicated surveys. Furthermore, this work does not address metric-based meta-learning, as it contributes little to meta-RL and the Adaptive Agent. Those specifically interested in this sub-field are directed to other surveys such as [167] or [72]. By contrast, the central contribution of this survey is a chronological presentation of the landmark developments culminating in generalist agents such as DeepMind's Adaptive Agent (ADA), using the task-based meta-learning paradigm as a unifying framework to consolidate the essential knowledge. For this purpose, this work is organized as follows:

- It derives meta-learning from standard supervised learning (Section 2.1) and systematically transfers the resulting notions and formulas to the meta-RL setting (Section 2.2). As part of this unified mathematical framework, Section 2.3 carefully defines the most important meta-learning performance measures that are treated informally in the literature. Additionally, the mathematical derivation of Bayesian Reinforcement Learning - which is not a central concept of this work - can be found in Appendix A.
- In Section 3, this work applies the formal paradigm as a consistent interpretive lens to present the landmarks on the path from early meta-learners to ADA. The paradigm of Section 2 serves as the organizing thread: each new algorithm is situated within the same formalism to clarify how the field evolved. The resulting timeline begins with the simpler family of gradient-based meta-learners (Section 3.1), then presents memory-based algorithms. The memory-based sequence starts with the simpler class of RNN-based meta-learners (Section 3.2) and then gradually increases in complexity until the Adaptive Agent (Section 3.5), is finally presented.
- It analyzes ADA as a representative meta-RL instance of a large-scale generalist agent; thereby discussing the key scaling and enhancement techniques (e.g., distillation, automated curriculum learning), and positions these mechanisms within the formal paradigm to explain how they contribute to ADA's capabilities.

Finally, this work examines emergent capabilities, interpretability challenges, and open problems in Section 4, before giving an outlook framed by the "Three Pillars of General Intelligence", and concluding in Section 5.

2 Paradigm

This section formally introduces meta-learning and meta-RL by comparing them to standard machine and reinforcement learning respectively. It, thereby, creates a consistent structure of terms, notions and performance measures (in Section 2.3 used throughout the rest of the entire work.

2.1 Meta-Learning

In standard machine learning a learner f_θ with parameters θ is trained to solve a particular task T of the form [1]

$$T := (\mathcal{L}, \mathcal{X}, \mu, \mathbb{T}, h), \tag{1}$$

by minimizing the loss function $\mathcal{L} : \mathcal{X} \to \mathbb{R}$ on some training data X_{train} out of the observation space $\mathcal{X} \subseteq \mathbb{R}^d$. The transition $\mathbb{T} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ represents the transition from one observation to another [2], while μ denotes the initial distribution and h the horizon of the task.

Example 1 - Image Classification:

Let X be a labeled image set out of the observation space $\mathcal{X}$ of images showing cats or dogs, and let $X_{\text{train}}, X_{\text{val}}$ and X_{test} be disjoint subsets of X with a split of 80% training data and 10% validation and test data. Then, the task T to classify images between 0 (dog) or 1 (cat) is formally defined by defining the components in (1), i.e.,:

- One can select any loss function $\mathcal{L}$ that is suitable for classification, e.g., the cross entropy loss.
- Since all images in X_{train} are equally likely to be sampled, the initial distribution μ is the uniform distribution conditioned on X_{train}.
- The transition can be defined as the probability of sampling image pair $(x_1, x_2) \in \mathcal{X} \times \mathcal{X}$. However, the concrete definition depends on the specific type of sampling (with or without replacement). item Since each shot in image classification consists of only one image, the horizon h equals 1 [a].

The goal is to find the optimal function f^* correctly classifying any image as cat or dog [b] by minimizing $\mathcal{L}_\theta$ w.r.t. θ. The notation $\mathcal{L}_\theta$ denotes that observations $x \in \mathcal{X}$ are processed or collected via the function f_θ. The learner f_θ can be any suitable machine learning model, e.g., a convolutional neural network processing images. This work focuses on deep learning so that the parameters θ are assumed to always correspond to the weights of a deep neural network throughout. The validation and test sets X_{val} and X_{test} contain labeled cat and dog images excluded from training. They allow one to evaluate whether the learner f_θ truly learned to distinguish cats from dogs or only overfitted by memorizing the images in X_{train}. Validation runs during training to monitor progress; testing runs once after training to report final performance. But this way of testing the model f_θ does not provide any information if the same learner could also distinguish between cats and horses since the latter class does not exist in the task's observation space $\mathcal{X}$.

[a] In sampling-based tasks, such as classification,horizon, transition, and initial distribution play a rather theoretical role. Therefore, they are only shown here for the sake of completeness. Example 2 illustrates the case of a sequence-based task, where those components are more meaningful.

[b] The optimal function f_T^* is not necessarily unique, but can, without loss of generality, be assumed as one particular function, as each optimal function, is per definition, as good as any other.

The loss function $\mathcal{L}$ as well as all other task-specific components, such as observation space $\mathcal{X}$ or transition $\mathbb{T}$, remain unchanged throughout the whole training and testing process. As a consequence, the model f_θ is tailored to solve the task T, while it generally fails to generalize to out-of-domain tasks, i.e., to tasks with different loss, transition, or observation space [166]. However, for many tasks the number of training examples available is not sufficient for learning, and sometimes training a (standard) learner for a new but very similar task fully from scratch is too costly in terms of computation power or training time.

For these reasons, meta-learning - unlike standard machine learning - seeks to capture general knowledge across a family of similar tasks, enabling rapid, task-specific adaptation, i.e., adaptation to a task within K shots rather than with a full training [3]. Modifying the observation space $\mathcal{X}$ of Example 1 to contain labeled images of several different animals, but only ten images per class, one obtains a meta-learning problem. With the same training-validation-test split as in Example 1, this means eight images per class

[1] This formalization of a task is inspired by [42], [166], [125].

[2] Note that this is the most general way of defining the transition function. In many cases, it is defined as the probability distribution of observing observation pairs $(x_1, x_2) \in \mathcal{X} \times \mathcal{X}$. Hence, one must, technically, define the transition as $\mathbb{T} : \sigma(\mathcal{X} \times \mathcal{X}) \to [0, 1]$. However, as all observations $x \in \mathcal{X}$ should generally be measurable, the σ function is redundant notation

[3] If only one shot or even no shot is used for fine-tuning, one speaks of one-shot or zero-shot learning, respectively.

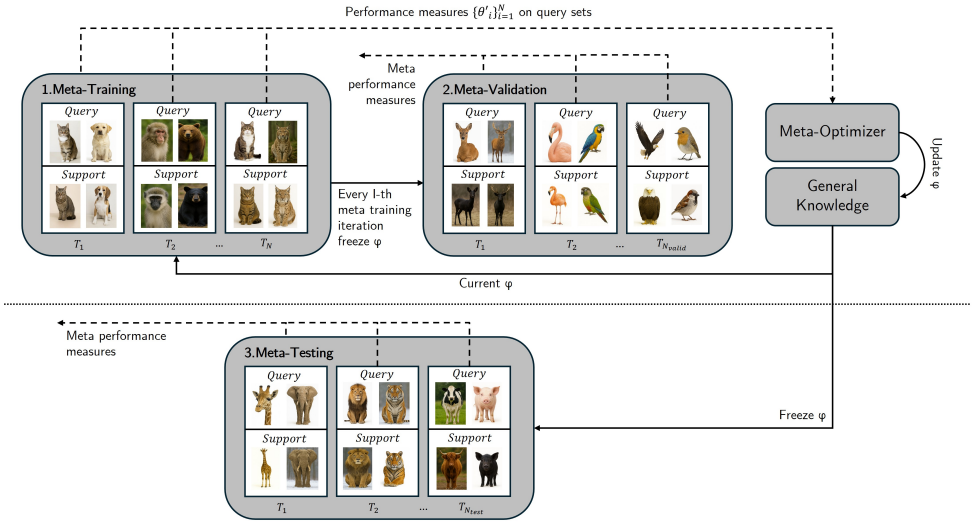

Figure 1: Meta-learning of 2-way 1-shot animal classification tasks. The current meta-knowledge φ is the prior for one-shot learning of each particular classification task. During meta-training, the meta-optimizer receives all N query set losses of the adapted models to update meta-knowledge φ. Meta-validation evaluates the training progress on new classification problems every l meta-epochs, while meta-testing on unseen classifications takes place after meta-training.

for training and only one for validation and testing. This small amount of training data does not suffice for training a standard learner properly. However, on a meta-level, distinguishing between all these different animals does not differ much from the task in Example 1. Quite the opposite is the case since, on an abstract level, classifying dogs vs. cats requires similar high-level skills as classifying cats vs. horses: The model needs to identify the creature in the picture first, recognize shapes of noses, ears, or other body parts, examine the silhouette, etc. In other words, classifying cats vs. horses is just another task from the task distribution over different animal classification tasks of the form presented in Example 1. In this way, the meta-task to classify any animal can be separated into several simplified tasks that share some common, high-level structure and belong to the same "meta-problem" of classifying animals. The following paragraphs formalize the meta-learning paradigm, while Figure 1 applies it to the meta-problem of classifying different animals.

The Meta-Learning Paradigm

Formally, the meta-learning paradigm [158] consists of a distribution $p(T)$ over tasks of the form

$$T_i := (\mathcal{L}_i, \mathcal{X}, \mu_i, \mathbb{T}_i, h). \tag{2}$$

Within this framework, each task of the form (2) may possess its own loss function, transition dynamics, or initial distribution, while the observation space $\mathcal{X}$ and the horizon h are generally assumed to be identical across tasks drawn from the same distribution p (see e.g., [42], [125]). This work follows this convention, although it is not a necessary condition. However, some works specifically focus on domain generalization [88], [165]. The definition of the observation space highly depends, among other factors, on the underlying problem, and so does the meaning of the notion "'identical"': In the meta-task of classifying animals, it can be defined to contain all images of animals that can potentially occur within the distribution $p(T)$ of different animal classification tasks. But the format (e.g., grey-scale vs. RGB images) needs to be identical. For other task distributions, e.g., distributions of RL tasks as discussed in Section 2.2, defining the observation space might mean something completely different.

In the task-based paradigm, there are two components of what must be learned from a theoretical point of view: Common knowledge over all tasks T_i in $p(T)$, and task-specific knowledge gained through few-shot fine-tuning. Hence, the formal learning paradigm consists of two stages, the more abstract meta-level and the few-shot standard learning of a particular task $T_i \sim p(T)$. This paradigm is explicitly applied to all gradient-based meta-learning algorithms presented in Section 3.1. However, even for the memory-based meta-learners presented in Sections 3.2 and 3.4 that do not explicitly divide learning into meta-learning

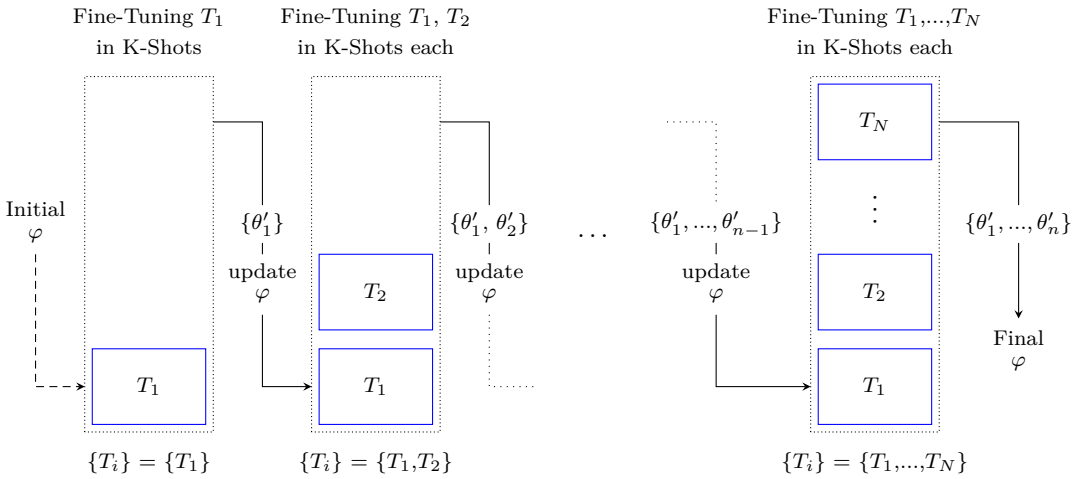

Figure 2: General Meta-Training. In each iteration a new task T_i is sampled from the family $p(T)$. The meta-variable φ is the prior for individual K shot fine-tuning of each task. The resulting parameters θ'_i of each task are used to update φ.

and task-specific standard learning, the meta-training paradigm can be formalized as (implicitly) two-stage by introducing a meta-variable φ encoding common knowledge over all tasks. As a consequence, the training-validation-test split is also two-stage. On meta-task level, certain tasks get explicitly excluded from the meta-training task pool to function as validation throughout meta-training or as test tasks after meta-training. The corresponding evaluation takes place on the stage of task-specific standard learning, which is why it is often referred to as inner learning. However, since every task T_i from the distribution $p(T)$ is assumed to be a few-shot learning task, a task-specific validation set X^i_{val} is not required. This way, standard few-shot learning is mimicked within each task T_i while meta-training provides every such task-specific fine-tuning with a meaningful prior to enable fast adaptation.

Meta-Training and Meta-Testing

The corresponding meta-training scheme aims to optimize φ to be the best prior for fast inner learning. As highlighted in Figure 2, this means yielding task-specific parameters from the general knowledge φ and measure their test set performance after K shots of task-level adaptation. The corresponding optimization problem in each meta-training iteration is

$$\text{Minimize} \quad \mathbb{E}_{T_i \sim p(T)} \mathcal{L}_{\text{meta}} \left(\theta^*_{T_i}(\varphi),\ \varphi,\ X^i_{\text{test}} \right) \quad \text{wrt. } \varphi. \tag{3}$$

The notation $\theta_i(\varphi)$ highlights the relationship between φ and the initial parameters of the inner learning, while $\mathcal{L}_{\text{meta}}$ denotes the meta loss function and $\theta^*_i(\cdot)$ the optimal task-specific parameters for Task T_i when starting inner learning with meta-parameter φ. However, the optimal task-specific parameters $\theta^*_i(\cdot)$ are rather a theoretical component of the formalization of meta-training taken from [166] than a practical implementation.

Inner learning aims to approximate the task-specific optimal parameters θ^*_i representing the optimal solution of the task T_i. Thus, the resulting parameters must be denoted differently to distinguish them from the optimal ones. Throughout this work, the respective notation is $\theta'_i := \theta'_i(\varphi)$ for parameters trained in inner learning starting from $\theta_i(\varphi)$ i.e., from initial parameters θ yielded from the prior φ. The resulting optimization problem for inner learning is

$$\text{Minimize} \quad \mathcal{L}_i \left(\theta_i, \theta_i(\varphi), X^i_{\text{train}} \right) \quad \text{wrt. } \theta_i. \tag{4}$$

The choice of the meta-loss $\mathcal{L}_{\text{meta}}$ and the inner learning determines the respective meta-learning framework. This holds for all algorithms presented in Section 3, although the meta-loss is not explicitly given for the memory-based meta-learners in Section 3.2. Moreover, the task sampling throughout meta-training is algorithm-specific. Figure 2 shows a scheme where tasks are iteratively drawn from $p(T)$ up to N tasks, but other algorithms might directly start with N tasks and re-sample them, respectively. The

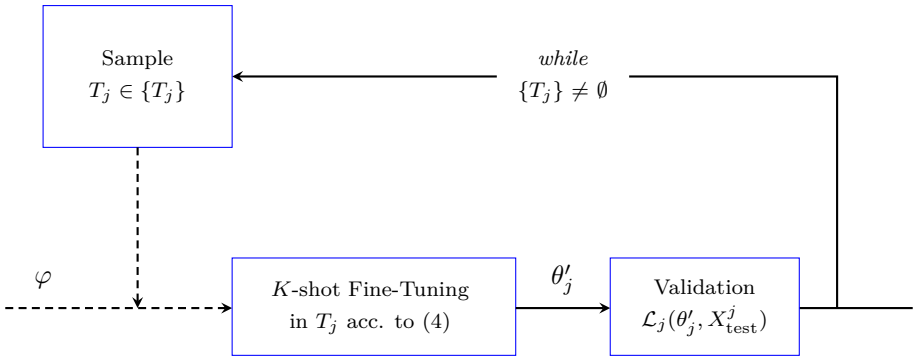

Figure 3: General Meta-Testing Paradigm. For each T_j sampled from the set of test tasks the parameters $\theta_j(\varphi)$ are fine-tuned in K shots, before the resulting θ'_j get evaluated on the task's test set via the task-specific loss $\mathcal{L}_j$ to yield the performance.

numbers N, N_{val} and N_{test} of training, validation, and testing tasks are themselves hyperparameters of meta-training.

After meta-training a meta-model f_φ, meta-testing aims to evaluate the trained model on unseen test tasks. However, evaluating the quality of a meta-model means examining how well it boosts task-specific learning. Hence, the task-specific parameters $\theta_j(\varphi)$ are adapted within K shots of inner learning for each test task, while only the meta-variable φ is fixed throughout the whole meta-testing process. Evaluating the correspondingly adapted task-level parameters θ'_j on the task-level test set X^j_{test} yields the task-level performance required for the meta-performance measures presented in the next section. Figure 3 shows this iterative meta-testing scheme.

2.2 Meta-Reinforcement Learning

This subsection transfers the general meta-learning paradigm to that of meta-Reinforcement Learning (meta-RL). Mimicking the structure of Section 2.1, it starts with presenting the standard reinforcement learning (RL) paradigm and thereafter derives the meta-RL paradigm from it.

Standard Reinforcement Learning

Standard RL is a special case of the standard machine learning paradigm, where an agent interacts with an environment $(\mathbb{A}, \mathbb{S}, R)$ by selecting an action $a \in \mathbb{A}$ based on the state $s \in \mathbb{S}$ and receiving the next state $s' \in \mathbb{S}$ and a reward signal $r \in \mathbb{R}$. Example 2 illustrates this agent-environment interaction, where the transition of a general task (1) extends to a distribution $\mathbb{T} : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to \mathbb{R}$ to additionally denote the effect of the agent's action on the subsequent state. The reward function $R : \mathbb{A} \times \mathbb{S} \to \mathbb{R}$ and the discount factor $\gamma \in [0, 1]$ determine the task loss $\mathcal{L}$. The reward as well as the transition function must fulfill the Markov property so that the standard task (1) forms a Markov Decision Process (MDP)

$$M := (\mathbb{A}, \mathbb{S}, R, \gamma, \mathbb{T}, \mu, h), \tag{5}$$

with action space $\mathbb{A}$, state space $\mathbb{S}$, episode horizon h [4] and initial state distribution μ. However, as the underlying MDP M_i determines the corresponding RL task T_i, these terms will be used synonymously throughout this work.

Example 2 - Racing games:

Considering a racing game, where the agent's goal is to drive in a way that finishes a racing track as fast as possible, the different components of the underlying MDP (5) are defined as follows:

- The set of possible actions $\mathbb{A}$ contains all directions in which the racer can move, its speed, and whether or not to brake.
- The state space $\mathbb{S}$ contains all states the racer can possibly find itself in. Additional to the

[4] RL problems do not have to be episodic, but $h < \infty$ is a rather practical condition.

current racing track (as pixels), it may include information on the racer's speed as well as the time since the start of the race or other useful information a player might have.

- The initial distribution over states μ refers to what the racer "sees" in its starting position. If that position is deterministic, μ is deterministic, too.
- The reward function can easily be modeled as one when reaching the goal and zero otherwise.
- The discount factor γ must be smaller than one to encourage the agent to finish the racing track as fast as possible.
- The transition models the effect any action $a \in \mathbb{A}$ has on any state s [a]. Since most racing tracks are entirely deterministic, this connection is also likely to be deterministic.
- Each race is an episode. Hence, the time the agent needs to finish the racing track determines the horizon h, which is therefore dynamic.

The strategy π is the theoretical decision rule of the agent. It is representative of the agent, as it determines, how the racer actually drives at any time in the race.

[a] Technically, the general transition introduced in the previous section is the probability of observing tupel $(s, a, s') \in \mathbb{S} \times \mathbb{A} \times \mathbb{S}$. However, in most cases this simplified version suffices.

The loss $\mathcal{L}$ of a RL task is often defined as the value function [5]

$$\mathcal{L} := -\mathbb{E}^{s_0 \sim \mu}_{\pi_\theta(\cdot)} \sum_{t=0}^{h} \gamma^t r_{t+1} \tag{6}$$

where $r_{t+1} = R(s_t, a_t)$ and $\mathbb{E}^{s_0 \sim \mu}_{\pi_\theta(\cdot)}$ denotes the expectation under the assumptions that s_0 is drawn from the initial distribution μ and every action a_t is taken according to policy $\pi_\theta(s_t)$. Correspondingly, in a RL task, the goal is to find an action selection strategy $\pi : \mathbb{S} \to \mathbb{A}$ that maximizes the reward collected throughout an episode of agent-environment interaction.

As this work only considers deep RL, the policy π_θ is defined by the agent's neural network weights θ. There are almost as many rules for updating θ as there are RL algorithms, and not all of them explicitly minimize (6) [149], [48]. For example, the family of function approximation RL algorithms aims to approximate the loss (6) (or modifications of it) by optimizing θ to best approximate the loss. Afterwards, they achieve the loss minimization by acting accordingly.

To learn the optimal strategy π^*_i for a particular MDP M_i, an agent must explore the environment to uncover new actions and their potential rewards. At the same time, it must exploit its gained knowledge to maximize immediate rewards. This trade-off is called the exploration-exploitation dilemma, and it motivates the Bayesian RL paradigm presented in Appendix A.

The Meta-Reinforcement Learning Paradigm

Interacting with an environment can be extremely financially expensive e.g., in robotics, trading or transportation. For such problems, an agent must be trained within a simulation and, thereafter, adapt as fast as possible to the real world. Moreover, environmental dynamics often change over time e.g., when shocks permanently increase market prices, when a robot is assigned to a different task within the same physical environment or when a racing track becomes more slippery due to heavy rain. In such scenarios, a standard RL agent must often be trained fully from scratch, although it has already gained a lot of knowledge about the environment that is still valid. For these reasons, meta-RL aims to collect general knowledge over a family of similar MDPs.

One can easily modify Example 2 into such a meta-RL problem by parameterizing the racing track's slipperiness. For each different value of slipperiness the transition $\mathbb{T}$ of the underlying MDP slightly changes, while the overall structure of state and action spaces and even reward remain unchanged. Such

[5] The agent does not necessarily have to only update its parameters at the end of an episode, although this is assumed throughout this paper. However, generalizing the value function to the expected future reward starting from any state s_t at any time t only requires for an index shift within the sum.

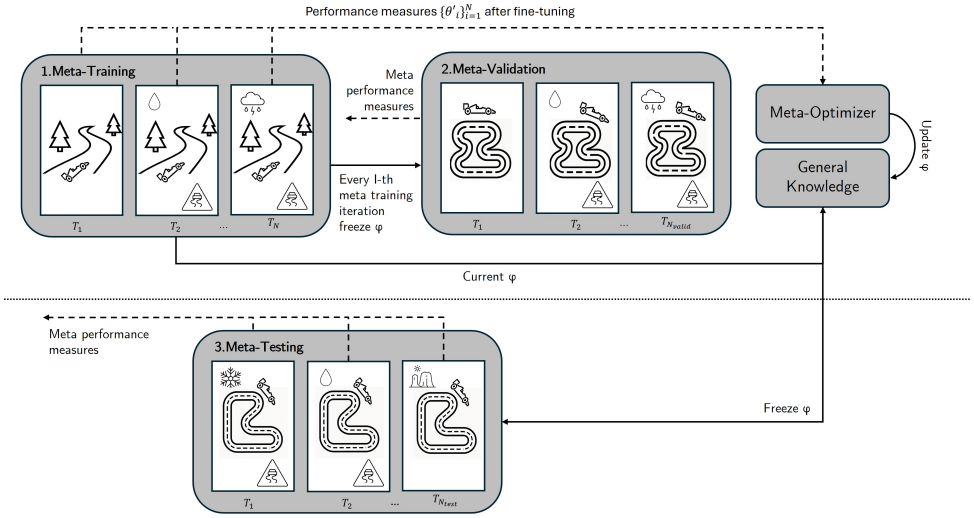

Figure 4: Meta-Reinforcement learning to race on tracks with varying weather conditions. Starting from the meta-knowledge φ, K episodes of fine-tuning on the particular racing track yield performance measures. The meta-optimizer uses these measures to update meta-knowledge φ. Meta-validation evaluates the training progress on unseen racing tracks every l meta-epochs. After meta-training, the meta-policy adapts to test tracks to evaluate the quality of prior φ.

families of MDPs are called parametric. Acting optimally in any of these environments requires almost the same high-level skills: The agent still needs to learn driving dynamics, when to brake and which direction to drive to, just with a different level of drifting in curves due to a change in slipperiness. In other words, the meta-problem to finish a randomly slippery racing track as fast as possible (see Figure 4 for an illustration), remains the same.

Formally, the meta-RL paradigm - analogous to general meta-learning - consists of a distribution $p(M)$ over MDPs of the form

$$M_i := (\mathbb{S}, \mathbb{A}, R_i, \mathbb{T}_i, \mu_i, \gamma, h). \tag{7}$$

The MDPs from the same family $\{M_i\}_{M_i \sim p(M)}$ normally vary with respect to their underlying dynamics $\mathbb{T}_i$ or reward function R_i, while sharing some structure like state and action space - which, together, can be interpreted as the observation space $\mathcal{X}$ of a general task (2). However, these assumptions are more practical than a theoretical necessity.

Collecting knowledge within a family of MDPs (possibly) means two similar but different things: Collecting general knowledge about structures like state or action space shared within the family (see sections 3.2 and 3.4), or identifying the current MDP as fast as possible to adjust the agent's policy π_θ accordingly (see sections C and 3.3). For racing MDPs of the form presented in Example 2 the former means learning how to process the pixels shown as the current state, remembering general information such as the route of the race, and understanding how to drive a car on an abstract level. The latter corresponds to the exploration/exploitation dilemma on MDP-level. It requires determining as quickly as possible how slippery the racing track actually is in order to adjust the style of driving accordingly. As a consequence, meta-RL is therefore a two-stage process: On meta-level common knowledge over all MDPs is collected, while on MDP-level the current MDP is identified to adjust the MDP-specific policy. The training-validation-test split is analogous to that of meta-learning.

Meta-Training and Meta-Testing

Both the inner optimization (4) as well as the outer optimization (3) are straightforward to adjust to the meta-RL training scheme by introducing a meta-variable φ representing meta-knowledge. The only core structural difference is in the MDP-specific training and testing data X^i_{train} and X^i_{test}: As the agent has to interact with its environment to observe the subsequent state and reward, sampling an observation set $X \in \mathcal{X}$ always depends on the current policy. one typically refers to this as "collecting experience" of the form $X = \{s_t, a_t, r_{t+1}, s_{t+1}\}_{t=0}^h$. Consequently, in the meta-RL paradigm, gathering data from a particular MDP M_i, in K shots, means collecting K episodes of experience within the environment

following the strategy $\pi_{\theta_i(\varphi)}$ that might or might not change between episodes depending on the inner RL algorithm.

The meta-testing scheme in meta-RL is also analogous to the general meta-testing scheme. The only key difference is in the MDP-level testing of the adapted policy π'_{θ_j} as it interacts with its environment for another K_{test} episodes after inner learning without further adaptation. The accumulated (and normalized) reward gained throughout these test episodes represents the test loss $\mathcal{L}^j_{\text{test}}$ required for calculating the performance measures presented in the previous section.

2.3 Performance Measures

In contrast to standard learning, where performance is typically evaluated by measuring a learner's loss on unseen test data, assessing a meta-learner's performance is more challenging (see the Appendix of [142] for a full portfolio of respective plots and tables). Task-level test performance alone is often insufficient to fully characterize adaptation capability during meta-learning. Instead, various measures must be used to draw a more detailed picture of the adaptation capability the model f_φ developed throughout meta-training. However, in many works (e.g., [125], [112], [34]), the notion "performance" exclusively refers to the total loss collected within the task-level test batches of test tasks. Similarly, the notion "asymptotic performance" only refers to the task-level test loss after a full, standard training [6] (see e.g., [125], where it is only implicitly defined). Other measures such as adaptation speed or sample-efficiency are only implicitly shown in tables or graphs (see, e.g., [200], [103], [142]). This makes it difficult to immediately understand results, particularly for newcomers, and impossible to quantify performance across different works. To address these issues, the following paragraphs define, motivate, and discuss the most important meta-learning performance measures.

Generalization: In standard learning, generalization refers to the ability of a learner to perform well on unseen data after training. It is generally quantified by the generalization error, i.e., the difference between the learner's performance on the training set and its performance on the test set:

$$\mathcal{L}_{\text{gen}}(\theta) := \mathcal{L}(\theta, X_{\text{test}}) - \mathcal{L}(\theta, X_{\text{train}}). \tag{8}$$

Analogously, the generalization ability of a meta-learner f_φ can be quantified by the accumulated generalization error over the unseen meta-test tasks:

$$\mathcal{L}^{\text{acc}}_{\text{gen}}(\varphi) := \mathcal{L}^{\text{meta}}_{\text{test}}(\varphi) - \mathcal{L}^{\text{meta}}_{\text{train}}(\varphi), \tag{9}$$

where $\theta'_K(\varphi)$ denotes the adapted parameters after k training shots on the respective task T_j starting from prior φ. The accumulated train and test losses are defined as the sum over the respective (standard learning) train and test losses

$$\mathcal{L}^{\text{meta}}_{\text{train}}(\cdot) := \sum_{T_j \sim p(T)} \mathcal{L}(\cdot, X^j_{\text{train}}), \qquad \mathcal{L}^{\text{meta}}_{\text{test}}(\cdot) := \sum_{T_j \sim p(T)} \mathcal{L}(\cdot, X^j_{\text{test}}),$$

which must be normalized to avoid differently scaled losses throughout different tasks.

In RL, however, comparing results is even more difficult, since observed rewards depend on the current state, the time, the actions taken by the agent, and how the reward function is modelled. Normalizing rewards (e.g. through min-max-scaling) does not equalize the distribution of potential rewards throughout different reward functions, which are, potentially, designed to encode completely different objectives (potentialy implying different optimal strategies). Probably, this is why all meta-RL algorithms presented along the timeline in section 3 examine reward curves on different tasks individually. However, it is still possible to gain a basic understanding of a learner's generalization ability as long as tasks from the same distribution have similar reward functions that represent similar objectives. The generalization performance is simply not quantifiable.

The accumulated generalization error (9) measures the generalization performance at task level. In contrast, one measures meta-generalization by abstracting (9) to the meta-generalization error

$$\mathcal{L}^{\text{meta}}_{\text{gen}}(\varphi) := \mathcal{L}^{\text{test}}_{\text{gen}}(\varphi) - \mathcal{L}^{\text{train}}_{\text{gen}}(\varphi), \tag{10}$$

[6] The notion "asymptotic" implies $K \to \infty$, which, from a theoretical point of view, is what happens in standard learning.

where $\mathcal{L}_{\text{gen}}^{\text{train}}$ and $\mathcal{L}_{\text{gen}}^{\text{test}}$ denote the accumulated generalization error (9) summed over all meta-training tasks or meta-test tasks, respectively. The meta-generalization error evaluates how well the model f_φ generalizes on the meta-level by taking its task-specific generalization capability on the training tasks into account. A good meta-generalization means that φ boosts task-specific learning equally well for training and test tasks, while a bad meta-generalization hints φ to overfit to the training tasks in such a way, that fine-tuning has the maximal success, but this knowledge cannot be transferred to other tasks from the same distribution. The latter is called meta-overfitting.

Adaptation speed: Adaptation speed refers to the rate at which a learner can effectively learn new tasks T_j from a limited number of training steps. It is typically measured by tracking task-specific metrics over test task training iterations and examining the slope of the gained curve (see e.g., [42], Figures 2 and 5, [103], Figure 5, [142], Figure 4). A high slope indicates fast adaptation, and vice versa. In the context of meta-learning, one typically tracks adaptation performance during the individual training of each test task T_j, i.e., the task-specific training loss $\mathcal{L}_{\text{train}}^j$ [7]. This way, one yields an adaptation function $\mu_{\varphi,T_j}^{\text{adapt}} : K \to \mathbb{R}$ with

$$\mu_{\varphi,T_j}^{\text{adapt}}(k) := \mathcal{L}_{\text{train}}^j \left(\theta_j^{(k)}, X_{\text{train}}^{j,(k)} \right), \tag{11}$$

whose derivative w.r.t. the number k of inner learning epochs (e.g., gradient descent steps) is the adaptation speed on the task T_j. Calculating (11) for each test task T_j and each fine-tuning iteration k yields the meta-adaptation performance. However, since the adaptation function (11) is generally not known, one must approximate it by empirically calculating it for different values of k (see, e.g., [42], Tables 1 and 2). In RL, this typically means evaluating (11) episodewise (see, e.g., [200], Figure 5).

Sample-Efficiency: Since gathering data is often difficult or expensive, it is also desirable to develop meta-learners that adapt to unseen situations without the need for large amounts of data. Such learners are referred to as sample-efficient. In contrast to adaptation speed, which measures how fast a model can learn and improve within a certain amount of training iterations regardless of the samples needed, sample-efficiency is about learning effectively from fewer examples regardless of the training iterations required for extracting the data's information. It can be measured by [8]

$$\mu_{\varphi,T_j}^{\text{sample eff}}(S) := \mathcal{L}_{\text{train}}^j \left(\theta_j^{(S)}, X_{X \subseteq X_{\text{train}}^j}^{|X|=S} \right), \tag{12}$$

where $\theta_j^{(S)}$ denotes the parameters $\theta_j(\varphi)$ fine-tuned with S observations. Similar to adaptation, one is typically more interested in the rate at which performance increases for additional samples, i.e., in the (empirical) derivative of (12). In classification, for example, the sample efficiency is directly determined by varying the number K of shots (see e.g., [42], [123], Tables 1 and 5), while, in RL, it typically corresponds to the amount of experience, i.e., number of timesteps, collected by the task-specific agent.

Out-of-distribution: While generalization refers to a learner's ability to perform well on unseen data drawn from the same distribution as the training data, out-of-distribution (OOD) performance specifically evaluates how well a model can handle data that comes from a different distribution. In other words, while generalization measures how well the learner can apply learned patterns to new examples within the same context, OOD assesses a learner's ability to generalize to situations not represented during training. Consequently, high OOD performance indicates robustness and adaptability, while poor performance can lead to failures in real-world applications where conditions may vary significantly from the training scenarios. On meta-level, this means defining the test tasks as the family $\{T_j\}_{T_j \not\sim p(T)}$ of tasks, that are not drawn from the task distribution $p(T)$. Then, measuring the performance means calculating all the metrics mentioned above on these OOD tasks.

3 The Timeline of meta-Reinforcement Learning Landmarks

This section traces the evolution of meta-RL algorithms up to DeepMind's Adaptive Agent, starting with the class of gradient-based meta-learners and then moving to memory-based approaches. Each

[7] Note that, in theory, any task-specific loss can be used for evaluating the adaptation speed. For example, using the task-specific generalization (8) one examines, how fast a model gains generalization capabilities within one particular task T_j. However, this is rather complicated and, hence, none of the works presented in this survey utilize such a sophisticated version of adaptation speed.

[8] Note that, similar to adaptation speed, sample-efficiency can be measured with any kind of task-specific loss.

subsection presents the landmark algorithm of the respective class on the timeline towards ADA, describing the related literature, meta-training and meta-testing schemes, and performance analyses for each landmark algorithm in clearly distinguished paragraphs. Following the previously established paradigm as a guiding framework, each landmark introduces a new concept, allowing the overall complexity to grow gradually towards ADA. Except for gradient-based methods (Section 3.1), the rest of this section focusses entirely on meta-RL. For gradient-based meta-learning, an additional paragraph hence derives the corresponding meta-RL variant from the more general meta-learning approach. The broader family of memory-based (black-box) meta-RL algorithms splits into two subsections: RNN-based methods (Section 3.2) and transformer-based methods (Section 3.4). In between, Section 3.3 discusses task inference, as its landmark (VariBAD) is an RNN-based algorithm extended by Bayesian inference to model task uncertainty. The section closes with ADA, the current state-of-the-art meta-RL algorithm, a generalist, transformer-based agent that integrates common techniques for boosting meta-training such as distillation and auto-curriculum learning.

Table 1 summarizes the advantages and drawbacks of the components presented along the landmarks in this section.

3.1 Gradient-based Meta-Learning

Deep neural networks are gradient-based models, i.e., optimized with stochastic gradient descent (SGD) or modifications like Adam [77] or AdamW [98]. But these optimizers often converge slowly or even fail to converge. Convergence depends heavily on the starting point: starting near the optimum can yield rapid progress, while random initialization often requires many more iterations or leads to a poor local minimum. A second issue is the choice of the learning rate. A rate that is too high can cause instability or oscillation, while choosing it too low can significantly slow down learning [106].

Both these problems could be tackled, if it were possible to a priori place the parameters θ in a sufficiently narrow area around the optimal solution θ^*. Not only would learning be more stable and directed towards that optimum, but a small learning rate would also suffice to approach it within a few steps. This is the main idea of gradient-based meta-learning [9].

Model-Agnostic Meta-Learning

Model-Agnostic Meta-Learning (MAML) [42] is the foundational gradient-based meta-learning algorithm and one of the earliest and simplest landmarks in the evolution of meta-learning and meta-RL. The main idea is to place the meta-level prior φ in a region that is promising for all tasks drawn from the distribution $p(T)$, so that starting each task-specific fine-tuning from that prior, i.e., setting $\theta_i(\varphi) = \varphi$ for all θ_i, K gradient descent steps maximize task-specific performance during fine-tuning. It is broadly applicable to any machine-learning problem where the loss function is sufficiently smooth to compute gradients. As a consequence, many works have successfully adapted the MAML paradigm to various machine learning domains such as neural architecture search [173], regression [140], multi-object tracking [23], time-series classification [174], and semi-supervised learning [17], while applying it to various fields like medicine [159], [4], [160], [126], [107], biomass energy production [191], or fault diagnoses in bearings [94].

Meta-Training and Meta-Testing

As the meta-training of MAML directly derives from the general meta-training scheme presented in Section 2.1, it consists of a meta-level and a task-specific stage. The meta-loss $\mathcal{L}_{\text{meta}}$ is defined as the sum over all task-specific losses $\mathcal{L}_i$ on the respective test sets X^i_{test}:

$$\mathcal{L}_{\text{meta}} := \sum_{i=1}^{N} \mathcal{L}_i \left(\theta'_i,\ \varphi,\ X^i_{\text{test}}\right)$$

[9] Although initialization schemes, such as the Glorot [53], He [65], or LeCun [82] initializations, already stabilize initial training, they, in contrast to gradient-based meta-learning, do not actively place model parameters in an area which is promising for all tasks of a task distribution $p(T)$.

with θ_i' denoting the task-specific parameters resulting from the individual inner update steps (14). For $K = 1$, the corresponding meta-level gradient descent update is

$$\varphi' = \varphi - \beta \nabla_{\varphi} \mathcal{L}_{\text{meta}} \tag{13}$$

with pre-defined meta-learning rate β. This way, the meta-level gradient descent update of φ implicitly optimizes φ w.r.t. the performance of the inner learning (4), that, itself, is solved by K gradient descent steps of the form

$$\theta_i' = \varphi - \alpha \nabla_{\varphi} \mathcal{L}_i \left(\varphi, X^i_{\text{train}} \right), \tag{14}$$

with a pre-defined learning rate α[10], and the meta-level parameters φ as the corresponding task-level prior. Figure 5 shows one iteration of the MAML meta-training scheme with $K = 1$. However, as the meta-testing of MAML fully follows the scheme presented in Figure 3, no additional figure shows the MAML meta-testing scheme.

To avoid the computation of second-order gradients, [42] suggest First-Order (FO) MAML, a more efficient approach that omits second-order derivatives. Although this modification does not guarantee global convergence [40] (a limitation that motivated further modifications like Hessianfree MAML [40] or Implicit MAML [124]), FO-MAML is much easier to implement and computationally less expensive than memory-based meta-learning algorithms presented in the subsequent sections [42].

The MAML Meta-RL Scheme

Besides the fact that task-specific training and test sets must be represented by K episodes of MDP-specific agent-environment interaction, the general MAML structure of embedding task-specific gradients in the meta-level gradient descent remains the same in meta-RL. However, the gradient descent steps (14) and (13) require gradients of the task-specific losses, which are the task-specific expected returns (6) in meta-RL. But these value functions are not differentiable due to unknown environment dynamics and the agent's impact on them, so that the original MAML meta-RL paradigm can only apply policy-gradient methods [150]. This makes MAML on-policy. While the exact computation of the task-level policy gradient depends on the respective task-level algorithm, the meta-update of φ requires policy gradients $\nabla_{\varphi} \pi_{\theta_i'}$ of the updated parameters θ_i' on the respective task-specific test episodes X^i_{test}. In this way, the underlying policy-gradient algorithm is extended to the meta-level, i.e. to updating φ with respect to the performance of the fine-tuned parameters θ_i' on test sets X^i_{test}. But this meta-update scheme consists of second-order policy gradients which are difficult to derive, what motivates meta-RL-specific first-order MAML extensions such as Taming-MAML [97] or DICE [47].

Many state-of-the-art standard RL algorithms, such as PPO [139] or TRPO [138], utilize an off-policy learning scheme to learn from many policies at once. This motivates the PEARL [125] algorithm, the landmark gradient-based off-policy meta-RL algorithm used as a baseline for various sophisticated meta-RL approaches. It utilizes a replay buffer for offline meta-level updates while task-level policy gradients are not required. Instead, inner learning is informed via embeddings of the currently collected experience. However, PEARL does not directly contribute to the development path towards the Adaptive Agent, so that the curious reader is referred to Appendix C for a full derivation of PEARL's paradigm and meta-training scheme.

Performance Analysis

Assuming that MAML's model architecture consists of a sufficiently deep neural network with ReLU activations and that the loss function is either cross-entropy or mean squared error (MSE), [43] prove that MAML serves as a universal function approximator for any training set X^i_{train} and test set X^i_{test} within an arbitrary task T_i. They show, furthermore, that MAML converges linearly to a global optimum under MSE loss when using an over-parameterized deep neural network along with a SGD optimizer like Adam [77]. This implies that a deeper model, with at least one hidden layer, is necessary, even if single tasks can be addressed using a shallower or linear model [6]. Empirical findings support that MAML's meta-training can achieve 100% training accuracy (indicating global convergence) on simple task distributions (such as Omniglot) when appropriate hyperparameters are utilized [42], [173]. However, [123] show

[10] [42] state that the learning rate α can also be meta-learned, which is addressed in α-MAML [10]

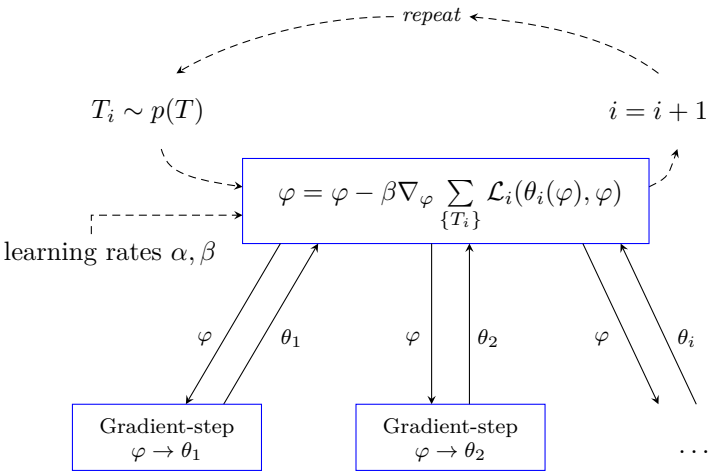

Figure 5: The MAML meta-training scheme.

that MAML learns effective feature representations rather than a rapidly adaptable prior: During task-specific training, the initial layers of the underlying network exhibit minimal changes, suggesting that the fundamental feature representations remain stable. Probably, this is why, MAML is more stable than other landmarks, especialy VariBAD (see Section 3.3), which is illustrated, e.g., on Meta-World [186] Tasks (see [103], Figure 4), where MAML achieves almost 100% success rate, despite high task-uncertainty.

The number of different tasks to be learned simultaneously in the MAML paradigm is naturally bounded by the fact that model size cannot be increased indefinitely. Moreover, the MAML paradigm assumes the optimal parameters $\theta_1^*, \theta_2^*, \ldots$ of the different tasks $T_1, T_2, \ldots$ to be sufficiently close to each other, so that a general prior φ (i.e., a global optimum) can be found from which each optimal solution can be sufficiently well approximated within K gradient descent steps. But as parameter vectors are likely to be high-dimensional, this is a rather strong assumption. In meta-RL, the time MAML requires for learning RL tasks, scales with their complexity [116] since more complex tasks require better exploration and exploitation. Empirically, this is demonstrated on Mujoco [161] tasks, such as HalfCheetahVel (see [103], Figures 4 and 5), where MAML achieves an average reward of -150 (averaged over different seeds), while memory-based algorithms, such as VariBAD (see Section 3.3, TrMRL (see Section 3.4, and RL^2 (see Section 3.2) achieve around -25 average reward starting from first episode. This, however, suggests a worse performance capability rather than a lower adaptation speed, since the rate of learning at the beginning is not significantly lower than in the other algorithms. This gap of performance is even more significant on OOD-tasks, such as illustrated in [103], Figure 6, where MAML achieves only -500 reward, while memory-based landmarks achieve a reward of approximately -100. These results provide the main motivation for the development of MAML extensions such as Robust MAML [108] and XB-MAML [85], which are specifically designed to improve generalization to tasks that are slightly out of distribution."

3.2 Memory-based Meta-RL

In meta-RL, all MDPs are drawn from a meta-distribution $p(M)$ over MDPs that are distinct but similar. Memorizing similarities facilitates the adaptation to a new MDP that requires a similar strategy. In this respect, the identification of the current task T_i, i.e., the hidden dynamics of the corresponding MDP M_i, plays a crucial role in learning the task-specific optimal behaviour π_i^*. However, during MDP-level learning, this identification can only rely on the experience

$$c_t^i := \{(s_j, a_j, r_{j+1}, s_{j+1})\}_{j=1}^t \tag{15}$$

gathered until the current time t in a particular MDP M_i. Such sequences are called context and numerous meta-learning algorithms such as PEARL [125], DREAM [95], [58], MetaCure [188], or CAVIA [199], incorporate it into their paradigm.

How well the environment was explored is determined by the quality of the context [112], while a good representation of that context is substantial for good exploitation.

In Example 2, the current context is all segments of the racing track the racer has visited so far. Here, the ability to process the whole context sequence representing the racing track, rather than just single transitions, significantly helps to cover the dependencies between different sections of the race and to assign the reward - potentially occuring only once at the end of the race - to the whole route. Such problems, where time dependencies are important and rewards are sparse, motivate the use of memory architectures, as they are context-based sequence learners just by design.

The simplest memory architecture is that of a Recurrent Neural Network (RNN). RNNs maintain a hidden state h that acts like a memory mechanism by storing information about past experience. They are Turing-complete universal computers [143]. However, even in standard RL, RNN-based agents are very sensitive to architectural choices like their initialization, the choice of the underlying RL algorithm, or the length of the context that is captured by the hidden state h [109]. The latter is, moreover, highly MDP-specific [109], which limits the flexibility of RNN-based meta-RL algorithms and makes them struggle to generalize between tasks [8], [12]. This motivates additional modifications, such as employing a hypernetwork [8] to learn initializations of the MDP-specific RNN weights, or incorporating Bayesian inference into the RNN-based meta-learners (as presented in Section 3.3). Nevertheless, the following paragraphs describe RL^2, the earliest and simplest memory-based meta-RL algorithm, that serves as the main reference for black-box (i.e., memory-based) meta-RL in many surveys, e.g., [9], [169], and as a baseline for most gradient-based and memory-based meta-learners.

Fast Reinforcement Learning via Slow Reinforcement Learning

The main idea of RL^2 is to combine a standard RL algorithm with an RNN-based agent [11]. The corresponding RNN weights update only once every K episodes of MDP-specific learning, which aligns with the notion of "slow learning", while the activation of the RNN's hidden states by the current context serves as an implicit fast adaptation scheme throughout these K episodes. This way, RL^2 can still be (theoretically) differentiated into an inner and an outer learning stage [34], although it consists of only one RNN representing the policy. The general knowledge φ is implicitly collected by updating the weights of the RNN once every K episodes. The goal to Bayes-optimally trade-off exploration and exploitation is also implicit: The objective of maximizing rewards over K episodes results in the implicit need to explore the current MDP as fast as possible in order to maximize rewards in the subsequent episodes.

Meta-Training and Meta-Testing

The RL^2 meta-training scheme differs significantly from the general meta-RL meta-training scheme: One meta-episode is defined as interacting with a particular MDP M_i for K episodes. The experiences X^i_{train} update the policy parameters θ in the standard RL manner on meta-level. For standard RL algorithms such as TRPO [138] or PPO [139], this means to sample single transitions (s, a, r, s') from the replay buffer β_i containing all experience in X^i_{train}, and use the corresponding batch for updating θ in several epochs of gradient descent.

Throughout MDP-specific learning, the parameters θ remain unchanged. Instead, the hidden state h_t of the RNN updates through the gathered experience. It resets at the beginning of each MDP-specific training but gets preserved throughout the K inner learning episodes - which is the major difference from simply modifying standard RL algorithms with an RNN agent. Conditioning the policy $\pi_\theta(\cdot|h_t)$ on that hidden state, one yields an MDP-specific context-based adaptation scheme, since the hidden state h_t functions as the representation z of the context c. An MDP-specific training-test split does not exist, as the inner learning performance is validated throughout MDP-specific training.

The meta-testing scheme of RL^2 is analogous to the meta-RL meta-testing presented in Section 2.2. The policy parameters θ are fixed, while the hidden state adapts during K episodes of MDP-specific learning.

Performance Analysis

RL^2 was experimentally shown to be Bayes-optimal on simple benchmark tasks like N-arm bandit problems [104]. The experiments indicated the Bayes-optimal solution to be a fixed point of meta-training.

[11] The originally published algorithm uses TRPO [138] as the standard RL algorithm, although [125] show RL^2 to perform even better with more sophisticated RL algorithms like Proximal Policy Optimization (PPO) [139].

However, the task distributions used in [104] were manually constructed so that calculating a Bayes-optimal solution was tractable. Hence, the performance on broader, more complex task distributions is not guaranteed. In fact, RL^2 was empirically shown to achieve very poor asymptotic performance on locomotion tasks ([58], Figure 4), Mujoco benchmark tasks ([200], Figure 6, [103], Figure 4 and 5), and the MetaWorld environment ([103], Figure 5). On all these tasks, the performance of RL^2 slightly increases at the beginning, but then it hardly learns anything else. This indicates poor generalization, as stated by [12], probably, since the RNN has to cover all architectures theoretically needed for the optimal strategy π_i^* of each MDP M_i of the MDP distribution $p(M)$. However, RL^2 exhibits much better OOD performance (around -150 reward almost from the beginning) than MAML (not better than -400 reward) on Mujoco tasks (such as HalfCheetahVel) [103], Figure 6. Most likely, this suprisingly good OOD performance results from the high sample-efficiency and adaptation speed, that [200] demonstrated for RL^2 on some Mujoco tasks (see Figures 5 and 6), but, reviewing the Figures presented in the literature, it is impossible to ultimately determine what the reason is.

3.3 Task-Inference Meta-RL

During learning, there is uncertainty about which MDP the agent is currently acting in. The actions taken by the current agent determine which experiences it gathers, but unseen states might be more informative about the current MDP and hence other actions might have been better for exploration. At the same time, the exploration-exploitation dilemma leads to poor reward during exploration, which is nevertheless required for exploiting the gained knowledge in order to maximize future rewards. As a consequence, many works, such as [146], [112], [95], or [188], separate exploration from exploitation by using distinct exploration and exploitation networks. Instead, the uncertainty about the underlying MDP can also be incorporated into the meta-RL framework by utilizing the Bayesian Reinforcement Learning (BRL) paradigm introduced in Appendix A. Thereby, the context updates the belief over the current MDP M_i a posteriori, while the corresponding posterior distribution informs the decisions of the agent. Algorithms following this principle are referred to as task-inference methods (see, e.g., [9]). They appear both in the family of gradient-based meta-learners [109], [12], [58], [188], and in the broader family of memory-based approaches [200], [14]. This subsection discusses Variational Bayes-Adaptive Deep RL (VariBAD) [200], the landmark task-inference method on the development path towards the Adaptive Agent.

Variational Bayes-Adaptive Deep RL

The main idea of VariBAD is to extend the memory-based meta-RL framework presented in Section 3.2 to include task uncertainty by incorporating a Variational AutoEncoder (VAE). The corresponding VAE architecture consists of a context-encoder RNN f_φ that outputs a variational distribution $p_\varphi(z|c_t^i)$, and an ordinary neural network decoder g_ψ with two heads: One representing a common transition function $\mathbb{T}$, the other a common reward R. The policy $\pi_\theta(\cdot|s, p_\varphi(z|c))$ is, in contrast to RL^2, an ordinary deep neural network parameterized by θ. Both the policy and the decoder receive the output distribution $p_\varphi(z|c)$ from the context-encoder f_φ, so that, in practice, both these components are different heads of the same architecture receiving the encoder output distribution p_φ as input.

Meta-Training and Meta-Testing

Following the context-based meta-training scheme presented in Section 3.2, the meta-parameters φ, θ, and ψ do not adjust throughout the K episodes of MDP-level training, while inner learning exclusively corresponds to the implicit change in the VAE distribution $p_\varphi(\cdot|c_t^i)$ resulting from updated context c_T^i. This way, VariBAD combines RL^2 with the PEARL algorithm (which is presented in Appendix C), since the VAE distribution directly incorporates task uncertainty into the model-based task-level decision-making. Additionally, the replay buffer β stores the resulting trajectories $\{c_t^i\}_{t=1,\dots,h_i}^{i=1,\dots,N}$ for meta-level gradient descent updates [12]. The corresponding meta-loss $\mathcal{L}_{\text{meta}}$ consists of two major components:

1. The RL loss which simply is the accumulated loss over all trajectories, and

[12] Although the original publication [200] utilizes on-policy algorithms, [33] present an off-policy modification, that is based on leveraging offline data collected by MDP-specific agents in order to extract meta-knowledge for meta-updates of φ, θ, and ψ.

2. The ELBO loss, which itself consists of two components, a reconstruction loss for the decoder and a KL-divergence keeping the encoder output p_φ close to its update[13].

Since the encoder output distribution p_φ functions as the meta-learned prior, task inference takes place at the output stage of the VAE encoder. The posterior is updated at each time step via variational inference. However, the corresponding scheme is rather technical and hence not shown here. Instead, the curious reader is referred to the original work in [200] or other works about task-inference like [133]. The meta-testing of VariBAD is analogous to the context-based meta-testing (Section 3.2).

Performance Analysis

VariBAD outperforms RL^2 in both a dynamic grid world and the MuJoCo benchmark (see [200], Figure 5). During the first five million time steps of the dynamic GridWorld environment, it achieves around twice as much reward, which indicates a two times higher sample-efficiency than RL^2 on that benchmark. Thereby, VariBAD exhibits almost Bayes-optimal exploratory behaviour (maximally 14% worse performance on the worst performance task in Figure 4). On Mujoco test tasks, it adapts within one single episode (see [200], Figure 5). Although RL^2 exhibits slightly better sample-efficiency at the beginning of fine-tuning on some of the Mujoco test tasks, VariBAD always adapts within one single episode, while other algorithms, such as PEARL (see Appendix C) cannot even solve some of the test tasks within the first two episodes. On other test tasks, VariBAD achieves three times as much reward as PEARL, i.e., exhibits a three times better adaptation speed, although PEARL shows significantly better sample-efficiency (see 3.3, Figure 6). This implies that VariBAD "takes time" to explore in early episodes, i.e. observes more frames to yield better episode reward at the cost of lower sample-efficiency. However, the adaptation speed of RL^2 on these tasks is similar to that of VariBAD (see [200], Figure 5b), which indicates a big role of the underlying RNN architecture and aligns with the performance analysis in Section 3.2.

The Bayes-optimality of VariBAD has neither been demonstrated nor proven across more sophisticated task distributions. In the MetaWorld environment, it exhibits unstable behavior throughout test task fine-tuning, i.e., the rate of successful performance oscilates between 0.5 and 0.8 (see [103], Figure 4), although it thereby outperforms TrMRL, the transformer-based landmark presented in Section 3.4. However, in OOD Mujoco tasks, VariBAD's instability leads to a much worse performance (between -100 and -300 total reward) than that of TrMRL (constantly around -100 reward), see [103], Figure 6. According to [103], this indicates worse generalization than implied by the original publication [200]. These issues led to modifications of VariBAD, such as HyperX [201], which enhances meta-exploration by distilling policies from random networks to improve performance in distributions of sparsely rewarded MDPs, and MELTS [14], specifically designed for effective zero-shot performance on non-parametric task distributions through task-clustering. Both these methods utilize modifications that are also used in the Adaptive Agent (i.e., distillation and task-selection), which are hence described in Section 3.5.

3.4 Transformer-based Meta-RL

The Transformer network architecture was first proposed by [168] as a language-to-language model. As induced by its title, [168] show that "attention is all you need" by outperforming all other state-of-the-art language-to-language algorithms by a good margin. This success motivated the development of many Transformer modifications further boosting performance, e.g., the Transformer-XL [27] used in the Adaptive Agent. Simultaneously, several researchers developed modifications of the initial Transformer for tasks different from language-to-language translation, e.g., Vision Transformers for computer vision [63], visual and semantic segmentation [89], [178], [147], forecasting [93], and RL [69]. This way, they applied Transformers to various fields like medicine [178], e.g., for RNA prediction [21], fault detection, e.g., in manufacturing [177], bioinformatics [190], automated driving [70], and many other application fields. In RL, Transformers close the gap between simulation and reality in locomotion control [79], act beneficially in various IoT environments like Smart-Homes or industrial buildings [131], and learn from a vast amount of (sub)-optimal demonstrations [96], [86], [128]. Since each of the various extensions and modifications of the vanilla transformer builds upon the attention mechanism proposed in the original publication [168], the original Transformer architecture is broadly presented and discussed in Appendix D, while this section focuses on the utilization of Transformer architectures for meta-RL.

[13] For a detailed derivation of the ELBO-loss and all corresponding components, the curious reader is referred to [133].

Transformers are Meta-Learners

Instead of translating a sentence from one language into another, a context-based RL agent "translates" context trajectories of the form (18) into the current action a_t. Such a "translation" does not require a decoder as the current action can directly be derived from a policy head. But it is likely that the single transitions of a context trajectory have strong dependencies to each other or the time they occurred, so that it is additionally desirable to contextualize them. This motivates using the Transformer architecture, e.g., in simple, gradient-based meta-learning algorithms like MAML (Section 3.1). Hence, different works present MAML modifications for detecting faults in bearings [90], improving short-term load forecasting in scenarios where different clients require federated learning [41], and forecasting stock prices [25]. However, Transformers additionally have a demonstrable long-term memory of up to 1500 steps into the past [110], which particularly motivates to use transformers in memory-based meta-learning [103], [56], [179], [142], [141].

TrMRL [103] is such a Transformer architecture tailored for meta-RL. It extends RL^2 by a Transformer architecture i.e., by self-attention, and fulfills all necessary properties of a meta-learner [103], i.e.,

1. The fast adaptation of the multi-head attention serves as a task representation mechanism, since each self-attention head contextualizes the embeddings of the context c_t^i collected in the current MDP M_i up until the current time step t.
2. the meta-learned model weights function as a long-term memory by design, through which the respective MDP can be identified and acted upon accordingly.

They hypothesize that any MDP can be represented by a distribution over working memories:

$$z_t = \sum_{t=1}^{T} \alpha_t f_\theta(o_t) \tag{16}$$

where o_t is a single transition of the form $(s_t, a_t, r_{t+1}, s_{t+1})$, T is the trajectory length, f_θ is a learnable, arbitrary linear function, and α_t are coefficients summing up to 1. In fact, the context scores of the last transformer encoder sub-block naturally have the form of such a task representation when given the context trajectory c_t^i of the form (18) as input at a particular time point t: Reformulating the self-attention softmax term (21) for a particular transition o_t and the multiplication with the corresponding value vector v_t, one yields representations for the coefficients α and the linear function f in (16) - the former through rewriting the softmax term, the latter by a sum over entries of the value vector. For the rather technical full derivation of that MDP representation property, the interested reader is referred to the original publication [103]. For the scope of this work, it is sufficient to conclude that it creates the implicit objective to "learn to make a distinction among the tasks in the embedding space" [103], i.e., the implicit goal to learn how to identify tasks through the collected experience. In fact, [103] additionally prove that any Transformer self-attention layer minimizes the task uncertainty based on the current context c_t^i, i.e., that every Transformer sub-block finds the best representation of the gathered information. In each Transformer sub-block, the previous representation of c_t^i is recombined through the dense layers before the next self-attention layer persists in the form of a task representation. In this way, the representation of c_t^i is refined accross episodes, which "resembles a memory reinstatement operation" [103], i.e., an operation to reactivate long-term memory in order to identify the current task.

Meta-Training and Meta-Testing

In RL, transformers are often unstable during training [69], especially in the beginning [103]. And while more sophisticated algorithms like AMAGO [56] stabilize training by incorporating off-policy learning into their Transformer architecture, TrMRL addresses this problem right at the beginning of meta-training by utilizing T-Fixup initialization [71]. This initialization scheme is especially designed to avoid learning rate variation and layer normalization during early training, and the ablation studies of [103] particularly show its usefulness. Besides that, the meta-training and meta-testing schemes of TrMRL directly derive from the RL^2 schemes provided in Section 3.2.

Performance Analysis

As implicitly shown in the performance analysis paragraphs of the previous subsections, TrMRL outperforms MAML, VariBAD, and, most importantly, RL^2 in the MuJoCo environment w.r.t. sample-efficiency, adaptation speed and particularly OOD performance (see [103], Figures 4, 5 and 6). This superior performance is even more significant in the more sophisticated Meta-World environment (see [103], Figures 4, 5 and 6), where it achieves equal or higher success rates than the other landmarks in all except the first episodes (in first episode, VariBAD is slightly better). The only exceptions are MuJoCo tasks with high task uncertainty, e.g., Meta-World-ML1-Push-v2, where VariBAD oscilates between a success rate of 0.5 and 0.8, while TrMRL remains stable at a value of 0.5 (see [103], Figure 4). This motivates combining a Transformer-based meta-RL agent with Bayesian inference such as in PSBL [179], or ADA.

3.5 The Adaptive Agent

Transformers still struggle with "credit assignment" [110], i.e., with assigning current actions to rewards later on in the planning horizon - a problem that even increases when the data complexity is high [110]. This particularly motivates the use of model-based RL, where the agent can base its decisions on the predictions of the dynamics model. The only landmark algorithm on the development path towards ADA utilizing model-based RL for inner learning is VariBAD (Section 3.3). It additionally models task uncertainty into its paradigm and, hence, [103] already identify the possibilities of combining the VariBAD paradigm with a transformer architecture. The Adaptive Agent is this transformer-based modification of VariBAD. It combines a large Transformer architecture with self-supervised learning techniques in order to develop a generalist agent applicable to a vast number of "downstream tasks", i.e., test tasks significantly different from the meta-training tasks - perhaps even OOD. In this respect, it builds upon other, similar approaches aiming for a generalist agent, i.e., RL foundation models, like GATO [128] or the work of [157].

Analogous to VariBAD, ADA consists of a policy π_θ, a decoder g_ψ, and a context encoder f_φ whose output is the variational distribution $p_\varphi(z|c_t^i)$ representing task uncertainty. However, the ADA paradigm modifies these components as follows:

- Prior to the Transformer architecture, a ResNet architecture preprocesses the visual 3D observations received from the XLand environment.
- The memory-based context encoder is a Transformer architecture; [14] more precisely, the Transformer-XL architecture [27], a demonstrably more efficient and powerful modification of the Vanilla Transformer whose architecture allows it to model long-range dependencies.
- The architecture following the Transformer-XL mimics the Muesli algorithm [66], a computationally efficient, demonstrably powerful RL approach that combines model-based and model-free RL. This requires a critic network estimating the value (6) of the current state as well as a dynamics model for planning. The latter predicts policy and critic network outputs along with future rewards and transitions.

In addition to these modifications, the number K of adaptation episodes varies (i.e., $K = \{1,\ 2,\ \ldots,\ 6\}$). Hence, k is an additional input to the context encoder.

Meta-Training and Meta-Testing

In contrast to TrMRL, the ADA meta-training scheme does not consist of a particular initialization scheme. Instead, layer normalization and gating techniques stabilize the transformer during training. However, as those are rather detailed technical details, the interested reader is referred to the original work [156]. Besides that, the VariBAD meta-training scheme is extended by two self-supervised learning techniques: 1) An automatic curriculum (ACL) selecting tasks on the frontier of the current agent's capabilities for efficient learning, and 2) distillation for kickstarting the training process. The subsequent subsections describe both these techniques in more detail. But before, the following paragraphs summarize the performance analysis provided in the original work [156].

[14] Note that theoretically any Transformer architecture can be used that is suitable for RL.

Performance Analysis

The Adaptive Agent achieves human-like few- and zero-shot performance (in terms of generalization and adaptation speed) for single and multi-agent tasks on an even more complex and dynamic extension of the XLAND environment [157], a "vast open-ended task space with sparse rewards" for single and multi-agent tasks. In multi-agent downstream tasks, the different agents clearly share labor and actively cooperate to improve their reward, while in single-agent tasks it is able to use one single episode of expert demonstrations in order to significantly improve performance [156]. This makes ADA superior to the work of [157] and other generalist agents like GATO [128] which exclusively learn from demonstrations. However, ADA is not generally superior to human-level performance, as there are single- and multi-agent downstream tasks neither humans nor ADA are able to solve.

ADA's transformer memory length, i.e., the length of the context c used as model input, as well as its model and task pool sizes, scale demonstrably well, particularly when being scaled together with each other [156]. For example, increasing the task pool, i.e., sampling a larger number of distinct tasks from a task distribution with more complex tasks, always has a positive effect on the agent's median performance (i.e., the median of the performance over all tasks), especially in the few-shot setting. But this effect is even bigger when model size or memory also increase. The same holds for the model size and the transformer memory length: larger models as well as models with longer context windows obtain better median performance in any K-shot setting, particularly when the task pool also increases. On a log-log plot, the effect of increasing the model or context window size additionally scales roughly linearly with the number K of shots. But only if the task pool is sufficiently large. Otherwise the model overfits the small task pool. All these findings also hold for the 20th quantile i.e., in the 20,This indicates a more stable training and underpins the work of [16] which identify scaling as the key factor of what makes foundation models so powerful.

Although ADA is capable of handling varying context lengths, it, like most RL Transformers, assumes the dimension of observations to be constant throughout tasks. However, as ADA particularly utilizes an additional observation encoder prior to its transformer architecture, it suffices to exchange only that encoder when observation spaces change. This motivates AMAGO [56], a more flexible approach that particularly utilizes an observation encoder mapping each context to a fixed-sized representation, so that the encoder is the only required architectural change across experiments. This way, AMAGO is particularly designed to incorporate off-policy actor-critic RL in combination with distinct, dynamic, long-term contexts, so that it is applicable for multi-task RL.

Since approaches like AMAGO, GATO, and ADA clearly are foundation models, they are computationally expensive during meta-training [142], which motivates hierarchical transformer architectures like HTrMRL [141] or ECET [142]. They use additional self-attention sub-blocks to process cross-episode data in order to further increase sample efficiency and reduce model complexity. In contrast to the intra-episode self-attention sub-blocks utilized in TrMRL, AMAGO, and ADA, these cross-episode blocks set whole episodes of experience in context to each other, so that the context mechanism of self-attention is also leveraged at episode level. Among other model reduction techniques like distillation, this is a promising direction for future work in the field of foundation models and generalist agents.

3.5.1 Automated Curriculum Learning:

Automated Curriculum Learning (ACL) leverages the idea from human learning that training tasks should not be too hard or too easy for the current level of learning. It originates from curriculum learning (CL), [28], where curricula are hand-crafted to guide the learning process of a standard learner (e.g., a neural network) on a single task. Such curricula were successfully applied in several supervised learning tasks like NLP, computer vision, medicine, Neural Architecture Search, and RL (see e.g., [175], [145], [60], for a detailed overview). The main idea of ACL is to automatically prioritize tasks at the frontier of the agent's capabilities during training, instead of manually selecting them. This ensures that the agent is consistently challenged and can progressively improve its skills [119], so that sample efficiency and generalization performance can be significantly improved [30], [74], [73], [175].

There are several ACL methods (see e.g., [119] for a detailed overview). One of the earliest approaches is Teacher-Student Learning [101], where a second model (the teacher) is simply used to select subtasks for the standard learner (the student) during training. For meta-RL, there also exist various techniques, e.g.,

- Simple, but quite static methods like domain randomization (e.g., [171]), where environments are generated randomly, but independently of the current policy's capabilities.
- More flexible approaches like min-max adversarial (e.g., [172]), where environments are created adversarially, i.e., to challenge the agent as much as possible.
- Sophisticated paradigms like min-max regret, e.g., Protagonist Antagonist Induced Regret Environment Design [30], where an adversary aims to maximize the regret between the protagonist (the agent) and its antagonist (an opponent). The latter is assumed to be optimal.

For min-max regret, the regret is defined as the difference between the value of the protagonist's and the antagonist's action. This way, the adversary selects the simplest tasks the protagonist is not yet able to solve [30]. This makes min-max-regret superior to min-max-adversarial, which, due to its objective, tends to generate tasks too difficult for the agent or even unsolvable [30]. However, ADA leverages and compares two other ACL methods during meta-training:

1. NOOB-Filtering [157] maintains a control policy that takes no actions. After rolling out ADA and the control policy for ten episodes in a new task, it is selected for meta-training if: ADA's performance is neither too good nor too bad, the variance among trials is sufficiently high across episodes to indicate proper learning, and The control policy performs poorly enough to indicate the relevance of proper decision making.
2. Robust Prioritized Level Replay (PLR) [74] maintains a "level buffer" of tasks with high learning potential. It samples tasks from that buffer with a probability of p, and otherwise samples a new task from the task distribution $p(T)$.

   The task's regret estimates its learning potential. A task is added to the level buffer, if its regret is higher than those of the other level buffer tasks. This constantly refines the level buffer. In ADA, several fitness metrics like policy and critic losses approximate the task regret.

In comparison to meta-training with uniformly sampled tasks, both no-op filtering and PLR, improve ADA's sample efficiency and generalization as well as its few- and zero-shot performance [156]. Additionally, PLR slightly outperforms NOOB filtering, especially when the number K of adaptation episodes is higher. However, any other approach from the ones described above might work just as well.

3.5.2 Distillation

The notion of distillation was first proposed by [67]. It generally refers to model compression techniques transferring knowledge from a large, pre-trained teacher model T [15] to a smaller, more efficient student model S [132]. Hence, distillation can be viewed as a transfer learning approach [100]. Classically, the student is trained to learn from the teacher's demonstrations, i.e., it learns to predict the teacher's softmax output distribution (in the discrete case) that is smoothed by choosing a sufficiently high softmax temperature parameter in order to yield soft targets [132], [100], [67]. In RL, one of the best-known distillation approaches is policy distillation [132] (also known as imitation learning), where the student policy is trained to imitate the teacher's action distribution. Other widely used approaches include value function distillation and dynamic reward-guided distillation (see [180] for further details).

Generally, the two main questions in selecting a knowledge distillation scheme are: 1) What knowledge to transfer and 2) which architectures to choose for teacher and student models [100], [54]. ADA uses dynamic reward-guided distillation, where the student policy is guided by the teacher policy rather than particularly mimicking it. It contains an additional distillation loss throughout the first four billion meta-updates that consists of the KL-divergence between student and teacher policy action distributions, as well as a L^2-regularization term like in [136]. This way, the student policy can also explore states different from those visited by the corresponding teacher model, which was trained from scratch with a model size of 23 million parameters.

For comparison, [156] train four models: A large (256 million parameters) and a small (23 million parameters) model, each with and without distillation. They show the larger distilled model to strongly

[15] It is also possible to use multiple teachers for different subtasks [136]. However, these teachers can be seen as one teacher model consisting of different task-specific components.

outperform the smaller one as well as the large ordinary model, while the smaller ordinary model outperforms the larger one. This not only indicates a significant positive effect of distillation on model performance, but also highlights its necessity for large-scale (foundation) models.

4 Discussion

Meta-learning describes the paradigm of learning how to learn. However, given the vast array of potential tasks, the question naturally arises: What general knowledge should be acquired? Earlier algorithms primarily focused on meta-learning the skill to make decisions in RL environments by pre-training model weights (MAML), and by basing the decisions on the current context (PEARL and RL^2) or the current context-based belief about which task they are acting in (VariBAD) with manually designed architecture modifications. Transformer-based agents are meta-learners by design. Although they were not specifically designed to be meta-learners, these models learn how to learn just by emergence.

The path towards general intelligence often looks like this [16]: While earlier approaches are hand-designed for particular purposes, later ones yield more general, unsupervised, and unguided intelligence, learning different skills emergently. The consequence is a shift towards "homogeneity", i.e., the use of a single model architecture for several different tasks or task distributions [16]. In other words, as soon as a model architecture generalizes better, it is used for more general purposes. The path towards the Adaptive Agent presented in Section 3 exemplifies this development. Even within the transformer-based development path, one can clearly observe this shift towards homogeneity: While early transformer-based meta-RL architectures like TrMRL require hand-designed architectures for different observation spaces, generalist agents like ADA and AMAGO abstract this problem by observation-space-specific encoder blocks. Observing this development scheme (as outlined by [16] and [26]), suggests how future developments might look like: As, from a theoretical point of view, any skill can be meta-learned, meta-learning does not have to solely focus on task-specific adaptation. The easiest example is meta-learning hyperparameters, - an approach that appeared quite early in the meta-learning timeline, with algorithms such as α-MAML [10] or PEARL [125]. But even far more complicated skills such as neural architecture search, curriculum design or pre-designing populations of evolutionary algorithms, can be meta-learned. The last of the following subsections discusses such approaches and sets them in context to the path towards general intelligence.

Relevance of Meta-Learning

With the shift from manually designed, relatively simple to understand meta-learning algorithms like MAML or PEARL to largely scaled, blackbox foundation models like GATO [128], ADA [156], or AMAGO [56], the question arises whether the paradigm of meta-learning is actually still of relevance. Large-scale foundation models from companies like DeepMind, OpenAI, or Meta can be applied to a vast amount of in- and out-of-distribution downstream tasks of different types - as e.g., done for the LAMA model in [130] for RL tasks - without even thinking about data quality, distribution shifts, or the meta-training paradigm. However, such foundation models require a vast amount of computational power, even for simple downstream rollouts without model updates. During training, they necessitate scaling of their model size, the task pool, and the task complexity [156], [16]. Such an amount of data and computation power is not available for all researchers, companies and people - even when techniques like distillation and ACL decrease the model size and boost the training progress. Moreover, in recent years there has been a significant shift from publicly available models with revealed source code trained on open-source data towards secretly training models of unknown sizes and shapes on largely collected datasets that are hidden from the public - and, as such, hidden from public control and revision. But how models behave highly depends on the data they were trained on, so that this privatization of model training comes with a high risk of unexpected, harmful behaviour [16], along with the societal risk of an increasing gap between institutions with a high amount of computational power and data and others who cannot utilize large resources.

Modern research can, regardless of the available computation power, focus on different niches of application [162] as well as on developing more efficient architectures like hierarchy transformers [141], [142], or on gaining a better understanding of blackbox models, their behaviour, the reasons for their failures, and the societal and ethical impact their application has [162]. The latter can support the development

of even better general problem solvers [26], so that collaboration between university researchers and Big Tech companies is probably beneficial for both [162], [16].

The Need for Specialized Meta-Learners

In application, agents must often be deployed on small edge devices with a very limited amount of computational power that does not allow for generalist agents but might benefit from fast adapting meta-models. Such problems likely cover only a very narrow, low-dimensional task distribution that does not require generalist agents. In other words, largely scaled transformer-based architectures often go far beyond what is needed to solve a particular problem. This likely explains why many meta-learning applications still rely on simple, sample-efficient gradient-based algorithms like MAML, even though they often have poorer OOD performance (see e.g., the applications for medicine [159], [4], [160], [126], [107], biomass energy production [191], or fault diagnosis [94]). However, understanding the advantages and drawbacks of the various components of the landmark algorithms presented in the previous section can support applied researchers in manually designing the best fitting meta-learner for their particular problem. For example, autonomous driving requires extremely fast adaptation, very good zero-shot performance and high robustness and reliability, while sample efficiency does not really matter, especially during training. In such an application, gradient-based algorithms are most likely the wrong choice, while memory-based Bayesian inference learners like VariBAD or ADA are much more promising. In contrast, medical applications typically have a very limited amount of data available per patient, so that they particularly require high sample efficiency. For such applications, gradient-based meta-learners are a good first choice, as they are simple and more sample-efficient. This especially holds true for off-policy meta-RL algorithms like PEARL. To assist the reader in selecting a meta-learning algorithm for application, Table 1 provides an overview of the main advantages and drawbacks of the developments discussed along the timeline of the previous section.

Comparability

There is a lack of comparability among the landmark algorithms presented in the timeline of the previous section. The utilized performance measures differ between the different works and are often not clearly defined, an issue this work aims to address by defining general performance measures in Section 2.3. In addition, different works utilize different benchmarks with different properties and of different complexity. Thereby, the development of those benchmarks follows the development of meta-RL algorithms itself: Early landmarks like MAML, RL^2, or PEARL were mainly tested on simple grid-world environments, the Atari benchmark [11], or the MuJoCo engine [161]. But [39] identified that those benchmarks are too simple to evaluate actual adaptation, which motivates more generalistic, complex, and sparsely rewarded environments like Meta-World [186], Crafter [62] or XLand [157]. This lack of homogeneity makes it difficult for researchers and developers to compare different algorithms and modifications. However, this is a general problem of ongoing research and development and can hence, only be tackled by several empirical studies.

The Three Pillars of the Path Towards General Intelligence

The development path towards human-like or even superior generally intelligent agents consists of three pillars [26]:

- Meta-learning algorithms that can be viewed as the "evolution of intelligence" itself,
- meta-NAS [118], [68] creating the "physical body" of the gained "intelligence," and
- the generation of more meaningful, complex, and challenging environments that function as the "world" the "physical body" of the learned "intelligence" is acting in.

While research in the meta-learning community primarily focused on the pillar of learning algorithms and, secondly, on meta-NAS approaches, meta-environment research is very sparse [26]. However, as the growing complexities and capabilities of recent developments come with the necessity of a further increase in complexity for benchmark environments, the attention most recently shifts towards this research field

by combining meta-RL with self-supervised learning techniques. This is reflected in the ADA paradigm, where initial learning is guided by a teacher model (Distillation) and an automated curriculum (ACL) selecting tasks on the frontiers of the agent's capabilities. The latter is, however, a skill that can be meta-learned, so that a (teacher) model learns to create a personalized curriculum for any student model [120], [181]. The corresponding Meta-ACL paradigm is inspired by classroom scenarios, where a teacher teaches several students at once, ideally with respect to their personal capabilities and special needs.

Additionally, meta-design of environments can be inspired by examining the evolution on Earth [26]. Instead of developing algorithms generating open-ended environments like XLand [157] and meaningful rewards, one can aim for agents that explore out of curiosity [3], i.e., on their own, or through intrinsic reward (see e.g., [188]) in environments providing a high variety of different niches to adapt to. The former corresponds to techniques like novelty search [87], while algorithms like quality diversity [121] can be utilized to generate high-variety environments.

Considering the current work of Big Tech companies, a very likely future path of meta-learning and foundation model development will be towards combining these pillars. The most recent landmark of that kind is the Evolution Transformer [80] of DeepMind. It combines the pillars of meta-learning and meta-NAS by learning how to design the population of RL agents (individuals) for evolutionary RL algorithms. In other words, on top of the evolutionary approach, the Evolution Transformer learns how the population of agents (breed of creatures) must be designed to have the highest chance of survival as a whole, i.e., it focuses on the continued existence of the species rather than on maximization the performance of single individuals.

Combining the different pillars results, along with a potential increase in model capability, in several potential ethical, societal, and economic concerns that are very difficult to predict. When models are "taught to play god" - even in a very limited world like the currently existing environments - it is essential to monitor their behaviour, the basis on which they make their decisions and the harm they can potentially cause from different views like law [20], ethics, economy and various others. In this respect, the future development might be examined and analyzed analogously to that of foundation models in [16], where a large group of various researchers of several domains and backgrounds collaborate in order to identify potential risks, challenges and benefits of the current shift towards large-scale foundation models.

5 Conclusion

This comprehensive survey provides the timeline of meta-RL developments from foundational algorithms like MAML and RL^2 to ADA, a large-scale generalist agent. Through detailed mathematical formalizations of meta-learning and meta-RL paradigms, this work addressees the lack of detailed formalism within the literature and lays the groundwork for understanding the paradigms and training schemes of the landmark algorithms presented along that timeline. Based on that formalized knowledge, it highlights the paradigm shift towards transformer architectures and large-scale foundation models through the usage of self-supervised learning techniques like distillation and automated curriculum learning. Looking ahead, this survey examines the three pillars of general intelligence and identifies a trend towards the automated generation of environments through meta-learning of self-supervision and evolutionary approaches. Besides offering valuable insights, this highlights the constantly growing need for collaborative, interdisciplinary teams of researchers that permanently analyze the behaviour of generalist agents and their ethical implications for our society.

Acknowledgments

This work was funded by the German Federal Ministry of Education and Research (BMBF) under the project "Reinforcement Learning for Cognitive Energy Systems (RL4CES)" with the funding code *01IS22063 A*. Additionally this work was supported by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) and the Ministry of Trade, Industry & Energy (MOTIE) of the Republic of Korea (No. RS-2024-00509239).

References

[1] Multitask Learning. *Machine Learning*, 28(1).

[2] Jay Alammar. The illustrated transformer. `http://jalammar.github.io/illustrated-transformer/`, 2018. Accessed: 2025-07-01.

[3] Ferran Alet, Martin F. Schneider, Tomas Lozano-Perez, and Leslie Pack Kaelbling. Meta-learning curiosity algorithms, March 2020. ADS Bibcode: 2020arXiv200305325A.

[4] Aqilah M. Alsaleh, Eid Albalawi, Abdulelah Algosaibi, Salman S. Albakheet, and Surbhi Bhatia Khan. Few-Shot Learning for Medical Image Segmentation Using 3D U-Net and Model-Agnostic Meta-Learning (MAML). *Diagnostics*, 14(12):1213, January 2024. Number: 12 Publisher: Multidisciplinary Digital Publishing Institute.

[5] Elena Arcari, Maria Vittoria Minniti, Anna Scampicchio, Andrea Carron, Farbod Farshidian, Marco Hutter, and Melanie N. Zeilinger. Bayesian Multi-Task Learning MPC for Robotic Mobile Manipulation. *IEEE Robotics and Automation Letters*, 8(6):3222–3229, June 2023.

[6] Sébastien Arnold, Shariq Iqbal, and Fei Sha. When MAML Can Adapt Fast and How to Assist When It Cannot. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pages 244–252. PMLR, March 2021. ISSN: 2640-3498.

[7] Marcos Barcina-Blanco, Jesus L. Lobo, Pablo Garcia-Bringas, and Javier Del Ser. Managing the unknown in machine learning: Definitions, related areas, recent advances, and prospects. *Neurocomputing*, 599:128073, September 2024.

[8] Jacob Beck, Matthew Thomas Jackson, Risto Vuorio, and Shimon Whiteson. Hypernetworks in Meta-Reinforcement Learning. In *Proceedings of The 6th Conference on Robot Learning*, pages 1478–1487. PMLR, March 2023. ISSN: 2640-3498.

[9] Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A Survey of Meta-Reinforcement Learning, January 2023. arXiv:2301.08028 [cs].

[10] Harkirat Singh Behl, Atılım Guneş Baydin, and Philip H. S. Torr. Alpha MAML: Adaptive Model-Agnostic Meta-Learning, May 2019. arXiv:1905.07435 [cs].

[11] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47:253–279, June 2013.

[12] Eseoghene Ben-Iwhiwhu, Jeffery Dick, Nicholas A. Ketz, Praveen K. Pilly, and Andrea Soltoggio. Context meta-reinforcement learning via neuromodulation. *Neural Networks*, 152:70–79, August 2022.

[13] Jinbo Bi, Tao Xiong, Shipeng Yu, Murat Dundar, and R. Bharat Rao. An Improved Multi-task Learning Approach with Applications in Medical Diagnosis. In Walter Daelemans, Bart Goethals, and Katharina Morik, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 117–132, Berlin, Heidelberg, 2008. Springer.

[14] Zhenshan Bing, Yuqi Yun, Kai Huang, and Alois Knoll. Context-Based Meta-Reinforcement Learning With Bayesian Nonparametric Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(10):6948–6965, October 2024.

[15] Benjamin Bischke, Patrick Helber, Joachim Folz, Damian Borth, and Andreas Dengel. Multi-Task Learning for Segmentation of Building Footprints with Deep Neural Networks. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 1480–1484, September 2019. ISSN: 2381-8549.

[16] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon,

John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the Opportunities and Risks of Foundation Models, August 2021. ADS Bibcode: 2021arXiv210807258B.

[17] Rinu Boney and Alexander Ilin. Semi-Supervised Few-Shot Learning with MAML. January 2018.

[18] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[19] Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, Tony Wang, Samuel Marks, Charbel-Raphaël Segerie, Micah Carroll, Andi Peng, Phillip Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J. Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Bıyık, Anca Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback, September 2023. arXiv:2307.15217 [cs].

[20] Alan Chan, Carson Ezell, Max Kaufmann, Kevin Wei, Lewis Hammond, Herbie Bradley, Emma Bluemke, Nitarshan Rajkumar, David Krueger, Noam Kolt, Lennart Heim, and Markus Anderljung. Visibility into AI Agents. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '24, pages 958–973, New York, NY, USA, June 2024. Association for Computing Machinery.

[21] Mayank Chaturvedi, Mahmood A. Rashid, and Kuldip K. Paliwal. Transformers in RNA structure prediction: A review. *Computational and Structural Biotechnology Journal*, 27:1187–1203, January 2025.

[22] Shreyas Chaudhari, Pranjal Aggarwal, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, Karthik Narasimhan, Ameet Deshpande, and Bruno Castro da Silva. RLHF Deciphered: A Critical Analysis of Reinforcement Learning from Human Feedback for LLMs. *ACM Comput. Surv.*, June 2025. Just Accepted.

[23] Jiayi Chen and Chunhua Deng. MAML MOT: Multiple Object Tracking Based on Meta-Learning. In *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 4542–4547, October 2024.

[24] Shijie Chen, Yu Zhang, and Qiang Yang. Multi-Task Learning in Natural Language Processing: An Overview. *ACM Comput. Surv.*, 56(12):295:1–295:32, July 2024.

[25] Yunzhu Chen, Neng Ye, Wenyu Zhang, Jiaqi Fan, Shahid Mumtaz, and Xiangming Li. Meta-LSTR: Meta-Learning with Long Short-Term Transformer for futures volatility prediction. *Expert Systems with Applications*, 265:125926, March 2025.

[26] Jeff Clune. AI-GAs: AI-generating algorithms, an alternate paradigm for producing general artificial intelligence, January 2020. arXiv:1905.10985 [cs].

[27] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics.

[28] DONALD Dansereau. The Development of a Learning Strategies Curriculum1. In HAROLD F. O'neil, editor, *Learning Strategies*, pages 1–29. Academic Press, January 1978.

[29] Marc Peter Deisenroth, Peter Englert, Jan Peters, and Dieter Fox. Multi-task policy search for robotics. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3876–3881, May 2014. ISSN: 1050-4729.

[30] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design, 2021.

[31] Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. Sharing Knowledge in Multi-Task Deep Reinforcement Learning, January 2024. arXiv:2401.09561 [cs].

[32] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[33] Ron Dorfman, Idan Shenfeld, and Aviv Tamar. Offline Meta Reinforcement Learning – Identifiability Challenges and Effective Data Collection Strategies. In *Advances in Neural Information Processing Systems*, volume 34, pages 4607–4618. Curran Associates, Inc., 2021.

[34] Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. RL$ˆ2$: Fast Reinforcement Learning via Slow Reinforcement Learning, November 2016. arXiv:1611.02779 [cs, stat].

[35] Worrawat Duanyai, Weon Keun Song, Poom Konghuayrob, and Manukid Parnichkun. Event-triggered model reference adaptive control system design for SISO plants using meta-learning-based physics-informed neural networks without labeled data and transfer learning. *International Journal of Adaptive Control and Signal Processing*, 38(4):1442–1456, 2024. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/acs.3758.

[36] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural Architecture Search: A Survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[37] Linus Ericsson, Henry Gouk, Chen Change Loy, and Timothy M. Hospedales. Self-Supervised Representation Learning: Introduction, advances, and challenges. *IEEE Signal Processing Magazine*, 39(3):42–62, May 2022.

[38] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, February 2017. Publisher: Nature Publishing Group.

[39] Rasool Fakoor, Pratik Chaudhari, Stefano Soatto, and Alexander J. Smola. Meta-Q-Learning, April 2020. arXiv:1910.00125 [cs, stat].

[40] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the Convergence Theory of Gradient-Based Model-Agnostic Meta-Learning Algorithms. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, pages 1082–1092. PMLR, June 2020. ISSN: 2640-3498.

[41] Changsen Feng, Liang Shao, Jiaying Wang, Youbing Zhang, and Fushuan Wen. Short-term Load Forecasting of Distribution Transformer Supply Zones Based on Federated Model-Agnostic Meta Learning. *IEEE Transactions on Power Systems*, 40(1):31–45, January 2025.

[42] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, August 2017.

[43] Chelsea Finn and Sergey Levine. Meta-Learning and Universality: Deep Representations and Gradient Descent can Approximate any Learning Algorithm, February 2018. arXiv:1710.11622 [cs].

[44] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic Model-Agnostic Meta-Learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[45] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-Shot Visual Imitation Learning via Meta-Learning. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 357–368. PMLR, October 2017. ISSN: 2640-3498.

[46] Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, Brian Ichter, Danny Driess, Jiajun Wu, Cewu Lu, and Mac Schwager. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*, 44(5):701–739, April 2025. Publisher: SAGE Publications Ltd STM.

[47] Jakob Foerster, Gregory Farquhar, Maruan Al-Shedivat, Tim Rocktaeschel, Eric Xing, and Shimon Whiteson. DiCE: The Infinitely Differentiable Monte Carlo Estimator. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1529–1538. PMLR, July 2018. ISSN: 2640-3498.

[48] Vincent Francois-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An Introduction to Deep Reinforcement Learning. *Foundations and Trends in Machine Learning*, 11(3-4):219–354, 2018. arXiv:1811.12560 [cs, stat].

[49] Yuan Gao, Haoping Bai, Zequn Jie, Jiayi Ma, Kui Jia, and Wei Liu. MTL-NAS: Task-Agnostic Neural Architecture Search Towards General-Purpose Multi-Task Learning. pages 11543–11552, 2020.

[50] Brian Gaudet, Richard Linares, and Roberto Furfaro. Adaptive guidance and integrated navigation with reinforcement meta-learning. *Acta Astronautica*, 169:180–190, April 2020.

[51] Hassan Gharoun, Fereshteh Momenifar, Fang Chen, and Amir H. Gandomi. Meta-learning Approaches for Few-Shot Learning: A Survey of Recent Advances. *ACM Comput. Surv.*, 56(12):294:1–294:41, July 2024.

[52] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *ArXiv*, abs/1609.04436, 2015.

[53] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[54] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge Distillation: A Survey. *International Journal of Computer Vision*, 129(6):1789–1819, June 2021.

[55] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting Gradient-Based Meta-Learning as Hierarchical Bayes, January 2018. arXiv:1801.08930 [cs].

[56] Jake Grigsby, Linxi Fan, and Yuke Zhu. AMAGO: Scalable In-Context Reinforcement Learning for Adaptive Agents. October 2023.

[57] Jie Gui, Tuo Chen, Jing Zhang, Qiong Cao, Zhenan Sun, Hao Luo, and Dacheng Tao. A Survey on Self-Supervised Learning: Algorithms, Applications, and Future Trends. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9052–9071, December 2024.

[58] Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-Reinforcement Learning of Structured Exploration Strategies. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[59] Abhishek Gupta, Justin Yu, Tony Z. Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, Thomas Devlin, and Sergey Levine. Reset-Free Reinforcement Learning via Multi-Task Learning: Learning Dexterous Manipulation Behaviors without Human Intervention. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6664–6671, May 2021. ISSN: 2577-087X.

[60] Kashish Gupta, Debasmita Mukherjee, and Homayoun Najjaran. Extending the Capabilities of Reinforcement Learning Through Curriculum: A Review of Methods and Applications. *SN Computer Science*, 3(1):28, October 2021.

[61] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018.

[62] Danijar Hafner. Benchmarking the Spectrum of Agent Capabilities. December 2021.

[63] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao. A Survey on Vision Transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):87–110, January 2023.

[64] Kai He, Nan Pu, Mingrui Lao, and Michael S. Lew. Few-shot and meta-learning methods for image understanding: a survey. *International Journal of Multimedia Information Retrieval*, 12(2):14, June 2023.

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *CoRR*, abs/1502.01852, 2015.

[66] Matteo Hessel, Ivo Danihelka, Fabio Viola, Arthur Guez, Simon Schmitt, Laurent Sifre, Theophane Weber, David Silver, and Hado Van Hasselt. Muesli: Combining Improvements in Policy Optimization. In *Proceedings of the 38th International Conference on Machine Learning*, pages 4214–4226. PMLR, July 2021. ISSN: 2640-3498.

[67] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[68] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-Learning in Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5149–5169, September 2022. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[69] Shengchao Hu, Li Shen, Ya Zhang, Yixin Chen, and Dacheng Tao. On Transforming Reinforcement Learning With Transformers: The Development Trajectory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):8580–8599, December 2024.

[70] Zhongxu Hu, Yiran Zhang, Yang Xing, Yifan Zhao, Dongpu Cao, and Chen Lv. Toward Human-Centered Automated Driving: A Novel Spatiotemporal Vision Transformer-Enabled Head Tracker. *IEEE Vehicular Technology Magazine*, 17(4):57–64, December 2022.

[71] Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving Transformer Optimization Through Better Initialization. In *Proceedings of the 37th International Conference on Machine Learning*, pages 4475–4483. PMLR, November 2020. ISSN: 2640-3498.

[72] Mike Huisman, Jan N. van Rijn, and Aske Plaat. A survey of deep meta-learning. *Artificial Intelligence Review*, 54(6):4483–4541, August 2021.

[73] Allan Jabri, Kyle Hsu, Abhishek Gupta, Ben Eysenbach, Sergey Levine, and Chelsea Finn. Unsupervised Curricula for Visual Meta-Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[74] Minqi Jiang, Edward Grefenstette, and Tim Rocktaeschel. Prioritized Level Replay. In *Proceedings of the 38th International Conference on Machine Learning*, pages 4940–4950. PMLR, July 2021. ISSN: 2640-3498.

[75] Lars Johannsmeier, Malkin Gerchow, and Sami Haddadin. A Framework for Robot Manipulation: Skill Formalism, Meta Learning and Adaptive Control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5844–5850, May 2019. ISSN: 2577-087X.

[76] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Huellermeier. A Survey of Reinforcement Learning from Human Feedback, December 2023. ADS Bibcode: 2023arXiv231214925K.

[77] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[78] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[79] Hang Lai, Weinan Zhang, Xialin He, Chen Yu, Zheng Tian, Yong Yu, and Jun Wang. Sim-to-Real Transfer for Quadrupedal Locomotion via Terrain Transformer. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5141–5147, May 2023.

[80] Robert Lange, Yingtao Tian, and Yujin Tang. Evolution Transformer: In-Context Evolutionary Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '24 Companion, pages 575–578, New York, NY, USA, August 2024. Association for Computing Machinery.

[81] Alessandro Lazaric and Mohammad Ghavamzadeh. Bayesian Multi-Task Reinforcement Learning. page 599. Omnipress, June 2010.

[82] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[83] Hung-yi Lee, Shang-Wen Li, and Ngoc Thang Vu. Meta Learning for Natural Language Processing: A Survey, July 2022. arXiv:2205.01500 [cs].

[84] Hung-yi Lee, Ngoc Thang Vu, and Shang-Wen Li. Meta Learning and Its Applications to Natural Language Processing. In David Chiang and Min Zhang, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Tutorial Abstracts*, pages 15–20, Online, August 2021. Association for Computational Linguistics.

[85] Jae-Jun Lee and Sung Whan Yoon. XB-MAML: Learning Expandable Basis Parameters for Effective Meta-Learning with Wide Task Coverage. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, pages 3196–3204. PMLR, April 2024. ISSN: 2640-3498.

[86] Jonathan Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. Supervised Pretraining Can Learn In-Context Reinforcement Learning. *Advances in Neural Information Processing Systems*, 36:43057–43083, December 2023.

[87] Joel Lehman and Kenneth O. Stanley. Novelty Search and the Problem with Objectives. In Rick Riolo, Ekaterina Vladislavleva, and Jason H. Moore, editors, *Genetic Programming Theory and Practice IX*, pages 37–56. Springer, New York, NY, 2011.

[88] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to Generalize: Meta-Learning for Domain Generalization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), April 2018. Number: 1.

[89] Xiangtai Li, Henghui Ding, Haobo Yuan, Wenwei Zhang, Jiangmiao Pang, Guangliang Cheng, Kai Chen, Ziwei Liu, and Chen Change Loy. Transformer-Based Visual Segmentation: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10138–10163, December 2024.

[90] Xianze Li, Hao Su, Ling Xiang, Qingtao Yao, and Aijun Hu. Transformer-based meta learning method for bearing fault identification under multiple small sample conditions. *Mechanical Systems and Signal Processing*, 208:110967, February 2024.

[91] Xiaoxu Li, Zhuo Sun, Jing-Hao Xue, and Zhanyu Ma. A concise review of recent few-shot meta-learning methods. *Neurocomputing*, 456:463–468, October 2021.

[92] Pengchen Liang, Bin Pu, Haishan Huang, Yiwei Li, Hualiang Wang, Weibo Ma, and Qing Chang. Vision Foundation Models in Medical Image Analysis: Advances and Challenges, February 2025. arXiv:2502.14584 [eess].

[93] Bryan Lim, Sercan O. Arık, Nicolas Loeff, and Tomas Pfister. Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, October 2021.

[94] Jian Lin, Haidong Shao, Xiangdong Zhou, Baoping Cai, and Bin Liu. Generalized MAML for few-shot cross-domain fault diagnosis of bearing driven by heterogeneous signals. *Expert Systems with Applications*, 230:120696, November 2023.

[95] Evan Z. Liu, Aditi Raghunathan, Percy Liang, and Chelsea Finn. Decoupling Exploration and Exploitation for Meta-Reinforcement Learning without Sacrifices. In *Proceedings of the 38th International Conference on Machine Learning*, pages 6925–6935. PMLR, July 2021. ISSN: 2640-3498.

[96] Hao Liu and Pieter Abbeel. Emergent Agentic Transformer from Chain of Hindsight Experience. In *Proceedings of the 40th International Conference on Machine Learning*, pages 21362–21374. PMLR, July 2023. ISSN: 2640-3498.

[97] Hao Liu, Richard Socher, and Caiming Xiong. Taming MAML: Efficient unbiased meta-reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4061–4071. PMLR, May 2019. ISSN: 2640-3498.

[98] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. September 2018.

[99] Gabriel Maicas, Andrew P. Bradley, Jacinto C. Nascimento, Ian Reid, and Gustavo Carneiro. Training Medical Image Analysis Systems like Radiologists. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*, pages 546–554, Cham, 2018. Springer International Publishing.

[100] Amir M. Mansourian, Rozhan Ahmadi, Masoud Ghafouri, Amir Mohammad Babaei, Elaheh Badali Golezani, Zeynab Yasamani Ghamchi, Vida Ramezanian, Alireza Taherian, Kimia Dinashi, Amirali Miri, and Shohreh Kasaei. A Comprehensive Survey on Knowledge Distillation, March 2025. arXiv:2503.12067 [cs].

[101] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–Student Curriculum Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3732–3740, September 2020.

[102] Daniel G. McClement, Nathan P. Lawrence, Michael G. Forbes, Philip D. Loewen, Johan U. Backstroem, and R. Bhushan Gopaluni. Meta-Reinforcement Learning for Adaptive Control of Second Order Systems. In *2022 IEEE International Symposium on Advanced Control of Industrial Processes (AdCONIP)*, pages 78–83, August 2022.

[103] Luckeciano C Melo. Transformers are Meta-Reinforcement Learners. *arXiv preprint arXiv:2206.06614*, 2022.

[104] Vladimir Mikulik, Grégoire Delétang, Tom McGrath, Tim Genewein, Miljan Martic, Shane Legg, and Pedro Ortega. Meta-trained agents implement Bayes-optimal agents. In *Advances in Neural Information Processing Systems*, volume 33, pages 18691–18703. Curran Associates, Inc., 2020.

[105] Michael Moor, Oishi Banerjee, Zahra Shakeri Hossein Abad, Harlan M. Krumholz, Jure Leskovec, Eric J. Topol, and Pranav Rajpurkar. Foundation models for generalist medical artificial intelligence. *Nature*, 616(7956):259–265, April 2023. Publisher: Nature Publishing Group.

[106] Kamil Nar and Shankar Sastry. Step Size Matters in Deep Learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[107] Tarun Naren, Yuanda Zhu, and May Dongmei Wang. COVID-19 diagnosis using model agnostic meta-learning on limited chest X-ray images. In *Proceedings of the 12th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, BCB '21, pages 1–9, New York, NY, USA, August 2021. Association for Computing Machinery.

[108] Thanh Nguyen, Tung Luu, Trung Pham, Sanzhar Rakhimkul, and Chang D. Yoo. Robust Maml: Prioritization Task Buffer with Adaptive Learning Process for Model-Agnostic Meta-Learning. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3460–3464, June 2021. ISSN: 2379-190X.

[109] Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent Model-Free RL Can Be a Strong Baseline for Many POMDPs, June 2022. arXiv:2110.05038 [cs].

[110] Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. When Do Transformers Shine in RL? Decoupling Memory from Credit Assignment. *Advances in Neural Information Processing Systems*, 36:50429–50452, December 2023.

[111] Shuteng Niu, Yongxin Liu, Jian Wang, and Houbing Song. A Decade Survey of Transfer Learning (2010–2020). *IEEE Transactions on Artificial Intelligence*, 1(2):151–166, October 2020. Conference Name: IEEE Transactions on Artificial Intelligence.

[112] Ben Norman and Jeff Clune. First-Explore, then Exploit: Meta-Learning to Solve Hard Exploration-Exploitation Trade-Offs. *Advances in Neural Information Processing Systems*, 37:27490–27528, December 2024.

[113] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler,

Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, C. J. Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. GPT-4 Technical Report, March 2024. arXiv:2303.08774 [cs].

[114] OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with Large Scale Deep Reinforcement Learning, December 2019. arXiv:1912.06680 [cs].

[115] Santisudha Panigrahi, Anuja Nanda, and Tripti Swarnkar. A Survey on Transfer Learning. In Debahuti Mishra, Rajkumar Buyya, Prasant Mohapatra, and Srikanta Patnaik, editors, *Intelligent and Cloud Computing*, pages 781–789, Singapore, 2021. Springer.

[116] Emilio Parisotto, Soham Ghosh, Sai Bhargav Yalamanchi, Varsha Chinnaobireddy, Yuhuai Wu, and Ruslan Salakhutdinov. Concurrent Meta Reinforcement Learning, March 2019. arXiv:1903.02710 [cs].

[117] Ramakanth Pasunuru and Mohit Bansal. Continual and Multi-Task Architecture Search, June 2019. arXiv:1906.05226 [cs].

[118] Gean Trindade Pereira. Meta-Learning applied to Neural Architecture Search. Towards new interactive learning approaches for indexing and analyzing images from expert domains, 2024.

[119] Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic Curriculum Learning For Deep RL: A Short Survey, May 2020. arXiv:2003.04664 [cs].

[120] Rémy Portelas, Clément Romac, Katja Hofmann, and Pierre-Yves Oudeyer. Meta Automatic Curriculum Learning, September 2021. arXiv:2011.08463 [cs].

[121] Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. Quality Diversity: A New Frontier for Evolutionary Computation. *Frontiers in Robotics and AI*, 3, July 2016. Publisher: Frontiers.

[122] Alireza Rafiei, Ronald Moore, Sina Jahromi, Farshid Hajati, and Rishikesan Kamaleswaran. Meta-learning in Healthcare: A Survey. *SN Computer Science*, 5(6):791, August 2024.

[123] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML, February 2020. arXiv:1909.09157 [cs, stat].

[124] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-Learning with Implicit Gradients. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[125] Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables, March 2019. arXiv:1903.08254 [cs, stat].

[126] Kanishka Ranaweera and Pubudu N. Pathirana. Optimizing Diagnosis in Sparse Data Environments: A Model Agnostic Meta Learning Approach. In *2024 46th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1–4, July 2024. ISSN: 2694-0604.

[127] Harish Ravichandar, Athanasios S. Polydoros, Sonia Chernova, and Aude Billard. Recent Advances in Robot Learning from Demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(Volume 3, 2020):297–330, May 2020. Publisher: Annual Reviews.

[128] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A Generalist Agent, November 2022. arXiv:2205.06175 [cs].

[129] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions. *ACM Comput. Surv.*, 54(4):76:1–76:34, May 2021.

[130] Micah Rentschler and Jesse Roberts. RL + Transformer = A General-Purpose Problem Solver, January 2025. arXiv:2501.14176 [cs].

[131] Gaith Rjoub, Saidul Islam, Jamal Bentahar, Mohammed Amin Almaiah, and Rana Alrawashdeh. Enhancing IoT Intelligence: A Transformer-based Reinforcement Learning Methodology. In *2024 International Wireless Communications and Mobile Computing (IWCMC)*, pages 1418–1423, May 2024. ISSN: 2376-6506.

[132] Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation, 2016.

[133] Noor Sajid, Philip J. Ball, Thomas Parr, and Karl J. Friston. Active Inference: Demystified and Compared. *Neural Computation*, 33(3):674–712, March 2021.

[134] Jürgen Schmidhuber. *Evolutionary Principles in Self-referential Learning: On Learning how to Learn: the Meta-meta-meta...-hook*. 1987.

[135] Jürgen Schmidhuber, Jieyu Zhao, and Marco Wiering. Shifting Inductive Bias with Success-Story Algorithm, Adaptive Levin Search, and Incremental Self-Improvement. *Machine Learning*, 28(1):105–130, July 1997.

[136] Simon Schmitt, Jonathan J. Hudson, Augustin Zidek, Simon Osindero, Carl Doersch, Wojciech M. Czarnecki, Joel Z. Leibo, Heinrich Kuttler, Andrew Zisserman, Karen Simonyan, and S. M. Ali Eslami. Kickstarting deep reinforcement learning, 2018.

[137] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised Pre-training for Speech Recognition, September 2019. arXiv:1904.05862 [cs].

[138] John Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

[139] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. arXiv:1707.06347 [cs].

[140] Snigdha Sen and Pavan Chakraborty. Evaluating Efficacy of MAML Based Approach on Regression Using Astronomical Imbalanced Dataset. In *2024 International Conference on Computational Intelligence and Network Systems (CINS)*, pages 1–8, November 2024.

[141] Gresa Shala, André Biedenkapp, and Josif Grabocka. Hierarchical Transformers are Efficient Meta-Reinforcement Learners, February 2024. arXiv:2402.06402 [cs].

[142] Gresa Shala, André Biedenkapp, Pierre Krack, Florian Walter, and Josif Grabocka. Efficient Cross-Episode Meta-RL. October 2024.

[143] Hava T. Siegelmann and Eduardo D. Sontag. On the computational power of neural nets. In *Proceedings of the fifth annual workshop on Computational learning theory*, Turing, pages 440–449, New York, NY, USA, July 1992. Association for Computing Machinery.

[144] Arturo Daniel Sosa-Ceron, Hugo Gustavo Gonzalez-Hernandez, and Jorge Antonio Reyes-Avendaño. Learning from Demonstrations in Human–Robot Collaborative Scenarios: A Survey. *Robotics*, 11(6):126, December 2022. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.

[145] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum Learning: A Survey. *International Journal of Computer Vision*, 130(6):1526–1565, June 2022.

[146] Bradly Stadie, Ge Yang, Rein Houthooft, Peter Chen, Yan Duan, Yuhuai Wu, Pieter Abbeel, and Ilya Sutskever. The Importance of Sampling inMeta-Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[147] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for Semantic Segmentation, May 2021. ADS Bibcode: 2021arXiv210505633S.

[148] Richard S. Sutton. A History of Meta-gradient: Gradient Methods for Meta-learning, February 2022. ADS Bibcode: 2022arXiv220209701S.

[149] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning series. The MIT Press, Cambridge, Massachusetts, 2. edition, 2018.

[150] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[151] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper With Convolutions. pages 1–9, 2015.

[152] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A Survey on Deep Transfer Learning. In Věra Kůrková, Yannis Manolopoulos, Barbara Hammer, Lazaros Iliadis, and Ilias Maglogiannis, editors, *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 270–279, Cham, 2018. Springer International Publishing.

[153] Mingxing Tan and Quoc Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6105–6114. PMLR, May 2019. ISSN: 2640-3498.

[154] Yanchao Tan, Carl Yang, Xiangyu Wei, Chaochao Chen, Weiming Liu, Longfei Li, Jun Zhou, and Xiaolin Zheng. MetaCare++: Meta-Learning with Hierarchical Subtyping for Cold-Start Diagnosis Prediction in Healthcare Data. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, pages 449–459, New York, NY, USA, July 2022. Association for Computing Machinery.

[155] Suigu Tang, Xiaoyuan Yu, Chak Fong Cheang, Yanyan Liang, Penghui Zhao, Hon Ho Yu, and I Cheong Choi. Transformer-based multi-task learning for classification and segmentation of gastrointestinal tract endoscopic images. *Computers in Biology and Medicine*, 157:106723, May 2023.

[156] Adaptive Agent Team, Jakob Bauer, Kate Baumli, Satinder Baveja, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, Vibhavari Dasagi, Lucy Gonzalez, Karol Gregor, Edward Hughes, Sheleem Kashem, Maria Loks-Thompson, Hannah Openshaw, Jack Parker-Holder, Shreya Pathak, Nicolas Perez-Nieves, Nemanja Rakicevic, Tim Rocktaeschel, Yannick Schroecker, Jakub Sygnowski, Karl Tuyls, Sarah York, Alexander Zacherl, and Lei Zhang. Human-Timescale Adaptation in an Open-Ended Task Space, January 2023. arXiv:2301.07608 [cs].

[157] Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-Fitt, Valentin Dalibard, and Wojciech Marian Czarnecki. Open-Ended Learning Leads to Generally Capable Agents, July 2021. arXiv:2107.12808 [cs].

[158] Sebastian Thrun and Lorien Pratt. Learning to Learn: Introduction and Overview. In Sebastian Thrun and Lorien Pratt, editors, *Learning to Learn*, pages 3–17. Springer US, Boston, MA, 1998.

[159] Yuanyi Tian. A Meta-Learning Prediction Model for Identifying miRNA-Disease Associations Based on MAML. In *2024 5th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, pages 1073–1077, March 2024.

[160] Yuanyi Tian, Quan Zou, ChunYu Wang, and Cangzhi Jia. MAMLCDA: A Meta-Learning Model for Predicting circRNA-Disease Association Based on MAML Combined With CNN. *IEEE Journal of Biomedical and Health Informatics*, 28(7):4325–4335, July 2024.

[161] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[162] Julian Togelius and Georgios N. Yannakakis. Choose Your Weapon: Survival Strategies for Depressed AI Academics [Point of View]. *Proceedings of the IEEE*, 112(1):4–11, January 2024.

[163] Lovre Torbarina, Tin Ferkovic, Lukasz Roguski, Velimir Mihelcic, Bruno Sarlija, and Zeljko Kraljevic. Challenges and Opportunities of Using Transformer-Based Multi-Task Learning in NLP Through ML Lifecycle: A Position Paper. *Natural Language Processing Journal*, 7:100076, June 2024.

[164] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models, February 2023. ADS Bibcode: 2023arXiv230213971T.

[165] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples, April 2020. arXiv:1903.03096 [cs].

[166] Richa Upadhyay. Sharing to learn and learning to share : Fitting together metalearning and multi-task learning. 2023. Publisher: Luleå University of Technology.

[167] Joaquin Vanschoren. Meta-Learning: A Survey, October 2018. ADS Bibcode: 2018arXiv181003548V.

[168] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[169] Anna Vettoruzzo, Mohamed-Rafik Bouguelia, Joaquin Vanschoren, Thorsteinn Roegnvaldsson, and KC Santosh. Advances and Challenges in Meta-Learning: A Technical Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(7):4763–4779, July 2024. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[170] Nelson Vithayathil Varghese and Qusay H. Mahmoud. A Survey of Multi-Task Deep Reinforcement Learning. *Electronics*, 9(9):1363, September 2020. Number: 9 Publisher: Multidisciplinary Digital Publishing Institute.

[171] Riccardo Volpi, Diane Larlus, and Gregory Rogez. Continual Adaptation of Visual Representations via Domain Randomization and Meta-Learning. pages 4443–4453, 2021.

[172] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to Unseen Domains via Adversarial Data Augmentation. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[173] Haoxiang Wang, Yite Wang, Ruoyu Sun, and Bo Li. *Global Convergence of MAML and Theory-Inspired Neural Architecture Search for Few-Shot Learning*, pages 9797–9808. 2022.

[174] Jiahao Wang, Zutong Sun, Hang Yan, and Hao Chen. MAML-TSC: Model-Agnostic Meta-Learning for Time Series Classification. In *2024 7th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*, pages 1043–1049, August 2024.

[175] Xin Wang, Yudong Chen, and Wenwu Zhu. A Survey on Curriculum Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):4555–4576, September 2022.

[176] Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson. RL-VLM-F: Reinforcement Learning from Vision Language Foundation Model Feedback, June 2024. arXiv:2402.03681 [cs].

[177] Haiyue Wu, Matthew J. Triebe, and John W. Sutherland. A transformer-based approach for novel fault detection and fault classification/diagnosis in manufacturing: A rotary system application. *Journal of Manufacturing Systems*, 67:439–452, April 2023.

[178] Hanguang Xiao, Li Li, Qiyuan Liu, Xiuhong Zhu, and Qihang Zhang. Transformers in medical image segmentation: A review. *Biomedical Signal Processing and Control*, 84:104791, July 2023.

[179] Tengye Xu, Zihao Li, and Qinyuan Ren. Meta-Reinforcement Learning Robust to Distributional Shift Via Performing Lifelong In-Context Learning. June 2024.

[180] Zeqiu Xu, Jiani Wang, Xiaochuan Xu, Peiyang Yu, Tianyi Huang, and Jingyuan Yi. A Survey of Reinforcement Learning-Driven Knowledge Distillation: Techniques, Challenges, and Applications. In *2025 IEEE 6th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, pages 1–6, April 2025.

[181] Zifan Xu, Yulin Zhang, Shahaf S. Shperberg, Reuth Mirsky, Yuqian Jiang, Bo Liu, and Peter Stone. Model-Based Meta Automatic Curriculum Learning. In *Proceedings of The 2nd Conference on Lifelong Learning Agents*, pages 846–860. PMLR, November 2023. ISSN: 2640-3498.

[182] Weirui Ye, Yunsheng Zhang, Haoyang Weng, Xianfan Gu, Shengjie Wang, Tong Zhang, Mengchen Wang, Pieter Abbeel, and Yang Gao. Reinforcement Learning with Foundation Priors: Let the Embodied Agent Efficiently Learn on Its Own, October 2024. arXiv:2310.02635 [cs].

[183] Shuolei Yin, Yejing Xi, Xun Zhang, Chengnuo Sun, and Qirong Mao. Foundation Models in Agriculture: A Comprehensive Review. *Agriculture*, 15(8):847, January 2025. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute.

[184] Wenpeng Yin. Meta-learning for Few-shot Natural Language Processing: A Survey, July 2020. arXiv:2007.09604 [cs].

[185] Jun Yu, Yutong Dai, Xiaokang Liu, Jin Huang, Yishan Shen, Ke Zhang, Rong Zhou, Eashan Adhikarla, Wenxuan Ye, Yixin Liu, Zhaoming Kong, Kai Zhang, Yilong Yin, Vinod Namboodiri, Brian D. Davison, Jason H. Moore, and Yong Chen. Unleashing the Power of Multi-Task Learning: A Comprehensive Survey Spanning Traditional, Deep, and Pretrained Foundation Model Eras, April 2024.

[186] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning. In *Proceedings of the Conference on Robot Learning*, pages 1094–1100. PMLR, May 2020. ISSN: 2640-3498.

[187] Wenhao Yu, Visak CV Kumar, Greg Turk, and C. Karen Liu. Sim-to-Real Transfer for Biped Locomotion. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3503–3510, November 2019. ISSN: 2153-0866.

[188] Jin Zhang, Jianhao Wang, Hao Hu, Tong Chen, Yingfeng Chen, Changjie Fan, and Chongjie Zhang. MetaCURE: Meta Reinforcement Learning with Empowerment-Driven Exploration. In *Proceedings of the 38th International Conference on Machine Learning*, pages 12600–12610. PMLR, July 2021. ISSN: 2640-3498.

[189] Shaoting Zhang and Dimitris Metaxas. On the challenges and perspectives of foundation models for medical image analysis. *Medical Image Analysis*, 91:102996, January 2024.

[190] Shuang Zhang, Rui Fan, Yuti Liu, Shuang Chen, Qiao Liu, and Wanwen Zeng. Applications of transformer-based language models in bioinformatics: a survey. *Bioinformatics Advances*, 3(1):vbad001, January 2023.

[191] Yi Zhang, Yanji Hao, Yu Fu, Yijing Feng, Yeqing Li, Xiaonan Wang, Junting Pan, Yongming Han, and Chunming Xu. GAN-MAML strategy for biomass energy production: Overcoming small dataset limitations. *Applied Energy*, 387:125568, June 2025.

[192] Yingying Zhang, Xian Wu, Quan Fang, Shengsheng Qian, and Changsheng Xu. Knowledge-Enhanced Attributed Multi-Task Learning for Medicine Recommendation. *ACM Trans. Inf. Syst.*, 41(1):17:1–17:24, January 2023.

[193] Yu Zhang and Qiang Yang. An overview of multi-task learning. *National Science Review*, 5(1):30–43, January 2018.

[194] Yu Zhang and Qiang Yang. A Survey on Multi-Task Learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, December 2022. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

[195] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744, December 2020.

[196] Yue Zhou, Houjin Chen, Yanfeng Li, Qin Liu, Xuanang Xu, Shu Wang, Pew-Thian Yap, and Dinggang Shen. Multi-task learning for segmentation and classification of tumors in 3D automated breast ultrasound images. *Medical Image Analysis*, 70:101918, May 2021.

[197] Zhuangdi Zhu, Kaixiang Lin, Anil K. Jain, and Jiayu Zhou. Transfer Learning in Deep Reinforcement Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):13344–13362, November 2023. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[198] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE*, 109(1):43–76, January 2021. Conference Name: Proceedings of the IEEE.

[199] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast Context Adaptation via Meta-Learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 7693–7702. PMLR, May 2019. ISSN: 2640-3498.

[200] Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. VariBAD: A Very Good Method for Bayes-Adaptive Deep RL via Meta-Learning, February 2020. arXiv:1910.08348 [cs, stat].

[201] Luisa M. Zintgraf, Leo Feng, Cong Lu, Maximilian Igl, Kristian Hartikainen, Katja Hofmann, and Shimon Whiteson. Exploration in Approximate Hyper-State Space for Meta Reinforcement Learning. In *Proceedings of the 38th International Conference on Machine Learning*, pages 12991–13001. PMLR, July 2021. ISSN: 2640-3498.

A Bayesian Reinforcement Learning

Bayesian Reinforcement Learning (BRL) provides a sophisticated approach to tackle the exploration/-exploitation dilemma by allowing agents to quantify uncertainty. It is the groundwork for the task-representation algorithms presented in sections C and 3.3. The main idea of BRL is to leverage the gathered information of the agent together with Bayesian methods in order to enable the agent to adapt its strategy as more information becomes available. For this purpose, one extends the notion of a MDP of the form (5) to that of a Bayes-Adaptive MDP (BAMDP)[16], i.e., a tuple

$$M' := (\mathbb{A}, \mathbb{S}', R', \gamma, \mathbb{T}', \mu', h'), \tag{17}$$

with a hyperstate-space $\mathbb{S}' = \mathbb{S} \times B$ as the extension of the original state space $\mathbb{S}$ by the belief space B capturing the possible parameters of a model of the environment dynamics $\mathbb{T}$ and R[17]. The augmented transition and reward functions, as well as the initial distribution and horizon, are defined analogously to the hyper-state space (see [200], [52] for further details). In terms of Example 2, this means, for example, to capture a belief of the current slipperiness of the track while collecting more experience to make this belief more precise.

The current belief $b_t := p_t = p(R', \mathbb{T}'|c_t) \in B$ is defined as the posterior of the dynamics model parameters given the current experience (also named context)

$$c_t := \{s_i, a_i, r_{i+1}, s_{i+1}\}_{i=0}^{t} \tag{18}$$

[16] In [52] it is done for discrete state and action spaces, but it is straight forward to adapt this for continuous spaces

[17] As all BRL-based algorithms presented in Section 3 use model-based approaches, model-free BRL is not discussed in this work.

and a prior distribution $p_0 = p(R', \mathbb{T}')$ over the unknown reward and transition functions $\mathbb{T}$ and R. The prior is often chosen as a normal distribution, while one computes the posterior p_t according to the Bayesian rule after collecting experience c_t:

$$p_t = p(R, T|c_t) = \frac{p(c_t|R, T)p(R, T)}{\int p(c_t|x)p(x)dx}. \tag{19}$$

The posterior encodes the agent's uncertainty about model environment parameters. However, its computation is generally intractable so that the respective BRL algorithm must approximate it accordingly. Since the agent aims to select the best possible action in each time step based on the current belief, while simultaneously refining the posterior belief whenever new information becomes available, it implicitly trades-off exploration and exploitation this way.

One calls agents Bayes-optimal, if they optimally trade-off exploration and exploitation. Throughout early learning, the value (6) of the corresponding Bayes-optimal policy is typically lower than that of the optimal policy π^*, since the agent must, at first, explore the environment before finding the optimal dynamics model parameters. In Example 2, the Bayes-optimal behaviour initially requires driving and braking a few times to figure out how slippery the track is, although this increases the racing time in the first training episode.

B Terminology

In the literature, meta-learning is often confused with the notions of transfer learning (TL) and multi-task learning (MTL). Although these concepts share quite some similarities - there are even algorithms exhibiting overlapping characteristics among them [166] - the confusion mainly arises from a lack of clear definition. Therefore, the following paragraphs broadly introduce TL and MTL focusing on a clearly defined paradigm. For a more detailed consideration, the reader is, nevertheless, referred to other works like [166] particularly focusing on meta-learning, MTL, TL and their overlaps.

B.1 Transfer Learning

The significance of the concept of Transfer Learning notably heightened following its integration with deep neural networks. Landmark advancements in computer vision, exemplified by architectures such as AlexNet [78], GoogleNet [151], and EfficientNet [153], are pre-trained on the extensive ImageNet dataset before employing them to tasks such as image classification, segmentation, and object detection. These models demonstrated substantial improvements over traditional machine learning algorithms in terms of adaptability, sample efficiency, and few-shot performance. Motivated by these achievements, transfer learning has also been effectively applied in various other domains, including speech recognition (e.g. Wav2Vec [137]), healthcare applications [38], and reinforcement learning (RL). The latter itself distributes into various applications as knowledge transfer is used to close the gap between simulation and reality in robotics [187], learn from demonstrations to incorporate expert knowledge [127], [144], or adapt to new rules and gaming mechanics in game play using reward shaping [114]. All these various sub fields of TL and transfer RL present distinct challenges and are hence areas of ongoing research. Interested readers are thus encouraged to explore specific surveys on TL [111], [198], [152], [115], and transfer RL [197], [195] for deeper insights.

Transfer Learning Paradigm

The main idea of TL is to extensively pre-train a model F_θ on a source task $T_1 := \{\mathcal{L}_1, \mathcal{X}_1, \mu, \mathbb{T}_1, h\}$ and transfer the acquired knowledge by fine-tuning f_θ on a related target task $T_2 := \{\mathcal{L}_2, \mathcal{X}_2, \mu, \mathbb{T}_2, h\}$. Mirroring human learning, this approach leverages knowledge from pre-training to enhance few-shot performance and learning speed on the target task T_2. For example, one can extensively practice inline skating in the summer to prepare for skiing when it snows for only a few days during winter. Therefore, the source task T_1 typically contains abundant, well-labeled data $\mathcal{X}_1$, while the target task's data $\mathcal{X}_2$ often is sparse, of low quality or expensive to obtain. Additionally, the goal encoded by $\mathcal{L}_1$ can differ from that represented by the loss function $\mathcal{L}_1$ or there may be a change in domain dynamics $\mathbb{T}$ that needs to be learned quickly. The latter is of particular importance in RL problems, where changes in environment

dynamics can cause severe issues; for example when an agent is supposed to drive a car and the terrain becomes significantly more slippery. A change in the loss function can, however, become problematic if the loss $\mathcal{L}_2$ is considerably more computationally expensive to calculate as this slows down learning.

One can also define the TL paradigm with the possibility of several source tasks. However, this is rather a technical difference as one can define $T_1 = \bigcup_{i=1}^{N} T_1^i$, where T_1^i are the different source tasks. If, however, more than one target task exists, one typically uses the notion "downstream tasks". This is the case in more recent developments, where TL, along with a vast amount of data and self-supervised learning approaches, led to much more powerful and broadly applicable models, namely foundation models.

Foundation Models

Foundation models serve as a common basis from which many task-specific models are built through adaptation [16], i.e. they are pre-trained by a vast amount of source tasks as described above, before being fine-tuned to different downstream tasks. As a consequence, foundation models are inherently incomplete: they rather serve as a "foundation" for fine-tuning to various applications than as static general problem solvers. This way, the adaptation of foundation models follows the meta-testing paradigm described in Section 2.1, while their training is that of a TL algorithm not following the meta-training paradigm of Section 2.1. As they are meant to be trained on a distribution $p(T)$ of training tasks rather than one single source task before being fine-tuned to a (potentially different) distribution of downstream tasks $p_{\text{test}}(T)$, they can be viewed as the "meta-level version" of TL.

TL is the key technique that makes foundation models possible, but scale is what makes them so powerful [16]. This fact probably arises from the universal approximator property of neural networks: larger models can represent more complex functions of higher dimension, although they also tend to overfit more easily. The latter is the reason why besides model size, the data (e.g., the task pool in meta-learning) is also required to scale [156], [16]. However, as the amount of data increases, the data quality can not necessarily be maintained. Instead, self-supervised learning techniques [57], [37] are used to learn patterns in available, but potentially unlabeled, data, so that foundation models are often defined as "large-scale transfer-learning models pre-trained via self-supervised learning" [16].

There was a paradigm shift towards foundation models that get fine-tuned for downstream tasks within the recent years [16]. For example, almost all state-of-the-art NLP models are now adapted from one of a few foundation models, e.g., Bert [32], GPT3 [18], GPT4 [113], LLAMA [164]. Similarly, foundation models were developed for various applications like agriculture [183], medicine [189], [92], [105], or robotics [46]. In RL, foundation models have been used for reward engineering [182], [176], e.g., to generate rewards from only text description and the visual observations yielded from the environment [176]. Additionally, [130] successfully fine-tune the LLAMA [164] foundation model on RL downstream tasks without RL-specific training with remarkable results. As a consequence, several works combine NLP and RL paradigms into RL from Human Feedback (RLHF), which is itself an ongoing and vast research field. We refer interested readers to particular RLHF surveys like [76], [19], [22].

B.2 Multi-Task Learning

The concept of Multi-Task Learning (MTL) was first introduced by [1] as the formalization of the idea of training models on multiple tasks simultaneously. The main idea is to train one single model among several similar tasks simultaneously in order to find valuable, general feature representations that can be leveraged to improve task-specific performances.

Applying the paradigm of MTL to different neural network architectures like Transformers [163] or Bayesian networks [81], ..., led to remarkable successes in natural language processing [24], neural architecture search [49], [117], segmentation [15], [196], [155], medicine [13], [192], [196], and robotics [5], [59], [29], to list only a few. We, however, refer interested readers to MTL-focused surveys like [194], [193], [185] or Multi-Task-RL (MTRL) specific works like [170] or [31] for a broader overview of MTL and MTRL algorithms. This section rather focuses on MTL and MTRL paradigms to help readers distinguish them from TL and meta-learning or transfer-RL and meta-RL respectively.

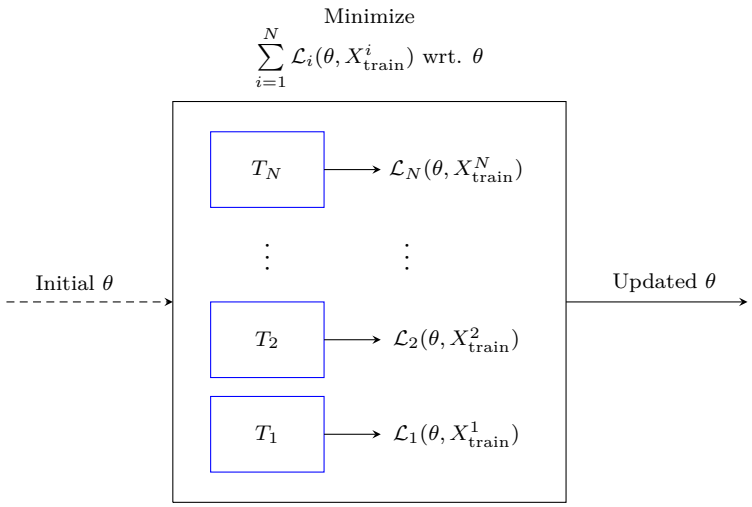

Figure 6: General Multi-Task-Learning Paradigm:

Multi-Task Learning Paradigm

Formally, in MTL a model is jointly trained to solve a given number N of tasks $(T_1, T_2, \ldots, T_n)$ sharing some common structure. The corresponding training process mimics standard learning on task-specific level, but with parameters being shared throughout the tasks. A meta-level does not exist and hence there is no two-staged training process. Instead, the MTL paradigm aims to solve all N tasks equally well. Figure 6 visualizes the resulting optimization problem. It can be formalized as

$$\text{Minimize} \quad \sum_{i=1}^{N} \mathcal{L}_i(\theta_i, X^i_{\text{train}}) \quad \text{wrt. } \theta \tag{20}$$

where $\theta_i \in \mathbb{R}^{d_i}$ denotes the task-specific parameters reserved for each individual task.

The task specific parts of the model typically are task-specific heads with one or more layers, so that the model parameters θ consist of task-specific parameters θ_i for each task T_i as well as common knowledge denoted by φ, but all in one single model. The performance of the MTL model is evaluated on task-level only, i.e. on the task-specific test sets X^i_{test}. A meta-test on unseen tasks does not take place.

In contrast to the meta-learning paradigm described in Section 2.1, the MTL paradigm does not treat the parameters θ_i of each task as individual parameter sets. They are rather considered as the different parts of one bigger set of parameters $\theta = \{\theta_0, \theta_1, \ldots, \theta_N\}$ with $\theta_0 \in \mathbb{R}^{d_0}$ denoting the common knowledge over all tasks. A deep neural network can, for example, learn some feature representations shared between all different tasks to extract the relevant information of its data before fine-tuning to the respective task. Such a NN theoretically consists of different blocks: A general part encoding all the information shared between tasks, and a smaller head for each individual task.

Multi Task Reinforcement Learning

In Multi-Task Reinforcement Learning (MTRL) one develops a policy for several similar but different tasks at once. Analogous to the general MTL paradigm, the common knowledge is about how to process the information the current state provides, which action has which effect on the environment and what are general dynamics shared among tasks. At the same time, rewards and transitions differ between the different MDPs, what requires for task-specific heads to decide for an action out of the (commonly) processed state information. Since the notion of a policy is rather a theoretical concept, this means the same neural network can function as the agent in the different MDPs with a task-specific head for each MDP. A second part of potentially common knowledge is task identification. However, compared to the more general meta-RL paradigm, this is rather simple: The model always faces the same N tasks. Hence, most algorithms simply use one-hot encoding.

In the literature, MTRL is often confused with Multi-Agent learning - where multiple agents act within the same environment while trying to increase individual or collective rewards - or Multi-Actor Learning - where multiple agents with the exact same goal collect experiences and share them for faster learning. But these two scenarios correspond to the single-task setting. The former extends the notion of single-task to multiple agents, while the latter only uses multiple instances of the environment in order to parallelize training.

B.3 Comparison

For differentiating between the three different paradigms of TL, MTL and Meta-Learning, it is important to understand that they are answers of increasing levels of abstraction [166] to the question of how to exploit knowledge from previously learned tasks to solve new ones in only a few shots. TL, as the solution with the lowest level of abstraction, simply transfers the knowledge gained in one task (the source task) to a second one (the target task), while Meta-Learning, as the solution with the highest level of abstraction, generally tries to extract the core information to learn any task of a distribution p in a few shots. In other words, Meta-Learning also extracts information from the source task(s) to use them as a prior for the target task(s), but this information can be much more general, and it results from the outer optimization rather than from solving the source task(s) in particular - a difference becoming even more unclear when foundation models get pre-trained on several source tasks in order to be meta-tested afterwards.

The most confusion in the literature is between MTL and meta-learning [166] since both aim to solve several tasks as well as possible rather than exclusively focusing on one single target task (like in TL). While meta-learning iteratively and sequentially learns a number N of tasks not necessarily fixed and, afterwards, tests the thus acquired knowledge on test tasks unseen during training, MTL trains a fixed number N of tasks simultaneously and without testing the resulting model parameters θ on unseen tasks. The trained MTL model is, instead, evaluated on the validation set X^i_{test} of each of the learned tasks, $T_1, \dots, T_N$ which corresponds to the standard machine learning test phase but for N tasks at the same time. In this way, MTL rather focuses on best solving the learned tasks than on adapting to new ones as fast and good as possible.

MTL can consist of tasks of different families, such as classification, regression, or segmentation, while meta-learning is normally reduced to one family of tasks represented by the distribution p over tasks of the form (2). And, most importantly, MTL does not consist of an outer learner extracting prior knowledge (or meta representations). In this specific aspect, MTL is more similar to transfer learning, where prior knowledge between tasks is only transported implicitly via the model parameters. However, the TL paradigm particularly requires a target task to transfer the gained knowledge to, while the MTL paradigm does not specifically include such an option. This further highlights the similarity between TL and meta-learning as well as the difference between meta-learning and MTL.

C Efficient Off-Policy Meta-RL

Efficient Off-Policy Meta-RL (PEARL) [125] is the landmark gradient-based task inference meta-RL algorithm and a frequently used benchmark for other meta-RL algorithms. In contrast to other extensions of MAML, such as α-MAML [10] or PMAML [44] , which only slightly modify the original paradigm, PEARL introduces a substantially different meta-learning scheme. It not only incorporates task inference (as also proposed in [44] or [55]), but also reformulates the meta-training scheme to off-policy learning. The main idea of PEARL is to infer a MDP-level latent context variable z_i via Bayesian updates that encodes an MDP M_i based on the current context (18). This context variable can either represent function approximations like the value function (6) (model-free approach), or the dynamics $\mathbb{T}_i$ and R_i of the current MDP M_i (model-based approach). It is sampled from a prior network $p_\varphi(z_i|c_i)$ at the beginning of each RL episode, whose weights φ_z are learned at meta-level. During inner learning, the prior network updates implicitly through the context and thereby approximates the corresponding Bayesian posterior update (19).

The policy $\pi(\cdot|z_i)$ is a neural network with meta-learned weights φ. According to the BRL paradigm, it depends on the context variable z_i, i.e., on the belief about the current MDP. Hence, it implicitly adapts to a particular MDP through the context, so that a MDP-level gradient descent update of the policy weights φ is not required. This makes PEARL off-policy: the policy π_φ is only updated at meta-level, but

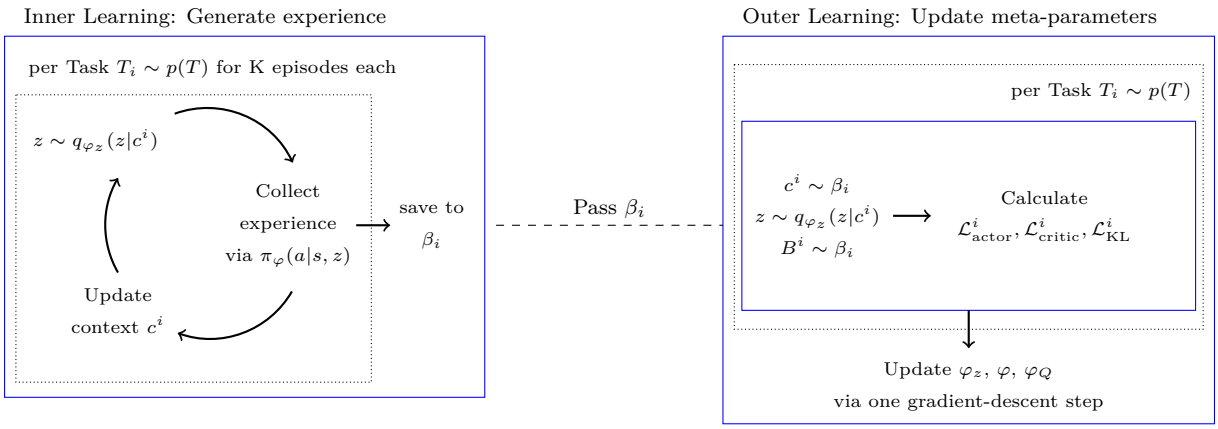

Figure 7: The PEARL meta-training scheme. In each MDP M_I, PEARL collects K episodes of experience and stores them in replay buffer β_i. It samples the context variable Z at the beginning of each episode. After inner learning, minibatches are sampled from the replay buffer to update meta-parameters φ, θ_π and θ_Q.

remains unchanged throughout MDP-level learning. Instead, the belief updates based on the gathered experience. In their original publication, [125] used Soft-Actor-Critic [61]. However, this is rather a design choice than a necessity of the PEARL paradigm, but it incorporates a critic Q_{φ_Q} into the framework whose parameters φ_Q must also be meta-learned via meta-gradient descent.

Meta-Training and Meta-Testing

Figure 7 illustrates the meta-training scheme of PEARL. At the beginning of each RL episode, z_i is sampled from the prior distribution $q_{\varphi_z}(\cdot|c_i)$ conditioned on the current context c^i. Then, the policy $\pi_\varphi(\cdot|z_i)$ gathers one RL episode of experience c^i_h, and a MDP-specific replay buffer β_i stores this experience for meta-level loss calculation. a MDP-specific test set X^i_{test} is not required[18], As all meta-losses are calculated on the experience gathered and stored this way.

The meta-level updates of φ, φ_z and φ_Q are one meta-gradient descent step of the form (13), with the meta-losses for the actor π_φ and critic Q_{φ_Q} defined as the sum of the respective MDP-level losses. To stabilize training, the meta-loss for φ_z contains a KL-divergence penalty

$$\mathcal{L}^i_{\text{KL}} := \mathrm{D}_{\text{KL}}(p(z_i \mid c^i_t), r(z_i))$$

with context c^i_t sampled from replay buffer β_i. The baseline distribution r is a hyperparameter of PEARL and chosen as a normal distribution in the original publication. The corresponding meta-loss for φ_z is

$$\mathcal{L}^{\varphi_z}_{\text{meta}} := \sum_{M_i} \left(\mathcal{L}^i_{\text{critic}} + \mathcal{L}^i_{\text{KL}}\right).$$

It, additionally, contains the MDP-level critic losses, because the functions frequently used for critics (like (6)) highly depend on which MDP the agent is currently acting in.

The meta-testing scheme of PEARL is analogous to the general meta-testing scheme presented in Section 2.2. However, as the experience gathered during inner learning are saved in the replay buffer, no test episodes are required for performance evaluation. But, similar to meta-training, this is rather a design choice of [125] than a necessity.

Performance Analysis

PEARL outperforms MAML and RL^2 in terms of performance, adaptation speed and sample-efficiency on the MuJoCo benchmark [161] - most likely due to its off-policy learning scheme. However, in contrast to more sophisticated meta-RL algorithms, such as VariBAD or TrMRL, it always requires two episodes of exploration (see, e.g., [200], Figure 6, or [103], Figure 5) before it performs comparatively well. This

[18] It is rather a design choice than a theoretical necessity to not sample another K episodes with fixed z_i or p_{φ_z}.

indicates a high adaptation speed and a high sample-efficiency, as well as poor zero-shot performance. The latter becomes even more severe in more complex tasks, such as Meta-World, where PEARL is not even able to solve tasks until the third episode ([103], Figures 4 and 5), which indicates poor exploration behaviour. This motivates more sophisticated gradient-based algorithms, such as DREAM [95], MAESN [58] and MetaCure [188], which are more tailored to optimally explore at the beginning of MDP-specific adaptation.

D Attention is all you need

The following paragraphs broadly discuss the architecture of the vanilla transformer [168]. The goal is to give the reader an intuition how self-attention "creates context" and "focuses" on important information on the example of language-to-language translation. For a detailed description of the vanilla transformer architecture the interested reader is nevertheless referred to the original paper or [2].

The vanilla transformer consists of an encoder and a decoder, which both divide into six 2- or 3-layer sub-blocks respectively[19]. Metaphorically speaking, the encoder translates an input sentence like

The child plays football because it likes it very much.

into "machine language", i.e. word embeddings, while the decoder translates these embeddings back into the desired output language. However, a word-by-word transformation from the input language into "'machine language "' and back into the output language preserves no information about what word the first "it" refers to and to which word the second "it" is related to. One needs a possibility to preserve the context between the words and, hence, the semantic structure of the sentence while encoding it. Similarly, while decoding, the syntactic structure of the sentence must be transformed so that it fits the grammar of the output language. For example, a German sentence has a totally different grammatical structure than its English counterpart with the same semantic meaning. In their original publication, [168] solved these problems using two techniques: Self-attention layers and positional encodings. The following paragraph discusses the former, while the latter is broadly discussed further below.

D.1 Self-Attention

In a self-attention layer, every word embedding x_i $(i = 1, 2, \dots, n)$ of a n-word input sentence x like the one above gets a query, key and value projection matrix Q_i, K_i and V_i assigned to it. These matrices are parameters of the self-attention and, as such, get learned during training. They project a word embedding x_i into query, key and value vectors q_i, k_i and v_i by multiplication. The "context" between words is then considered in the following way:

- Every word embedding x_i gets "compared" to every other word embedding x_j by multiplying every query vector q_i with every key vector k_j in a dot product. One can interpret the result as the context between the corresponding words of the input sentence since, mathematically, the dot product puts the query and key vectors in geometrical relationship to each other.
- For each word embedding x_i, the result of its query-key multiplication with every other word embedding (including itself) is a vector representing the attention scores of the word embedding x_i to all other word embeddings x_j. This vector of attention scores is fed into a softmax function to yield the respective attention weights summing up to one.
- At last, these "attention weights" are multiplied by the value vector v_i, resulting in a weighted sum over attention weights with weights v_i that directly depend on the learned projection V_i. Hence, the model can learn which attention scores $q_i \cdot k_j$ are more or less important by adjusting the values of the value projection matrix V_i respectively.

In practice, one typically simultaneously calculates all query-key pairs of the embedded input sentence x:

$$\text{Attention}(x) := \text{softmax}(\frac{QK^T}{\sqrt{d}})\ V, \tag{21}$$

[19] This number of six sub-blocks for encoder and decoder is rather an architectural choice by [168] than a necessity of the vanilla transformer.

with the hyperparameter d as the dimension of key, value and query vectors, $\sqrt{d}$ as the scaling factor and K, V and Q being the respective key, value and query matrices. The scaling factor $\sqrt{d}$ keeps the dot product logits small so that the soft-max does not slip into saturation. This maintains a proper gradient flow and thus stabilizes training.

Each encoder sub-block of the vanilla transformer consists of a self-attention layer so that the context between words (and hence the semantic structure of the sentence) is preserved throughout the whole embedding process. Encoding the above example sentence, the model can learn through the query-key multiplication $q_1 \cdot k_2$, that the first word "The" refers to the next word "child". At the same time, this connection is almost meaningless to the semantic of the whole sentence so that the model possibly learns to assign a very small value to the value vector v_1.

The self-attention layers of the decoder sub-blocks look slightly different to those described above. Since decoding the embeddings into output language is a sequential manner, no word can be set into context to words following later on in the sentence. Therefore, the corresponding query-key multiplications $q_i \cdot k_{i+a}$ with $a = 1, \ldots, n - i$ are set to minus infinity by construction so that the respective softmax results in zero attention weights.

In their original work, [168] implement multi-head-attention with the number of heads as a hyperparameter and show this to further boost performance. Multi-Head Attention duplicates the word embeddings x_i, projects them into the smaller subspace of each head [20], executes the Self-Attention of the form (21), , concatenates the outputs of all heads to one single output matrix, and projects them back into the original output dimension d by another linear layer transformation.

D.2 Positional Encoding

In the self-attention (21) the order of input tokens does not matter and hence the order of the words of the input sentence is not automatically preserved throughout encoding. However, the order of words in a sentence is quite important, especially as syntactic structure of input and output languages might differ, e.g. when translating English into German. This is why, [168] include a positional embedding for each word embedding x_i in the vanilla transformer architecture that has the same length as the embedding itself. The form of the positional encoding p_i is a hyperparameter of the vanilla transformer. In their original work, [168] combine different sin and cosine functions, but state that they have tried different positional encodings without a major loss in performance. Other transformer variants even learn this positional embedding function along with all the above model parameters.

[20] the queries, keys, values of each head are smaller-dimensional vectors.

Table 1: Advantages and disadvantages of the different components introduced along with the landmarks in Section 3. The last column lists algorithms containing the respective component.

Component	**Advantages**	**Drawbacks**	**Algos**
Gradient-based	• Easy to implement • Usable for meta-learning and meta-RL • Many open-source repos available	• Weak generalization and OOD performance • Only PG methods for meta-RL	MAML, FO-MAML
Context-based	• Dynamically adjust to tasks without model training • Benefitial in sparse-reward tasks • Implicit goal of base Bayes-optimal behaviour	• Highly dependent on the quality of context	CAVIA, DREAM, MAESN, Meta-Cure, PEARL, RL^2
RNN-based	• Sequence processing, context-based by design • Simple memory-based architecture	• Poor asymptotic performance • Sensitive to architecture, context length, and RL algorithm choice	RL^2, VariBAD
Bayesian Inference	• Explicit uncertainty quantification • Tailored for Bayes-optimal behaviour	• Complex implementation • Often exhibit unstable learning	MAESN, Meta-Cure, PEARL, VariBAD, MELTS
Model-based RL on Task-Level	• Decisions can be based on the dynamics model's predictions; better long-horizon planing	• Higher architectural and computational complexity	VariBAD, ADA
Transformer-based	• Powerful contextualization • Meta-learner per design • Very high sample efficiency and adaptation speed • Particularly high OOD performance	• Require vast amounts of data and computational resources for (meta-)training • Perform worse on tasks with high task uncertainty	TrMRL, ADA, AMAGO, GATO, ECET
ACL	• Improved meta-training efficiency	• May be costly to implement	MELTS, ADA
Distillation	• Reduce model size • Stabilize initial meta-training	• Potential loss of performance	HyperX, ADA
Generalist Agents	• Can handle a wide variety of tasks	• High computation cost even for rollouts	ADA, AMAGO, GATO, ECET