

---

# Meta-Learning by Adjusting Priors Based on Extended PAC-Bayes Theory

---

Ron Amit<sup>1</sup> Ron Meir<sup>1</sup>

## Abstract

In meta-learning an agent extracts knowledge from observed tasks, aiming to facilitate learning of novel future tasks. Under the assumption that future tasks are ‘related’ to previous tasks, representations should be learned in a way which captures the common structure across learned tasks, while allowing the learner sufficient flexibility to adapt to novel aspects of new tasks. We present a framework for meta-learning that is based on generalization error bounds, allowing us to extend various PAC-Bayes bounds to meta-learning. Learning takes place through the construction of a distribution over hypotheses based on the observed tasks, and its utilization for learning a new task. Thus, prior knowledge is incorporated through setting an experience-dependent prior for novel tasks. We develop a gradient-based algorithm which minimizes an objective function derived from the bounds and demonstrate its effectiveness numerically with deep neural networks. In addition to establishing the improved performance available through meta-learning, we demonstrate the intuitive way by which prior information is manifested at different levels of the network.

## 1. Introduction

Learning from examples is the process of inferring a general rule from a finite set of examples. It is well known in statistics (e.g., Devroye et al. (1996)) that learning cannot take place without prior assumptions. Recent work in deep neural networks has achieved significant success in using prior knowledge in the implementation of structural constraints, e.g., convolutions and weight sharing (LeCun

et al., 2015). However, often the relevant prior information for a given task is not clear, and there is a need for building it through learning from previous interactions with the world. Learning from previous experience can take several forms: **Continual Learning** (Kirkpatrick et al., 2017) - a learning agent is trained on a sequence of tasks, aiming to solve the current task while maintaining good performance on previous tasks. **Multi-Task Learning** (Caruana, 1997) - a learning agent learns how to solve several observed tasks, while exploiting their shared structure. **Domain Adaptation** (Ben-David et al., 2010) - a learning agent solves a ‘target’ learning task using ‘source’ tasks (both are observed, but usually the target is predominantly unlabeled).

We work within the framework of **Meta-Learning / Learning-to-Learn / Inductive Transfer** (Thrun & Pratt, 1997; Vilalta & Drissi, 2002)<sup>1</sup> in which a ‘meta-learner’ extracts knowledge from several observed tasks to facilitate the learning of *new tasks* by a ‘base-learner’ (see Figure 1). In this setup the meta-learner must generalize from a finite set of observed tasks. The performance is evaluated when learning related new tasks (which are unavailable to the meta-learner).

As a motivational example, consider the case in which a meta-learner observes many image classification tasks of natural images, and uses a CNN to learn each task. The meta-learner might learn a prior which fixes the lower layers of the network to extract generic image features, but allows variation in the higher layers to adapt to new classes. Thus, new tasks can be learned using fewer examples than learning from scratch (e.g., Yosinski et al. (2014)). Generally, other scenarios might instead benefit from sharing other parts of the network (e.g., Yin & Pan (2017)). In our framework the prior is automatically inferred from the observed tasks, rather than being manually inserted by the algorithm designer.

The notion of ‘task-environment’ was formulated by Baxter (2000). In analogy to the standard single-task learning where data is sampled from an unknown distribution, Baxter

---

<sup>1</sup>The Viterbi Faculty of Electrical Engineering, Technion - Israel Institute of Technology, Haifa, Israel. Correspondence to: Ron Amit <ronamit@campus.technion.ac.il>, Ron Meir <rmeir@ee.technion.ac.il>.

---

<sup>1</sup>In our setting all observed tasks are available simultaneously to the meta-learner. The setting in which task are observed sequentially is often termed as **Lifelong Learning** (Thrun, 1996; Alquier et al., 2017)

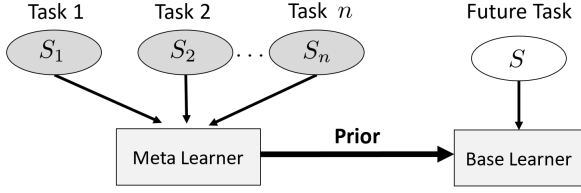


Figure 1. The meta-learner uses the data sets of the observed tasks  $S_1, \dots, S_n$  to infer ‘prior knowledge’ which in turn can facilitate learning in future tasks from the task environment (which are unobserved by the meta-learner).

suggested a setting where tasks are sampled from an unknown task distribution (environment), so that knowledge acquired from previous tasks can be used in order to improve performance on a novel task. **Baxter’s work not only provided an interesting and mathematically precise perspective for meta-learning, but also provided generalization bounds demonstrating the potential improvement in performance due to prior knowledge.**

In this paper we work within the framework formulated by Baxter (2000), and, following the setup in Pentina & Lampert (2014), provide generalization error bounds within the PAC-Bayes framework. These bounds are then used to develop a practical learning algorithm that is applied to neural networks, demonstrating the utility of our approach. The main contributions of this work are the following. (i) An **improved and tighter bound** in the theoretical framework of Pentina & Lampert (2014) derived using a technique which can extend different single-task PAC-Bayes bounds to the meta-learning setup. (ii) A principled meta-learning method and its implementation using probabilistic feedforward neural networks. (iii) Empirical demonstration of the performance enhancement compared to naive approaches as well as recent methods in this field.

**Related Work** While there have been many recent developments in meta-learning (e.g., Edwards & Storkey (2016); Andrychowicz et al. (2016); Finn et al. (2017)), most of them were not based on generalization error bounds, which is the focus of the present work. An elegant extension of generalization error bounds to meta-learning was provided by Pentina & Lampert (2014), mentioned above (extended in Pentina & Lampert (2015)). Their work, however, did not provide a practical algorithm applicable to deep neural networks. More recently, Dziugaite & Roy (2017) developed a single-task algorithm based on PAC-Bayes bounds that was demonstrated to yield good performance in simple classification tasks with deep networks. Other recent theoretical approaches to meta or multitask learning (e.g. Maurer (2005; 2009); Ruvolet & Eaton (2013); Maurer et al. (2016); Alquier et al. (2017)) provide increasingly general bounds but have not led to practical algorithms for neural networks.

## 2. Preliminaries: PAC-Bayes Learning

In the common setting for learning, a set of independent samples,  $S = \{z_i\}_{i=1}^m$ , from a space of examples  $\mathcal{Z}$ , is given, each sample drawn from an *unknown* probability distribution  $\mathcal{D}$ , namely  $z_i \sim \mathcal{D}$ . We will use the notation  $S \sim \mathcal{D}^m$  to denote the distribution over the full sample. In supervised learning, the samples are input/output pairs  $z_i = (x_i, y_i)$ . The usual learning goal is, based on  $S$ , to find a function  $h \in \mathcal{H}$ , where  $\mathcal{H}$  is the so-called hypothesis space, that minimizes the expected loss function  $\mathbb{E}\ell(h, z)$ , where  $\ell(h, z)$  is a *loss function* bounded in  $[0, 1]$ . As the distribution  $\mathcal{D}$  is unknown, learning consists of selecting an appropriate  $h$  based on the sample  $S$ . In classification  $\mathcal{H}$  is a space of classifiers mapping the input space to a finite set of classes. As noted in the Introduction, an inductive bias is required for effective learning. While in the standard approach to learning, described in the previous paragraph, one usually selects a single classifier (e.g., the one minimizing the empirical error), the PAC-Bayes framework, first formulated by McAllester (1999), considers the construction of a complete probability distribution over  $\mathcal{H}$ , and the selection of a single hypothesis  $h \in \mathcal{H}$  based on this distribution. Since this distribution depends on the data it is referred to as a *posterior distribution* and will be denoted by  $Q$ . We note that while the term ‘posterior’ has a Bayesian connotation, the framework is not necessarily Bayesian, and the posterior does not need to be related to the prior through the likelihood function as in standard Bayesian analysis. The PAC-Bayes framework has been widely studied in recent years, and has given rise to significant flexibility in learning, and, more importantly, to some of the best generalization bounds available Audibert (2010); McAllester (2013); Lever et al. (2013). The framework has been recently extended to the meta-learning setting by Pentina & Lampert (2014), and will be extended and applied to neural networks in the present contribution.

### 2.1. Single-task Problem Formulation

Following the notation introduced above we define the generalization error  $er(h, \mathcal{D}) \triangleq \mathbb{E}_{z \sim \mathcal{D}} \ell(h, z)$  and the empirical error  $\hat{er}(h, S) \triangleq (1/m) \sum_{j=1}^m \ell(h, z_j)$  used in the single-task setting. Since the distribution  $\mathcal{D}$  is unknown,  $er(h, \mathcal{D})$  cannot be directly computed. In the PAC-Bayesian setting the learner outputs a distribution over the entire hypothesis space  $\mathcal{H}$ , i.e, the goal is to provide a *posterior* distribution  $Q \in \mathcal{M}$ , where  $\mathcal{M}$  denotes the set of distributions over  $\mathcal{H}$ . The expected (over  $\mathcal{H}$ ) *generalization error* and *empirical error* are then given in this setting by averaging over the posterior distribution  $Q \in \mathcal{M}$ , namely  $er(Q, \mathcal{D}) \triangleq \mathbb{E}_{h \sim Q} er(h, \mathcal{D})$  and  $\hat{er}(Q, S) \triangleq \mathbb{E}_{h \sim Q} \hat{er}(h, S)$ , respectively.

## 2.2. PAC-Bayes Generalization Bound

In this section we introduce a PAC-Bayesian bound for the single-task setting. The bound will also serve us for the meta-learning setting in the next sections. PAC-Bayesian bounds are based on specifying some ‘prior’ reference distribution  $P \in \mathcal{M}$ , that must not depend on the observed data  $S$ . The distribution over hypotheses  $Q$  which is provided as an output from the learning process is called the posterior (since it is allowed to depend on  $S$ )<sup>2</sup>. The classical PAC-Bayes theorem for single-task learning was formulated by McAllester (1999).

**Theorem 1 (McAllester’s single-task bound)** *Let  $P \in \mathcal{M}$  be some prior distribution over  $\mathcal{H}$ . Then for any  $\delta \in (0, 1]$ , the following inequality holds uniformly for all posteriors distributions  $Q \in \mathcal{M}$  with probability at least  $1 - \delta$ ,*

$$er(Q, \mathcal{D}) \leq \hat{er}(Q, S) + \sqrt{\frac{D(Q||P) + \log \frac{m}{\delta}}{2(m-1)}}$$

where  $D(Q||P)$  is the Kullback-Leibler (KL) divergence,  $D(Q||P) \triangleq \mathbb{E}_{h \sim Q} \log \frac{Q(h)}{P(h)}$ .

Theorem 1 can be interpreted as stating that with high probability the expected error  $er(Q, \mathcal{D})$  is upper bounded by the empirical error plus a complexity term. Since, with high probability, the bound holds uniformly for all  $Q \in \mathcal{M}$ , it holds also for data dependent  $Q$ . By choosing  $Q$  that minimizes the bound we obtain a learning algorithm with generalization guarantees. Note that PAC-Bayes bounds express a trade-off between fitting the data (empirical error) and a complexity/regularization term (distance from prior) which encourages selecting a ‘simple’ hypothesis, namely one similar to the prior. The specific choice of  $P$  affects the bound’s tightness and so should express prior knowledge about the problem. Generally, we want the prior to be close to posteriors which can achieve low training error.

## 3. PAC-Bayes Meta-Learning

In this section we introduce the meta-learning setting. In this setting a meta-learning agent observes several ‘training’ tasks from the same task environment. The meta-learner must extract some common knowledge (‘learned prior’) from these tasks, which will be used for learning new tasks from the same environment. In the literature this setting is often called learning-to-learn, lifelong-learning, meta-

learning or bias learning (Baxter, 2000). We will formulate the problem and provide a generalization bound which will later lead to a practical algorithm. Our work extends Pentina & Lampert (2014) and establishes a potentially tighter bound. Furthermore, we will demonstrate how to apply this result practically to deep neural networks using stochastic learning.

### 3.1. Meta-Learning Problem Formulation

The meta-learning problem formulation follows Pentina & Lampert (2014). We assume all tasks share the sample space  $\mathcal{Z}$ , hypothesis space  $\mathcal{H}$  and loss function  $\ell : \mathcal{H} \times \mathcal{Z} \rightarrow [0, 1]$ . The learning tasks differ in the unknown sample distribution  $\mathcal{D}_t$  associated with each task  $t$ . The meta-learning agent observes the training sets  $S_1, \dots, S_n$  corresponding to  $n$  different tasks. The number of samples in task  $i$  is denoted by  $m_i$ . Each observed dataset  $S_i$  is assumed to be generated from an unknown sample distribution  $S_i \sim \mathcal{D}_i^{m_i}$ . As in Baxter (2000), we assume that the sample distributions  $\mathcal{D}_i$  are generated i.i.d. from an unknown tasks distribution  $\tau$ .

The goal of the meta-learner is to extract some knowledge from the observed tasks that will be used as prior knowledge for learning new (yet unobserved) tasks from  $\tau$ . The prior knowledge comes in the form of a distribution over hypotheses,  $P \in \mathcal{M}$ . When learning a new task, the *base learner* uses the observed task’s data  $S$  and the prior  $P$  to output a posterior distribution  $Q(S, P)$  over  $\mathcal{H}$ . We assume that all tasks are learned via the same learning process. Namely, for a given  $S$  and  $P$  there is a specific output  $Q(S, P)$ . Hence the base learner  $Q$  is a mapping:  $Q : \mathcal{Z}^m \times \mathcal{M} \rightarrow \mathcal{M}$ .<sup>3</sup>

The quality of a prior  $P$  is measured by the expected loss when using it to learn new tasks, as defined by,

$$er(P, \tau) \triangleq \mathbb{E}_{(\mathcal{D}, m) \sim \tau} \mathbb{E}_{S \sim \mathcal{D}^m} \mathbb{E}_{h \sim Q(S, P)} \mathbb{E}_{z \sim \mathcal{D}} \ell(h, z). \quad (1)$$

The expectation is taken w.r.t. (i) tasks drawn from the task environment, (ii) training samples, (iii) hypotheses drawn from the posterior which is learned based on the training samples and prior 4. a ‘test’ sample.

As described in section 2, in the single-task PAC-Bayes framework, the learner assumes a prior over hypotheses  $P(h)$ , then observes the training samples and outputs a posterior distribution over hypotheses  $Q(h)$ . In an analogous way, in the meta-learning PAC-Bayes framework, the meta-learner assumes a prior distribution over priors, a ‘hyper-prior’  $\mathcal{P}(P)$ , observes the training tasks, and then outputs a distribution over priors, a ‘hyper-posterior’  $\mathcal{Q}(P)$ .

We emphasize that while the hyper-posterior is learned using the observed tasks, the goal is to use it for learning new,

<sup>2</sup>As noted above, the terms ‘prior’ and ‘posterior’ might be misleading, since, this is not a Bayesian inference setting (the prior and posterior are not connected through the Bayes rule). However, PAC-Bayes and Bayesian analysis have interesting and practical connections, as we will see in the next sections (see also Germain et al. (2016)).

<sup>3</sup>In the next section we will use stochastic optimization methods as learning algorithms, but we can assume convergence to a same solution for any execution with a given  $S$  and  $P$ .

independent task from the environment. When encountering a new task, the learner samples a prior from the hyper-posterior  $Q(P)$ , and then use it for learning. Ideally, the performance of the hyper-posterior  $Q$  is measured by the expected generalization loss of learning new tasks using priors generated from  $Q$ . This quantity is denoted as the *transfer error*

$$er(Q, \tau) \triangleq \mathbb{E}_{P \sim Q} er(P, \tau). \quad (2)$$

While  $er(Q, \tau)$  is not computable, we can however evaluate the *empirical multi-task error*

$$\hat{er}(Q, S_1, \dots, S_n) \triangleq \mathbb{E}_{P \sim Q} \frac{1}{n} \sum_{i=1}^n \hat{er}(Q(S_i, P), S_i), \quad (3)$$

and use it to prove a PAC-Bayes style upper bound on the transfer error, that can be minimized over  $Q$ .

In the single-task PAC-Bayes setting one selects a prior  $P \in \mathcal{M}$  before seeing the data, and updates it to a posterior  $Q \in \mathcal{M}$  after observing the training data. In the present meta-learning setup, following Pentina & Lampert (2014), one selects an initial hyper-prior distribution  $\mathcal{P}$ , essentially a distribution over prior distributions  $P$ , and, following the observation of the data from all tasks, updates it to a hyper-posterior distribution  $Q$ . As a simple example, assume the initial prior  $P$  is a Gaussian distribution over neural network weights, characterized by a mean and covariance. A hyper distribution would correspond in this case to a distribution over the mean and covariance of  $P$ .

### 3.2. Meta-Learning PAC-Bayes Bound

In this section we present a novel bound on the transfer error in the meta-learning setup. The theorem is proved in section A.1 of the supplementary material.

**Theorem 2 (Meta-learning PAC-Bayes bound)** *Let  $Q : \mathcal{Z}^m \times \mathcal{M} \rightarrow \mathcal{M}$  be a base learner, and let  $\mathcal{P}$  be some pre-defined hyper-prior distribution. Then for any  $\delta \in (0, 1]$  the following inequality holds uniformly for all hyper-posterior distributions  $Q$  with probability at least  $1 - \delta$ ,<sup>4</sup>*

$$er(Q, \tau) \leq \frac{1}{n} \sum_{i=1}^n \mathbb{E}_{P \sim Q} \hat{er}_i(Q_i, S_i) + \frac{1}{n} \sum_{i=1}^n \sqrt{\frac{D(Q||\mathcal{P}) + \mathbb{E}_{P \sim Q} D(Q_i||P) + \log \frac{2nm_i}{\delta}}{2(m_i - 1)}} + \sqrt{\frac{D(Q||\mathcal{P}) + \log \frac{2n}{\delta}}{2(n - 1)}}, \quad (4)$$

where  $Q_i \triangleq Q(S_i, P)$ .

<sup>4</sup>The probability is taken over sampling of  $(\mathcal{D}_i, m_i) \sim \tau$  and  $S_i \sim \mathcal{D}_i^{m_i}$ ,  $i = 1, \dots, n$ .

Notice that the transfer error (2) is bounded by the empirical multi-task error (3) plus two complexity terms. The first is the average of the task-complexity terms of the observed tasks. This term converges to zero in the limit of a large number of samples in each task ( $m_i \rightarrow \infty$ ). The second is an environment-complexity term. This term converges to zero if infinite number of tasks is observed from the task environment ( $n \rightarrow \infty$ ). As in Pentina & Lampert (2014), our proof is based on two main steps. The second step, similarly to Pentina & Lampert (2014), bounds the generalization error at the task-environment level (i.e, the error caused by observing only a finite number of tasks) by the average generalization error in the observed tasks plus the environment-complexity term. The first step differs from Pentina & Lampert (2014). Instead of using a single joint bound on the average generalization error, we use a single-task PAC-Bayes theorem to bound the generalization error in each task separately (when learned using priors from the hyper-posterior), and then use a union bound argument. By doing so our bound takes into account the specific number of samples in each observed task (instead of their harmonic mean). Therefore our bound is better adjusted the observed data set.

Our proof technique can utilize different single-task bounds in each of the two steps. In section A.1 we use McAllester’s bound (Theorem 1), which is tighter than the lemma used in Pentina & Lampert (2014). Therefore, the complexity terms are in the form of  $\sqrt{\frac{1}{m} D(Q||P)}$  instead of  $\frac{1}{\sqrt{m}} D(Q||P)$  as in Pentina & Lampert (2014). This means the bound is tighter<sup>5</sup>. In section A.2 we demonstrate how our technique can use other, possibly tighter, single-task bounds. In Section 5 we will empirically evaluate the different bounds as meta-learning objectives and show that the improved tightness is critical for performance.

## 4. Meta-Learning Algorithm

As in the single-task case, the bound of Theorem 2 can be evaluated from the training data and so can serve as a minimization objective for a principled meta-learning algorithm. Since the bound holds uniformly for all  $Q$ , it is ensured to hold also for the minimizer of the objective  $Q^*$ . In this section we will derive a practical learning procedure that can be applied to a large family of differentiable models, including deep neural networks.

### 4.1. Hyper-Posterior Model

In this section we choose a specific form of hyper-posterior distribution  $Q$  which enables practical implementation. Given a parametric family of priors  $\{P_{\tilde{\theta}} : \tilde{\theta} \in \mathbb{R}^{N_P}\}$ ,

<sup>5</sup>E.g., Seldin et al. (2012) Theorems 5 and 6



$N_P \in \mathbb{N}$ , the space of hyper-posteriors consists of all distributions over  $\mathbb{R}^{N_P}$ . We will limit our search to a family of Gaussian distributions defined by  $\mathcal{Q}_\theta \triangleq \mathcal{N}(\theta, \kappa_Q^2 I_{N_P \times N_P})$ , where  $\kappa_Q > 0$  is a predefined constant. Notice that  $\mathcal{Q}$  appears in the bound (4) in two forms (i) divergence from the hyper-prior  $D(\mathcal{Q}||\mathcal{P})$  and (ii) expectations over  $P \sim \mathcal{Q}$ .

By setting the hyper-prior as Gaussian,  $\mathcal{P} = \mathcal{N}(0, \kappa_P^2 I_{N_P \times N_P})$ , where  $\kappa_P > 0$  is another constant, we get a simple form for the KL-divergence term,  $D(\mathcal{Q}_\theta||\mathcal{P}) = \frac{1}{2\kappa_P^2} \|\theta\|_2^2$ . Note that the hyper-prior acts as a regularization term which prefers solutions with small  $L_2$  norm.

The expectations can be approximated by averaging several Monte-Carlo samples of  $P$ . Notice that sampling from  $\mathcal{Q}_\theta$  means adding Gaussian noise to the parameters  $\theta$  during training,  $\tilde{\theta} = \theta + \varepsilon_P, \varepsilon_P \sim \mathcal{N}(0, \kappa_Q^2 I_{N_P \times N_P})$ . This means the learned parameters must be robust to perturbations, which encourages selecting solutions which are less prone to over-fitting.

## 4.2. Joint Optimization

The term appearing on the RHS of the meta-learning bound in (4) can be compactly written as

$$J(\theta) \triangleq \frac{1}{n} \sum_{i=1}^n J_i(\theta) + \Upsilon(\theta), \quad (5)$$

where we defined,

$$J_i(\theta) \triangleq \mathbb{E}_{\tilde{\theta} \sim \mathcal{Q}_\theta} \hat{e}r_i(Q_i(S_i, P_{\tilde{\theta}}), S_i) \quad (6)$$

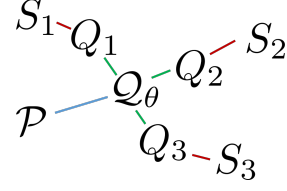
$$+ \sqrt{\frac{D(\mathcal{Q}_\theta||\mathcal{P}) + \mathbb{E}_{\tilde{\theta} \sim \mathcal{Q}_\theta} D(Q(S_i, P_{\tilde{\theta}})||P_{\tilde{\theta}}) + \log \frac{2nm_i}{\delta}}{2(m_i - 1)}},$$

$$\Upsilon(\theta) \triangleq \sqrt{\frac{D(\mathcal{Q}_\theta||\mathcal{P}) + \log \frac{2n}{\delta}}{2(n - 1)}}. \quad (7)$$

Theorem 2 allows us to choose *any* procedure  $Q(S_i, P) : \mathcal{Z}^{m_i} \times \mathcal{M} \rightarrow \mathcal{M}$  as a base learner. We will use a procedure which minimizes  $J_i(\theta)$  due to the following advantages: (i) It minimizes a bound on the generalization error of the observed task <sup>6</sup>. (ii) It uses the prior knowledge gained from the prior  $P$  to get a tighter bound and a better learning objective. (iii) As will be shown next, formulating the single task learning as an optimization problem enables joint learning of the shared prior and the task posteriors.

To formulate the single-task learning as an optimization problem, we choose a parametric form for the posterior of each task  $Q_{\phi_i}, \phi_i \in \mathbb{R}^{N_Q}$  (see section 4.3 for an explicit

<sup>6</sup>See section A.1 in the supplementary material.



**Figure 2. Joint optimization illustration.** The posterior of task  $i$ ,  $Q_i = Q_{\phi_i}$  is influenced by both the dataset  $S_i$  (through the empirical error term) and by the hyper-posterior  $\mathcal{Q}_\theta$  (through the task-complexity term). The hyper-posterior  $\mathcal{Q}_\theta$  is influenced by both the posteriors and by the hyper-prior  $\mathcal{P}$  (through the environment-complexity term).

example). The base-learning algorithm can be formulated as  $\phi_i^* = \operatorname{argmin}_{\phi_i} J_i(\theta, \phi_i)$ , where we abuse notation by denoting the term  $J_i(\theta)$  evaluated with posterior parameters  $\phi_i$  as  $J_i(\theta, \phi_i)$ .

The meta-learning problem of minimizing  $J(\theta)$  over  $\theta$  can now be written more explicitly,

$$\min_{\theta, \phi_1, \dots, \phi_n} \left\{ \frac{1}{n} \sum_{i=1}^n J_i(\theta, \phi_i) + \Upsilon(\theta) \right\}. \quad (8)$$

The optimization process is illustrated in Figure 2.

## 4.3. Distributions Model

In this section we make the meta-learning optimization problem (8) more explicit by defining a model for the posterior and prior distributions. First, we define the hypothesis class  $\mathcal{H}$  as a family of functions parameterized by a *weight vector*  $\{h_w : w \in \mathbb{R}^d\}$ . Given this parameterization, the posterior and prior are distributions over  $\mathbb{R}^d$ .

We will present an algorithm for any differentiable model <sup>7</sup>, but our aim is to use neural network (NN) architectures. In fact, we will use Stochastic NNs (Graves, 2011; Blundell et al., 2015) since in our setting the weights are random and we are optimizing their posterior distribution. The techniques presented next will be mostly based on Blundell et al. (2015). Next we define the posteriors  $Q_{\phi_i}, i = 1, \dots, n$ , and the prior  $P_\theta$  as factorized Gaussian distributions<sup>8</sup>,

$$P_\theta(w) = \prod_{k=1}^d \mathcal{N}(w_k; \mu_{P,k}, \sigma_{P,k}^2) \quad (9)$$

$$Q_{\phi_i}(w) = \prod_{k=1}^d \mathcal{N}(w_k; \mu_{i,k}, \sigma_{i,k}^2) \quad (10)$$

<sup>7</sup>The only assumption on  $\{h_w : w \in \mathbb{R}^d\}$  is that the loss function  $\ell(h_w, z)$  is differentiable w.r.t  $w$ .

<sup>8</sup>This choice makes optimization easier, but in principle we can use other distributions as long as the density function is differentiable w.r.t. the parameters.

where for each task, the posterior parameters vector  $\phi_i = (\mu_i, \rho_i) \in \mathbb{R}^{2d}$  is composed of the means and log-variances of each weight,  $\mu_{i,k}$  and  $\rho_{i,k} = \log \sigma_{P,k}^2$ ,  $k = 1, \dots, d$ .<sup>9</sup> The shared prior vector  $\theta = (\mu_P, \rho_P) \in \mathbb{R}^{2d}$  has a similar structure. Since we aim to use deep models where  $d$  could be in the order of millions, distributions with more parameters might be impractical.

Since  $Q_{\phi_i}$  and  $P_{\theta}$  are factorized Gaussian distributions the KL-divergence,  $D(Q_{\phi_i} \| P_{\theta})$ , takes a simple analytic form,

$$\frac{1}{2} \sum_{k=1}^d \left\{ \log \frac{\sigma_{P,k}^2}{\sigma_{i,k}^2} + \frac{\sigma_{i,k}^2 + (\mu_{i,k} - \mu_{P,k})^2}{\sigma_{P,k}^2} - 1 \right\}. \quad (11)$$

#### 4.4. Optimization Technique

As an underlying optimization method, we will use stochastic gradient descent (SGD). In each iteration, the algorithm takes a parameter step in a direction of an estimated negative gradient. As is well known, lower variance facilitates convergence and its speed. Recall that each single-task bound is composed of an empirical error term and a complexity term (6). The complexity term is a simple function of  $D(Q_{\phi_i} \| P_{\theta})$  (11), which can easily be differentiated analytically. However, evaluating the gradient of the empirical error term is more challenging.

Recall the definition of the empirical error,  $\hat{e}_r(Q_{\phi_i}, S_i) = \mathbb{E}_{w \sim Q_{\phi_i}} (1/m_i) \sum_{j=1}^{m_i} \ell(h_w, z_{i,j})$ . This term poses two major challenges. (i) The data set  $S_i$  could be very large making it expensive to cycle over all the  $m_i$  samples. (ii) The term  $\ell(h_w, z_j)$  might be highly non-linear in  $w$ , rendering the expectation intractable. Still, we can get an unbiased and low variance estimate of the gradient.

First, instead of using all of the data for each gradient estimation we will use a randomly sampled mini-batch  $S'_i \subset S_i$ . Next, we require an estimate of a gradient of the form  $\nabla_{\phi} \mathbb{E}_{w \sim Q_{\phi}} f(w)$  which is a common problem in machine learning. We will use the ‘re-parametrization trick’ (Kingma & Welling, 2013; Rezende et al., 2014) which is an efficient and low variance method<sup>10</sup>. The re-parametrization trick is easily applicable in our setup since we are using Gaussian distributions. The trick is based on describing the Gaussian distribution  $w \sim Q_{\phi_i}$  (9) as first drawing  $\varepsilon \sim \mathcal{N}(\bar{0}, I_{d \times d})$  and then applying the deterministic function  $w(\phi_i, \varepsilon) = \mu_i + \sigma_i \odot \varepsilon$  (where

<sup>9</sup>Note that we use  $\rho = \log \sigma^2$  as a parameter in order to keep the parameters unconstrained (while  $\sigma^2 = \exp(\rho)$  is guaranteed to be strictly positive).

<sup>10</sup>In fact, we will use the ‘local re-parameterization trick’ (Kingma et al., 2015) in which we sample a different  $\varepsilon$  for each data point in the batch, which reduces the variance of the estimate. To make the computation more efficient with neural-networks, the random number generation is performed w.r.t the activations instead of the weights (see Kingma et al. (2015) for more details.).

$\odot$  is an element-wise multiplication). Therefore, we get  $\nabla_{\phi} \mathbb{E}_{w \sim Q_{\phi}} f(w) = \nabla_{\phi} \mathbb{E}_{\varepsilon \sim \mathcal{N}(\bar{0}, I_{d \times d})} f(w(\phi_i, \varepsilon))$ .

The expectation can be approximated by averaging a small number of Monte-Carlo samples with reasonable accuracy. For a fixed sampled  $\varepsilon$ , the gradient  $\nabla_{\phi} f(w(\phi_i, \varepsilon))$  is easily computable with backpropagation.

In summary, the Meta-Learning by Adjusting Priors (MLAP) algorithm is composed of two phases. In the first phase (Algorithm 1, termed ‘meta-training’) several observed ‘training tasks’ are used to learn a prior. In the second phase (Algorithm 2, termed ‘meta-testing’) the previously learned prior is used for the learning of a new task (which was unobserved in the first phase). Note that the first phase can be used independently as a multi-task learning method. Both algorithms are described in pseudo-code in the supplementary material (section A.4)<sup>11 12</sup>.

## 5. Experimental Demonstration

In this section we demonstrate the performance of our transfer method with image classification tasks solved by deep neural networks. In image classification, the data samples,  $z \triangleq (x, y)$ , consist of an image,  $x$ , and a label,  $y$ . The hypothesis class  $\{h_w : w \in \mathbb{R}^d\}$  is the set of neural networks with a given architecture (which will be specified later). As a loss function  $\ell(h_w, z)$  we will use the cross-entropy loss.

While the theoretical framework is defined with a bounded loss, in our experiments we use an unbounded loss function in the learning objective. Still, we can have theoretical guarantees on a variation of the loss which is clipped to  $[0, 1]$ . Furthermore, in practice the loss function is almost always smaller than one.

We conduct two experiments with two different task environments, based on augmentations of the MNIST dataset (LeCun, 1998). In the first environment, termed *permuted labels*, each task is created by a random permutation of the labels. In the second environment, termed *permuted pixels*, each task is created by a permutation of the image pixels. The pixel permutations are created by a limited number of location swaps to ensure that the tasks stay reasonably related.

In both experiments, the meta-training set is composed of tasks from the environment with 60,000 training examples. Following the meta-training phase, the learned prior is used to learn a new meta-test task with fewer training samples (2,000). The network architecture used for the permuted-labels experiment is a small CNN with 2 convolutional-

<sup>11</sup>Code is available at: <https://github.com/ron-amit/meta-learning-adjusting-priors>.

<sup>12</sup>For a visual illustration of the algorithm using a toy example see section A.6 in the supplementary material.

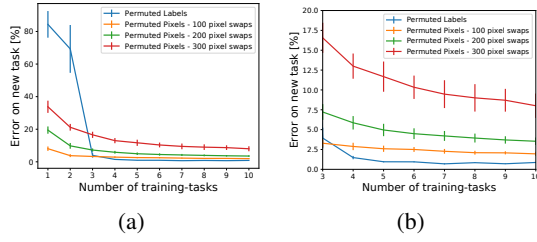


Figure 3. The average test error of learning a new task for different numbers of training-tasks and for different environments (average over 20 meta-test tasks). Figure (b) reproduces (a) starting with 3 tasks. Best viewed in color.

layers, a linear hidden layer and a linear output layer. In the permuted-pixels experiment we used a fully-connected network with 3 hidden layers and a linear output layer. See section A.5 for more implementation details.

We tested the generalization performance of learning a new ‘meta-test’ task when using the MLAP algorithm to transfer a prior from several ‘meta-training’ tasks in different task-environments, namely the permuted labels environment, and the permuted pixels environment with 100, 200, 300 pixel swaps. In Figure 3 we plot the average test error of learning a new task based on the number of training-tasks in the different environments. The results clearly show that the more tasks are used to learn the prior, the better the performance on the new task. For example, in the permuted labels case, a prior that is learned based on one or two tasks leads to negative transfer, i.e. worse results than standard learning from scratch (with no transfer), which achieves 2.27% error. However after observing 3 or more tasks, the transferred prior facilitates learning with lower generalization error. In the permuted pixels experiment, standard learning from scratch achieves 7.9% test error. The number of training tasks needed for positive transfer depends on the number of pixels swapped. A higher number of swaps means larger variation in the task environment and more training-tasks are needed to learn a beneficial prior.

Next we compare the average generalization performance in learning a new task when using the following methods. As a baseline, we measure the performance of learning **without transfer** from the meta-training tasks. **Scratch-D**: deterministic (standard) learning from scratch. **Scratch-S**: stochastic learning from scratch (using a stochastic network with no prior/complexity term).

Other methods transfer knowledge from only **one** of the meta-train tasks. **Warm-start**: Standard learning with initial weights taken from the standard learning of a single task from the meta-train set. **Oracle**: Same as the previous method, but some of the layers are frozen (unchanged from their initial value) depending on the experiment. In the per-

mutated labels experiment all layers besides the output are frozen. In the permuted pixels we freeze all layers except the input layer. We refer to this method as ‘oracle’ since the transfer technique is tailored to each task-environment, while the other methods are applied identically in any environment (and so must learn to adjust to the environment automatically).

Finally, we compare methods which transfer knowledge from **all** of the training tasks: **MLAP-M**: The objective is based on Theorem 2 - the meta-learning bound obtained using Theorem 1 (McAllester’s single-task bound). **MLAP-S**: The objective is based on the meta-learning bound derived from Seeger’s single-task bound (see section A.2 in the supplementary material, eq.(18)). **MLAP-PL**: In this method we use the main theorem of Pentina & Lampert (2014) as an objective for the algorithm, instead of Theorem 2. **MLAP-VB**: In this method the learning objective is derived from a Hierarchical Bayesian framework using variational Bayes tools<sup>13</sup>. **Averaged**: Each of the training tasks is learned in a standard way to obtain a weights vector,  $w_i$ . The learned prior is set as an isotropic Gaussian with unit variances and a mean vector which is the average of  $w_i, i = 1, \dots, n$ . This prior is used for meta-testing as in MLAP-S. **MAML**: The Model-Agnostic-Meta-Learning (MAML) algorithm by Finn et al. (2017). In MAML the base learner takes few gradient steps from an initial point,  $\theta$ , to adapt to a task. The meta-learner optimizes  $\theta$  based on the sum of losses on the observed tasks after base-learning. We tested several hyper-parameters and report the best results (see details in the supplementary material A.5).

Table 1 summarizes the results for the permuted labels experiment with 5 training-tasks and the permuted pixels experiment with 200 pixel swaps and 10 training-tasks.

In the permuted labels experiment the best results are obtained with the ‘oracle’ method. Recall that the oracle method has the ‘unfair’ advantage of a ‘hand-engineered’ transfer technique which is based on knowledge about the problem. In contrast, the other methods must automatically learn the task environment by observing several tasks.

The MLAP-M and MLAP-S variants of the MLAP algorithm improves considerably over learning from scratch and over the naive warm-start transfer. They even improve over the ‘oracle’ method in the permuted pixels experiment. The result of the MLAP-VB are close to the MLAP-M and MLAP-S variants. However the MLAP-PL variant performed much worse since the complexity terms are disproportionately large compared to the empirical error terms. This demonstrates the importance of using the tight generalization bound developed in our work as a learning objective.

<sup>13</sup>See section A.3 in the supplementary material for details. The explicit learning objective is in equation (23).

Table 1. Comparing the test error percentage of different learning methods on 20 test tasks (the  $\pm$  shows the 95% confidence interval) in the permuted labels and permuted pixels experiments (200 swaps).

METHOD	PERMUTED LABELS	PERMUTED PIXELS
SCRATCH-S	$2.27 \pm 0.06$	$7.92 \pm 0.22$
SCRATCH-D	$2.82 \pm 0.06$	$7.65 \pm 0.22$
WARM-START	$1.07 \pm 0.03$	$7.95 \pm 0.39$
ORACLE	$0.69 \pm 0.04$	$6.57 \pm 0.32$
MLAP-M	$0.908 \pm 0.04$	$3.4 \pm 0.18$
MLAP-S	$0.75 \pm 0.03$	$3.54 \pm 0.2$
MLAP-PL	$82.8 \pm 5.26$	$74.9 \pm 4.03$
MLAP-VB	$0.85 \pm 0.03$	$3.52 \pm 0.17$
AVERAGED	$2.72 \pm 0.08$	$7.63 \pm 0.36$
MAML	$1.16 \pm 0.07$	$3.77 \pm 0.8$

The results for the “averaged-prior” method are about the same as learning from scratch. Due to the high non-linearity of the problem, averaging weights was not expected to perform well.

The results of the MLAP algorithm are slightly better than MAML. Note that in MAML the meta-learning only infers an initial point for base-learning. Thus there is a trade-off in choosing the number of adaptation steps. Taking many gradient steps exploits a larger number of samples but the effect of the initial weights diminishes. Also, taking a large number of steps is computationally infeasible in meta-training. Therefore MAML is especially suited for few-shot learning, which is not the case in our experiment. In our method we infer a prior that serves both as an initial point and as a regularizer which can fix some of the weights, while allowing variation in others, depending on the amount of data. Recent work by Grant et al. (2018) showed that MAML can be interpreted as an approximate empirical Bayes procedure. This interesting perspective, differs from the present contribution that is based on generalization bounds within a non-Bayesian setting.

**Analysis of learned prior** Qualitative examination of the learned prior (Figure 4) affirms that it has indeed adjusted to each task environment. In the permuted-labels experiment the learned prior assigns low variance to the lower layers (fixed representation) and high variance to the output layer (which enable easy adjustment to different label permutations). As expected, in the permuted-pixels experiment the opposite phenomenon occurs. The mapping from the final hidden layer to the output becomes fixed, and the mapping from the input to the final hidden layer (representation) has more flexibility to change in light of the task data.

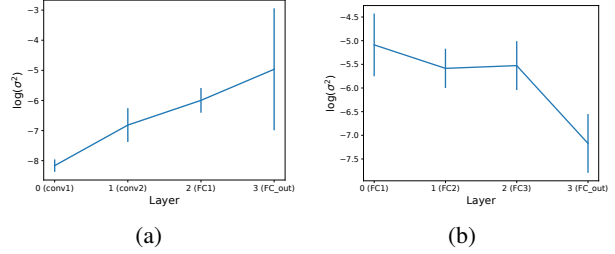


Figure 4. Log weight uncertainty ( $\log(\sigma^2)$ ) in each layer of the learned prior (average  $\pm$  STD). Higher value means higher variance/uncertainty. (a) Permuted labels experiment, (b) Permuted pixels experiment (200 swaps).

## 6. Discussion and Future Work

We have presented a framework for representational meta-learning, motivated by extended PAC-Bayes generalization bounds, and implemented through the adjustment of a learned prior, based on tasks encountered so far. The framework bears conceptual similarity to the empirical Bayes method while not being Bayesian, and is implemented at the level of tasks rather than samples (see Section A.3 in the supplementary material for details about a Bayesian perspective). Combining the flexibility of the approach, with the rich representational structure of deep neural networks, and learning through gradient based methods leads to an efficient procedure for meta-learning, as motivated theoretically and demonstrated empirically. While our experimental results are preliminary, we believe that our work attests to the utility of using rigorous performance bounds to derive learning algorithms, and demonstrates that tighter bounds indeed lead to improved performance.

There are several open issues to consider. First, the current version learns to solve all available tasks in parallel, while a more useful procedure should be sequential in nature. This can be easily incorporated into our framework by updating the prior following each novel task. Second, our method requires training stochastic models which is challenging due to the high-variance gradients. We would like to develop new methods within our framework which have more stable convergence and are easier to apply in larger scale problems. Third, there is much current effort in applying meta-learning ideas to reinforcement learning, for example, Teh et al. (2017) presents a heuristically motivated framework that is conceptually similar to ours. An interesting challenge would be to extend our techniques to derive meta-learning algorithms for reinforcement learning based on performance bounds.



## ACKNOWLEDGMENTS

We thank Asaf Cassel, Guy Tennenholtz, Baruch Epstein, Daniel Soudry, Elad Hoffer and Tom Zahavy for helpful discussions of this work, and the anonymous reviewers for their helpful comment. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. The work was partially supported by the Ollendorff Center of the Viterbi Faculty of Electrical Engineering at the Technion.

## References

- Alquier, P., Mai, T. T., and Pontil, M. Regret Bounds for Lifelong Learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 261–269, 2017.
- Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas, N. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 3981–3989, 2016.
- Audibert, J.-Y. *PAC-Bayesian aggregation and multi-armed bandits*. PhD thesis, Université Paris-Est, 2010.
- Baxter, J. A model of inductive bias learning. *J. Artif. Intell. Res. (JAIR)*, 12(149-198):3, 2000.
- Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.
- Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural network. In *International Conference on Machine Learning (ICML)*, pp. 1613–1622, 2015.
- Caruana, R. Multitask learning. *Machine Learning*, 28(1): 41–75, 1997.
- Devroye, L., Gyoörfi, L., and Lugosi, G. *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.
- Dziugaite, G. K. and Roy, D. M. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.
- Edwards, H. and Storkey, A. Towards a neural statistician. In *International Conference on Learning Representations (ICLR)*, 2016.
- Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, pp. 1126–1135, 2017.
- Germain, P., Bach, F., Lacoste, A., and Lacoste-Julien, S. PAC-Bayesian theory meets Bayesian inference. In *Advances In Neural Information Processing Systems (NIPS)*, pp. 1876–1884, 2016.
- Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T. Recasting gradient-based meta-learning as hierarchical Bayes. In *International Conference on Learning Representations (ICLR)*, 2018.
- Graves, A. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2348–2356, 2011.
- Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2013.
- Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2575–2583, 2015.
- Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, pp. 201611835, 2017.
- LeCun, Y. The mnist database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>, 1998.
- LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553):436, 2015.
- Lever, G., Laviolette, F., and Shawe-Taylor, J. Tighter PAC-Bayes bounds through distribution-dependent priors. *Theoretical Computer Science*, 473:4–28, 2013.
- Maurer, A. Algorithmic stability and meta-learning. *Journal of Machine Learning Research (JMLR)*, 6:967–994, 2005.
- Maurer, A. Transfer bounds for linear feature learning. *Machine learning*, 75(3):327–350, 2009.
- Maurer, A., Pontil, M., and Romera-Paredes, B. The benefit of multitask representation learning. *Journal of Machine Learning Research (JMLR)*, 17(81):1–32, 2016.
- McAllester, D. A PAC-Bayesian tutorial with a dropout bound. *arXiv preprint arXiv:1307.2118*, 2013.

- McAllester, D. A. PAC-Bayesian model averaging. In *Conference on Computational Learning Theory (COLT)*, pp. 164–170, 1999.
- Pentina, A. and Lampert, C. H. A PAC-Bayesian bound for lifelong learning. In *International Conference on Machine Learning (ICML)*, pp. 991–999, 2014.
- Pentina, A. and Lampert, C. H. Lifelong learning with non-iid tasks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1540–1548, 2015.
- Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, pp. 1278–1286, 2014.
- Ruvolo, P. and Eaton, E. ELLA: An efficient lifelong learning algorithm. In *International Conference on Machine Learning (ICML)*, pp. 507–515, 2013.
- Seldin, Y., Laviolette, F., Cesa-Bianchi, N., Shawe-Taylor, J., and Auer, P. PAC-Bayesian inequalities for martingales. *IEEE Transactions on Information Theory*, 58(12): 7086–7093, 2012.
- Teh, Y., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. Distal: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 4499–4509, 2017.
- Thrun, S. Is learning the n-th thing any easier than learning the first? In *Advances in neural information processing systems (NIPS)*, pp. 640–646, 1996.
- Thrun, S. and Pratt, L. *Learning To Learn*. Kluwer Academic Publishers, November 1997.
- Vilalta, R. and Drissi, Y. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2): 77–95, 2002.
- Yin, H. and Pan, S. J. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *AAAI Conference on Artificial Intelligence*, pp. 1640–1646, 2017.
- Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *Advances in neural information processing systems (NIPS)*, pp. 3320–3328, 2014.