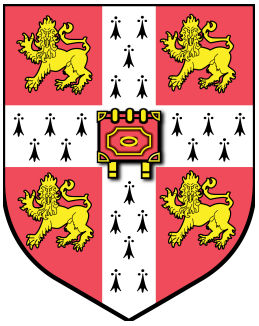
Meta-Learning: A PAC-Bayesian Perspective

Klemens Niklas Lothar Dieter Flöge



Department of Applied Mathematics
and Theoretical Physics
University of Cambridge

This Part III Essay is submitted for the degree of
Master of Advanced Study

May 4, 2023

Abstract

Meta-learning, also referred to as "learning to learn," involves leveraging past learning experiences to enhance the efficiency and efficacy of future learning. This approach is crucial as it enables machines to generalise and adapt better to tasks and transfer knowledge from related ones. Currently, the field is predominantly empirical, with researchers frequently proposing and comparing new algorithms and techniques on benchmark datasets. However, there is a lack of a comprehensive framework or theory that underpins all of these methods. Introducing PAC-Bayesian bounds for meta-learners results in a new class of algorithms known as PACOH that inhibit probabilistic performance guarantees. These algorithms not only demonstrate state-of-the-art performance on benchmark datasets but also enable the derivation of optimisation-based meta-learners such as MAML and REPTILE. PAC-Bayesian Meta-Learning is a potent framework that not only facilitates the development of new approaches but also places existing algorithms in context and within a theoretical framework. Additionally, the essay will demonstrate the performance and characteristics of the presented algorithms through numerical experiments.

Declaration

I declare that this essay is work done as part of the Part III Examination. I have read and understood both the University's statement on the Definition of Academic Misconduct and the Faculty Guidelines on Plagiarism and Academic Misconduct and have abided by them. This essay is the result of my own work, and except where explicitly stated otherwise, only includes material undertaken since the publication of the list of essay titles, and includes nothing which was performed in collaboration. No part of this essay has been submitted, or is concurrently being submitted, for any degree, diploma or similar qualification at any university or similar institution.

Part III Essay
Klemens Flöge
May 4, 2023

Summary

1 Introduction

Meta-learning is a modern field of machine learning and is concerned with teaching machines 'learning to learn'. Modern Deep Learning techniques perform very well on most machine learning problems if provided with huge amount of training data. Unfortunately, in most applications of interest training data is not abundant but rather quite limited. As will be shown with the example of few-shot-learning, meta-learning attempts to leverage information from data originating in different but related tasks to learn the general structure of the problem and then efficiently adapt to the new task with relatively few data samples.

Meta-learning is a behaviour which can be observed in humans. Most people including children will be able to recognise a person in a given picture after having seen them only once or twice before on a photo. This so called few-shot-learn is quite difficult for Neural Networks which usually require hundreds to thousands of examples of a given class in order to configure the massive parameter space. The intuition for meta-learning then comes from the idea that humans are only able to recognise a given person after seeing one or two images because they had years of experience in recognising faces in general. So adapting to a particular face is less of a challenge and only needs a few examples.

A good meta-learning algorithm should perform well on a variety of different but related tasks, such as recognising a given person in an image. This variety could be modelled as a distribution of tasks $\mathcal{T}$ called the environment, where during training a task $\tau_i = (\mathcal{D}_i, m_i) \sim \mathcal{T}$ is drawn. The learner is then presented with a dataset $S_i \sim \mathcal{D}_i^{m_i}$ of m_i samples drawn from the task distribution $\mathcal{D}$. Given a loss function $\mathcal{L}$ parameterised by θ the optimal model parameters are:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{\tau \sim \mathcal{T}}[\mathcal{L}_\theta(S)]. \tag{1}$$

This is very similar to standard supervised learning however one dataset is considered as one data sample. A dataset S contains pairs of feature vector and labels, $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and each label belongs to known label set $\mathcal{L}^{\text{label}}$. An example of a meta-learning problem as mentioned before is few-shot learning for regression or classification, which shall be the main focus of this essay. *K-shot N-class classification* consists of classifying N different classes based on K labelled examples from each class. An instance of this problem is depicted in the following Figure 1.

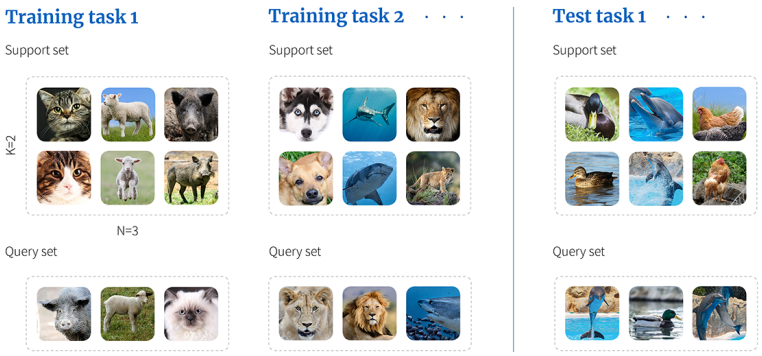


Figure 1: Example of $K = 2$ shot $N = 3$ class animal image classification problem, Zi et al. [43]

A task drawn from the environment consists of distinguishing between three different classes based on two labelled examples from each class. The classes presented in Figure 1 for training task 1 are cats, sheep and pig and in training task 2 dogs, sharks and lions. While the task 1 test set consists of ducks, foxes and dolphins. Meta learning algorithms should not learn how to distinguish all of these classes in general, but rather learn on how to distinguish three different classes of animals based on two examples of each class.

The field of meta-learning is still a largely empirical research field with a myriad of different architectures and approaches. In this essay the field will be examined through a PAC-Bayesian lens, which gives probabilistic guarantees on the performance of certain algorithms. The main application to keep in mind throughout this essay will be supervised few-shot-learning as introduced earlier. PAC-Bayesian meta learning is a very elegant theory in which certain performance guarantees for meta-learners can be proven. In Section 1 this essay will present different approaches to meta-learning. A focus will be placed on optimisation-based algorithms as these have gained much attention in recent years due to universality in applications, performance and ease of implementation. The main algorithms discussed from this class are MAML and REPTILE to a lesser extent. In Section 2 general supervised PAC-bounds will be stated. These are very useful as they provide probabilistic guarantees on the performance of a learning algorithm. This framework will then be extended to meta-learning following Rothfuss et al. [40]. Here theorems about the performance on general PAC-Bayesian Meta-Learners will be stated and proven. It will also be shown how the resulting bounds can be minimised to obtain optimal performance. Moreover, through the developed techniques it will be possible to derive modern optimisation-based algorithms such as MAML and REPTILE which demonstrates the universality of the PAC-Bayesian meta-learning framework. In Section 3 we will show how the theory developed in Section 2 can be used to built a class of state of the art algorithms called PACOH, which outperform most other meta-learning algorithms in certain tasks. In Section 4 some example applications and characteristics of the PACOH are shown.

1.1 Common Concerns and Approaches to Meta-Learning

There are a few common approaches to meta-learning using Deep Neural Networks which are model-based, metric-based and optimisation-based. In normal supervised learning we seek a model f_θ which for an input x outputs $y = f_\theta(x)$, a label for example. The model-based approach is to encode the task information in the internal states of the network. As such mainly Recurrent-Neural-Networks (RNNs) and variants are used in order to memorise previous tasks in this technique. In metric-based meta-learning the learner tries to approximate the posterior distribution through a kernel density estimation with the training dataset. So the learner attempts to find input similarities of the training data. Optimisation based techniques usually use standard Deep Neural Networks and then adapt then trough optimiser to best fit the meta learning paradigm. A summary of the three types of meta-learning algorithms is given by the table below.

	Metric-based	Model-based	Optimisation-based
Key idea	Input similarity	Internal task representation	Optimise for fast adaptation
Strength	Simple and effective	Flexible	More robust generalisability
How is $P_\theta(y\|\mathbf{x})$ modelled?	$\sum_{(\boldsymbol{x}_i,y_i)\in S_i} k_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{x}_i)\, y_i$	$f_\theta(\mathbf{x}, S)$	$P_{g_\phi(\theta,S^L)}(y\|\mathbf{x})$

This table taken was taken from Huisman et al. [21] and provides a high-level overview of the three deep meta-learning categories, i.e., metric-, model-, and optimisation-based techniques, and their main strengths and weaknesses. Recall that τ_i is a task, $S_i \sim \mathcal{D}_i$ the corresponding task distribution, $k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}_i)$ a kernel function returning the similarity between the two inputs $\boldsymbol{x}$ and $\boldsymbol{x}_i, y_i$ are the true labels for known inputs $\boldsymbol{x}_i, \theta$ are base-learner parameters, and $g_{\boldsymbol{\varphi}}$ is a (learned) optimiser with parameters φ. This is by no means meant to be a comprehensive summary of all meta-learning techniques and is just meant to inform that these approaches exists and that this essay is not meant to summarise the entire field of meta-learning. The interested reader is referred to review papers such as Huisman et al. [21] or Hospedales et al. [20].

A few common concerns to keep in mind when building, analysing or deploying meta-learning algorithms is the number of tasks needed for training, number of examples needed per task, speed of adoption of the model.

1.2 Optimisation-Based Meta-Learning

An algorithm that has recently gained much attraction in the field is called Model-Agnostic-Meta-Learning (MAML), with impressive performance and being very general to apply, while the core ideas are relatively simple to understand. Optimisation-based approaches to meta-learning attempt to learn a suitable initialisation of the model which can be fine tuned at test-time in order to fit the task at hand. A version of this idea precedes meta-learning and can be seen in pre-trained models. In a lot of Computer Vision or Natural Language Processing applications large models are trained on sizeable general datasets. The resulting model configuration is then used as an initialisation for the fine-tuning process on a much smaller dataset to adapt the pre-trained model to the desired task. In recent years optimisation-based meta-learning techniques such as MAML and REPTILE have gained much attraction with great performance on benchmark datasets such as Omniglot [26] and MiniImagenet [36]. First introduced in 2017 by Finn et al. [11] MAML prepares a deep learning model for fast adoption to new tasks through differentiating in the fine-tuning process to directly optimise performance. As a result the learner obtains a sensible gradient-based learning algorithm even when it receives out of sample data as pointed out by Nichol et al. [30]. The main intuition underlying the algorithm is that some internal representation are more transferable than others. The neural network might be able to learn a representation that is applicable to all tasks in the environment $\mathcal{T}$, as opposed to just one individual task. MAML attempts to find model parameters that are sensitive to changes in the tasks. A change in the direction of the gradient of the loss function then allows a small change in the model parameters lead to large improvements in the new tasks drawn from $\mathcal{T}$. Algorithm 1 implements MAML in pseudo code.

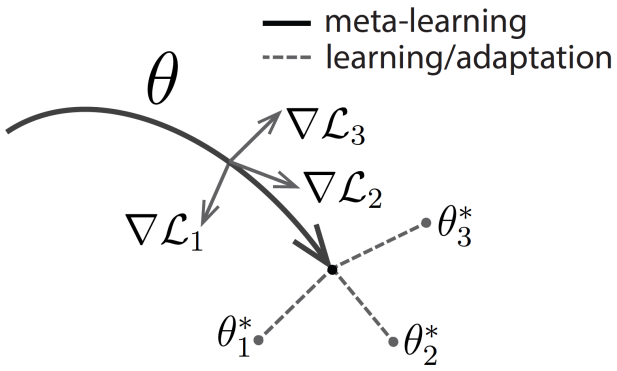


Figure 2: Parameter update of θ with three tasks τ_i during training according to the inner loop of the MAML algorithm illustrated visually. Figure taken from Finn et al. [11].

Figure 2 illustrates how the MAML algorithm optimises the meta-learner f_θ for a current representation $\theta \in \Theta$, for some Neural Network (NN) parameter space Θ. The MAML algorithm consists of two main stages. Firstly the inner loop, in which the algorithm adapts the model parameters according to one task $\tau_i \sim \mathcal{T}$. Secondly, the meta-step uses the information from a batch of examined tasks $\tau_1, ..., \tau_m$ to perform the meta-update. Given a loss function $\mathcal{L}$ and α, β as step-size hyperparameters. For the current task $\tau_i \sim \mathcal{T}$ MAML's inner loop will temporarily update the current model parameters θ to θ_i'. For example with one gradient update:

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\tau_i}(f_\theta)) = \theta - \alpha g_{\text{MAML}} \tag{2}$$

The left side of the above equation holds for one gradient update, while the $g_{\text{MAML}} = \nabla_\theta \mathcal{L}_{\tau_i}(U_{\tau_i}^k)$ is a more general notation inspired by Nichol et al. [30]. Here $U_{\tau_i}^k$ corresponds to k gradient updates in the inner loop. So for $k = 1$, $g_{\text{MAML}} = \nabla_\theta \mathcal{L}_{\tau_i}(f_\theta))$. According to Antoniou et al. [3] choosing a larger number of inner gradient steps can sometimes lead to better performance, but usually $k = 1$ is chosen.

The objective for our meta learner is given by

$$\min_\theta \sum_{\tau_i \sim \mathcal{T}} \mathcal{L}_{\tau_i}(f_{\theta_i'}). \tag{3}$$

The meta optimisation step then uses Stochastic Gradient Descent (SGD) to update the model as follows.

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\tau_i \sim \mathcal{T}} \mathcal{L}_{\tau_i}(f_{\theta_i'}) \tag{4}$$

This algorithm only assumes that the model f is parameterised by θ and that it is possible to perform gradient descent updates. As such MAML is very general and easy to implement. The main computational drawback of the algorithm is in the outer loop of Algorithm 1. The gradient operator in the meta-step uses second order derivatives when backpropagating the error through the meta-objective. As this Hessian-vector computation is quite expensive some techniques have been proposed to reduce it. The authors themselves proposed simply to omit these higher order derivatives, in which they reported similar performance and a 33% reduction in computational

Algorithm 1 Model-Agnostic Meta-Learning according to Finn et al. [11]

Require: $\mathcal{T}$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialise θ
2: **while** not done **do**
3: Sample batch of tasks $\tau_i \sim \mathcal{T}$
4: **for** all τ_i **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\tau_i}(f_\theta)$ with respect to K examples Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\tau_i}(f_\theta)) = \theta - \alpha g_{\text{MAML}}$
6: **end for**
7: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\tau_i \sim \mathcal{T}} \mathcal{L}_{\tau_i}(f_{\theta_i'})$
8: **end while**

costs, Finn et al. [11]. According to Goodfellow et al. [18] ReLU activation function in deep neural networks are almost linear, so the second-order gradients tend to vanish, which could be a possible explanation as to why the first-order algorithm works comparably well. Returning to the inner loop update in equation (2), g_{MAML} can be written more generally as

$$g_{\text{MAML}} = \nabla_\theta \mathcal{L}_{\tau_i}(U_{\tau_i}^k(\theta)) = dU_{\tau_i}^k \nabla_\theta \mathcal{L}_{\tau_i}(\tilde{\theta}), \text{ where } \tilde{\theta} = U_{\tau_i}^k(\theta). \tag{5}$$

In equation (5) dU_{τ_i} is the Jacobian matrix of the update operation. The update operation is equivalent to the addition of a series of k gradient vectors to the original parameter θ, $U_{\tau_i}^k(\theta) = \theta + g_1 + ... + g_k$. Depending on the optimiser, such as Adam, these gradients might be rescaled however the following conclusion remains unchanged as pointed out by Nichol et al. [30]. In First-Order MAML (FOMAML) as introduced by Finn et al. [11] these gradients g_i are all treated as constant, so the Jacobian $dU_{\tau_i}^k = I$ simply becomes the identity operation. The gradient update rule in the inner loop is then $g_{\text{FOMAML}} = \nabla_\theta \mathcal{L}_{\tau_i}(\tilde{\theta})$. In replacing g_{MAML} with g_{FOMAML} in Algorithm 1 the FOMAML algorithm is obtained.

Another First-Order modification of the MAML algorithm was introduced by Nichol et al. [30] and is named REPTILE. Unlike MAML this method does not unroll a computation graph or calculate second order derivatives. Instead REPTILE performs standard SGD on each task in the given batch and then averages the resulting connection vectors to these resulting points. The meta-update rule of REPTILE works as follows for a batch of n tasks $\tau_1, ..., \tau_n$,

$$\theta \leftarrow \theta + \epsilon \left(\frac{1}{n} \sum_{i=1}^{n} U_{\tau_i}(\theta) - \theta \right), \tag{6}$$

where $U_{\tau_i}(\theta)$ denotes k steps of some optimiser such as SGD or Adam with respect to task τ_i. The rest of the procedure is detailed in Algorithm 2. REPTILE was introduced as a sophisticated first-order alternative to MAML. This algorithm finds suitable parameter configurations $U_{\tau_i}(\theta)$ for each task in the batch and then averages them for the parameter update instead of using second-order derivatives. MAML and REPTILE will both be revisited in Section 2.3 when their gradient meta-updates will be derived from PAC-Bayesian meta-learning bounds. Moreover, MAML will be included for completeness in Section 4 for some qualitative numerical experiment.

Algorithm 2 Reptile according to Nichol et al. [30]

Require: $\mathcal{T}$: distribution over tasks
Require: ϵ: step size hyperparameters
1: randomly initialise θ
2: **while** not done **do**
3: Sample batch of tasks $\tau_i \sim \mathcal{T}$, corresponding to loss $\mathcal{L}_{\tau_i}(f_{\tilde{\theta}})$ on weight vectors $\tilde{\theta}$
4: $\tilde{\theta} = U^k_{\tau_i}(\theta)$, denoting k steps of SGD or Adam
5: Update $\theta \leftarrow \theta + \epsilon(\tilde{\theta} - \theta)$
6: **end while**

Other popular more Bayesian variants of the above are the Bayesian MAML (BMAML) Kim et al. [23], Neural Processes Garnelo et al. [15] and PAC-Bayesian meta-learning algorithm (MLAP) Amit and Meir [2]. Due to the limited scope of this work it will not be possible to go through all of them in any detail, but the interested reader is referred to the corresponding articles.

2 Meta Learning in the PAC-Bayesian Framework

As previously mentioned the research field is of an empirical nature. A significant proportion of articles published propose a new algorithm or a modification to some existing technique and then report the performance on some benchmark dataset, such as Omniglot (Lake et al.[26]) or MiniImagenet (Ravi and Larochelle [36]). No unifying theoretical framework exists in which meta-learning algorithms can be analysed. Therefore there exists a gap between the impressive empirical success of meta-learning algorithms and theoretical explanations as to why these algorithms perform so well. Probably Approximately Correct (PAC) Bayesian learning delivers a rigorous framework to analyse the probabilistic generalisation of algorithms in standard supervised learning. In most settings the number of tasks available for meta-training is small. In this environment many previously mentioned algorithms such as MAML and REPTILE suffer from overfitting on the meta-training tasks, Yin et al. [44]. Rothfuss et al. [40], [39] showed how PAC-Bayesian meta-learning can mitigate this issue, as its probabilistic performance is guaranteed by Theorem 2.3. This section will first introduce general PAC-Bayesian bounds and then apply them to the meta-learning paradigm. Theorem 2.3 which gives probabilistic guarantees on the performance of PAC-Bayesian meta-learners is stated and proven. In addition, further mathematical statements will follow in order to optimise the PAC-bound in Theorem 2.3. Moreover, this framework will be able derive MAML and REPTILE from PAC-Bayesian bounds for meta-learning as demonstrated in Ding et al. [10]. As such this section outlines the main theory and lays the groundwork for constructing efficient algorithms based on PAC-Bayesian meta-learning which will be done in Section 3. The main contribution of the essay in this section is the comprehensive overview of presenting the results of Rothfuss et al. [40] and Ding et al. [10] in conjunction with each other. Moreover the proof of Theorem 2.3 which is central to the work in [40] was flawed in its final step. This has been corrected by myself in correspondence with the authors Rothfuss et al. and presented in this work.

2.1 Introduction to PAC-Bayesian Bounds

Firstly some notation and preliminaries will be introduced where mainly Rothfuss et al. [39] is followed. There is a data distribution $\mathcal{D}$ over a domain $\mathcal{Z}$ from which m observations are sampled to form the dataset $S = \{z_i\}_{i=1}^m$, $z_i \sim \mathcal{D}$. Where $S \sim \mathcal{D}^m$ indicates the i.i.d. sampling from $\mathcal{D}$. In supervised learning, which is the main focus of this essay, a training sample $z_i = (x_i, y_i)$ is tuple consisting of input features $x_i \in \mathcal{X}$ and target labels $y_i \in \mathcal{Y}$. In general the learning algorithm attempts to find a hypothesis $h \in \mathcal{H}$ with $h : \mathcal{X} \to \mathcal{Y}$ for some hypothesis space $\mathcal{H}$, which enables prediction for new inputs $x \sim \mathcal{D}$. For a loss function $l : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}$ the chosen hypothesis attempts to minimise the expected error under the data distribution, which is denoted as $\mathcal{L}(h, \mathcal{D}) = \mathbb{E}_{z^* \sim \mathcal{D}}\, l(h, z^*)$. Unfortunately $\mathcal{D}$ is unknown is practice which results in $\mathcal{L}(h, \mathcal{D})$ being intractable to compute. However, it is possible to compute the empirical error $\hat{\mathcal{L}}(h, S) = \frac{1}{m} \sum_{i=1}^m l(h, z_i)$ [1]. One approach would be to minimise $\hat{\mathcal{L}}(h, S)$ and hope that the expected error is minimised. In general however this is likely to lead to significant overfitting on the test data. The main idea of PAC-learning is to give

[1] Of course the data should be split in a training and test dataset with $S^{\text{train}}, S^{\text{test}} \subset S$ and $S^{\text{train}} \cup S^{\text{test}} = S$, but this would make the notation more cluttered so it will not be stated explicitly again.

probabilistic guarantees on bounding the unknown expected loss with the empirical loss. In order to do so one needs to essentially bound the difference between the two quantities $\Delta \geq |\mathcal{L}(h, \mathcal{D}) - \hat{\mathcal{L}}(h, S)|$ with probability $1 - \delta$, for $\delta \in (0, 1]$. PAC-Bayesian bounds then have the following form.

$$\mathcal{L}(h, \mathcal{D}) \leq \hat{\mathcal{L}}(h, S) + \Delta \tag{7}$$

These kind of bounds then hold for some large probability $1 - \delta$. PAC-Bayesian machine learning attempts to find the hypothesis $h \in \mathcal{H}$ for which the RHS is as small as possible, so for which the bound is tightest. Therefore this kind of learning is Probably Approximately Correct (PAC) which is where the name comes from.

To extend these ideas into the Bayesian framework the predictors are randomised and the learning attempts to find a hypothesis in the set of all probability measures on the hypothesis space, denoted by $\mathcal{M}(\mathcal{H})$. The two probability measures of interest are the Prior $P \in \mathcal{M}(\mathcal{H})$ and Posterior $Q \in \mathcal{M}(\mathcal{H})$. In the PAC-Bayesian framework it is assumed that the Prior P is independent of the observed data, while the Posterior may depend on it, Rothfuss et al. [39]. This stands in stark contrast to standard Bayesian Inference which is founded on the much stronger assumption that Prior and Posterior are closely related through Bayes' theorem. Throughout this essay it will be assumed that the Kullback-Leibler (KL) divergence $D_{KL}(Q||P)$ exists [2]. Also define in analogy to the expected error the Gibbs error for a randomised predictor Q as $\mathcal{L}_{\text{Gibbs}}(Q, \mathcal{D}) = \mathbb{E}_{h\sim Q}\, \mathcal{L}(h, \mathcal{D})$ and the empirical counterpart $\hat{\mathcal{L}}_{\text{Gibbs}}(Q, S) = \mathbb{E}_{h\sim Q}\, \hat{\mathcal{L}}(h, S)$.

Definition 2.1 (Centred Cumulant-Generating Function) *For a random Variable X with a distribution ν and a real-valued function f the centred cumulant-generating function is defined as*

$$\Psi_{\nu, f(\cdot)} = \ln \mathbb{E}_{X\sim\nu} \left[e^{t(f(X) - \mathbb{E}[f(X)])} \right]. \tag{8}$$

The Centred CGF is defined by the logarithm of the standard centred moment-generating function and measures how much a random variable $f(X)$, $X \sim \nu$ deviates from its mean. Using these preliminaries and definition it is possible to bound the Gibbs error $\mathcal{L}_{\text{Gibbs}}(Q, \mathcal{D}) = \mathbb{E}_{h\sim Q}\, \mathcal{L}(h, \mathcal{D})$ by its empirical counterpart $\hat{\mathcal{L}}_{\text{Gibbs}}(Q, S) = \mathbb{E}_{h\sim Q}\, \hat{\mathcal{L}}(h, S)$ in the following PAC-Bayesian bound.

Theorem 2.1 (Alquier et al. [1]) *Given a data distribution $\mathcal{D}$, hypothesis space $\mathcal{H}$, loss function $l(h, z)$, prior P, confidence level $\delta \in (0, 1]$, and $\beta > 0$ with probability at least $1 - \delta$ over samples $S \sim \mathcal{D}^m$, we have for all $Q \sim \mathcal{M}(\mathcal{H})$*

$$\mathcal{L}_{Gibbs}(Q, \mathcal{D}) \leq \hat{\mathcal{L}}_{Gibbs}(Q, S) + \frac{1}{\beta} \left[D_{KL}(Q||P) + \ln \frac{1}{\delta} + \Psi(\beta, m) \right], \tag{9}$$

with $\Psi(\beta, m) = \ln \mathbb{E}_{h\sim P} \mathbb{E}_{S\sim\mathcal{D}^m} \exp \left[\beta \left(\mathcal{L}(h, \mathcal{D}) - \hat{\mathcal{L}}(h, S) \right) \right]$

To get a better feeling for the terms in this bound consider the KL-divergence term. It would be

[2] This is not so much of a restriction as any probability distributions p of interest could be expressed using Kernel Density (KDE) estimation $p \approx \sum_i \delta(\boldsymbol{v}_i)$. Through the use of the delta function δ, KDE is supported over the whole space.

minimised if $Q = P$ is chosen. This is however not likely to yield good result as we assumed earlier that the prior P is independent of the data S. As such the empirical Gibbs error $\hat{\mathcal{L}}_{\text{Gibbs}}(Q, S)$ is likely to become very large for $Q = P$. The bound in (9) holds with probability $1 - \delta$, as such the $\frac{1}{\beta} \ln \frac{1}{\delta}$ can be interpreted as a penalising term for smaller confidence levels $\delta \in (0, 1]$.

Larger values for β would result in decreasing importance of the terms $D_{KL}(Q||P) + \ln \frac{1}{\delta}$ however centred cumulant-generating-function $\Psi(\beta, m)$ scales in β. Here a choice of $\beta = \sqrt{m}$ might be convenient as it results in the decrease of the KL-complexity term $D_{KL}(P||Q)$ as the number of training samples m increases, Rothfuss et al. [39]. The $\Psi(\beta, m)$ is a log moment-generating-function measuring the deviation of the empirical and expected errors. The dependence on the expected error $\mathcal{L}(h, \mathcal{D})$ makes it unknown in practice. If further assumptions on the loss function are made it is however still possible to bound this term tightly. For example, if $l(\cdot)$ is bounded in $[a, b]$:

$$\Psi(\beta, m) \leq \frac{\beta^2 (b - a)^2}{8m}, \tag{10}$$

which can be obtained through a simple application of Hoeffding's lemma (Lemma A.4). The loss function $l(\cdot)$ is considered be sub-Gamma with variance factor s^2 and sale parameter c, for a given prior P and data distribution $\mathcal{D}$ if it can be described by a sub-gamma random variable $X = \mathcal{L}(h, \mathcal{D}) - l(h, z)$. For reference the definition of sub-gamma random variables have been included in Definition A.2. The sub-Gamma assumption can be used to obtain the following

$$\Psi(\beta, m) \leq \frac{\beta^2 s^2}{8m(1 - c\beta/m)}. \tag{11}$$

Following Germain et al. [17] it can be shown in the case of linear regression $f_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}$ and squared loss $l(h, z) = (y - \boldsymbol{w} \cdot \boldsymbol{x})^2$ then the loss function can be considered valid and the above bounds hold. The sub-gaussian loss $l(\cdot)$ with variance factor s^2 can be understood in term of a scaling limit $c^+ \to 0$ of the above sub-gamma case. So we obtain

$$\Psi(\beta, m) \leq \frac{\beta^2 s^2}{8m}. \tag{12}$$

All three of these results could be derived more thoroughly, but something analogous is done in the final part of the proof of Theorem 2.3 so we will refrain from it here. The main point to keep in mind is that the bound in (9) can be made explicit and tight if assumptions on the loss function are made. Moreover, these assumptions are quite general and a large class of possible loss functions belong to it. Examples of loss functions which are bounded: Hinge loss, Logistic loss; Sub-Gamma: Huber loss; Sub-Gaussian: Squared loss and Hinge squared loss.

In order for our learning algorithm to achieve the best performance on unseen data it should seem natural to find a posterior distribution $Q \in \mathcal{M}(\mathcal{H})$ for which the RHS of equation (9) is minimised. The following lemma provides the solution to such a minimisation problem.

Lemma 2.2 (Cantoni [7]) *Let $\mathcal{H}$ be the hypothesis space, $g : \mathcal{H} \to \mathbb{R}$ a (loss) function, $Q \in \mathcal{M}(\mathcal{H})$ and $P \in \mathcal{M}(\mathcal{H})$ probability measures over $\mathcal{H}$. Then for any $\beta > 0$ and $h \in \mathcal{H}$ the posterior*

$$Q^*(h) \coloneqq \frac{P(h)e^{-\beta g(h)}}{\mathbb{E}_{h\sim P}\left[e^{-\beta g(h)}\right]}, \tag{13}$$

is the minimising probability density of

$$\arg\min_{Q\in\mathcal{M}(\mathcal{H})} \beta\, \mathbb{E}_{h\sim Q}[g(h)] + D_{KL}(Q||P). \tag{14}$$

This probability measure is called the Gibbs Posterior Q^*. Now we would like to use this lemma to find a Q^{opt} which minimises the terms on the RHS in the bound (9). For this we fix P, S, m, δ and can consider only the terms which depend on the posterior Q,

$$\begin{aligned} Q^{\text{opt}}(h) &= \arg\min_{Q\in\mathcal{M}(\mathcal{H})} \hat{\mathcal{L}}_{\text{Gibbs}}(Q, S) + \frac{1}{\beta} D_{KL}(Q||P) && (15)\\ &= \arg\min_{Q\in\mathcal{M}(\mathcal{H})} \beta\, \mathbb{E}_{h\sim Q}[\hat{\mathcal{L}}(h, S)] + D_{KL}(Q||P) && (16)\\ &= \frac{P(h)\, e^{-\beta\hat{\mathcal{L}}(h,S)}}{\mathbb{E}_{h\sim P}\left[e^{-\beta\hat{\mathcal{L}}(h,S)}\right]} && (17)\\ &= Q^*(h), && (18) \end{aligned}$$

where Lemma 2.2 is applied in the third equality. As a result one can see that the minimising posterior for the bound in Theorem 2.1 is the Gibbs posterior. As this Gibbs posterior will be very important later for minimising the meta-learning PAC-Bayesian bounds we state it again in slightly different form

$$Q^*(h) = P(h)e^{-\beta\hat{\mathcal{L}}(h,S)} / \, Z_\beta(S, P), \tag{19}$$

where $Z_\beta(S, P) = \int_{\mathcal{H}} P(h)e^{-\beta\hat{\mathcal{L}}(h,S)}dh$ is a normalisation constant. As mentioned before so far we have only assumed that the prior P is independent of the data and that the posterior Q may depend on it. As such the posterior $Q^*(h)$ is not necessarily connected to the prior P through Bayes Theorem and should be thought of as a 'pseudo-posterior' or 'generalised-posterior'. The generalised Bayesian framework introduced so far can be described as model-free, Geudj [19]. The connection between this generalised PAC-Bayesian framework and standard Bayes becomes clear in the probabilistic setting if we consider the negative log-likelihood $l(\cdot)$ as our loss, $l(h, z_i) \coloneqq -\log p(z_i|h)$ the optimal Gibbs posterior, becomes

$$Q^*(h) = \frac{P(h)e^{-\beta\hat{\mathcal{L}}(h,S)}}{\mathbb{E}_{h\sim P}[e^{-\beta\hat{\mathcal{L}}(h,S)}]} \tag{20}$$

$$= \frac{P(h)e^{-\beta\frac{1}{m}\sum_{i=1}^{m} l(h,z_i)}}{Z_\beta(S,P)} \tag{21}$$

$$= \frac{P(h)e^{-\beta\frac{1}{m}\sum_{i=1}^{m} -\log p(z_i|h)}}{Z_\beta(S,P)} \tag{22}$$

$$= \frac{P(h)\left(\prod_{i=1}^{m} p(z_i|h)\right)^{\beta/m}}{Z_\beta(S,P)} \tag{23}$$

$$= \frac{P(h)p(S|h)^{\beta/m}}{Z_\beta(S,P)}, \tag{24}$$

which coincides with the generalised Bayesian Posterior $Q^*(h;S,P)$. Here the normalisation constant $Z_\beta(S,P) = \int_{\mathcal{H}} P(h)e^{-\beta\hat{\mathcal{L}}(h,S)}dh = \int_{\mathcal{H}} P(h)\left(\prod_{j=1}^{m} p(z_j|h)\right)^{\beta/m} dh$ is called the generalised marginal likelihood. In the case the $\beta = m$ the standard Bayesian posterior is recovered.

The procedure in the PAC-Bayesian framework is bounding the general error by the empirical error and then to minimise the upper bound. In Section 2.2 these ideas will be revisited and extended to the meta-learning setting. Moreover, the negative log-likelihood loss function $l(h,z_i) := -\log p(z_i|h)$ will be used in the probabilistic model so that the Gibbs posterior $Q^*(h;S,P)$ coincides with the generalised Bayesian posterior.

2.2 PAC-Bayesian Meta-Learning Bounds

The aim of this section is to adapt the previously introduced theory of PAC-Bayesian bounds to the meta-learning framework. The overview of the meta-learning framework is depicted in the following Figure 3.

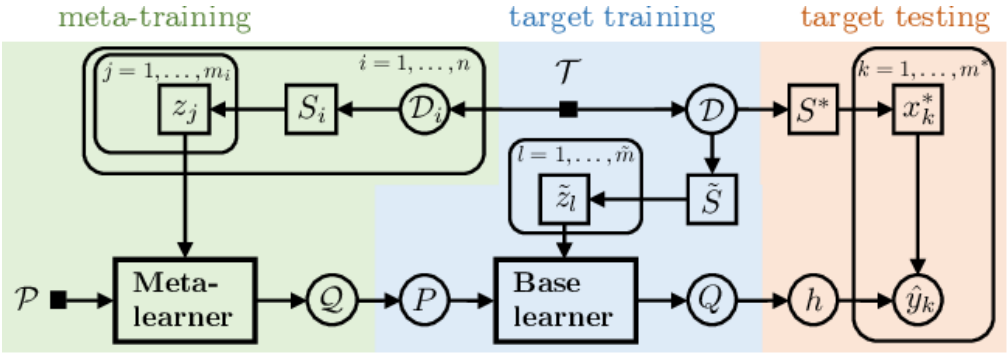


Figure 3: Overview Meta-Learning framework as presented in Rothfuss et al. [40] with environment $\mathcal{T}$, task distributions $\mathcal{D}_i$, target prior P, target posterior Q, dataset S, data point $z = (x,y)$, hyper-prior $\mathcal{P}$ and hyper-posterior $\mathcal{Q}$

A few modifications are necessary to the preliminaries introduced at the beginning of the previous Subsection 2.1. Since it is now of concern to learn a probability measure Q from a prior distribution P, the base learner $Q(h;S,P)$ denoted as a mapping $Q : \mathcal{Z}^m \times \mathcal{M}(\mathcal{H}) \to \mathcal{M}(\mathcal{H})$ defines a density

over the hypothesis space $\mathcal{H}$. This base-learner Q is the model that will be used during the testing or meta-testing phase. As such this meta-learning framework aims to learn the Prior P of the base-learner in a data-driven manner through the n statistically related tasks $\{\tau_1, ..., \tau_n\}$, where $\tau_i = (\mathcal{D}_i, S_i)$. Here $\mathcal{D}_i$ is the data distribution for task τ_i and $S_i \sim \mathcal{D}_i^{m_i}$ is the dataset generated by drawing m_i samples.

All tasks share the same data-domain $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, hypothesis space $\mathcal{H}$ and loss function $l(\theta, z)$. In order to learn this prior P the meta-learner starts with a Hyper-prior $\mathcal{P} \in \mathcal{M}(\mathcal{M}(\mathcal{H}))$, which is a probability distribution over all possible priors (a probability measure over probability measures). Over the course of training on the n datasets $S_1, ..., S_n$, where is $S_i \sim \tau_i$ respectively, the meta-learner outputs a Hyper-posterior $\mathcal{Q} \in \mathcal{M}(\mathcal{M}(\mathcal{H}))$ over priors. For the target training and testing a prior $P \sim \mathcal{Q}$ is then drawn. One can one define the expected Gibbs error when sampling priors P from the Hyper-posterior, this is known as the transfer-error.

$$\mathcal{L}_{\text{transfer}}(\mathcal{Q}, \mathcal{T}) = \mathbb{E}_{P\sim\mathcal{Q}}\left[\mathbb{E}_{(\mathcal{D},m)\sim\mathcal{T}}[\mathbb{E}_{S\sim\mathcal{D}^m}[\mathcal{L}_{\text{Gibbs}}(Q(h; S, P), \mathcal{D})]]\right] \tag{25}$$

Like the expected error defined in Section 2.1 the transfer-error is unknown in practice so the empirical multi-task error is used.

$$\hat{\mathcal{L}}_{\text{multi}}(\mathcal{Q}, S_1, ..., S_n) = \mathbb{E}_{P\sim\mathcal{Q}}\left[\frac{1}{n}\sum_{i=1}^{n}\hat{\mathcal{L}}_{\text{Gibbs}}(Q(h; S_i, P), S_i)\right] \tag{26}$$

The following Theorem shows how it is possible to bound the true transfer error $\mathcal{L}(\mathcal{Q}, \mathcal{T})$ by the empirical multi-task error $\hat{\mathcal{L}}(\mathcal{Q}, S_1, ..., S_n)$. This Theorem will for the basis of the analysis in this section and will provide probabilistic performance guarantees for the PACOH algorithms developed in Section 3.

Theorem 2.3 (Rothfuss et al. [40]) *Let $Q : \mathcal{Z}^m \times \mathcal{M}(\mathcal{H}) \to \mathcal{M}(\mathcal{H})$ be a base learner, $\mathcal{P} \in \mathcal{M}(\mathcal{M}(\mathcal{H}))$ some fixed hyper-prior, $\lambda, \beta > 0$ and $\lambda \geq \sqrt{n}$, $\beta \geq \sqrt{\tilde{m}}$. For any confidence level $\delta \in (0, 1]$ the inequality*

$$\begin{aligned}\mathcal{L}_{transfer}(\mathcal{Q}, \mathcal{T}) \leq \hat{\mathcal{L}}_{multi}(\mathcal{Q}, S_1, ..., S_n) + \left(\frac{1}{\lambda} + \frac{1}{n\beta}\right) D_{KL}(\mathcal{Q}||\mathcal{P}) \\ + \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\mathbb{E}_{P\sim\mathcal{Q}}[D_{KL}(Q(h; S_i, P)||P)] + C(\delta, \lambda, \beta).\end{aligned} \tag{27}$$

holds uniformly over all hyper-posteriors $\mathcal{Q} \in \mathcal{M}(\mathcal{M}(\mathcal{H}))$ with probability $1 - \delta$. In two special cases this term can be bounded as follows

Case I: If the loss function is bounded in $[a, b]$ and assume that the harmonic mean $\tilde{m} = (\sum_{i=1}^{n} m_i^{-1})^{-1}$ is independent of the environment $\mathcal{T}$, e.g. $\mathbb{E}_{\mathcal{T}}[m_i] = \mathbb{E}_{\mathcal{T}}[m]$, we can bound $C(\delta, \lambda, \beta)$ by

$$C(\delta, \lambda, \beta) \leq \left(\frac{\beta}{8n\tilde{m}} + \frac{\lambda}{8n}\right)(b - a)^2 - \frac{1}{\sqrt{n}}\ln\delta. \tag{28}$$

Case II: If the loss function is sub-gamma with variance factor s_I^2 and scale parameter c_I for the data distributions $\mathcal{D}_i$ and s_{II}^2, c_{II} for the task distribution $\mathcal{T}$. Furthermore if $\forall i : m_i = m$ we have

$$C(\delta, \lambda, \beta) \leq \frac{\beta s_I^2}{2m(1 - ((c_I \beta)/m)} + \frac{\lambda s_{II}^2}{2n(1 - ((c_{II} \lambda)/n)} - \frac{1}{\sqrt{n}} \ln \delta. \tag{29}$$

Here, $C(\delta, \lambda, \beta)$ is a rather complicated expression involving CGFs and will be specified in the proof later. It is worth noting that the form of Theorem 2.3 will be slightly different to the statement in Rothfuss et al. [40]. The condition $\lambda \geq \sqrt{n}, \beta \geq \sqrt{\tilde{m}}$ was not mentioned in [40] but used in the proof so it is included here. Furthermore, the proof in [40] contained a technical error, which resulted in the change of $C(\delta, \lambda, \beta)$. Moreover, in Case I the assumption $\mathbb{E}_{\mathcal{T}}[m_i] = \mathbb{E}_{\mathcal{T}}[m]$ had to be added and the resulting bound has a different form. If $\forall i : m_i = m$ then $\tilde{m} = m/n$ and the original bound from Rothfuss et al. [40] is recovered. For Case II the assumption $\forall i : m_i = m$ had to be added.

This theorem is useful in the sense that it allows the unknown transfer error to be bounded by the empirical multi-task error in the general meta-learning framework. It holds for any potentially sub-optimal hyper-posterior $\mathcal{Q}$ and base-learner Q. In Theorem 2.3 $\lambda, \beta > 0$ are tunable parameters. Following Rothfuss et al. [39] and Ding et al. [10] the following choices are reasonable:

a) In case $\lambda \propto \sqrt{n}$, $\beta \propto \sqrt{\tilde{m}}$ the bound is consistent in the sense that $|\mathcal{L}_{\text{transfer}}(\mathcal{Q}, \mathcal{T}) - \hat{\mathcal{L}}_{\text{multi}}(\mathcal{Q}, S_1, ..., S_n)| \to 0$ as $n, m \to \infty$. Moreover, the generalisation gap $|\mathcal{L}_{\text{transfer}} - \hat{\mathcal{L}}_{\text{multi}}|$ is at least $O(\frac{1}{\sqrt{\tilde{m}}})$

b) For $\lambda \propto n, \beta \propto \tilde{m}$ the generalisation gap $|\mathcal{L}_{\text{transfer}} - \hat{\mathcal{L}}_{\text{multi}}|$ is at least $O(\frac{1}{\tilde{m}})$ and the KL-divergence terms decay faster which makes the bound more applicable in cases of smaller samples sizes. Unfortunately however the $C(\delta, \lambda, \beta)$ term does not converge to zero, so there remains a gap in the bound.

In order to not loose track in this section the proof of Theorem 2.3 will be deferred to the end of the section as will the proofs of the following Corollary 2.4 and Proposition 2.5. Similar to the procedure in the previous Section 2.1, now that the PAC bound has been stated we would like to find the hyper-posterior $\mathcal{Q}$ and base-learner Q which make the bound in (27) as tight as possible. Using this PAC-Optimal hyper-posterior $\mathcal{Q}$ to draw priors P for the base-learner will result (PAC) optimal performance guarantees, with respect to Theorem 2.3. Minimising this bound means that the meta-learner has probabilistic guarantees on the difference between training and expected error. As such Theorem 2.3 is very useful in the way that it gives theoretical motivation and performance guarantees on an algorithm which approximates said PAC-optimal Hyper-posterior.

The result in Theorem 2.3 holds for an arbitrary hyper-posterior $\mathcal{Q}$ and base-learner $Q(h; S, P)$. In order to achieve optimal performance guarantees we will first consider the terms on the RHS in (27) which involve the base-learner, these are the empirical multi-task error and the empirical average of KL-divergences of the base-learner and prior.

$$\hat{\mathcal{L}}_{\text{multi}}(\mathcal{Q}, S_1, ..., S_n) + \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\mathbb{E}_{P\sim\mathcal{Q}}[D_{KL}(Q(h; S_i, P)||P)] \tag{30}$$

which can be simplified to

$$\mathbb{E}_{P\sim\mathcal{Q}}\left[\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{h\sim Q}[\hat{\mathcal{L}}(h, S_i)] + D_{KL}(Q(h; S_i, P)||P))\right] \tag{31}$$

The summand looks very similar to the form which we have already seen in Lemma 2.2 and therefore the Gibbs posterior $Q^*(h; S_i, P)$ is used as the base-learner. If this base learner is assumed the previous bound (27) can be restated as the following Corollary.

Corollary 2.4 (Rothfuss et al. [40]) *When using a Gibbs posterior* $Q^*(h; S_i, P) = P(h)\frac{\exp(-h\hat{\mathcal{L}}(h,S_i))}{Z_\beta(S_i,P)}$ *as a base-learner, under the assumptions of Theorem 2.3, we have with probability at least* $1-\delta$

$$\mathcal{L}_{transfer}(\mathcal{Q}, \mathcal{T}) \leq -\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\mathbb{E}_{P\sim\mathcal{Q}}[\ln Z_\beta(S_i, P)] + \left(\frac{1}{\lambda} + \frac{1}{n\beta}\right)D_{KL}(\mathcal{Q}||\mathcal{P}) + C(\delta, \lambda, \beta), \tag{32}$$

where

$$Z_\beta(S_i, P) = \mathbb{E}_{h\sim P}[\exp(-\beta\hat{\mathcal{L}}(h, S_i))] = \int_{\mathcal{H}} P(h)\, e^{-\beta\,\hat{\mathcal{L}}(h,S_i)}dh \tag{33}$$

denotes the normalisation constant.

The bound in (27) holds for any potentially sub-optimal posterior $Q \in \mathcal{M}(\mathcal{H}))$. Since Lemma 2.2 applies for any loss function in the general Bayesian setting the bound in (32) is at least as tight as (27), so we have $\mathcal{L}_{\text{transfer}}(\mathcal{Q}, \mathcal{T}) \leq (32) \leq (27)$. One of the main contributions of Rothfuss et al. [40] was that the bound in (32) does not depend on $Q(h; S_i, P)$. This means in order to minimise this bound one can directly optimise for the hyper-posterior $\mathcal{Q}$, which is much easier than the previous two-level optimisation problem of attempting to find $\mathcal{Q}$ and $Q(h; S_i, P)$ simultaneously in Amit and Meir [2].

Further examining the bound presented in (32). The first term of the bound corresponds to the expected generalised marginal log-likelihood under the hyper-posterior $\mathcal{Q}$. This term provides a measure of how well the model is likely to perform on new data. The second term, which is the KL-divergence between the hyper-posterior $\mathcal{Q}$ and the hyper-prior $\mathcal{P}$, acts as a meta-level regulariser as pointed out by Rothfuss et al. [40]. It penalises the complexity of the hyperparameters and encourages the model to favour simpler models. This is because the more complex a model is, the more it diverges from the hyper-prior, and the higher the KL-divergence becomes. Thus, the KL-divergence term is a way to balance the trade-off between model complexity and generalisation performance. Moreover, as the sample size grows larger and larger, the KL-divergence term approaches zero. This is compatible with the idea that a regulariser should be relatively important if the sample size is small, as there is less data to learn from, and should asymptotically vanish as the sample size and model complexity increase. In other words, as the model sees more data, the regularisation term becomes less important and the model can learn more complex patterns from the data.

To summarise, the bound presented in (32) strikes a balance between model complexity and generalisation performance by using a regularisation term that becomes less important as the sample size grows larger.

Now the only thing left to do is to find the optimal hyper-posterior $\mathcal{Q}^*$ for which the bound in (32) is as tight as possible. Then we will have the Gibbs posterior as an optimal base learner and this $\mathcal{Q}^*$ as the optimal hyper-posterior. Building an algorithm to approximate both of these would give a PAC-Optimal meta-learning algorithm, again with respect to the bound in Theorem 2.3. Fortunately there exists a closed for solution for this $\mathcal{Q}^*$, which is stated in the following Proposition.

Proposition 2.5 (PAC-Optimal Hyper-Posterior Rothfuss et al. [40]) *Given a hyper-prior $\mathcal{P}$ and datasets $S_1, ..., S_n$, the hyper-posterior minimising the meta-learning bound in (32) is given by*

$$\mathcal{Q}^*(P) = \frac{\mathcal{P}(P) \exp\left(\frac{\lambda}{n\beta+\lambda} \sum_{i=1}^n \ln Z_\beta(S_i, P)\right)}{Z^{II}(S_1, ..., S_n, \mathcal{P})}, \tag{34}$$

with $Z^{II}(S_1, ..., S_n, \mathcal{P}) = \mathbb{E}_{P\sim\mathcal{P}}\left[\exp\left(\frac{\lambda}{n\beta+\lambda} \sum_{i=1}^n \ln Z_\beta(S_i, P)\right)\right]$

The constants n, λ, β are all known. In analogy to standard Bayesian statistics the hyper-prior $\mathcal{P}$ is chosen before the meta-training begins. Therefore the unknowns in equation (34) are the generalised marginal log-likelihoods $\ln Z_\beta(S_i, P)$ and the normalising constant $Z^{II}(S_1, ..., S_n, \mathcal{P})$. There is unfortunately no hope to compute or approximate $Z^{II}(S_1, ..., S_n, \mathcal{P})$ due to the difficult integral. Therefore Bayesian Inference methods such as Stein Variational Gradient Descent (SVGD) or Variational Inference (VI) are required in order to approximate $\mathcal{Q}^*$. In order to apply these methods however we need to analytically compute or approximate the generalised marginal log-likelihoods $\ln Z_\beta(S_i, P)$.

Theorem 2.3 states probabilistic meta-learning guarantees for any (potentially sub-optimal) base-learner Q and hyper-posterior $\mathcal{Q}$. Then for Corollary 2.4 a Gibbs posterior $Q^*(h; S_i, P)$ was assumed as a base-learner, from which then the PAC-Optimal hyper-posterior $\mathcal{Q}^*$ was derived in Proposition 2.5. Recall from Section 2.1 that the Gibbs posterior $Q^*(h; S_i, P)$ coincides with the generalised Bayesian posterior if the negative log-likelihood loss function is used, $l(h, z_{ij}) = -\ln p(z_{ij}|h)$. Therefore the PACOH algorithms in Section 3 will all use Bayesian models with a negative log-likelihood loss function. The hyper-posterior $\mathcal{Q} \in \mathcal{M}(\mathcal{M}(\mathcal{H}))$ is a probability distribution over probability distributions on the hypothesis space $\mathcal{H}$. In order to use this theory to construct actual algorithms a parametric space of priors Φ is considered. So $\phi \in \Phi$ fully specifies a prior P_ϕ. The hyper-posterior $\tilde{\mathcal{Q}}$ is then approximated either using K particles $\{P_{\phi_1}, ..., P_{\phi_K}\}$ with $\phi_k \in \mathcal{P}$ or through a parametric variational family $\mathcal{F} = \{\tilde{\mathcal{Q}}_v |\ v \in \mathcal{U}\} \subset \mathcal{M}(\Phi)$ with its parameter vector $v \in \mathcal{U}$. How this is done will be the subject of Section 3.1. So far we have left unspecified how the prior P_ϕ is parameterised and how the marginal log-likelihoods $\ln Z_\beta(S_i, P_\phi)$ are computed or approximated. Both of these depend on the Bayesian model used, in Section 3 Gaussian Processes (GPs) and Bayesian Neural Networks (BNNs) will be considered.

2.2.1 Proof of Theorem 2.3

Proof. The proof of Theorem 2.3 closely follows the one presented in Rothfuss et al. [39] and will be done in three steps. An additional quantity called the expected Multi-task error is also needed. Since the notation of all the error functions is quite overloaded and the proof interchanges them frequently a reminder of the previously introduced error functions is given.

- General supervised learning setting
  - expected error: $\mathcal{L}(h, \mathcal{D}) = \mathbb{E}_{z^* \sim \mathcal{D}}\, l(h, z^*)$
  - empirical error: $\hat{\mathcal{L}}(h, S) = \frac{1}{m} \sum_{i=1}^{m} l(h, z_i)$
- PAC-Bayesian supervised learning setting
  - Gibbs error: $\mathcal{L}_{\text{Gibbs}}(Q, \mathcal{D}) = \mathbb{E}_{h \sim Q}\, \mathcal{L}(h, \mathcal{D}) = \mathbb{E}_{h \sim Q} \mathbb{E}_{z^* \sim \mathcal{D}}\, l(h, z^*)$
  - empirical Gibbs error: $\hat{\mathcal{L}}_{\text{Gibbs}}(Q, S) = \mathbb{E}_{h \sim Q}\, \hat{\mathcal{L}}(h, S) = \frac{1}{m} \mathbb{E}_{h \sim Q} \sum_{i=1}^{m} l(h, z_i)$
- PAC-Bayesian meta-learning setting
  - transfer-error $\mathcal{L}_{\text{transfer}}(\mathcal{Q}, \mathcal{T}) = \mathbb{E}_{P \sim \mathcal{Q}} \left[\mathbb{E}_{(\mathcal{D}, m) \sim \mathcal{T}} [\mathbb{E}_{S \sim \mathcal{D}^m} [\mathcal{L}_{\text{Gibbs}}(Q(h; S, P), \mathcal{D})]] \right]$

$$\mathcal{L}_{\text{transfer}}(\mathcal{Q}, \mathcal{T}) = \mathbb{E}_{P \sim \mathcal{Q}}\, \mathbb{E}_{(\mathcal{D}, m) \sim \mathcal{T}} \mathbb{E}_{S \sim \mathcal{D}^m} \mathbb{E}_{h \sim Q(h; S, P)} \mathbb{E}_{z^* \sim \mathcal{D}}\, l(h, z^*) \tag{35}$$

  - empirical multi-task error: $\hat{\mathcal{L}}_{\text{multi}}(\mathcal{Q}, S_1, ..., S_n) = \mathbb{E}_{P \sim \mathcal{Q}} \left[\frac{1}{n} \sum_{i=1}^{n} \hat{\mathcal{L}}_{\text{Gibbs}}(Q(h; S_i, P), S_i) \right]$

$$\hat{\mathcal{L}}_{\text{multi}}(\mathcal{Q}, S_1, ..., S_n) = \mathbb{E}_{P \sim \mathcal{Q}} \left[\frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{h_i \sim Q_i} \frac{1}{m_i} \sum_{j=1}^{m_i} l(h_i, z_{ij}) \right] \tag{36}$$

  - expected multi-task error: $\tilde{\mathcal{L}}_{\text{multi}}(\mathcal{Q}, \mathcal{D}_1, ..., \mathcal{D}_n) = \mathbb{E}_{P \sim \mathcal{Q}} \left[\frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{S \sim D_i^{m_i}}\, \mathcal{L}_{\text{Gibbs}}(Q(h; S, P), \mathcal{D}_i) \right]$

$$\tilde{\mathcal{L}}_{\text{multi}}(\mathcal{Q}, \mathcal{D}_1, ..., \mathcal{D}_n) = \mathbb{E}_{P \sim \mathcal{Q}} \left[\frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{S \sim D_i^{m_i}} \mathbb{E}_{h \sim Q(h; S, P)} \mathbb{E}_{z^* \sim \mathcal{D}_i}\, l(h, z^*) \right] \tag{37}$$

Here (35), (36) and (37) are the written out definitions. It can be seen that the expected multi-task error is kind of an in-between the transfer and multi-task error through the replacement of $\mathbb{E}_{(\mathcal{D}, m)}$ with the sum $\frac{1}{n} \sum_{i=1}^{n}$ and $\mathcal{D}_i$. As such the expected multi-task error can be thought of as the transfer error in case the n observed tasks and their respective distributions $\mathcal{D}_i$ have been fixed. This will provide a useful link between the three quantities which will be seen later. The proof also heavily relies on the following Lemma 2.6 which is invoked twice.

Lemma 2.6 (Change of Measure inequality, Picard-Weibel and Guedj [32]) *let f be a random variable taking values in a set A and let $X_1, ..., X_l$ be independent random variables, with $X_k \in A$ and distribution μ_k. For functions $g_k : A \times A \to \mathbb{R}$. $k = 1, ..., l$, let $\xi_k(f) = \mathbb{E}_{X_k \sim \mu_k}[g_k(f, X_k)]$ denote the expectation of g_k under X_k for a fixed $f \in A$. Then for any distributions $\rho, \pi \in \mathcal{M}(A)$ and any $\lambda > 0$, we have that*

$$\mathbb{E}_{f \sim \rho} \left[\sum_{k=1}^{l} \xi_k(f) - g_k(f, X_k) \right] \leq \frac{1}{\lambda} \left(D_{KL}(\rho || \pi) + \ln \mathbb{E}_{f \sim \pi} \left[e^{\lambda (\sum_{k=1}^{l} \xi_k(f) - g_k(f, X_k))} \right] \right). \tag{38}$$

Some intuition as to why one might expect a Lemma of this form is that it gives an inequality for a change in measure for a random quantity $f \sim \rho$. In Bayesian learning the system changes the distribution from a prior to a posterior, so π will correspond to the prior and ρ to the posterior. The notation will unfortunately be slightly more cumbersome as we will require a hyper-prior and a prior for each of the n tasks which are all changed for their respective posteriors. Lemma 2.6 will initially be used by fixing the task distributions on the task-level. The Lemma will then be used again on the meta-level to draw the tasks. Both of these steps will involve the expected multi-task error, which is then eliminated by combining the two inequalities to receive the final bound. The proof strategy is as follows:

Proof strategy for Theorem 2.3:

- Step 1 (Task specific generalisation): Lemma 2.6 is applied in order to bound the difference of $\tilde{\mathcal{L}}_{\text{multi}}(\mathcal{Q}, \mathcal{D}_1, ..., \mathcal{D}_n)$ and $\hat{\mathcal{L}}_{\text{multi}}(\mathcal{Q}, S_1, ..., S_n)$. This bound applies on the task level for $\tau_1, ..., \tau_n$.
- Step 2 (Task environment generalisation): Lemma 2.6 is used again on the meta-level to bound $\mathcal{L}_{\text{transfer}}(\mathcal{Q}, \mathcal{T})$ and $\tilde{\mathcal{L}}_{\text{multi}}(\mathcal{Q}, \mathcal{D}_1, ..., \mathcal{D}_n)$
- Step 3 (Bounding residual terms): Combining the results of the first two previous steps will result in a random product of cumulant-generating functions which will need to be bounded again to achieve the desired result.

Step 1 (Task specific generalisation) The learning algorithm $Q : \mathcal{Z}^{m_i} \times \mathcal{M}(\mathcal{H}) \to \mathcal{M}(\mathcal{H})$ outputs a posterior $Q(h; S_i, P)$ over hypotheses $h \in \mathcal{H}$ given a prior P and a dataset $S_i \sim \mathcal{D}_i^{m_i}$ of size m_i. For the observed tasks $\tau_i = (\mathcal{D}_i, m_i)$, $i = 1, ..., n$ we now wish to bound the generalisation error of the learning algorithm.

In order to apply Lemma 2.6 to the union of all training sets $S' = \cup_{i=1}^n S_i$ with $l = \sum_{i=1}^n m_i$. So each X_k corresponds to one data point, i.e. $X_k = z_{ij}$ and $\mu_k = \mathcal{D}_i$. Further we set $f = (P, h_1, ..., h_n)$ to the tuple of one prior and n base hypotheses. So in this two-level hypothesis h_i is the hypothesis for the supervised task τ_i while P constitutes a hypothesis of the meta-learning problem. Correspondingly define two-level joint-hypotheses $\pi = (\mathcal{P}, P^n) = \mathcal{P} \prod_{i=1}^n P$ and $\rho = (\mathcal{Q}, Q^n) = \mathcal{Q} \prod_{i=1}^n Q_i$, with $Q_i = Q(h; S_i, P)$. Lemma 2.6 allows for a change in measure for a random quantity f. In this case initially $f \sim \rho$ so we have some hyper-prior $\mathcal{P}$ as well as a prior P for the n tasks. We define $g_k(f, X_k) = \frac{1}{nm_i} l(h_i, z_{ij})$ as the summand in the empirical multi-task error, which can be seen in (36). Now we wish to apply Lemma 2.6 with the quantities we have just defined. We will apply the Lemma now with $\gamma > 0$ as the constant to retrieve the following.

$$\mathbb{E}_{f \sim (\mathcal{Q}, Q^n)} \left[\sum_{k=1}^{l} \mathbb{E}_{z_{ij} \sim \mathcal{D}_i} \left[\frac{1}{nm_i} l(h_i, z_{ij}) \right] - \frac{1}{nm_i} l(h_i, z_{ij}) \right] \leq \frac{1}{\gamma} D_{KL}((\mathcal{Q}, Q^n) || (\mathcal{P}, P^n)) + \frac{1}{\gamma} \ln \mathbb{E}_{f \sim (\mathcal{P}, P^n)} \left[e^{\gamma (\sum_{k=1}^{l} \mathbb{E}_{z_{ij} \sim \mathcal{D}_i} \left[\frac{1}{nm_i} l(h_i, z_{ij}) \right] - \frac{1}{nm_i} l(h_i, z_{ij}))} \right] \quad (39)$$

This can be stated as.

$$\mathbb{E}_{f\sim(\mathcal{Q},Q^n)}\left[\sum_{k=1}^{l}\mathbb{E}_{z_{ij}\sim\mathcal{D}_i}\left[\frac{1}{nm_i}l(h_i,z_{ij})\right]\right]\leq\mathbb{E}_{f\sim(\mathcal{Q},Q^n)}\left[\sum_{k=1}^{l}\frac{1}{nm_i}l(h_i,z_{ij})\right]$$
$$+\frac{1}{\gamma}D_{KL}((\mathcal{Q},Q^n)||(\mathcal{P},P^n))+\frac{1}{\gamma}\ln\mathbb{E}_{P\sim\mathcal{P}}\mathbb{E}_{h\sim P}\left[e^{\gamma(\sum_{k=1}^{l}\mathbb{E}_{z_{ij}\sim\mathcal{D}_i}\left[\frac{1}{nm_i}l(h,z_{ij})\right]-\frac{1}{nm_i}l(h,z_{ij}))}\right]. \tag{40}$$

Here $i=1,...,n$ is an iterator over the tasks, while $j=1,...,m_i$ is an iterator over the samples from task τ_i. Lastly, $k=1,...,l$ iterates through all the samples. Using these we will now take (40) apart to simplify each term, starting with the LHS.

$$\begin{aligned}
\mathbb{E}_{f\sim(\mathcal{Q},Q^n)}\left[\sum_{k=1}^{l}\mathbb{E}_{z_{ij}\sim\mathcal{D}_i}\left[\frac{1}{nm_i}l(h_i,z_{ij})\right]\right] &= \frac{1}{n}\mathbb{E}_{f\sim(\mathcal{Q},Q^n)}\sum_{i=1}^{n}\sum_{j=1}^{m_i}\mathbb{E}_{z_{ij}\sim\mathcal{D}_i}\frac{1}{m_i}l(h_i,z_{ij})\\
&= \frac{1}{n}\mathbb{E}_{f\sim(\mathcal{Q},Q^n)}\sum_{i=1}^{n}\sum_{j=1}^{m_i}\frac{1}{m_i}\mathbb{E}_{z^*\sim\mathcal{D}_i}l(h_i,z^*)\\
&= \frac{1}{n}\mathbb{E}_{f\sim(\mathcal{Q},Q^n)}\sum_{i=1}^{n}\frac{m_i}{m_i}\mathbb{E}_{z^*\sim\mathcal{D}_i}l(h_i,z^*)\\
&= \frac{1}{n}\mathbb{E}_{P\sim\mathcal{Q}}\mathbb{E}_{(h_1,...,h_n)\sim(Q_1,...,Q_n)}\sum_{i=1}^{n}\mathbb{E}_{z^*\sim\mathcal{D}_i}l(h_i,z^*)\\
&= \frac{1}{n}\mathbb{E}_{P\sim\mathcal{Q}}\sum_{i=1}^{n}\mathbb{E}_{(h_1,...,h_n)\sim(Q_1,...,Q_n)}\mathbb{E}_{z^*\sim\mathcal{D}_i}l(h_i,z^*)\\
&= \frac{1}{n}\mathbb{E}_{P\sim\mathcal{Q}}\sum_{i=1}^{n}\mathbb{E}_{h_i\sim Q_i}\mathbb{E}_{z^*\sim\mathcal{D}_i}l(h_i,z^*)\\
&= \mathbb{E}_{P\sim\mathcal{Q}}\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{S\sim\mathcal{D}^{m_i}}\mathcal{L}_{\text{Gibbs}}(Q(h;S,P),\mathcal{D}_i)\\
&= \tilde{\mathcal{L}}_{\text{multi}}(\mathcal{Q},\mathcal{D}_1,...,\mathcal{D}_n).
\end{aligned}$$

Similarly we can bound the first term in the RHS of (40).

$$\begin{aligned}
\mathbb{E}_{f\sim(\mathcal{Q},Q^n)}\left[\sum_{k=1}^{l}\frac{1}{nm_i}l(h_i,z_{ij})\right] &= \frac{1}{n}\mathbb{E}_{f\sim(\mathcal{Q},Q^n)}\sum_{i=1}^{n}\sum_{j=1}^{m_i}\frac{1}{m_i}l(h_i,z_{ij})\\
&= \mathbb{E}_{P\sim\mathcal{Q}}\left[\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{h_i\sim Q_i}\sum_{j=1}^{m_i}\frac{1}{m_i}l(h_i,z_{ij})\right]\\
&= \hat{\mathcal{L}}_{\text{multi}}(\mathcal{Q},S_1,...,S_n)
\end{aligned}$$

Where the definition (36) was used in the last equality.

$$\begin{aligned}\frac{1}{\gamma}D_{KL}((\mathcal{Q},Q^n)||(\mathcal{P},P^n)) &= \frac{1}{\gamma}\mathbb{E}_{P\sim\mathcal{Q}}\left[\mathbb{E}_{h\sim Q_i}\left[\ln\frac{\mathcal{Q}(P)\prod_{i=1}^n Q_i(h)}{\mathcal{P}(P)\prod_{i=1}^n P_i(h)}\right]\right]\\ &= \frac{1}{\gamma}\mathbb{E}_{P\sim\mathcal{Q}}\left[\ln\frac{\mathcal{Q}(P)}{\mathcal{P}(P)}\right]+\sum_{i=1}^n\mathbb{E}_{P\sim\mathcal{Q}}\left[\mathbb{E}_{h\sim Q_i}\left[\ln\frac{Q_i(h)}{P_i(h)}\right]\right]\\ &= \frac{1}{\gamma}D_{KL}(\mathcal{Q}||\mathcal{P})+\frac{1}{\gamma}\sum_{i=1}^n\mathbb{E}_{P\sim\mathcal{Q}}\left[D_{KL}(Q_i||P)\right]\end{aligned}$$

In the last term on the RHS of (40) it is possible to simplify the term in the exponent as follows

$$\gamma\left(\sum_{k=1}^{l}\mathbb{E}_{z_{ij}\sim\mathcal{D}_i}\left[\frac{1}{nm_i}l(h,z_{ij})\right]-\frac{1}{nm_i}l(h,z_{ij})\right)=\frac{\gamma}{n}\left(\sum_{k=1}^{n}\mathcal{L}(h,\mathcal{D}_i)-\hat{\mathcal{L}}(h,S_i)\right)$$

Where the definitions of expected and empirical error were used. Combining all of these four simplifications into the bound (40) we obtain

$$\begin{aligned}\tilde{\mathcal{L}}_{\text{multi}}(\mathcal{Q},\mathcal{D}_1,...,\mathcal{D}_n) \leq \hat{\mathcal{L}}_{\text{multi}}(\mathcal{Q},S_1,...,S_n)+\frac{1}{\gamma}D_{KL}(\mathcal{Q}||\mathcal{P})\\ +\frac{1}{\gamma}\sum_{i=1}^n\mathbb{E}_{P\sim\mathcal{Q}}\left[D_{KL}(Q_i||P)\right]+\frac{1}{\gamma}\underbrace{\ln\mathbb{E}_{f\sim(\mathcal{P},P^n)}\left[e^{\frac{\gamma}{n}\left(\sum_{k=1}^n\mathcal{L}(h,\mathcal{D}_i)-\hat{\mathcal{L}}(h,S_i)\right)}\right]}_{\Gamma^{\text{I}}(\gamma)}\end{aligned}\tag{41}$$

The bound in (41) bounds the difference between the expected and empirical multi-task error. This completes the first step of the proof and we will return to this bound in step 3.

Step 2 (Task environment generalisation After bounding the difference between the expected and empirical multi-task error in Step 1 we now wish to bound the difference between the expected multi-task and transfer error in order to combine both results later. In this step Lemma 2.6 is applied again but on the meta-level. As such, the tasks are the random variables with $X_k\sim\tau_i$, $l=n$ and $\mu_k\sim\tau$. In the meta-level the learner updates the hyper-prior to the hyper-posterior over the course of the learning process. Therefore we set $\rho=\mathcal{Q}$, $\pi=\mathcal{P}$, $f=P$ and $g_k(f,X_k)=\frac{1}{n}\mathcal{L}_{\text{Gibbs}}(Q_i,\mathcal{D}_i)$. Through applying Lemma 2.6 with $\lambda>0$ the following bound is obtained.

$$\begin{aligned}\mathbb{E}_{P\sim\mathcal{Q}}\left[\sum_{i=1}^n\mathbb{E}_{\tau_i\sim\mathcal{T}}\left[\frac{1}{n}\mathcal{L}_{\text{Gibbs}}(Q_i,\mathcal{D}_i)\right]-\frac{1}{n}\mathcal{L}_{\text{Gibbs}}(Q_i,\mathcal{D}_i)\right] \leq \frac{1}{\lambda}D_{KL}(\mathcal{Q}||\mathcal{P})\\ +\frac{1}{\lambda}\ln\mathbb{E}_{P\sim\mathcal{P}}\left[e^{\lambda\left(\sum_{i=1}^n\mathbb{E}_{\tau_i\sim\mathcal{T}}\left[\frac{1}{n}\mathcal{L}_{\text{Gibbs}}(Q_i,\mathcal{D}_i)\right]-\frac{1}{n}\mathcal{L}_{\text{Gibbs}}(Q_i,\mathcal{D}_i)\right)}\right]\end{aligned}\tag{42}$$

Again separating the terms on the LHS individually yields.

$$\mathbb{E}_{P\sim\mathcal{Q}}\left[\sum_{i=1}^{n}\mathbb{E}_{\tau_i\sim\mathcal{T}}\left[\frac{1}{n}\mathcal{L}_{\text{Gibbs}}(Q_i,\mathcal{D}_i)\right]\right] \leq \mathbb{E}_{P\sim\mathcal{Q}}\left[\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_{\text{Gibbs}}(Q_i,\mathcal{D}_i)\right] + \frac{1}{\lambda}D_{KL}(\mathcal{Q}||\mathcal{P}) + \frac{1}{\lambda}\ln\mathbb{E}_{P\sim\mathcal{P}}\left[e^{\lambda\left(\sum_{i=1}^{n}\mathbb{E}_{\tau_i\sim\mathcal{T}}\left[\frac{1}{n}\mathcal{L}_{\text{Gibbs}}(Q_i,\mathcal{D}_i)\right]-\frac{1}{n}\mathcal{L}_{\text{Gibbs}}(Q_i,\mathcal{D}_i)\right)}\right] \quad (43)$$

Again simplifying the terms individually,

$$\begin{aligned}\mathbb{E}_{P\sim\mathcal{Q}}\left[\sum_{i=1}^{n}\mathbb{E}_{\tau_i\sim\mathcal{T}}\left[\frac{1}{n}\mathcal{L}_{\text{Gibbs}}(Q_i,\mathcal{D}_i)\right]\right] &= \mathbb{E}_{P\sim\mathcal{Q}}\left[\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{(\mathcal{D},m)\sim\mathcal{T}}\mathbb{E}_{S\sim\mathcal{D}^m}\left[\mathcal{L}_{\text{Gibbs}}(Q_i,\mathcal{D}_i)\right]\right]\\ &= \mathbb{E}_{P\sim\mathcal{Q}}\left[\frac{n}{n}\mathbb{E}_{(\mathcal{D},m)\sim\mathcal{T}}\mathbb{E}_{S\sim\mathcal{D}^m}\left[\mathcal{L}_{\text{Gibbs}}(Q_i,\mathcal{D}_i)\right]\right]\\ &= \mathcal{L}_{transfer}(\mathcal{Q},\mathcal{T}).\end{aligned}$$

Lastly the second term on the LHS is recognised as the expected multi-task error.

$$\mathbb{E}_{P\sim\mathcal{Q}}\left[\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_{\text{Gibbs}}(Q_i,\mathcal{D}_i)\right] = \tilde{\mathcal{L}}_{\text{multi}}(\mathcal{Q},\mathcal{D}_1,...,\mathcal{D}_n)$$

Combining these simplifications the bound in (43) becomes

$$\mathcal{L}_{transfer}(\mathcal{Q},\mathcal{T}) \leq \tilde{\mathcal{L}}_{\text{multi}}(\mathcal{Q},\mathcal{D}_1,...,\mathcal{D}_n) + \frac{1}{\lambda}D_{KL}(\mathcal{Q}||\mathcal{P}) + \frac{1}{\lambda}\underbrace{\ln\mathbb{E}_{P\sim\mathcal{P}}\left[e^{\frac{\lambda}{n}\left(\sum_{i=1}^{n}\mathbb{E}_{(\mathcal{D},S)\sim\mathcal{T}}[\mathcal{L}_{\text{Gibbs}}(Q(h;S,P),\mathcal{D})]-\mathcal{L}_{\text{Gibbs}}(Q(h;S_i,P),\mathcal{D}_i)\right)}\right]}_{\Gamma^{II}(\lambda)} \quad (44)$$

The above equation (44) is the desired bound between the transfer and expected multi-task error which concludes the second step of the proof.

Step 3 (Bounding the cumulant-generating functions In the final step we first combine the results the previous two steps. We do this by using the bound on the $\tilde{\mathcal{L}}_{\text{multi}}(\mathcal{Q},\mathcal{D}_1,...,\mathcal{D}_n)$ from (41) and plugging it into the bound (44) to obtain

$$\mathcal{L}_{transfer}(\mathcal{Q},\mathcal{T}) \leq \hat{\mathcal{L}}_{\text{multi}}(\mathcal{Q},S_1,...,S_n) + \left(\frac{1}{\gamma}+\frac{1}{\lambda}\right)D_{KL}(\mathcal{Q}||\mathcal{P}) + \frac{1}{\gamma}\sum_{i=1}^{n}\mathbb{E}_{P\sim\mathcal{Q}}\left[D_{KL}(Q_i||P)\right] + \frac{1}{\gamma}\Gamma^{I}(\gamma) + \frac{1}{\lambda}\Gamma^{II} \quad (45)$$

When choosing $\gamma = n\beta$ this bound looks quite similar to (27) that which we are trying to proof. The only thing left to do in this final step is to bound the random quantities $\frac{1}{\gamma}\Gamma^{I}(\gamma) + \frac{1}{\lambda}\Gamma^{II}(\lambda)$. It is worth reminding ourselves that the randomness in $\Gamma^{I}(\gamma)$ is governed by the random data points

z_{ij} which are i.i.d. sampled from $\mathcal{D}_i$. The randomness in $\Gamma(\lambda)$ is determined by the random tasks sampled from the environment $\mathcal{T}$. Throughout this step of the proof a few common inequalities such as Jensen and Markov are used, these have been included for the sake of completeness in the Appendix A. First we factor out $\sqrt{n}$ from λ and γ to obtain

$$\frac{1}{\gamma}\Gamma^I(\gamma) + \frac{1}{\lambda}\Gamma^{II}(\lambda) = \frac{1}{\sqrt{n}}\left(\frac{\sqrt{n}}{\gamma}\Gamma^I(\gamma) + \frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)\right) \tag{46}$$

Proceed in bounding $\frac{\sqrt{n}}{\gamma}\Gamma^I(\gamma) + \frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)$ on the RHS. For this we will use Markov's Inequality as stated in Theorem A.1 with $\phi(t) = \exp(t)$ as the non-nonngative and non-decreasing function and $X = \frac{\sqrt{n}}{\gamma}\Gamma^I(\gamma) + \frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)$ as a random variable.

$$\mathbb{P}\left(e^{\frac{\sqrt{n}}{\gamma}\Gamma^I(\gamma)+\frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)} > e^t\right) \leq \frac{\mathbb{E}\left[e^{\frac{\sqrt{n}}{\gamma}\Gamma^I(\gamma)+\frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)}\right]}{e^t} \tag{47}$$

Define $\delta = \mathbb{P}\left(e^{\frac{\sqrt{n}}{\gamma}\Gamma^I(\gamma)+\frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)} > e^t\right)$ as the probability on the LHS. Now rewrite the equation as follows

$$e^t \leq \frac{\mathbb{E}\left[e^{\frac{\sqrt{n}}{\gamma}\Gamma^I(\gamma)+\frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)}\right]}{\delta}. \tag{48}$$

By construction is $e^{\frac{\sqrt{n}}{\gamma}\Gamma^I(\gamma)+\frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)} \leq e^t$ with probability $1-\delta$. So the following bound holds with probability $1-\delta$.

$$e^{\frac{\sqrt{n}}{\gamma}\Gamma^I(\gamma)+\frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)} \leq \frac{\mathbb{E}\left[e^{\frac{\sqrt{n}}{\gamma}\Gamma^I(\gamma)+\frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)}\right]}{\delta}, \tag{49}$$

applying logarithms and multiplying by $\frac{1}{\sqrt{n}}$ results in,

$$\frac{1}{\gamma}\Gamma^I(\gamma) + \frac{1}{\lambda}\Gamma^{II}(\lambda) \leq \frac{1}{\sqrt{n}}\ln\mathbb{E}\left[e^{\frac{\sqrt{n}}{\gamma}\Gamma^I(\gamma)+\frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)}\right] - \frac{1}{\sqrt{n}}\ln\delta, \tag{50}$$

which again holds with probability $1-\delta$. Next we bound the expectation term on the RHS. As mentioned before the randomness in $\Gamma^I(\gamma)$ comes from the i.i.d. data samples $z_{ij} \sim \mathcal{D}_i$. The randomness of $\Gamma^{II}(\lambda)$ originates in the tasks themselves $\tau_i \sim \mathcal{T}$. When applying Lemma 2.6 in Step 1 of the proof these task distributions $\mathcal{D}_1, ..., \mathcal{D}_n$ were assumed to be fixed. The following step of bounding $\mathbb{E}\left[e^{\frac{\sqrt{n}}{\gamma}\Gamma^I(\gamma)+\frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)}\right]$ was done by Rothfuss et al. [40], [39] and Ding et al. [10].

$$\mathbb{E}\left[e^{\frac{\sqrt{n}}{\gamma}\Gamma^I(\gamma)+\frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)}\right] = \mathbb{E}_{\mathcal{D}_1,...,\mathcal{D}_n}\left[e^{\frac{\sqrt{n}}{\gamma}\Gamma^I(\gamma)}\right]\mathbb{E}_{\mathcal{T}}\left[e^{\frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)}\right] \tag{51}$$

The two terms were then bounded separately using Jensen's inequality. Unfortunately this is incorrect as the $\Gamma^I(\gamma)$ depends on the task distribution $\mathcal{D}_1, ..., \mathcal{D}_n$ and these depend on the sampled task $\tau_i \sim \mathcal{T}$. As such it is not possible to pull the first term on the RHS of (51) out of the expectation over the environment $\mathcal{T}$. As a result the proof needs to be changed from [40]. In correspondence

with the authors Rothfuss et al. the following adaption of the proof was developed.

Firstly, conditioning on the sigma-algebra generated the n task distributions $\sigma(\mathcal{D}_1, ..., \mathcal{D}_n)$ inside the expectation.

$$\mathbb{E}\left[e^{\frac{\sqrt{n}}{\gamma}\Gamma^I(\gamma)+\frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)}\right] = \mathbb{E}\left[\mathbb{E}_{\mathcal{D}_1,...,\mathcal{D}_n}\left[e^{\frac{\sqrt{n}}{\gamma}\Gamma^I(\gamma)+\frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)} \mid \sigma(\mathcal{D}_1, ..., \mathcal{D}_n)\right]\right] \tag{52}$$

$$= \mathbb{E}\left[e^{\frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)}\mathbb{E}_{\mathcal{D}_1,...,\mathcal{D}_n}\left[e^{\frac{\sqrt{n}}{\gamma}\Gamma^I(\gamma)} \mid \sigma(\mathcal{D}_1, ..., \mathcal{D}_n)\right]\right], \tag{53}$$

where the second step was possible as the randomness in $\Gamma^{II}(\lambda)$ originates in the tasks themselves $\tau_i \sim \mathcal{T}$, so by conditioning on the tasks it just becomes a scalar and can be pulled outside of the inner expectation. If $\lambda \geq \sqrt{n}$ and $\gamma \geq n\sqrt{m}$. The exponents $\sqrt{n}/\lambda$ and $\sqrt{n}/\gamma$ are smaller or equal to one. The function $\psi : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ with $\psi(x) = x^a$ is concave for $0 \leq a \leq 1$. As such we can apply Jensen inequality (Theorem A.2) to move the inner expectation inside of the exponent in equation (53). Moreover, for each $i = 1, ..., n$ denote the meta-task generalisation error V_i^{II} as the i.i.d. realisations of the random variable $\mathbb{E}_{(\mathcal{D},S)\sim\mathcal{T}}\left[\mathcal{L}_{\text{Gibbs}}(Q(S,P),\mathcal{D})\right] - \mathcal{L}_{\text{Gibbs}}(Q(h;S_i,P),\mathcal{D}_i)$. Similarly write the in-task generalisation error as V_{ij}^I for i.i.d. realisations of $\mathcal{L}(h_i,\mathcal{D}_i) - l(h,z_{ij})$ with $i = 1, ..., n$ and $j = 1, ..., m$. These are the random terms in Γ^I and Γ^{II}.

$$(53) \leq \mathbb{E}_{\mathcal{T}}\left[e^{\frac{\sqrt{n}}{\lambda}\Gamma^{\text{II}}(\lambda)}\left(\mathbb{E}_{\mathcal{D}_1,...,\mathcal{D}_n}\left[e^{\Gamma^{\text{I}}(\gamma)}\right]\right)^{\frac{\sqrt{n}}{\gamma}}\right] \tag{54}$$

$$= \mathbb{E}_{\mathcal{T}}\left[\left(\mathbb{E}_{\mathcal{P}}\left[e^{\frac{\lambda}{n}\sum_{i=1}^n V_i^{\text{II}}}\right]\right)^{\frac{\sqrt{n}}{\lambda}} \cdot \left(\mathbb{E}_{\mathcal{P}}\mathbb{E}_P\mathbb{E}_{\mathcal{D}_1,...,\mathcal{D}_n}\left[e^{\frac{\gamma}{nm}\sum_{i=1}^n\sum_{j=1}^{m_i} V_{ij}^{\text{I}}}\right]\right)^{\frac{\sqrt{n}}{\gamma}}\right] \tag{55}$$

$$= \mathbb{E}_{\mathcal{T}}\left[\left(\mathbb{E}_{\mathcal{P}}\left[\prod_{i=1}^n e^{\frac{\lambda}{n}V_i^{\text{II}}}\right]\right)^{\frac{\sqrt{n}}{\lambda}} \cdot \left(\mathbb{E}_{\mathcal{P}}\mathbb{E}_P\mathbb{E}_{\mathcal{D}_1,...,\mathcal{D}_n}\left[\prod_{i=1}^n\prod_{j=1}^{m_i} e^{\frac{\gamma}{nm_i}V_{ij}^{\text{I}}}\right]\right)^{\frac{\sqrt{n}}{\gamma}}\right] \tag{56}$$

As the realisation V^{I} and V^{II} are i.i.d. we can follow the arguments by Germain et al. [17] (proof of Corollary 4) to obtain

$$(53) \leq \underbrace{\mathbb{E}_{\mathcal{T}}\left[\left(\mathbb{E}_{\mathcal{P}}\left[\prod_{i=1}^n e^{\frac{\lambda}{n}V_i^{\text{II}}}\right]\right)^{\frac{\sqrt{n}}{\lambda}} \cdot \left(\left[\prod_{i=1}^n\prod_{j=1}^{m_i}\mathbb{E}_{\mathcal{P}}\mathbb{E}_P\mathbb{E}_{\mathcal{D}_i} e^{\frac{\gamma}{nm_i}V_{ij}^{\text{I}}}\right]\right)^{\frac{\sqrt{n}}{\gamma}}\right]}_{\Upsilon(\lambda,\gamma)}. \tag{57}$$

Combining our initial bound at the beginning of Step 3 (45) with the bound for the cumulant moment-generating-functions (50) results in

$$\mathcal{L}_{transfer}(\mathcal{Q},\mathcal{T}) \leq \hat{\mathcal{L}}_{\text{multi}}(\mathcal{Q},S_1,...,S_n) + \left(\frac{1}{n\beta}+\frac{1}{\lambda}\right) D_{KL}(\mathcal{Q}||\mathcal{P}) + \frac{1}{\gamma}\sum_{i=1}^{n}\mathbb{E}_{P\sim\mathcal{Q}}\left[D_{KL}(Q_i||P)\right] + \frac{1}{\sqrt{n}}\ln\mathbb{E}\left[e^{\frac{\sqrt{n}}{\gamma}\Gamma^{I}(\gamma)+\frac{\sqrt{n}}{\lambda}\Gamma^{II}(\lambda)}\right] - \frac{1}{\sqrt{n}}\ln\delta. \quad (58)$$

Now we substitute for the product expectation term in the bound (57) and replace $\gamma = n\beta$. It is worth noting that replacing $\gamma = n\beta$ is not restricting $\beta > 0$ in any way.

$$\mathcal{L}_{transfer}(\mathcal{Q},\mathcal{T}) \leq \hat{\mathcal{L}}_{\text{multi}}(\mathcal{Q},S_1,...,S_n) + \left(\frac{1}{n\beta}+\frac{1}{\lambda}\right) D_{KL}(\mathcal{Q}||\mathcal{P}) + \frac{1}{n\beta}\sum_{i=1}^{n}\mathbb{E}_{P\sim\mathcal{Q}}\left[D_{KL}(Q_i||P)\right] + \underbrace{\frac{1}{\sqrt{n}}\ln\mathbb{E}_{\mathcal{T}}\left[\left(\mathbb{E}_{\mathcal{P}}\left[\prod_{i=1}^{n}e^{\frac{\lambda}{n}V_i^{\text{II}}}\right]\right)^{\frac{\sqrt{n}}{\lambda}}\cdot\left(\left[\prod_{i=1}^{n}\prod_{j=1}^{m_i}\mathbb{E}_{\mathcal{P}}\mathbb{E}_{P}\mathbb{E}_{\mathcal{D}_i}e^{\frac{\beta}{m}V_{ij}^{\text{I}}}\right]\right)^{\frac{1}{\beta\sqrt{n}}}\right] - \frac{1}{\sqrt{n}}\ln\delta}_{C(\delta,\lambda,\beta)} \quad (59)$$

This concludes the main part of the proof. All that is left is provide the bound for $C(\delta,\lambda,\beta)$ in cases of a bounded and sub-gamma loss functions.

Case I: bounded loss First, we bound the moment-generating function (MGF) corresponding to $\Gamma^{\text{I}}(\gamma)$, if the loss function $l\left(h_i,z_{ij}\right)$ is bounded in $[a,b]$. Furthermore by definition $\mathbb{E}(V_{ij}^{\text{I}}) = 0$. We can now apply Hoeffding's lemma (Lemma A.4) to the moment generating function on the right of (57).

$$\mathbb{E}\left[e^{\frac{\sqrt{n}}{\gamma}\Gamma^{\text{I}}(\gamma)+\frac{\sqrt{n}}{\lambda}\Gamma^{\text{II}}(\lambda)}\right] \leq \Upsilon(\lambda,\gamma) \quad (60)$$

$$= \mathbb{E}_{\mathcal{T}}\left[\left(\mathbb{E}_{\mathcal{P}}\left[\prod_{i=1}^{n}e^{\frac{\lambda}{n}V_i^{\text{II}}}\right]\right)^{\frac{\sqrt{n}}{\lambda}}\cdot\left(\mathbb{E}_{\mathcal{D}_1},...,\mathbb{E}_{\mathcal{D}_n}\left[e^{\Gamma^{\text{I}}(\gamma)}\right]\right)^{\frac{\sqrt{n}}{\gamma}}\right] \quad (61)$$

$$= \mathbb{E}_{\mathcal{T}}\left[\left(\mathbb{E}_{\mathcal{P}}\left[\prod_{i=1}^{n}e^{\frac{\lambda}{n}V_i^{\text{II}}}\right]\right)^{\frac{\sqrt{n}}{\lambda}}\cdot\left(\left[\prod_{i=1}^{n}\prod_{j=1}^{m_i}\mathbb{E}_{\mathcal{P}}\mathbb{E}_{P}\mathbb{E}_{\mathcal{D}_i}e^{\frac{\gamma}{nm_i}V_{ij}^{\text{I}}}\right]\right)^{\frac{\sqrt{n}}{\gamma}}\right] \quad (62)$$

$$\leq \mathbb{E}_{\mathcal{T}}\left[\mathbb{E}_{\mathcal{P}}\left[\prod_{i=1}^{n}e^{\frac{\lambda}{n}V_i^{\text{II}}}\right]^{\sqrt{n}/\lambda}\left(\prod_{i=1}^{n}\prod_{j=1}^{m_i}e^{\frac{\gamma^2(b-a)^2}{8n^2m_i^2}}\right)^{\sqrt{n}/\gamma}\right] \quad (63)$$

Simplifying further yields,

$$\mathbb{E}_{\mathcal{T}}\left[\mathbb{E}_{\mathcal{P}}\left[\prod_{i=1}^{n} e^{\frac{\lambda}{n}V_i^{\mathrm{II}}}\right]^{\sqrt{n}/\lambda}\left(\prod_{i=1}^{n}\prod_{j=1}^{m_i} e^{\frac{\gamma^2(b-a)^2}{8n^2m_i^2}}\right)^{\sqrt{n}/\gamma}\right] = \mathbb{E}_{\mathcal{T}}\left[\mathbb{E}_{\mathcal{P}}\left[\prod_{i=1}^{n} e^{\frac{\lambda}{n}V_i^{\mathrm{II}}}\right]^{\sqrt{n}/\lambda}\left(\prod_{i=1}^{n} e^{\frac{\gamma^2(b-a)^2}{8n^2m_i}}\right)^{\sqrt{n}/\gamma}\right] \tag{64}$$

$$= \mathbb{E}_{\mathcal{T}}\left[\mathbb{E}_{\mathcal{P}}\left[\prod_{i=1}^{n} e^{\frac{\lambda}{n}V_i^{\mathrm{II}}}\right]^{\sqrt{n}/\lambda}\left(e^{\left(\sum_{i=1}^{n}\frac{1}{m_i}\right)\frac{\gamma^2(b-a)^2}{8n^2}}\right)^{\sqrt{n}/\gamma}\right] \tag{65}$$

$$= \mathbb{E}_{\mathcal{T}}\left[\mathbb{E}_{\mathcal{P}}\left[\prod_{i=1}^{n} e^{\frac{\lambda}{n}V_i^{\mathrm{II}}}\right]^{\sqrt{n}/\lambda}\right] e^{\frac{\gamma(b-a)^2}{8n^{3/2}\tilde{m}}}. \tag{66}$$

With the definition of the harmonic mean $\tilde{m} = \left(\sum_{i=1}^{n}\frac{1}{m_i}\right)^{-1}$. We assume here that the harmonic mean $\tilde{m}$ is independent of the realisation of environment, so that $\forall i : \mathbb{E}_{\mathcal{T}}[m_i] = \mathbb{E}_{\mathcal{T}}[m]$. Crucially, the upper bound $e^{\frac{\gamma(b-a)^2}{8n^{3/2}\tilde{m}}} \geq \mathbb{E}_{\mathcal{D}_1}\ldots\mathbb{E}_{\mathcal{D}_1}\left[\left(e^{\Gamma^{\mathrm{I}}(\gamma)}\right)^{\frac{\sqrt{n}}{\gamma}}\right]$ no longer depends on the tasks which are sampled from $\mathcal{T}$ which allows us to move it outside the expectation under $\mathcal{T}$. Secondly, we follow analogous steps to bound the MGF corresponding to $\Gamma^{\mathrm{II}}(\lambda)$. Again, assuming that $\lambda \geq \sqrt{n}$, we use Jensen's inequality (Theorem A.2) to move the exponent outside of the outer expectation and follow the arguments of Germain et al. [17] (Proof of Corollary 4) again to obtain

$$\mathbb{E}\left[e^{\frac{\sqrt{n}}{\gamma}\Gamma^{\mathrm{I}}(\gamma)+\frac{\sqrt{n}}{\lambda}\Gamma^{\mathrm{II}}(\lambda)}\right] \leq e^{\frac{\gamma(b-a)^2}{8n^{3/2}\tilde{m}}}\left[\prod_{i=1}^{n}\mathbb{E}_{\mathcal{T}}\mathbb{E}_{\mathcal{P}}e^{\frac{\lambda}{n}V_i^{\mathrm{II}}}\right]^{\sqrt{n}/\lambda} \tag{67}$$

From the boundedness of the loss together with Hoeffding's lemma, it follows that

$$\mathbb{E}\left[e^{\frac{\sqrt{n}}{\gamma}\Gamma^{\mathrm{I}}(\gamma)+\frac{\sqrt{n}}{\lambda}\Gamma^{\mathrm{II}}(\lambda)}\right] \leq e^{\frac{\gamma(b-a)^2}{8n^{3/2}\tilde{m}}+\frac{\lambda(b-a)^2}{8\sqrt{n}}} \tag{68}$$

Finally we use Markov's inequality and $\gamma = n\beta$ to obtain that with probability at least $1-\delta$

$$\frac{1}{\gamma}\Gamma^{\mathrm{I}}(\gamma) + \frac{1}{\lambda}\Gamma^{\mathrm{II}}(\lambda) = \frac{1}{\sqrt{n}}\left(\frac{\sqrt{n}}{\gamma}\Upsilon^{\mathrm{I}}(\gamma) + \frac{\sqrt{n}}{\lambda}\Upsilon^{\mathrm{II}}(\lambda)\right) \tag{69}$$

$$\leq \frac{1}{\sqrt{n}}\ln\mathbb{E}\left[e^{\frac{\sqrt{n}}{\gamma}\Upsilon^{\mathrm{I}}(\gamma)+\frac{\sqrt{n}}{\lambda}\Upsilon^{\mathrm{II}}(\lambda)}\right] - \frac{1}{\sqrt{n}}\ln\delta \tag{70}$$

$$\leq \frac{1}{\sqrt{n}}\left(\frac{\gamma(b-a)^2}{8n^{3/2}\tilde{m}} + \frac{\lambda(b-a)^2}{8\sqrt{n}}\right) - \frac{1}{\sqrt{n}}\ln\delta \tag{71}$$

$$= \left(\frac{\beta}{8n\tilde{m}} + \frac{\lambda}{8n}\right)(b-a)^2 - \frac{1}{\sqrt{n}}\ln\delta \tag{72}$$

$$\tag{73}$$

Case II: sub-gamma loss First, we assume that $\forall i = 1,\ldots,n$, the random variables $V_{ij}^{\mathrm{I}} := \mathcal{L}(h,\mathcal{D}_i) - l(h_i,z_{i,j})$ are sub-gamma with variance factor s_{I}^2 and scale parameter c_{I} under the

two-level prior $(\mathcal{P}, P)$ and the respective data distribution $\mathcal{D}_i$. we also assume that the number of samples from each task is consant $\forall i : m_i = m$ That is, their moment generating function can be bounded by that of a Gamma distribution Gamma $\left(s_{\mathrm{I}}^2, c_{\mathrm{I}}\right)$:

$$\mathbb{E}_{z\sim\mathcal{D}_i}\mathbb{E}_{P\sim\mathcal{P}}\mathbb{E}_{h\sim P}\left[e^{\gamma(\mathcal{L}(h,\mathcal{D}_i)-l(h,z))}\right] \leq \exp\left(\frac{\gamma^2 s_{\mathrm{I}}^2}{2\left(1-c_{\mathrm{I}}\gamma\right)}\right) \quad \forall\gamma\in(0,1/c_{\mathrm{I}})$$

Second, we assume that the random variable $V_i^{\mathrm{II}} := \mathbb{E}_{(D,S)\sim\mathcal{T}}[\mathcal{L}(Q(P,S),\mathcal{D})] - \mathcal{L}\left(Q\left(P,S_i\right),\mathcal{D}_i\right)$ is sub-gamma with variance factor s_{II}^2 and scale parameter c_{II} under the hyper-prior $\mathcal{P}$ and the task distribution $\mathcal{T}$. That is, its moment generating function can be bounded by that of a Gamma distribution Gamma $\left(s_{\mathrm{II}}^2, c_{\mathrm{II}}\right)$:

$$\mathbb{E}_{(\mathcal{D},S)\sim\mathcal{T}}\mathbb{E}_{P\sim\mathcal{P}}\left[e^{\lambda\left(\mathbb{E}_{(D,S)\sim\mathcal{T}}[\mathcal{L}(Q(P,S),\mathcal{D})]-\mathcal{L}(Q(P,S),\mathcal{D})\right)}\right] \leq \exp\left(\frac{\lambda^2 s_{\mathrm{II}}^2}{2\left(1-c_{\mathrm{II}}\lambda\right)}\right) \quad \forall\lambda\in(0,1/c_{\mathrm{II}})$$

The two inequalities above can be used to effectively bound the difficult product expectation term, which will result in a tight bound for the constant $C(\delta,\lambda,\beta)$

$$\begin{aligned}
\mathbb{E}\left[e^{\frac{\sqrt{n}}{\gamma}\Gamma^{\mathrm{I}}(\gamma)+\frac{\sqrt{n}}{\lambda}\Gamma^{\mathrm{II}}(\lambda)}\right] &\leq \Upsilon(\lambda,\gamma) && (74)\\
&= \mathbb{E}_{\mathcal{T}}\left[\left(\mathbb{E}_{\mathcal{P}}\left[\prod_{i=1}^{n} e^{\frac{\lambda}{n}V_i^{\mathrm{II}}}\right]\right)^{\frac{\sqrt{n}}{\lambda}}\cdot\left(\mathbb{E}_{\mathcal{D}_1},...,\mathbb{E}_{\mathcal{D}_n}\left[e^{\Gamma^{\mathrm{I}}(\gamma)}\right]\right)^{\frac{\sqrt{n}}{\gamma}}\right] && (75)\\
&= \mathbb{E}_{\mathcal{T}}\left[\left(\mathbb{E}_{\mathcal{P}}\left[\prod_{i=1}^{n} e^{\frac{\lambda}{n}V_i^{\mathrm{II}}}\right]\right)^{\frac{\sqrt{n}}{\lambda}}\cdot\left(\left[\prod_{i=1}^{n}\prod_{j=1}^{m}\mathbb{E}_{\mathcal{P}}\mathbb{E}_{P}\mathbb{E}_{\mathcal{D}_i} e^{\frac{\gamma}{nm_i}V_{ij}^{\mathrm{I}}}\right]\right)^{\frac{\sqrt{n}}{\gamma}}\right] && (76)\\
&\leq \mathbb{E}_{\mathcal{T}}\left[\left(\mathbb{E}_{\mathcal{P}}\left[\prod_{i=1}^{n} e^{\frac{\lambda}{n}V_i^{\mathrm{II}}}\right]\right)^{\frac{\sqrt{n}}{\lambda}}\cdot\left(\prod_{i=1}^{n}\prod_{j=1}^{m}\exp\left(\frac{(\frac{\gamma}{nm})^2 s_{\mathrm{I}}^2}{2\left(1-c_{\mathrm{I}}\frac{\gamma}{nm}\right)}\right)\right)^{\frac{\sqrt{n}}{\gamma}}\right] && (77)\\
&= \mathbb{E}_{\mathcal{T}}\left[\left(\mathbb{E}_{\mathcal{P}}\left[\prod_{i=1}^{n} e^{\frac{\lambda}{n}V_i^{\mathrm{II}}}\right]\right)^{\frac{\sqrt{n}}{\lambda}}\cdot\left(\exp\left(\frac{\frac{nm}{1}\frac{\gamma^2}{n^2m^2}\frac{\sqrt{n}}{\gamma}s_{\mathrm{I}}^2}{2\left(1-c_{\mathrm{I}}\frac{\gamma}{nm}\right)}\right)\right)\right] && (78)\\
&= \mathbb{E}_{\mathcal{T}}\left[\left(\mathbb{E}_{\mathcal{P}}\left[\prod_{i=1}^{n} e^{\frac{\lambda}{n}V_i^{\mathrm{II}}}\right]\right)^{\frac{\sqrt{n}}{\lambda}}\right]\cdot\left(\exp\left(\frac{\frac{\gamma}{\sqrt{n}m}s_{\mathrm{I}}^2}{2\left(1-c_{\mathrm{I}}\frac{\gamma}{nm}\right)}\right)\right) && (79)\\
&\leq \exp\left(\frac{n(\frac{\lambda}{n})^2 s_{\mathrm{II}}^2}{2\left(1-c_{\mathrm{II}}(\frac{\lambda}{n})\right)}\frac{\sqrt{n}}{\lambda}\right)\cdot\left(\exp\left(\frac{\frac{\gamma}{\sqrt{n}m}s_{\mathrm{I}}^2}{2\left(1-c_{\mathrm{I}}\frac{\gamma}{nm}\right)}\right)\right) && (80)\\
&\leq \exp\left(\frac{\lambda s_{\mathrm{II}}^2}{2\sqrt{n}\left(1-(c_{\mathrm{II}}\lambda)/n\right)}\right)\cdot\left(\exp\left(\frac{\sqrt{n}\beta s_{\mathrm{I}}^2}{2m\left(1-(c_{\mathrm{I}}\beta)/m\right)}\right)\right) && (81)\\
& && (82)
\end{aligned}$$

This holds $\forall\gamma\in(0,1/c_{\mathrm{I}})$, $\forall\lambda\in(0,1/c_{\mathrm{II}})$ and with $\gamma = n\beta$. Returning to (50) and similar to Case I:

$$\frac{1}{\gamma}\Gamma^{\mathrm{I}}(\gamma)+\frac{1}{\lambda}\Gamma^{\mathrm{II}}(\lambda)=\frac{1}{\sqrt{n}}\left(\frac{\sqrt{n}}{\gamma}\Upsilon^{\mathrm{I}}(\gamma)+\frac{\sqrt{n}}{\lambda}\Upsilon^{\mathrm{II}}(\lambda)\right) \tag{83}$$

$$\leq\frac{1}{\sqrt{n}}\ln\mathbb{E}\left[e^{\frac{\sqrt{n}}{\gamma}\Upsilon^{\mathrm{I}}(\gamma)+\frac{\sqrt{n}}{\lambda}\Upsilon^{\mathrm{II}}(\lambda)}\right]-\frac{1}{\sqrt{n}}\ln\delta \tag{84}$$

$$\leq\frac{1}{\sqrt{n}}\ln\left[\exp\left(\frac{\lambda s_{\mathrm{II}}^2}{2\sqrt{n}\,(1-(c_{\mathrm{II}}\lambda)/n)}\right)\cdot\left(\exp\left(\frac{\sqrt{n}\beta s_{\mathrm{I}}^2}{2m\,(1-c_{\mathrm{I}}\beta/m)}\right)\right)\right]-\frac{1}{\sqrt{n}}\ln\delta \tag{85}$$

$$=\frac{\lambda s_{\mathrm{II}}^2}{2n\,(1-c_{\mathrm{II}}(\lambda/n))}+\frac{\beta s_{\mathrm{I}}^2}{2m\,(1-(c_{\mathrm{I}}\beta)/m)}-\frac{1}{\sqrt{n}}\ln\delta. \tag{86}$$

This concludes the proof of Theorem 2.3. □

2.2.2 Proof of Corollary 2.4

Proof. Corollary 2.4 states the PAC bound for the transfer error if the Gibbs posterior is used. In choosing the optimal Gibbs posterior $Q_i^*(h)=Q^*(h;S_i,P)=P(h)\,\exp(-h\hat{\mathcal{L}}(h,S_i))/Z_\beta(S_i,P)$ we can rewrite the terms involving the posterior or base learner in (27) as follows.

$$\hat{\mathcal{L}}_{\text{multi}}(\mathcal{Q},S_1,...,S_n)+\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\mathbb{E}_{P\sim\mathcal{Q}}[D_{KL}(Q_i^*||P)] \tag{87}$$

$$=\frac{1}{n}\sum_{i=1}^{n}\left(\mathbb{E}_{P\sim\mathcal{Q}}\,\mathbb{E}_{h\sim Q_i^*}\left[\hat{\mathcal{L}}(h,S_i)\right]+\frac{1}{\beta}\mathbb{E}_{P\sim\mathcal{Q}}[D_{KL}(Q_i^*||P)]\right) \tag{88}$$

$$=\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\left(\mathbb{E}_{P\sim\mathcal{Q}}\,\mathbb{E}_{h\sim Q_i^*}\left[\beta\hat{\mathcal{L}}(h,S_i)+\ln\frac{Q_i^*(h)}{P(h)}\right]\right) \tag{89}$$

$$=\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\left(\mathbb{E}_{P\sim\mathcal{Q}}\,\mathbb{E}_{h\sim Q_i^*}\left[\beta\hat{\mathcal{L}}(h,S_i)+\ln\frac{P(h)\exp(-h\hat{\mathcal{L}}(h,S_i))}{P(h)Z_\beta(S_i,P)}\right]\right) \tag{90}$$

$$=\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\left(-\mathbb{E}_{P\sim\mathcal{Q}}\ln Z_\beta(S_i,P)\right) \tag{91}$$

These simplifications allow us to rewrite (27) in Theorem 2.3 as follows

$$\mathcal{L}_{\text{transfer}}(\mathcal{Q},\mathcal{T})\leq-\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\mathbb{E}_{P\sim\mathcal{Q}}[\ln Z_\beta(S_i,P)]+\left(\frac{1}{\lambda}+\frac{1}{n\beta}\right)D_{KL}(\mathcal{Q}||\mathcal{P})+C(\delta,\lambda,\beta) \tag{92}$$

The terms involving the base learner Q as stated in (88) can be regarded as a loss function. As such it is possible to apply Lemma 2.2 with the conclusion that the Gibbs posterior $Q^*(h;S_i,P)$ is the minimiser of (88). $\forall P\in\mathcal{M}(\mathcal{H})$, $\forall i=1,...,n$:

$$Q_i^*(h)=\frac{P(h)e^{-h\hat{\mathcal{L}}(h,S_i)}}{Z_\beta(S_i,P)}=\underset{Q\in\mathcal{M}(\mathcal{H})}{\operatorname{argmin}}\,\mathbb{E}_{h\sim Q}\left[\hat{\mathcal{L}}(h,S_i)\right]+\frac{1}{\beta}D_{KL}(Q||P). \tag{93}$$

Rewrite (92)

$$\mathcal{L}_{\text{transfer}}(\mathcal{Q},\mathcal{T}) \leq -\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\mathbb{E}_{P\sim\mathcal{Q}}[\ln Z_\beta(S_i,P)] + \left(\frac{1}{\lambda}+\frac{1}{n\beta}\right)D_{KL}(\mathcal{Q}||\mathcal{P}) + C(\delta,\lambda,\beta) \tag{94}$$

$$= -\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{P\sim\mathcal{Q}}\left[\min_{Q\in\mathcal{M}(\mathcal{H})}\hat{\mathcal{L}}_{\text{Gibbs}}(Q,S_i)+\frac{1}{\beta}D_{KL}(Q||P)\right] + \left(\frac{1}{\lambda}+\frac{1}{n\beta}\right)D_{KL}(\mathcal{Q}||\mathcal{P}) + C(\delta,\lambda,\beta) \tag{95}$$

$$\leq \hat{\mathcal{L}}_{\text{multi}}(\mathcal{Q},S_1,...,S_n) + \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{P\sim\mathcal{Q}}\left[\hat{\mathcal{L}}_{\text{Gibbs}}(Q,S_i)+\frac{1}{\beta}D_{KL}(Q||P)\right] + \left(\frac{1}{\lambda}+\frac{1}{n\beta}\right)D_{KL}(\mathcal{Q}||\mathcal{P}) + C(\delta,\lambda,\beta), \tag{96}$$

which shows that the bound for Gibbs optimal base learners in (32) and (92) is tighter than (27) which is the bound in Theorem 2.3 which holds uniformly for all $Q \in \mathcal{M}(\mathcal{H})$. □

2.2.3 Proof of Proposition 2.5

Proof. The PAC bound using Gibbs-optimal base learners in (32) . The RHS of this bound can be treated as our objective function and reads

$$J(\mathcal{Q}) = -\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\mathbb{E}_{P\sim\mathcal{Q}}[\ln Z_\beta(S_i,P)] + \left(\frac{1}{\lambda}+\frac{1}{n\beta}\right)D_{KL}(\mathcal{Q}||\mathcal{P}) \tag{97}$$

$$= -\mathbb{E}_{P\sim\mathcal{Q}}\left[\frac{\lambda}{n\beta+\lambda}\sum_{i=1}^{n}\ln Z(S_i,P)\right] + D_{KL}(\mathcal{Q}||\mathcal{P}), \tag{98}$$

where all additive terms from (32) have been omitted that do not depend on $\mathcal{Q}$. In the second step we multiply with the scaling factor $\frac{\lambda n\beta}{n\beta+\lambda}$. As all the described transformations are monotone the minimising distribution of $J(\mathcal{Q})$ given by

$$\mathcal{Q} = \operatorname*{argmin}_{\mathcal{Q}\in\mathcal{M}(\mathcal{M}(\mathcal{H}))} J(\mathcal{Q}), \tag{99}$$

is also the minimiser of (32) which we seek. The objective function $J(\mathcal{Q})$ is a function of the hyper-posterior. The problem at hand looks structurally similar to Lemma 2.2. In order to apply it we need to specify the loss function etc. on the meta-level. As such take $A = \mathcal{M}(\mathcal{H})$, $g(a) = -\sum_{i=1}^{n}\ln Z(S_i,P)$, $\beta = \frac{1}{\sqrt{n\tilde{m}}+1}$ to show that the optimal hyper-posterior is

$$\mathcal{Q}^*(P) = \frac{\mathcal{P}(P)\exp\left(\frac{\lambda}{n\beta}\sum_{i=1}^{n}\ln Z_\beta(S_i,P)\right)}{Z_\beta^{II}(S_1,...,S_n,\mathcal{P})} \tag{100}$$

where $Z_\beta^{II}(S_1,...,S_n,\mathcal{P}) = \mathbb{E}_{P\sim\mathcal{P}}\left[\exp\left(\frac{\lambda}{n\beta}\sum_{i=1}^{n}\ln Z_\beta(S_i,P)\right)\right]$. □

2.3 Optimisation-Based Meta-Learning Revisited in the PAC-Bayesian Framework

In this section some shortfalls of the previously introduced PAC-Bayesian framework in the few-shot setting will be addressed. The PAC-bounds introduced so far do not yield good results for small test datasets if the training datasets are large. This exposes a gap in the lack of explainability of the current PAC-Bayesian theoretical framework of meta-learning and the very impressive performance of recent optimisation-based meta-learning algorithms. As such the previous bounds will be altered and then be used in order to derive the MAML and REPTILE algorithms.

As mentioned already and pointed out in Ding et al. [10] one potential drawback of the PAC-bound in Theorem 2.3 is the assumption that the number of training samples m_i from the observed tasks τ_i and the number of training examples m for the target task τ are drawn from the same distribution, $\mathbb{E}_{\mathcal{T}}[m_i] = \mathbb{E}_{\mathcal{T}}[m]$. This can be problematic as in most practical applications one would like to leverage larger training datasets in order to compensate for smaller training sample sizes. This is also the case in meta-learning benchmark challenges such as Imagenet [41] or BERT [9]. As discussed underneath the statement of Theorem 2.3 the PAC-bound is only able to produce a loose bound of $O(\frac{1}{m})$ which is not very effective for small sample sizes. This shows how the PAC-Bayesian meta-learning framework is unable to explain the generalisation performance of meta-learning algorithms (such as MAML and REPTILE) reported in practice.

In order to tackle this problem one can treat the target and observed task environments as different, with the target environment $\mathcal{T}$ and the training tasks $(\mathcal{D}_i, m_i) \sim \tilde{\mathcal{T}}$ drawn from the training environment.

Theorem 2.7 (Ding et al. [10]) *For a target task environment $\mathcal{T}$ and an observed task environment $\tilde{\mathcal{T}}$ where $\mathbb{E}_{\tilde{\mathcal{T}}}[\mathcal{D}] = \mathbb{E}_{\mathcal{T}}[\mathcal{D}]$ and $\mathbb{E}_{\tilde{\mathcal{T}}}[m] \geq \mathbb{E}_{\mathcal{T}}[m]$, let $\mathcal{P}$ be a fixed hyper-prior and $\lambda > 0$, $\beta > 0$, then with probability at least $1-\delta$ over samples $S_1 \in \mathcal{D}_1^{m_1}, ..., S_n \in \mathcal{D}_n^{m_n}$ where $(\mathcal{D}_i, m_i) \sim \tilde{\mathcal{T}}$, we have, for all base learner Q and hyper-posterior $\mathcal{Q}$*

$$\mathcal{L}_{transfer}(\mathcal{Q}, \mathcal{T}) \leq \hat{\mathcal{L}}_{multi}(\mathcal{Q}, S_1, ..., S_n) + \left(\frac{1}{\lambda} + \frac{1}{n\beta}\right) D_{KL}(\mathcal{Q}||\mathcal{P}) + \frac{1}{n}\sum_{i=1}^{n} \frac{1}{\beta} \mathbb{E}_{P\sim\mathcal{Q}}[D_{KL}(Q(h; S_i, P)||P)] + C(\delta, \lambda, \beta, n, m_i) + \Delta_\lambda\left(\mathcal{P}, \mathcal{T}, \tilde{\mathcal{T}}\right), \quad (101)$$

where $\Delta_\lambda\left(\mathcal{P}, \mathcal{T}, \tilde{\mathcal{T}}\right) = \frac{1}{\lambda} \ln \mathbb{E}_{P\sim\mathcal{P}}\, e^{\lambda\left(\mathcal{L}_{transfer}(P,\mathcal{T}) - \mathcal{L}_{transfer}(P,\tilde{\mathcal{T}})\right)}$

This bound is very similar to the bound (27) given in Theorem 2.3 except for the additional penalty term $\Delta_\lambda\left(\mathcal{P}, \mathcal{T}, \tilde{\mathcal{T}}\right)$. In case $\mathbb{E}_{\tilde{\mathcal{T}}}[m_i] \gg \mathbb{E}_{\mathcal{T}}[m]$ this decoupling of the training of the training and test environment appears beneficial, as $\tilde{m}$ is bigger than in Theorem 2.3, as a result the error asymptotics $O(\frac{1}{\tilde{m}})$ are smaller. Regrettably the additional term $\Delta_\lambda\left(\mathcal{P}, \mathcal{T}, \tilde{\mathcal{T}}\right)$ becomes larger with an increasing $\mathbb{E}_{\tilde{\mathcal{T}}}[\tilde{m}]$. A possible way to mitigate this problem is by considering only a subset $S_i' \sim \mathcal{D}^{m_i'}$ from S_i where m and m_i' follow the same distribution and $m_i' \leq m_i$. Then only this subset S_i' is used to train the base learner $Q(h; S_i', P)$. At the same time all examples of $S_i \sim \mathcal{D}_i^{m_i}$ are used for

evaluating the empirical Gibbs error $\hat{\mathcal{L}}_{\text{Gibbs}}(Q(h; S_i', P), S_i)$, so that the larger m_i in the empirical Gibbs error $\hat{\mathcal{L}}_{\text{Gibbs}}(Q(h; S_i', P), S_i)$ help tighten the generalisation gap. This revised strategy leads to the following bound.

Theorem 2.8 (Ding et al. [10]) *For a target task environment $\mathcal{T}$ and an observed task environment $\tilde{\mathcal{T}}$ where $\mathbb{E}_{\tilde{\mathcal{T}}}[\mathcal{D}] = \mathbb{E}_{\mathcal{T}}[\mathcal{D}]$ and $\mathbb{E}_{\tilde{\mathcal{T}}}[m] \geq \mathbb{E}_{\mathcal{T}}[m]$, let $\mathcal{P}$ be a fixed hyper-prior and $\lambda > 0$, $\beta > 0$, then with probability at least $1 - \delta$ over samples $S_1 \in \mathcal{D}_1^{m_1}, ..., S_n \in \mathcal{D}_n^{m_n}$ where $(\mathcal{D}_i, m_i) \sim \tilde{\mathcal{T}}$, and subsamples $S_1' \in \mathcal{D}_1^{m_1'} \subset S_1, ..., S_n' \in \mathcal{D}_n^{m_n'} \subset S_n$, where $\mathbb{E}[m_i'] = \mathbb{E}_{\mathcal{T}}[m_i]$ we have, for all base learner Q and hyper-posterior $\mathcal{Q}$*

$$\begin{aligned}\mathcal{L}_{transfer}(\mathcal{Q}, \mathcal{T}) \leq \mathbb{E}_{P\sim\mathcal{Q}}\left[\frac{1}{n}\sum_{i=1}^{n}\hat{\mathcal{L}}_{Gibbs}(Q(h; S_i', P), S_i)\right] + \left(\frac{1}{\lambda} + \frac{1}{n\beta}\right) D_{KL}(\mathcal{Q}||\mathcal{P}) \\ + \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\mathbb{E}_{P\sim\mathcal{Q}}[D_{KL}(Q(h; S_i', P)||P)] + C(\delta, \lambda, \beta, n, m_i)\end{aligned} \tag{102}$$

This theorem provides tight probabilistic guarantees for small m and can be seen as an improved version of Theorem 2.3. Ding et al. also use it to derive a PAC-Bayesian meta-learning algorithm called PACMAML. The proofs of both theorems are very similar to the one seen in Section 2.2 with the three steps of bounding the task-level, the meta-level and then combining both of these. The first step of bounding the task specific generalisation is in fact identical to the proof of Theorem 2.3. Some changes need to be made for the meta-level and correspondingly for step 3, but the overall structure of the proof is the same. It is important to note that Theorem 2.3 can be modified depending on the setting and opens up a whole new class of meta-learning PAC-Bayesian bounds and algorithms which are derived from those.

One major limitation of the work stated in this essay is that Theorems 2.3, 2.7 and 2.8 not take into account a data domain shift (e.g. Germain et al. [16]). The target and training environments are assumed to have the same underlying data generating mechanisms, which is not often the case in practice. Altering these bounds to take this shift into account requires additional assumptions about the target environments and is an important area of current research. In Section 4 we will see how the PACOH-NN algorithm performance can deteriorate significantly in the presence of distribution shift.

2.3.1 Derivation of MAML and REPTILE

The Theorems 2.7 and 2.8 serve not just as tight bounds in the setting of fewer training data samples, but they also can be used to derive the MAML and REPTILE algorithms. This is great as it provides us with a theoretical justification for these algorithms and highlights the broader context in which these popular meta-learning algorithms exist.

The following derivation is heavily based on the one presented in Ding et al. [10]. We start by considering the maximum-a-posteriori (MAP) approximations of the base-learners $Q_i(h)$, $i = 1, .., n$

and hyper-posterior $\mathcal{Q}(P)$ with Dirac (point) measures. Moreover, the isotropic Gaussian priors with variance parameters σ_0^2 and σ^2 for the hyper-prior $\mathcal{P}(P)$ and the prior $P(h)$ are chosen. The hypothesis is parameterised by a vector θ. To summarise

$$\begin{aligned}
\mathcal{P}(P) &= \mathcal{N}(\mathbf{p}|0, \sigma_0^2 I_k), \\
\mathcal{Q}(P) &= \delta(\mathbf{p} = \mathbf{p}_0), \\
P(P) &= \mathcal{N}(\mathbf{h}|\mathbf{p}, \sigma^2 I_k), \\
Q_i(h) &= \delta(\mathbf{h} = \mathbf{q}_i).
\end{aligned}$$

It is reasonable that Dirac measures for the hyper-posterior $\mathcal{Q}$ and base-learner Q_i are used, since MAML and REPTILE are used for point-estimate neural networks, as illustrates later in Figure 4 part (a). The goal of the MAP approximation is to find the optimal meta-parameters $\mathbf{p}_0$. These distributions allow us to write the two KL-divergence terms in the PAC-bounds (101) and (102) up to a constant as follows

$$\begin{aligned}
D_{KL}(\mathcal{Q}, \mathcal{P}) &= \int d\mathbf{p}\, \delta(\mathbf{p} = \mathbf{p}_0) \ln \frac{\delta(\mathbf{p} = \mathbf{p}_0)}{\mathcal{N}(\mathbf{p}|0, \sigma_0^2 I_k)} \\
&= \int d\mathbf{p}\, \delta(\mathbf{p} = \mathbf{p}_0) \left(\ln \delta(\mathbf{p} = \mathbf{p}_0) - \ln \mathcal{N}(\mathbf{p}|0, \sigma_0^2 I_k) \right) \\
&= \int d\mathbf{p}\, \delta(\mathbf{p} = \mathbf{p}_0) \left(\ln \delta(\mathbf{p} = \mathbf{p}_0) - \ln \left(\frac{1}{\sqrt{(2\pi)^k \det(\sigma_0^2 I_k)}} e^{-\frac{1}{2}(\mathbf{p}-0)^T \frac{1}{\sigma_0^2} I_k (\mathbf{p}-0)} \right) \right) \\
&= \int d\mathbf{p}\, \delta(\mathbf{p} = \mathbf{p}_0) \left(\ln \delta(\mathbf{p} = \mathbf{p}_0) - \ln \left(\frac{1}{\sqrt{(2\pi\sigma_0^2)^k \det(I_k)}} e^{-\frac{1}{2}\mathbf{p}^T \frac{1}{\sigma_0^2} I_k \mathbf{p}} \right) \right) \\
&= \int d\mathbf{p}\, \delta(\mathbf{p} = \mathbf{p}_0) \left(\ln \delta(\mathbf{p} = \mathbf{p}_0) + \frac{||\mathbf{p}^2||}{2\sigma_0^2} + \frac{k}{2} \ln(2\pi\sigma_0^2) \right) \\
&= \frac{||\mathbf{p}_0^2||}{2\sigma_0^2} + \frac{k}{2} \ln(2\pi\sigma_0^2) + c,
\end{aligned}$$

where $\int d\mathbf{p}\, \delta(\mathbf{p} = \mathbf{p}_0) \ln \delta(\mathbf{p} = \mathbf{p}_0) = 0$, k the dimension of $\mathbf{p}$ and c just the integration constant for an indefinite integral. Similarly,

$$\begin{aligned}
\mathbb{E}_{P\sim\mathcal{Q}}[D_{KL}(Q_i||P)] &= \int d\mathbf{p}\, \delta(\mathbf{p} = \mathbf{p}_0) \int d\mathbf{h}\, \delta(\mathbf{h} = \mathbf{q}_i) \ln \frac{\delta(\mathbf{h} = \mathbf{q}_i)}{\mathcal{N}(\mathbf{h}|\mathbf{p}, \sigma^2 I_k)} \\
&= \int d\mathbf{p}\, \delta(\mathbf{p} = \mathbf{p}_0) \int d\mathbf{h}\, \delta(\mathbf{h} = \mathbf{q}_i) \left(\frac{||\mathbf{h} - \mathbf{p}||^2}{2\sigma^2} + \frac{k}{2} \ln(2\pi\sigma^2) + \ln \delta(\mathbf{h} = \mathbf{q}_i) \right) \\
&= \int d\mathbf{p}\, \delta(\mathbf{p} = \mathbf{p}_0) \left(\frac{||\mathbf{q}_i - \mathbf{p}||^2}{2\sigma^2} + \frac{k}{2} \ln(2\pi\sigma^2) \right) \\
&= \frac{||\mathbf{p}_0 - \mathbf{q}_i||^2}{2\sigma^2} + \frac{k}{2} \ln(2\pi\sigma^2) + c',
\end{aligned}$$

where we again substituted the definition of a normal density function in the second equality. With all of these assumptions it is possible to write the RHS of PAC-Bayesian bound (PacB) in equation

(101) and equation (102) as follows

$$PacB(\mathbf{p}_0) = \frac{1}{n}\sum_{i=1}^{n} \hat{\mathcal{L}}_{Gibbs}(\mathbf{q}_i, S_i) + \frac{\tilde{\xi}||\mathbf{p}_0||^2}{2\sigma_0^2} + \frac{1}{n\beta}\sum_{i=1}^{n}\frac{||\mathbf{p}_0 - \mathbf{q}_i||^2}{2\sigma^2} + C', \tag{103}$$

where $\tilde{\xi} = \frac{1}{\lambda} + \frac{1}{n\beta}$ and C' is a constant the exact value of which depends on the bound used. Since we will be minimising the expression the exact value of the constant is not important and will not be computed. The base-learner $Q_i = \delta(\mathbf{h} = \mathbf{q}_i)$ is determined by its parameter vector $\mathbf{q}_i$, which can be a function of $\mathbf{p}_0$ and S_i for equation (101) (or $\mathbf{p}_0$ and S_i' for equation (102)). As such the only free variable in equation (103) is $\mathbf{p}_0$. One approach to find the MAP estimator of $PacB$ is through gradient descent on $\mathbf{p}_0$.

In case $\mathbf{q}_i = \mathbf{p}_0$ the gradient of equation (103) reduces to that of multi-task pre-training.

$$\lim_{\mathbf{q}_i \to \mathbf{p}_0} \frac{d(PacB)}{d\,\mathbf{p}_0} = \frac{\tilde{\xi}\mathbf{p}_0}{\sigma_0^2} + \frac{1}{n}\sum_{i=1}^{n}\frac{d}{d\,\mathbf{p}_0}\hat{\mathcal{L}}_{\text{Gibbs}}(\mathbf{p}_0, S_i) \tag{104}$$

Note that in equation (101) for a given $\mathbf{p}_0$ and S_i, there exists an optimal Dirac-base-learner $\mathbf{q}_i^*$ in the form

$$\mathbf{q}_i^* = \underset{\mathbf{q}_i}{\text{argmin}} \left[\hat{\mathcal{L}}_{Gibbs}(\mathbf{q}_i, S_i) + \frac{||\mathbf{p}_0 - \mathbf{q}_i||^2}{2\sigma^2}\right] \tag{105}$$

Given this optimal $\mathbf{q}_i^*$ the full derivative of $PacB$ with respect to $\mathbf{p}_0$ is given by

$$\frac{d(PacB)}{d\mathbf{p}_0} = \frac{\partial(PacB)}{\partial\mathbf{p}_0} + \left\langle \frac{\partial\mathbf{q}_i^*}{\partial\mathbf{p}_0}, \frac{\partial(PacB)}{\partial\mathbf{q}_i^*} \right\rangle \tag{106}$$

$$= \frac{\partial(PacB)}{\partial\mathbf{p}_0} \tag{107}$$

$$= \frac{\tilde{\xi}\mathbf{p}_0}{\sigma_0^2} + \frac{1}{n}\sum_{i=1}^{n}\frac{\mathbf{p}_0 - \mathbf{q}_i^*}{\beta\sigma^2}, \tag{108}$$

where the second equation is true because $\frac{\partial(PacB)}{\partial\mathbf{q}_i^*} = 0$ for the optimal base-learner $\mathbf{q}_i^* = 0$. The RHS of equation (108) can be rewritten as follows

$$\frac{\tilde{\xi}}{\sigma_0^2}\mathbf{p}_0 + \frac{1}{\beta\sigma^2}\left(\mathbf{p}_0 - \frac{1}{n}\sum_{i=1}^{n}\mathbf{q}_i^*\right) \tag{109}$$

This is equivalent to the meta-update of the REPTILE algorithm developed by Nichol et al. [30] and detailed in equation (6) and Algorithm 2, except that REPTILE does not solve for the optimal base learner $\mathbf{q}_i^*$, but whose inner loop in an approximate algorithm for the optimal Dirac-base-learner.

From the optimal condition, the base-learner $\mathbf{q}_i^*$ satisfies

$$\frac{\mathbf{p}_0 - \mathbf{q}_i^*}{\beta\sigma^2} = \nabla_{\mathbf{q}_i^*}\mathcal{L}_{\text{Gibbs}}(\mathbf{q}_i^*, S_i). \tag{110}$$

Equation (108) can now be rewritten in the form of an *implicit gradient*

$$\frac{d(PacB)}{d\mathbf{p}_0} = \frac{\tilde{\xi}\mathbf{p}_0}{\sigma_0^2} + \frac{1}{n}\sum_{i=1}^{n} \nabla_{\mathbf{q}_i^*} \mathcal{L}_{\text{Gibbs}}(\mathbf{q}_i^*, S_i) \tag{111}$$

In contrast the standard multi-task objective used the *explicit gradient*, with $\mathbf{q}_i^* = \mathbf{p}_0$ and

$$\frac{d(PacB)}{d\mathbf{p}_0} = \frac{\tilde{\xi}\mathbf{p}_0}{\sigma_0^2} + \frac{1}{n}\sum_{i=1}^{n} \nabla_{\mathbf{p}_0} \mathcal{L}_{\text{Gibbs}}(\mathbf{p}_0, S_i) \tag{112}$$

Lastly in case $\mathbf{q}_i$ is a few gradient steps away from $\hat{\mathcal{L}}(\mathbf{q}_i, S_i')$ with initialisation $\mathbf{q}_i = \mathbf{p}_0$, the gradient of equation (103) reduces to the meta-update of the MAML algorithm as outlined in equation (4) and Algorithm 1 as $\sigma^2 \to \infty$ [3].

$$\lim_{\sigma^2 \to \infty} \frac{d(PacB)}{d\,\mathbf{p}_0} = \frac{\tilde{\xi}\mathbf{p}_0}{\sigma_0^2} + \frac{1}{n}\sum_{i=1}^{n} \frac{d}{d\,\mathbf{p}_0} \hat{\mathcal{L}}_{\text{Gibbs}}(\mathbf{q}_i, S_i) \tag{113}$$

So the MAML meta-update rule corresponds to a flat prior P and MAP estimates of the hyper-posterior $\mathcal{Q}$ & base-learner Q. One thing to keep in mind is that $\mathbf{q}_i = f(\mathbf{p}_0, d\mathbf{q}_i/d\mathbf{p}_0)$ is a function of $\mathbf{p}_0$ and $d\mathbf{q}_i/d\mathbf{p}_0)$ so it involves higher-order derivatives with respect to $\mathbf{p}_0$ which results in computationally more intensive algorithms. This was already pointed out at the end of Section 1.2. The derivation of the meta-update rules of MAML & REPTILE from the PAC-Bayesian bounds in Theorem 2.7 and Theorem 2.8 shows that the meta-learning framework introduced in this section is not some arbitrary theoretical construction, but rather a foundation and extension of the impressive empirical performance exhibited by optimisation-based meta-learning algorithms. This should increase our confidence in these bounds and the algorithms derived from them in the next section.

[3] As mentioned by Ding et al. [10], a noteworthy difference is that the MAML algorithm assumes $S_i \cap S_i' = \emptyset$, while in the presented setting we assumes $S_i' \subset S_i$. However, Theorem 2.8 is still valid when $S_i \cap S_i = \emptyset$

3 The PACOH Algorithm

In Section 1 the general Meta-Learning paradigm was introduced. In addition it was shown how prominent optimisation-based algorithms like MAML & REPTILE can approach these types of problems in a more efficient way than vanilla machine learning techniques. Section 2 reviewed the basics of PAC-Bayesian Theory and showed how the expected error of a machine learning system can be bounded through its empirical test error with large probability. Following Rothfuss et al. [39] and predecessors these ideas are extended to a Meta-Learning framework. The majority of the previous section is concerned with stating the PAC-bound in Theorem 2.3 and then proving the closed form solution of its tightest version. The PACOH algorithm approximates this closed form solution of the hyper-posterior to meta-learn a prior for the base-learner, which is then deployed on the target task.

Even though this section is mainly concerned with the construction of the PACOH algorithm from Rothfuss et al. [40] based on the minimisation of the bound in Theorem 2.3. Understanding this procedure allows to derive a whole new class of algorithms which are build from PAC-Bayesian meta-learning bounds. For example the PACMAML algorithm can be derived from Theorem 2.8 following Ding et al. [10]. In order to understand the PACOH algorithm let us consider again the PAC-Bayesian framework introduced in Section 2.2 and visualised in Figure 3. The inputs of the Meta-learning system are the datasets $S_1, ..., S_n$ from the n training tasks and a hyper-prior $\mathcal{P} \in \mathcal{M}(\mathcal{M}(\mathcal{H}))$, which is a probability distribution over priors. The Meta-learner then outputs a hyper-posterior $\mathcal{Q} \in \mathcal{M}(\mathcal{M}(\mathcal{H}))$, which is again a probability distribution over priors. Proposition 2.5 provides us with a closed-form solution to the PAC-optimal Hyper-Posterior (PACOH) which is given by

$$\mathcal{Q}^*(P) = \frac{\mathcal{P}(P) \exp\left(\frac{\lambda}{n\beta+\lambda} \sum_{i=1}^n \ln Z_\beta(S_i, P)\right)}{Z^{II}(S_1, ..., S_n, \mathcal{P})},$$

with $Z^{II}(S_1, ..., S_n, \mathcal{P}) = \mathbb{E}_{P\sim\mathcal{P}}\left[\exp\left(\frac{\lambda}{n\beta+\lambda} \sum_{i=1}^n \ln Z_\beta(S_i, P)\right)\right]$.

Unfortunately the normalisation constant $Z^{II}(S_1, ..., S_n, \mathcal{P})$ cannot be calculated in practice because of the high-dimensional expectation, so $\mathcal{Q}^*(P)$ is only known up to a constant. Assuming we can calculate the $\ln Z_\beta(S_i, P)$ terms, the first step of the algorithm is to compute the hyper-posterior. During target training and target testing the algorithm draws a prior for the base-learner from $\mathcal{Q}^*(P)$. As discussed in the previous section if the negative log-likelihood is chosen as a loss function the optimal Gibbs posterior becomes the generalised Bayesian posterior. Choices for base-learners discussed here are the Bayesian models Gaussian Processes (GPs) and Bayesian Neural Networks (BNNs), yielding the PACOH-GP and PACOH-NN algorithms respectively. In essence the PACOH algorithm meta-learns a GP or BNN prior and then deploys that model on the target task.

3.1 Approximating the PACOH

In order to build a practical meta-learning algorithm when employing BNNs or GPs we assume a parametric family of priors $\{P_\phi | \ \phi \in \Phi\}$. So each prior P_ϕ is fully determined by its parameter

vector $\phi \in \Phi$. The exact choice for the parameter space Φ varies depending on the base-learner, so will be discussed later. Given a parametric model like this, the hyper-prior and -posterior become distribution over the parameter space, i.e. $\mathcal{P} \in \mathcal{M}(\Phi)$ and $\mathcal{Q} \in \mathcal{M}(\Phi)$. How exactly the priors P_ϕ are parameterised and the MLL functions $\ln Z_\beta(S_i, P)$ calculated depends on which Bayesian model is chosen as a base-learner. Both of these questions will be discussed in the following two sections fore the cases of GPs and BNNs.

Given the hyper-prior $\mathcal{P}$ and generalised marginal-log-likelihood (MLL) functions $\ln Z_\beta(S_i, P)$ we can compute the PACOH $\mathcal{Q}^*(P)$ using the definition in (34), up to the normalisation constant $Z^{II}(S_1, ..., S_n, \mathcal{P})$. The problem of trying to compute a posterior which is only known up a constant is well established in Bayesian statistics and as such we will use three methods for approximate inference such as Maximum A Posteriori, Variational Inference and Stein Variational Gradient Descent.

Maximum A Posteriori (MAP) This somewhat simplest method approximates the posterior as a Dirac-function by attempting to find its largest mode. $\phi^* = \arg\max_{\phi \in \Phi} Q^*(\phi)$

Variational Inference (VI) Following Blei et al. [6] VI turns an inference problem into an optimisation problem. It does so through restricting the space of considered hyper-posteriors to a parametric variational family $\mathcal{F} = \{\tilde{\mathcal{Q}}_v |\ v \in \mathcal{U}\} \subset \mathcal{M}(\Phi)$ where the variational posterior $\tilde{\mathcal{Q}}_v$ is fully determined by its parameter vector $v \in \mathcal{U}$. VI then minimises the 'distance'[4] or KL-divergence within this variational family and the PACOH

$$v^* = \arg\min_{v \in \mathcal{U}} D_{KL}(\tilde{\mathcal{Q}}_v || \mathcal{Q}^*) \tag{114}$$

The KL divergence allows us to pull out the normalisation constant. This means that the Z^{II} is just an additive constant and can be disregarded in the optimisation process of VI.

Stein Variational Gradient Descent (SVGD) First introduced by Liu and Wang [27] SVGD is a powerful algorithm which approximates the posterior through an ensemble of K particles which are updated based on functional gradient descent which minimise the KL-divergence. As such, the approximation exhibits the form $\tilde{\mathcal{Q}} = \frac{1}{K} \sum_{k=1}^{K} \delta(\phi_k)$. The particles are initialised by sampling from the hyper-prior $\phi_k \sim \mathcal{P}$. All particles are encoded in one $K \times dim(\phi)$ matrix $\boldsymbol{\phi} = [\phi_1, ..., \phi_K]$. SVGD transports this set of points to match the target distribution. It does so through functional gradient descent on $D_{KL}(\tilde{\mathcal{Q}} || \mathcal{Q}^*)$ in the reproducing kernel Hilbert space induced by a kernel function $k(\cdot, \cdot)$. The particle matrix $\boldsymbol{\phi}$ is updated as follows

$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \eta \boldsymbol{K} \nabla_{\boldsymbol{\phi}} \ln \mathcal{Q}^* + \nabla_{\boldsymbol{\phi}} \boldsymbol{K}, \tag{115}$$

with $\nabla_{\boldsymbol{\phi}} \mathcal{Q}^* = [\nabla_{\phi_1} \mathcal{Q}^*(\phi_1), ..., \nabla_{\phi_K} \mathcal{Q}^*(\phi_K)]^T$ being a matrix of stacked gradients, the kernel matrix $\boldsymbol{K} = [k(\phi_k, \phi_{k'}]_{k,k'}$ and the step size η of the SVGD update. It is worth noting that because of

[4] The KL-divergence is not a metric on the space of probability measures due to some obvious violated assumptions such as symmetry. However, it can be useful to think of it like a distance.

the logarithm the normalisation constant can be pulled out and then vanishes in the derivative. This method can be interpreted as follows, the $\eta \boldsymbol{K} \nabla_{\boldsymbol{\phi}} \ln \mathcal{Q}^*$ term directs particles that are closer to a mode of the posterior to move in that direction. So this term transports the particles to areas of higher probability. Lastly the Jacobian $\nabla_{\boldsymbol{\phi}} \boldsymbol{K}$ is important because it acts as a repulsive force between the particles. Without it all particles would eventually converge to the maximum of the posterior distribution, which would not capture the shape of Q^* very well.

An overview of these methods and their update rules can be found in Appendix A.2

3.2 Meta-Learning Gaussian Process Priors

In the derivation of the PAC-optimal hyper-posterior a Gibbs posterior $Q^*(h)$ was assumed to be the base-learner $Q : \mathcal{Z}^m \times \mathcal{M}(\mathcal{H}) \to \mathcal{M}(\mathcal{H})$. As discussed in Section 2.1 the Gibbs posterior coincides with the generalised Bayesian posterior if the negative log-likelihood $l(h, z_i) \coloneqq -\log p(z_i|h)$ is used as a loss function. In this Section we are concerned with the case that the Bayesian model Gaussian Process (GP) with negative log-likelihood is chosen as a base-learner. The posterior of the GP then coincides with the Gibbs posterior and moreover the GP model can be used for inference.

GPs are common tools in Bayesian Machine Learning and offer some closed-form expressions which will prove to be useful. Gaussian Processes are continuous time stochastic process $\{X_t : t \in \mathbb{R}_{>0}\}$ where for all finite number of indices $t_1 < ... < t_k$,

$$(X_{t_1}, ..., X_{t_k})^\top \tag{116}$$

is a multivariate Gaussian random variable. This concept is then used in Bayesian Machine Learning where GPs are regarded as distributions over functions $f(x)$ specified by a mean function $m(x)$ and a positive definite covariance function (kernel) $k(x, x')$, here x are the function values and (x, x') are all possible pairs with input domain $x' \in \mathcal{X}$. Then write

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')). \tag{117}$$

For all finite subsets of the input domain $\{x_1, ..., x_n\} \subset \mathcal{X}$, which is the test data, the marginal distribution is a multivariate Gaussian

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} = \mathcal{N} \left(\begin{bmatrix} m(x_1) \\ \vdots \\ m(x_n) \end{bmatrix}, \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & ... & k(x_n, x_n) \end{bmatrix} \right) \tag{118}$$

with mean vector $\mu = (m(x_1), ..., m(x_n))^T$ and covariance matrix $\Sigma_{i,j} = k(x_i, x_j)$. For a more comprehensive assessment of general Gaussian Processes the interested reader is referred to Rasmussen et al. [35].

Setup In the GP setup the data points are tuples $z_{i,j} = (x_{i,j}, y_{i,j})$. For the i-th dataset write $S_i = (\boldsymbol{X}_i, \boldsymbol{y}_i)$ with $\boldsymbol{X}_i = (x_{i,1}, ..., x_{i,m_i})^T$ and $\boldsymbol{y}_i = (y_{i,1}, ..., y_{i,m_i})^T$. As GPs are Bayesian methods

they require a prior $P_\phi(h) = \mathcal{GP}(h|\ m_\phi(x), k_\phi(x, x'))$ for which is determined by the mean function $m_\phi : \mathcal{X} \to \mathbb{R}$ and kernel $k_\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, for $\phi \in \Phi$. The PAC-Bayesian Meta-Learning framework introduced in Section 2.2 uses a Meta-Learner to learn a hyper-posterior over priors. As such the major component of the PACOH-GP algorithms is to learn a distribution over priors P_ϕ. In order to do that effectively in a computer one needs to parameterise this prior space or the mean & kernel functions. Following Fortuin et al. [14] m_ϕ and k_ϕ are instantiated as neural networks and the parameter vector ϕ is meta-learned. Unfortunately it would not be wise to parameterise k_ϕ directly as a NN because the positive-definiteness conditions of kernels would not be satisfied in general. As such a feature map ψ_ϕ on top of a squared exponential kernel is used, $k_\phi = \frac{1}{2}\exp(-||\psi_\phi(x) - \psi_\phi(x')||_2^2)$. In Rothfuss et al. [39] m_ϕ and ψ_ϕ fully-connected Neural Networks with 4 hidden layers and 32 neurons each were used in correspondence with tanh activation functions. With these definitions if we wanted to sample hypotheses h for input points $x_1, ..., x_n$ from prior P_ϕ, this is done as follows

$$\begin{bmatrix} h(x_1) \\ \vdots \\ h(x_n) \end{bmatrix} = \mathcal{N}\left(\begin{bmatrix} m_\phi(x_1) \\ \vdots \\ m_\phi(x_n) \end{bmatrix}, \begin{bmatrix} k_\phi(x_1, x_1) & \cdots & k_\phi(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k_\phi(x_n, x_1) & ... & k_\phi(x_,x_n) \end{bmatrix} \right) \tag{119}$$

Over the course of meta-training the parameters ϕ are updates to obtain the posterior. The free parameters λ, β in the PAC-Optimal Hyper-Posterior expression (34) are chosen as $\lambda = n$ and $\beta_i = m_i$. The empirical loss under the GP posterior Q^* coincides with the negative log-likelihood of the regression targets $\boldsymbol{y}_i$, i.e. $\hat{\mathcal{L}}(Q^*, S_i) = -\frac{1}{m_i}\ln p(\boldsymbol{y}|\ \boldsymbol{X}_i)$, Rothfuss et al. [40]. Negative log-likelihood loss functions are quite standard in Bayesian models and the pre-factor is useful for scaling tasks of different sample sizes and for some closed form expression later. A generic hyper-prior of a zero-centred, spherical Gaussian $\mathcal{P} = \mathcal{N}(0, \sigma_{\mathcal{P}}^2 I)$. This choice can be motivated by considering that any parameter space can be scaled to mean zero and that all priors are initially treated as uncorrelated.

As usual in GP regression a Gaussian likelihood is assumed $p(\boldsymbol{y}|\ \boldsymbol{h}) = \mathcal{N}(\boldsymbol{y}; h(\boldsymbol{X}_i), \sigma^2 I)$, with the observation noise variance σ^2. The loss function can then be written as follows,

$$\hat{\mathcal{L}}(Q^*, S_i) = -\frac{1}{m_i}\ln \mathcal{N}(\boldsymbol{y}; h(\boldsymbol{X}_i), \sigma^2 I) \tag{120}$$

$$= \frac{1}{m_i}\left(\boldsymbol{y} - h(\boldsymbol{X}_i)\right)^\top \frac{1}{\sigma^2}(\boldsymbol{y} - h(\boldsymbol{X}_i)) - \frac{1}{m_i}\ln(2\pi)^k|\sigma^2\, I| \tag{121}$$

$$= \frac{1}{m_i\sigma^2}\,||\boldsymbol{y} - h(\boldsymbol{X}_i)||^2 - \frac{1}{m_i}\ln(2\pi\sigma^2)^k, \tag{122}$$

with $k = \dim(\boldsymbol{y})$. As mentioned earlier the squared loss is sub-Gaussian. Therefore the i.i.d. realisation of the in-tasks generalisation error $V_{ij}^{\mathrm{I}} := \mathcal{L}\left(h, \mathcal{D}_i\right) - l\left(h_i, z_{i,j}\right)$ and the meta-tasks generalisation error $V_i^{\mathrm{II}} := \mathbb{E}_{(D,S)\sim\mathcal{T}}[\mathcal{L}(Q(P,S),\mathcal{D})] - \mathcal{L}\left(Q\left(P, S_i\right), \mathcal{D}_i\right)$ are both sub-gamma. As a result the tight bound for $C(\delta, \lambda, \beta)$ in equation (29) from Theorem 2.3 applies.

Algorithm The PACOH-GP algorithm consists of two parts. One is the meta-training phase to approximate the PAC-Optimal Hyper-Posterior which is detailed in Algorithm 3. The second part is the meta-testing, where the hyper-posterior is used to sample priors for the GP to be deployed on

the target task. Since the algorithm depends quite a bit on which inference method from Section 3.1 is used to approximate the PACOH $\mathcal{Q}^*$ we will detail the procedure here for SVGD with K particles, as the MAP is simply a special case of this with $K = 1$ and leave the VI to Appendix A.2 and Rothfuss et al. [39]. To keep the notation general we write generic operations, the details of which can be seen in a table which can be found in the Appendix.

The algorithm takes the datasets $S_1, ..., S_n$ from the n tasks and the hyper-prior $\mathcal{P}$ as input. The first step is to do an initial approximation of the PACOH with Init_Approx_Inference(). In SVGD this consists of sampling K particles from the hyper-prior $\tilde{\phi}_k \sim \mathcal{P}$ to obtain an initial estimate $\tilde{\mathcal{Q}} = \frac{1}{K}\sum_{i=1}^{K} \delta(\tilde{\phi}_k)$ of the PACOH $\mathcal{Q}^*$. Each of these K particles is then updated according to SVGD and the gradient information of $\mathcal{Q}^*$. The PAC-Optimal Hyper-posterior for particle k is defined as

$$\mathcal{Q}^*(P_{\phi_k}) = \mathcal{Q}^*(\phi_k) = \frac{\mathcal{P}(P_{\phi_k}) \exp\left(\sum_{i=1}^{n} \frac{1}{m_i+1} \ln Z_{m_i}(S_i, P_{\phi_k})\right)}{Z^{II}(S_1, ..., S_n, \mathcal{P})}, \tag{123}$$

according to (34), choices for λ, β and our parametrisation of the priors. SVGD requires some gradient information of this expression. roughly speaking this means that one would like to see the particles move to regions of larger probability under $\mathcal{Q}^*$. This of course is partially balanced with the repulsive term. In order to compute the SVGD update in (115) we need the gradient of the log of (123) with respect respect to ϕ_k, which is given by

$$\nabla_{\phi_k} \ln \mathcal{Q}^*(\phi_k) = \nabla_{\phi_k} \ln \mathcal{P}(\phi_k) + \sum_{i=1}^{n} \frac{1}{m_i + 1} \nabla_{\phi_k} \ln Z_{m_i}(S_i, P_{\phi_k}). \tag{124}$$

As a result of the logarithm in the SVGD update rule the uncomputable normalisation constant Z^{II} vanishes. The first term in (124) is known as we defined the hyper-prior. In order to calculate the (generalised) marginal log-likelihood (MLL) $\ln Z_\beta(S_i, P_\phi)$ with $\lambda = n$, $\beta_i = m_i$ consider the following.

$$\ln Z_{\beta_i}(S_i, P_\phi) = \ln \int_{\mathcal{H}} P_\phi(h) \left(\prod_{j=1}^{m_i} p(z_{ij}|h)\right)^{\frac{\beta_i}{m_i}} dh \tag{125}$$

$$= \ln \int_{\mathcal{H}} P_\phi(h) \left(p(\boldsymbol{y}|\ \boldsymbol{X}_i, h)\right) dh \tag{126}$$

$$= \ln p(\boldsymbol{y}|\ \boldsymbol{X}_i, \phi). \tag{127}$$

Fortunately this log probability can be computed analytically following Rasmussen et al. [35],

$$\ln p(\boldsymbol{y}|\ \boldsymbol{X}_i, \phi) = -\frac{1}{2}(\boldsymbol{y} - m_{\boldsymbol{X},\phi})^T \tilde{K}_{\boldsymbol{X},\phi}^{-1}(\boldsymbol{y} - m_{\boldsymbol{X},\phi}) - \frac{1}{2}\ln|\tilde{K}_{\boldsymbol{X},\phi}| - \frac{m_i}{2}\ln 2\pi, \tag{128}$$

where $\tilde{K}_{\boldsymbol{X},\phi} = K_{\boldsymbol{X},\phi} + \sigma^2 I$ with the kernel matrix $K_{\boldsymbol{X},\phi} = (k_\phi(x_i, x_j))_{i,j=1}^{m_i}$ and mean vector $m_{\boldsymbol{X},\phi} = (m_\phi(x_1), ..., m_\phi(x_n))^T$. As detailed in Algorithm 3 for each of the n tasks expression (128) is evaluated to obtain the $\ln Z_{m_i}(S_i, P_{\phi_k})$. These terms can then be used to compute $\nabla_{\phi_k} \ln \mathcal{Q}^*(\phi_k)$. However in each iteration the MLL is computed $\mathcal{O}(K \cdot n)$ times which can be costly given that the matrix $\tilde{K}_{\boldsymbol{X},\phi}^{-1}$ needs to be inverted every time, which requires $\mathcal{O}(m_i^3)$ operations. As a result

it is advantageous to consider mini-batching on the task level in order to calculate an estimate $\tilde{\nabla}_\phi \ln \mathcal{Q}^*(\phi)$. In each iteration a mini-batch of $H \leq n$ is datasets $S_1, ..., S_H$ is sampled to form an unbiased estimate of the hyper-posterior score using the following expression.

$$\tilde{\nabla}_\phi \ln \mathcal{Q}^*(\phi) = \nabla_\phi \ln \mathcal{P}(\phi) + \frac{n}{H} \sum_{h=1}^{H} \frac{1}{m_h + 1} \nabla_{\phi_k} \ln Z_{m_h}(S_h, P_{\phi_k}) \tag{129}$$

The approximate hyper-posterior $\tilde{\mathcal{Q}}$ is then updated by applying the SVGD update rule (115) to all K particles, which can be written out for each particle as

$$\phi_k \leftarrow \phi_k + \eta_t \psi^*(\phi) \text{ with } \psi^*(\phi) = \frac{1}{K} \sum_{l=1}^{K} [k(\phi_l, \phi) \nabla_{\phi_l} \ln \mathcal{Q}^*(\phi_l) + \nabla_{\phi_l} k(\phi_l, \phi)]. \tag{130}$$

This process is repeated until the particles converge or until the maximum number of iterations is reached.

Algorithm 3 PACOH-GP - Approximate Inference of $\mathcal{Q}^*$ according to Rothfuss et al. [39]

Require: hyper-prior $\mathcal{P}$, dataset $S_1, ..., S_n$, step-size η

$\tilde{\mathcal{Q}} \leftarrow$ Init_Approx_Inference()
while not converged **do**
  $\{\phi_1, ..., \phi_k\} \leftarrow$ Sample_Prior_Params($\tilde{\mathcal{Q}}$)
  for $k = 1, ..., K$ **do**
    for $i = 1, ..., n$ **do**
      $\ln Z_{m_i}(S_i, P_{\phi_k}) \leftarrow -\frac{1}{2}(\boldsymbol{y}_i - m_{\boldsymbol{X}_i,\phi_k})^T \tilde{K}^{-1}_{\boldsymbol{X}_i,\phi_k}(\boldsymbol{y}_i - m_{\boldsymbol{X}_i,\phi_k}) - \frac{1}{2} \ln |\tilde{K}_{\boldsymbol{X}_i,\phi_k}| - \frac{m_i}{2} \ln 2\pi$
    end for
    $\nabla_{\phi_k} \ln \mathcal{Q}^* \leftarrow \nabla_{\phi_k} \ln \mathcal{P} + \sum_{i=1}^{n} \frac{1}{m_i+1} \nabla_{\phi_k} \ln Z_{m_i}(S_i, P_{\phi_k})$
  end for
  $\tilde{\mathcal{Q}} \leftarrow$ Approx_Inference_Update($\tilde{\mathcal{Q}}, \nabla_{\phi_k} \ln \mathcal{Q}^*, \eta$)
end while

Algorithm 3 returns an estimate of the PAC-Optimal Hyper-posterior $\tilde{\mathcal{Q}}$. In the PAC-Bayesian Meta-learning setup visualised in Figure 3 the base-learner (GP in this case) is presented with a dataset $\tilde{S} = (\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{y}})$ from a previously unseen task $\mathcal{T}$ and prior $P_\phi \sim \tilde{\mathcal{Q}}$ while outputting a GP posterior Q as a result of its inference. The resulting predictive distribution $\hat{p}(y^*|\ x^*, \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{y}}, \phi)$ is again a Gaussian distribution, Rasmussen et al. [35]. $\tilde{\mathcal{Q}}$ is a distribution over priors so the resulting posterior Q of the base learner will depend on the prior $P_\phi \sim \tilde{\mathcal{Q}}$. In order to obtain the predictive distribution we perform an empirical average. Firstly draw a set of priors from the hyper-posterior $\phi_1, ..., \phi_n \sim \tilde{\mathcal{Q}}$ then form the equally weighted mixture

$$\hat{p}(y^*|\ x^*, \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{y}}, \mathcal{Q}) = \mathbb{E}_{\phi \sim \mathcal{Q}} \left[\hat{p}(y^*|\ x^*, \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{y}}, \phi) \right] \approx \frac{1}{n} \sum_{i=1}^{n} \hat{p}(y^*|\ x^*, \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{y}}, \phi_k). \tag{131}$$

As the GP base-learner was chosen, equation (131) corresponds to a mixture of Gaussians and the average prediction is the average of all models under the priors $\phi_1, ..., \phi_n$. PACOH-GP-MAP $K = 1$ is a special case of SVGD and here $\hat{p}(y^*|\ x^*, \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{y}}, \mathcal{Q}) \approx \hat{p}(y^*|\ x^*, \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{y}}, \phi^{\text{MAP}})$ is a single Gaussian.

3.3 Meta-Learning Bayesian Neural Network Priors

In the PACOH-NN variant of the algorithm the base learner $Q : \mathcal{Z}^m \times \mathcal{M}(\mathcal{H}) \to \mathcal{M}(\mathcal{H})$ is again chosen as the Gibbs posterior Q^* with a negative log-likelihood loss function. As previously explained this means that the Gibbs posterior coincides with the generalised Bayesian posterior for a probabilistic model. In this version we will consider the case where this model is a Bayesian Neural Network (BNN). BNNs can be thought of as the Bayesian version of standard fully-connected Neural Networks. Instead of learning a specific configuration of weights and biases they learn a probability distribution over these weights or nodes. As shown in Figure 4 there a few different choices for stochastic NNs in this work we will consider BNNs that learn a probability distribution over the weights and biases, so (c) in Figure 4.

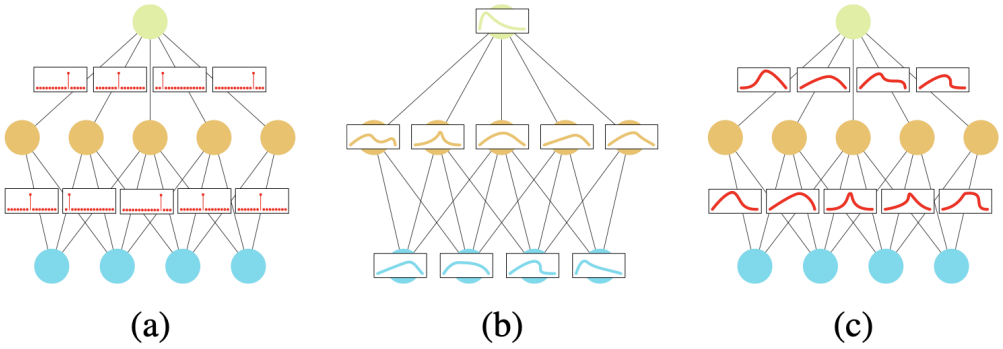


Figure 4: (a) standard Point estimate Neural Network. (b) and (c) are stochastic Neural Networks, where (b) provides learns a probability distribution over nodes and (c) over the weights of the network. This Figure was taken from Jospin et al. [22]

For a more thorough introduction of BNNs the interested reader is referred to the Deep Learning tutorial from Jospin et al. [22]. The main objective of PACOH-NN is to use the training tasks to meta-learn BNN priors. Learning BNN priors in general is challenging due to the large number of dimensions of the weight space and complex mappings between weights and functions as pointed out by Fortuin [13]. The PAC-Bayesian meta-learning setup introduced in Section 2 however is very general and applies to BNNs as well if the negative log-likelihood is used as a loss function.

Setup Following Rothfuss et al. [40], let $h_\theta : \mathcal{X} \to \mathcal{Y}$ be a function parameterised by a Neural Network with weights $\theta \in \Theta$. The BNN then outputs a probability distribution over all of these weights as shown in (c) in Figure 4. Using this NN mapping one can define the conditional distribution for regression again as $p(y|\ x, \theta) = \mathcal{N}(y|\ h_\theta(x), \sigma^2\, I)$. By similar arguments to the GP case the tight bound for $C(\delta, \lambda, \beta)$ in equation (29) from Theorem 2.3 applies as well. Here σ^2 is referred to as the noise variance and $\ln \sigma$ is a learnable parameter. As such an hypothesis consists consists of a tuple $h = (\theta, \ln \sigma)$. The logarithm is used in order to avoid positivity constraint on possible h and simplify the optimisation, since the logarithm is a bijective function this does not impact the chosen σ. In classification the conditional probability is $p(y|\ x, \theta) = \text{Categorical}(\text{softmax}(h_\theta(x)))$. The loss function is chosen as the negative log-likelihood $l(\theta, z) = -\ln p(y|\ x, \theta)$.

As a next step we need to define a family of priors $\{P_\phi : \phi \in \Phi\}$ over the NN parameters $\theta \in \Theta$. For notational convenience these are chosen as Block diagonal Gaussians $P_{\theta_l} = \mathcal{N}(\mu_{P_k}, \text{diag}(\sigma^2_{P_k}))$ with $\phi = (\mu_{P_k}, \ln \sigma_{P_k})$. This choice might seem arbitrary or restricting but any parametric distribution distribution that supports re-parameterised sampling and has tractable log-density could be used here (e.g. normalising flows from Rezende and Mohamed [37]). Identical to the last section we choose the hyper-prior over prior parameters ϕ as a zero-centred spherical Gaussian $\mathcal{P} = \mathcal{N}(0, \sigma^2_{\mathcal{P}} I)$. Again we choose $\lambda = n$ and $\beta_i = m_i$.

Approximating the marginal log-likelihood Before the algorithm can be introduced one needs to consider the computation of the marginal log-likelihood (MLL) $\ln Z_{\beta_i}(S_i, P_\phi) = \ln \mathbb{E}_{h_\theta \sim P_\phi} e^{-\beta_i \hat{\mathcal{L}}(h_\theta, S_i)} = \ln \mathbb{E}_{\theta \sim P_\phi} e^{-\beta_i \hat{\mathcal{L}}(\theta, S_i)}$ which is needed in order to calculate the PACOH $\mathcal{Q}^*$. Calculating $\ln Z_{\beta_i}(S_i, P_\phi)$ directly is difficult due to several reasons. Firstly, unlike for Gaussian Processes, there is no convenient analytical expression available. Secondly, computing the partition function involves taking a high dimensional expectation over Θ, which can be computationally challenging. Additionally, numerical instabilities arise when computing $e^{-\beta_i \hat{\mathcal{L}}(\theta, S_i)}$ for large m_i and thus large β_i. This is because the exponential term becomes very small, and taking the logarithm produces large numbers, leading to numerical overflow or underflow errors. In order to overcome these problems the reparameterisation developed by Kingma and Welling [24] and the Log Sum expectation (LSE) is used, which is defined as follows

$$\text{LSE}_{l=1}^{L}(x_l) = \ln e^{x_1} + ... + e^{x_L} - \ln L \tag{132}$$

The LSE operator is nice because it allows us to normalise a vector of log probabilities effectively and results in numerical stability. For some more details on how the normalisation for the LSE operator works see Appendix A.3. The reparameterisation trick allows us to derive an alternative method to sample from some conditional distribution $q_\phi(z|\ x)$. It works by considering some random variable z that is distributed according to that conditional distribution $z \sim q_\phi(z|\ x)$. It is often possible to consider an additional variable ϵ such that $z \sim g_\phi(\epsilon, x)$. For some vector valued functions q_ϕ, g_ϕ parameterised by ϕ. This will prove helpful as the Monte Carlo estimate of the expectation of $q_\phi(z|\ x)$ is differentiable. Since we want to estimate $\nabla_\phi \ln Z_\beta(S_i, P_\phi)$ in order to compute $\mathcal{Q}^*$. The LSE operator will form the Monte-Carlo estimator and with the reparametrisation trick the differential is guaranteed. More specifically, draw L samples $\theta_l = f(\phi_k, \epsilon_l) = \mu_{P_k} + \sigma_{P_k} \odot \epsilon_l$, $\epsilon_l \sim \mathcal{N}(0, I)$. The generalised MLL estimate is given by

$$\ln \hat{Z}_\beta(S_i, P_\phi) \coloneqq \text{LSE}_{l=1}^{L}\left(-\beta_i \hat{\mathcal{L}}(\theta_l, S_i)\right) - \ln L \tag{133}$$

Computing the differential w.r.t. ϕ

$$\nabla_\phi \ln \hat{Z}_\beta(S_i, P_\phi) = \nabla_\phi \left[\mathrm{LSE}_{l=1}^L \left(-\beta_i \hat{\mathcal{L}}(f(\phi_k, \epsilon_l), S_i) \right) - \ln L \right] \tag{134}$$

$$= \nabla_\phi \left[\ln \left(\sum_{l=1}^L e^{-\beta_i \hat{\mathcal{L}}(f(\phi_k, \epsilon_l), S_i)} \right) \right] \tag{135}$$

$$= \frac{1}{\sum_{l=1}^L e^{-\beta_i \hat{\mathcal{L}}(f(\phi_k, \epsilon_l), S_i)}} \sum_{l=1}^L \left(e^{-\beta_i \hat{\mathcal{L}}(f(\phi_k, \epsilon_l), S_i)} (-\beta_i) \nabla_\phi \hat{\mathcal{L}}(f(\phi_k, \epsilon_l), S_i) \right) \tag{136}$$

$$= -\beta_i \sum_{l=1}^L \underbrace{\frac{e^{-\beta_i \hat{\mathcal{L}}(f(\phi_k, \epsilon_l), S_i)}}{\sum_{l=1}^L e^{-\beta_i \hat{\mathcal{L}}(f(\phi_k, \epsilon_l), S_i)}}}_{\text{softmax}} \underbrace{\nabla_\phi f(\phi, \epsilon_l)^\top}_{\text{re-param. Jacobian}} \underbrace{\nabla_{\theta_l} \hat{\mathcal{L}}(\theta_l, S_i)}_{\text{score}}. \tag{137}$$

Unfortunately $\ln \hat{Z}_\beta(S_i, P_\phi)$ is not an unbiased estimator, but it is consistent. As $\ln Z_\beta(S_i, P_\phi)$ is replaced with our estimator $\ln \hat{Z}_\beta(S_i, P_\phi)$ it needs to be verified that we still minimise a valid PAC-Bayesian upper bound for the transfer error. The following proposition ensures this.

Proposition 3.1 (Rothfuss et al. [40]) *In expectation, replacing* $\ln Z_\beta(S_i, P_\phi)$ *in (32) by the Monte-Carlo estimator* $\ln \hat{Z}_\beta(S_i, P_\phi) \coloneqq \ln \frac{1}{L} \sum_{l=1}^L e^{-\beta_i \hat{\mathcal{L}}(\theta_l, S_i)}$*, still yields a valid upper bound of the transfer error. In particular it holds that*

$$\mathcal{L}_{transfer}(\mathcal{Q}, \mathcal{T}) \leq -\frac{1}{n} \sum_{i=1}^n \frac{1}{\beta} \mathbb{E}_{P \sim \mathcal{Q}}[\ln Z_\beta(S_i, P)] + \left(\frac{1}{\lambda} + \frac{1}{n\beta} \right) D_{KL}(\mathcal{Q} || \mathcal{P}) + C(\delta, \lambda, \beta) \tag{138}$$

$$\leq -\frac{1}{n} \sum_{i=1}^n \frac{1}{\beta} \mathbb{E}_{P \sim \mathcal{Q}} \left[\mathbb{E}_{\theta_1, \ldots, \theta_L \sim P} \left[\ln \hat{Z}_\beta(S_i, P) \right] \right] + \left(\frac{1}{\lambda} + \frac{1}{n\beta} \right) D_{KL}(\mathcal{Q} || \mathcal{P}) + C(\delta, \lambda, \beta) \tag{139}$$

Proof. Firstly, we show that

$$\begin{aligned}
\mathbb{E}_{\theta_1, \ldots, \theta_L \sim P} \left[\ln \hat{Z}_\beta(S_i, P) \right] &= \mathbb{E}_{\theta_1, \ldots, \theta_L \sim P} \left[\ln \frac{1}{L} \sum_{l=1}^L e^{-\beta_i \hat{\mathcal{L}}(\theta_l, S_i)} \right] \\
&\leq \ln \frac{1}{L} \sum_{l=1}^L \mathbb{E}_{\theta_l \sim P} e^{-\beta_i \hat{\mathcal{L}}(\theta_l, S_i)} \\
&= \ln \mathbb{E}_{\theta \sim P} \left[e^{-\beta_i \hat{\mathcal{L}}(\theta, S_i)} \right] \\
&= \ln Z_\beta(S_i, P).
\end{aligned}$$

The inequality in the second line followed directly from the concavity of the logarithm and Jensen's inequality for concave functions, which is stated for reference in Theorem A.2. From the above inequality follows Proposition 3.1 directly. □

Using the MC estimator still yields a valid upper bound, but (139) is potentially looser than the original bound. Moreover, by the law of large numbers, $\ln \hat{Z}(S_i, P) \to \ln Z(S_i, P)$ as $L \to \infty$. So for

large sample sizes L the original bound (32) is recovered.

Algorithm Identical to PACOH-GP the meta-training and testing stages are separated. With the established setup and MLL estimator Algorithm 4 details the meta-training stage of PACOH-NN in order to approximate $\mathcal{Q}^*$. It follows very similar steps to PACOH-GP, with the main difference being the replacement of the MLL $\ln Z_{\beta_i}(S_i, P_{\phi_k})$ with the described MC estimator $\ln \hat{Z}_{\beta_i}(S_i, P_{\phi_k})$. Analogously to (129) it is convenient to use mini-batching in order get an estimate $\tilde{\nabla}_\phi \ln Q^*(\phi)$ using $n_{bs} \leq n$ tasks and adjusting the truncated sum by a factor of $\frac{n}{n_{bs}}$.

Algorithm 4 PACOH-NN - Approximate Inference of $\mathcal{Q}^*$ according to Rothfuss et al. [39]

Require: hyper-prior $\mathcal{P}$, dataset $S_1, ..., S_n$, step-size η

$\tilde{\mathcal{Q}} \leftarrow$ Init_Approx_Inference()
while not converged **do**
  $\{\phi_1, ..., \phi_k\} \leftarrow$ Sample_Prior_Params($\tilde{\mathcal{Q}}$)
  for $k = 1, ..., K$ **do**
    $\{\theta_1^k, ..., \theta_L^k\} \sim P_{\phi_k}$
    for $i = 1, ..., n$ **do**
      $\ln \hat{Z}_{\beta_i}(S_i, P_{\phi_k}) \leftarrow \text{LSE}_{l=1}^{L}(-\beta_i \hat{\mathcal{L}}(\theta_l^k, S_i)) - \ln L$
    end for
    $\nabla_{\phi_k} \ln \hat{\mathcal{Q}}^* \leftarrow \nabla_{\phi_k} \ln \mathcal{P}(\phi_k) + \sum_{i=1}^{n} \frac{\lambda}{n\beta_i + \lambda} \nabla_{\phi_k} \ln \hat{Z}_{\beta_i}(S_i, P_{\phi_k})$
  end for
  $\tilde{\mathcal{Q}} \leftarrow$ Approx_Inference_Update($\tilde{\mathcal{Q}}, \nabla_{\phi_k} \ln \mathcal{Q}^*, \eta$)
end while

If the mini-batched version in correspondence with SVGD is used to approximate $\mathcal{Q}^*$ the algorithm transports K particles each iteration to approximate the hyper-posterior. In each forward step of the algorithm L NN-parameters (of dimensionality $|\Theta|$) are sampled per prior particle and deployed on n_{bs} tasks to estimate the score of $\mathcal{Q}^*$. This results in a space complexity of $\mathcal{O}(|\Theta|K + L)$ and a time complexity of $\mathcal{O}(K^2 + KLn_{bs})$ for a single iteration of the algorithm. In most experiments done in Section 4 20 000 iterations of the algorithm are done, but this can very depending on application and computational resources at hand. Running Algorithm 4 results in an estimate of the PAC-optimal hyper-posterior $\tilde{\mathcal{Q}}$ from which BNN priors can be sampled.

Similar to PACOH-GP the second part of the algorithm is the meta-testing stage, where the system is presented with an unseen learning task $\tau = (\mathcal{D}, m) \sim \mathcal{T}$ and dataset $\tilde{S} \sim \mathcal{D}^m$. Identical to meta-testing in the GP case we expect our resulting inference to depend upon the prior sampled $P_\phi \sim \tilde{\mathcal{Q}}$. Therefore the meta-testing is commenced by sampling a set of K priors from the hyper-posterior $\{P_{\phi_1}, ..., P_{\phi_K}\} \sim \tilde{\mathcal{Q}}$. BNNs as Bayesian base-learners use a prior $P_\phi(h)$ and dataset $\tilde{S}$ to output a posterior $Q^*(h; \tilde{S}, P_\phi)$. Unfortunately this posterior is for BNNs not analytically available and a rather complicated expression. Similar to the discussion earlier in Section 3.1 $Q^*(h; \tilde{S}, P_\phi)$ is approximated using SVGD, which is a standard choice for BNNs. In this case L particles approximate $Q^*(h; \tilde{S}, P_{\phi_k})$ for each of the K drawn priors P_{ϕ_k}. Algorithm 5 details the procedure. The resulting set of $K \cdot L$ NN parameters can then be used for inference on a desired input x^* in the following way

$$\tilde{p}(y^*|\ x^*, \tilde{S}) = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{L}\left[\sum_{l=1}^{L} p\left(y^*|\ h_{\theta_l^k}(x^*)\right)\right]$$
$$= \frac{1}{K \cdot L}\sum_{k=1}^{K}\sum_{l=1}^{L} p\left(y^*|\ h_{\theta_l^k}(x^*)\right).$$

Algorithm 5 PACOH-NN - meta-testing according to Rothfuss et al.[39]

Require: set of prior $\{P_{\phi_1}, ..., P_{\phi_K}\}$, target training data set $\tilde{S}$, evaluation point x^*
Require: kernel function $k(\cdot,\cdot)$, SVGD step size ν, number of particles K

for $k = 1, ..., K$ **do**
  $\{\theta_1^k, ..., \theta_L^k\} \sim P_{\phi_k}$
  while not converged **do**
    for $l = 1, ..., L$ **do**
      $\nabla_{\theta_l^k} Q^*(\theta_l^k) \leftarrow \nabla_{\theta_l^k} \ln P_{\phi_k}(\theta_l^k) + \beta\, \nabla_{\theta_l^k} \mathcal{L}(l, \tilde{S})$
    end for
    $\theta_l^k \leftarrow \theta_l^k + \frac{\nu}{L}\sum_{l'=1}^{L}\left[k(\theta_{l'}^k, \theta_l^k)\, \nabla_{\phi_{l'}^k} Q^*(\theta_{l'}^k) + \nabla_{\theta_{l'}^k} k(\theta_{l'}^k, \theta_l^k)\right]$
  end while
end for

Output: a set of NN parameters $\cup_{k=1}^{K}\{\theta_1^k, ..., \theta_L^k\}$

3.4 Discussion

One of the key advantages of PACOH-NN compared to previous methods for meta-learning BNN priors, such as in Pentina and Lampert [31] and Amit and Meir [2], is that it simplifies the previously nested optimisation problem into a stochastic optimisation problem. This results in increased stability and scalability of meta-learning, as there is no need to explicitly compute the task posteriors Q_i and mini-batching over tasks can be used. As a result, the space and compute complexity do not depend on the number of tasks n, unlike the MLAP algorithm from Amit and Meir [2], which has a memory footprint of $O(|\Theta| \cdot n)$ and becomes prohibitive for more than 50 tasks.

In addition, PACOH-NN includes a hyper-prior $\mathcal{P}$ that provides principled meta-level regularisation, which addresses overfitting to the meta-training tasks detailed in Yin et al. [44]. As shown in experiments done by Rothfuss et al. [39], PACOH-NN can do successful meta-learning with as few as 5 tasks. This is in contrast to other popular meta-learners such as MAML from Finn et al. [11], its Bayesian version BMAML [23] from Kim et al. and Neural Processes Garnelo et al. [15], which require a large number of tasks to generalise well on the meta-level, as noted by Qin et al. [34].

All of the algorithms discussed in this Section operate in the weight or parameter space $|\Theta|$ of the neural networks, such as the SVGD or VI approximations. This can be a severe limitation as NN weight spaces are heavily overparametrised, as mentioned by Baldassi et al. [4]. For example every single mode of the posterior is degenerate in the sense that many parameter configurations correspond to the same NN. This is partially due to the inherent symmetries of a neural network.

The authors Rothfuss et al. have proposed a functional extension of PACOH called F-PACOH [38], where the gradient updates are done in the output space of the neural networks and then backpropagated using the chain rule to arrive at an optimal parameter configuration. Due to the limited scope of this essay these interesting ideas will not be discussed here any further but the intrigued reader is referred to Rothfuss et al. [38] to see the functional extension of PACOH and to D'Angelo et al. [8] to see how SVGD can be converted to function space.

4 Numerical Experiments

This section will show some numerical results of the algorithms discussed so far with a particular emphasis on PACOH-NN. Modern state-of-the-art deep learning models such as the ones presented here are usually trained on vast datasets using large GPU computing clusters which were not available for this project. Therefore this section should be understood as more of a qualitative presentation as opposed to an exhaustive numerical study. Four experiments will be detailed. The first one is a slightly modified version of the standard sinusoids environment where the BNN, MAML, PACOH-NN, PACOH-GP algorithms are compared. Experiment 2 details a possible application for PACOH-NN-SVGD for a Sinc environment. In the third experiment a hyper-parameter investigation is done for PACOH-NN-SVGD for the Cauchy environment introduced in Rothfuss et al. [40]. Lastly, the problem of train/- test time distribution shift is illustrated for PACOH-NN-SVGD. The code for the experiments was adapted from Rothfuss et al. [40] and can be found in Flöge [12]

Experiment 1: 1D Noisy Sinusoids regression

As an initial qualitative experiment to compare some of the algorithms discussed in this essay the standard sinusoids environment for meta-learning will be used with a slight twist. We will preform one-dimensional function regression in the domain $[-4, 4]$. The task are differentiated through the random parameters $A \sim U[-4, 4]$ and $\phi \sim U[-2, 2]$ and

$$f_{A,\phi}(x) = A \cdot \sin(x - \phi) + 5 \tag{140}$$

The training dataset for each task in then created by sampling $x_i \sim \mathcal{N}(0, 1)$ and $y_i \sim f_{A,\phi}(x_i) + \epsilon_j$, where $\epsilon_j \sim \mathcal{N}(0, 0.1^2)$. For the meta test data $x_i \sim U(-4, 4)$. The main difference to the standard sinusoids environment is that the training data is sampled from a standard normal $\mathcal{N}(0, 1)$ so the algorithms see few to no data on the ends of the domain. The data training data is plotted in the following Figure 5

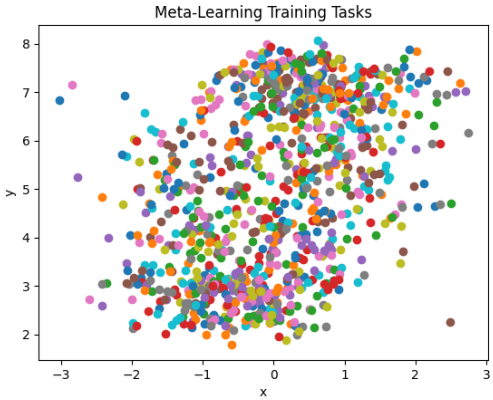


Figure 5: Experiment 1 training data

We use 200 training tasks with 5 samples per task. During the meta-test stage 20 tasks are sampled with 5 context samples [5] and 200 test samples. The posterior of a Vanilla BNN, MAML, PACOH-GP and PACOH-NN are visualised in Figure 6. We trained the algorithms for 20 000 iterations and 300

[5] context samples refer to labelled examples from a respective task. This is the few shot learning

meta test iterations. A task batch size of 10 was used for MAML.

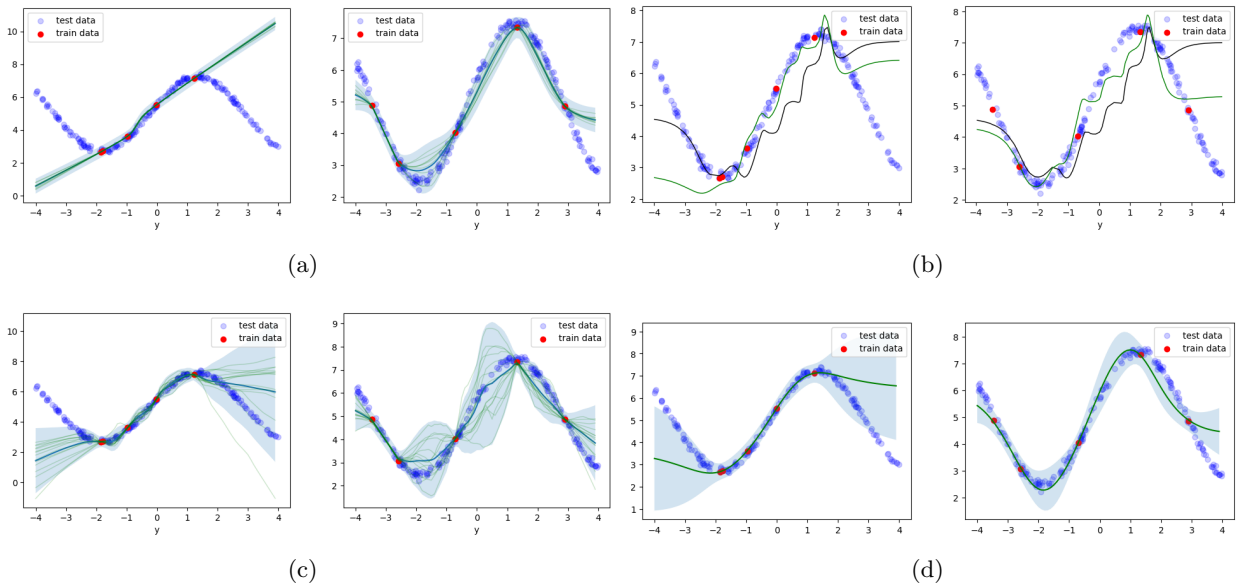


Figure 6: Noisy sinusoids environment for four algorithms. Two test tasks were plotted with red points representing the context samples and blue the test samples. (a) Vanilla BNN (b) MAML (c) PACOH-NN-SVGD (d) PACOH-GP-MAP

In Figure 6 the resulting posterior for two test tasks are plotted for each of the four algorithms. The green line corresponds to the mean posterior and the blue areas to the confidence interval around the prediction. Please note that since MAML optimises point-estimate Neural Networks as visualised in Figure 4 the concept of a confidence interval does not apply. The red points refer to the context samples the algorithms received form that task and the blue points are the test points. It can be seen in (a) that the Vanilla BNN does not pick-up on the sinusoidal structure of the data at all. The MAML (b) does not perform so bad, but on the ends of the domain where no test data was present the algorithm's performance deteriorates. (c) PACOH-NN-SVGD is doing relatively well at capturing the true line within its confidence, (d) PACOH-GP-MAP appears to be smoother and less adaptive than PACOH-NN in its confidence intervals which is likely due to the lack of flexibility of a MAP estimate.

Experiment 2: 2D Sinc function regression

The function $\text{Sinc} : \mathbb{R} \to \mathbb{R}$ is formally defined by

$$\text{Sinc}(x) = \frac{\sin(x)}{x} \tag{141}$$

These are ubiquitous in science and engineering. In communications theory they can be used to represent the impulse response of an ideal low-pass filter. Low-pass-filters are electrical circuits which are commonplace in almost every signal processing unit. In optics and quantum mechanics sinc functions can be used to model the interference pattern of a double slit experiment. As such if we consider two dimensional detector plate the two-dimensional regression problem for sinc function

is an important application as outlined in Bishop and Tipping [5].

Here we will extend this experiment by not just considering Additive White Gaussian Noise (AWGN) on the interference of two sinc functions, but rather a more complicated Gaussian process prior noise term. For this we use the kernel $k(x, x') = \exp\left(\frac{||x-x'||^2}{l}\right)$ with $l = 0.2$.

$$h(x) = \text{Sinc}\left(||x - \mu_1||_2\right) + 2\text{Sinc}(||x - \mu_2||_2), \tag{142}$$

$$f(x) = h(x) + g(x), \text{ with } g \sim \mathcal{GP}(0, k(x, x'). \tag{143}$$

The target data is then generated as follows $x = \min(\max(\tilde{x}, -10), 10)$,

$$y \sim \mathcal{N}(f(x), 0.05^2). \tag{144}$$

The mean function $h(x)$ is depicted in Figure 7.

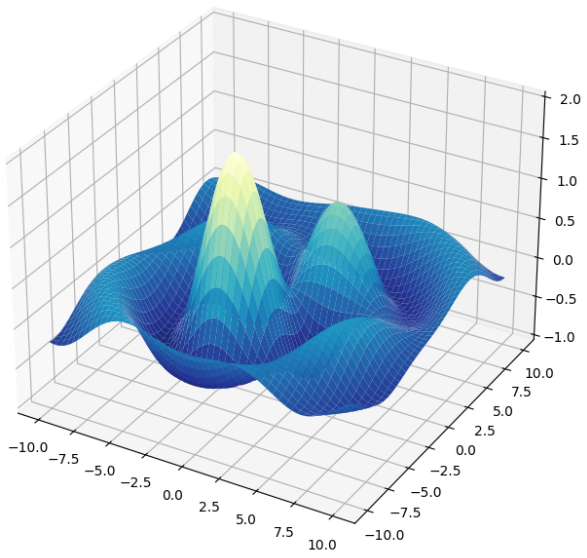


Figure 7: Mean function $h(x)$ for the sinc environment. The plot was generated for $\mu_1 = (1, 3)$ and $\mu_2 = (-2, -2)$

We will use the PACOH-NN-SVGD algorithm with number of iteration 20000, ReLu activation function and 20 context samples.

In order to evaluate the performance of the algorithm the average Root-Mean-Squared-Error (RMSE) is used. Given a dataset $(x^*, y^*)_{j=1}^{m_i} = S_j^*$. The RMSE for datset S_i^* is defined as follows

$$\text{RMSE}(S_i^*) = \sqrt{\frac{1}{|S_i^*|} \sum_{(x^*, y^*) \in S_i^*} (y^* - \hat{y})^2}, \tag{145}$$

where $\hat{y}$ is the prediction of the model corresponding to input x^*. If we have n test tasks with datasets $S_1^*, ..., S_n^*$ the the average RMSE is the averaged RMSE of all training tasks.

Figure 8: Training loss of PACOH-NN-SVGD with respect to number of iterations during training in the Sinc environment.

$$\text{average RMSE} = \frac{1}{n}\sum_{i=1}^{n}\text{RMSE}(S_i^*) = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\frac{1}{|S_i^*|}\sum_{(x^*,y^*)\in S_i^*}(y^*-\hat{y})^2} \tag{146}$$

In addition to the average RMSE, calibration is a crucial process for probabilistic predictors that generate a probability distribution $\hat{p}\left(y_i \mid x_i\right)$ of predicted target values y_i when presented with a new input x_i. One would like this predicted probability to behave like the true underlying probability, this is then referred to as a calibrated model. Calibration error for regression is defined as follows. Corresponding to the predictive density, we denote a predictor's cumulative density function (CDF) as $\hat{F}\left(y_j \mid x_j\right) = \int_{-\infty}^{y_j} \hat{p}\left(y \mid x_i\right) dy$. For confidence levels $0 \leq q_h < \ldots < q_H \leq 1$, we can compute the corresponding empirical frequency

$$\hat{q}_h = \frac{\left|\left\{y_j \mid \hat{F}\left(y_j \mid x_j\right) \leq q_h, j = 1, \ldots, m\right\}\right|}{m} \tag{147}$$

based on dataset $S = \{(x_i, y_i)\}_{i=1}^m$ of m samples. If we have calibrated predictions we would expect that $\hat{q}_h \to q_h$ as $m \to \infty$. Similar to Kuleshov et al. [25], we can define the calibration error as a function of residuals $\hat{q}_h - q_h$, in particular,

$$\text{calibration error} = \frac{1}{H}\sum_{h=1}^{H}\left|\hat{q}_h - q_h\right|. \tag{148}$$

Note that while Kuleshov et al. [25] report the average of squared residuals $|\hat{q}_h - q_h|^2$, following Rothfuss et al. [40] we report the average of absolute residuals $|\hat{q}_h - q_h|$ in order to preserve the units and keep the calibration error easier to interpret. The confidence levels $0 \leq q_1 < ... < q_H \leq 1$ are chosen equally spaced in $[0, 1]$.

For our experiment set $\mu_1 = (0, 0)^\top$ and μ_2 is drawn from a truncated normal with domain $[-10, 10]$ and variance $\sigma^2 = 3$. This can be interpreted as aligning the detector plate with one of the sources and then being flexible with the location of the second source. For 20 training tasks and 20 context

samples per task we get an average RMSE of 0.3606 ± 0.0861 and a calibration error 0.410 ± 0.0146. Which is relatively good for such a complicated environment.

Experiment 3: Hyper-parameter tuning in Cauchy environment
Detailed in Rothfuss et al. [40] the Cauchy environment is a 2D regression environment very similar to the sinc environment detailed in the previous experiment. Here the mean function is given by an unnormalised mixture of two Cauchy distributions, i.e.

$$m(x) = \frac{6}{\pi \cdot (1 + ||x - \mu_1||_2^2} + \frac{3}{\pi \cdot (1 + ||x - \mu_2||_2^2)}, \tag{149}$$

with $\mu_1 = (-1, -1)^\top$ and $\mu_2 = (2, 2)^\top$. The kernel function is again defined as $k(x, x') = \exp\left(\frac{||x-x'||^2}{l}\right)$ with $l = 0.2$. The tasks are then sampled from the task environment as follows

$$f(x) = h(x) + g(x), \text{ with } g \sim \mathcal{GP}(0, k(x, x'). \tag{150}$$

The samples from each task are generated according by drawing x from a truncated normal distribution and then applying additive Gaussian noise with standard deviation 0.05 to the function values $f(x)$. For $x = \min(\max(\tilde{x}, -10), 10)$,

$$\tilde{x} \sim \mathcal{N}(0, 2.5^2), \quad y \sim \mathcal{N}(f(x), 0.05^2). \tag{151}$$

On this environment Rothfuss et al. [40] reported the following results for different meta-learning algorithms.

Algorithm	average RMSE	calibration error
Vanilla GP	0.275 ± 0.000	0.087 ± 0.000
Vanilla BNN (Liu & Wang, [27])	0.327 ± 0.008	0.055 ± 0.006
MLL-GP ((Fortuin & Ratsch, [14])	0.216 ± 0.003	0.059 ± 0.003
MLAP (Amit & Meir, [2])	0.219 ± 0.004	0.086 ± 0.015
MAML (Finn et al., [11])	0.219 ± 0.004	N/A
BMAML (Kim et al., [23])	0.225 ± 0.004	0.061 ± 0.007
NP (Garnelo et al., [15])	0.224 ± 0.008	0.057 ± 0.009
PACOH-GP-SVGD (Rothfuss et al., [40])	0.209 ± 0.008	0.056 ± 0.004
PACOH-NN-SVGD (Rothfuss et al., [40])	$\mathbf{0.195 \pm 0.001}$	$\mathbf{0.046 \pm 0.001}$

For this comparison the algorithms were trained with $m_i = 20$ context samples on each of the $n = 20$ tasks sampled from the Cauchy environment. It is important to note that the MAML algorithm operates on point estimate NNs, which means that the concept of calibration-error does not apply. It can be seen that the PACOH algorithms outperform all the other listed meta-learners or Bayesian models. Especially PACOH-NN-SVGD has the best performance in terms of average RMSE and calibration error. The authors Rothfuss et al. demonstrated similar impressive performance on other meta-learning datasets such as SwissFel (Milne et al. [42]), Physionet (Silva et al. [33]), Berkeley-Sensor (Madden [28]) and Omniglot (Lake et al. [26]). This showcases the impressive state

of the art performance of the PACOH algorithms in meta-learning problems.

The methods introduced in this essay have a lot of tuning parameters which need to be specified. In order to investigate PACOH-NN-SVGD further a hyperparameter study is done. The main parameters with their default configurations for this experiment are:

- Number of particles K to approximate hyper-posterior, $K = 3$
- Number of particles K to approximate BNN posterior, $L = 5$
- Size of hidden layers, $(32, 32, 32, 32)$
- learning rate, $\text{lr} = 1.5 \cdot 10^{-3}$
- number of meta-train iterations, 20 000
- number of samples per task, $m = 20$
- number of tasks, $n = 20$
- Standard deviation of hyper-prior, $\sigma_{\mathcal{P}} = 0.12$

The average RMSE for the Cauchy environment goes to zero with increasing numbers of training tasks n for PACOH-algorithms as demonstrated in Rothfuss et al. [40]. With the hyperparameters specified as above the average RMSE is 0.2071 ± 0.0194 and calibration error 0.0489 ± 0.0248. In the PACOH-NN algorithm SVGD inference is used twice. Firstly K particles are used to approximate the hyper-posterior $\mathcal{Q}^*$ and secondly L particles are used to approximate the posterior Q^*. As such the dependence of the average RMSE and calibration error with respect to the number of particles K and L is investigated. the results are displayed in Figure 9.

It can be seen in Figure 9 that the average RMSE and calibration error barely change with respect to the number of hyper-posterior and posterior-particles. There is a slight negative trend initially on all the plots, as on would expect more particles to better represent the shape of the distribution. For the MAP $K = 1$ or $L = 1$ the standard deviation for the average RMSE and calibration error tend to be larger. This is likely a result of the fact that the approximation of $\mathcal{Q}^*$ and Q^* by a Dirac-measure does not capture the shape of these distributions well. The overall change in error is quite small. Possible explanations are that the number of meta-train iterations was too small to transport the particles properly or that the standard-deviation of the hyper-prior $\sigma_{\mathcal{P}}$ is too small. The first explanation seems unlikely for the RMSE and Calibration barely change even with respect to the MAP estimate. To investigate the second possible explanation a little further, the standard-deviation of the hyper-prior was increased to $\sigma_{\mathcal{P}} = 1$.

Number of Hyper-posterior particles	Average RMSE	Calibration error
5	0.2114 ± 0.0203	0.0434 ± 0.0204
10	0.2074 ± 0.0212	0.0411 ± 0.0183

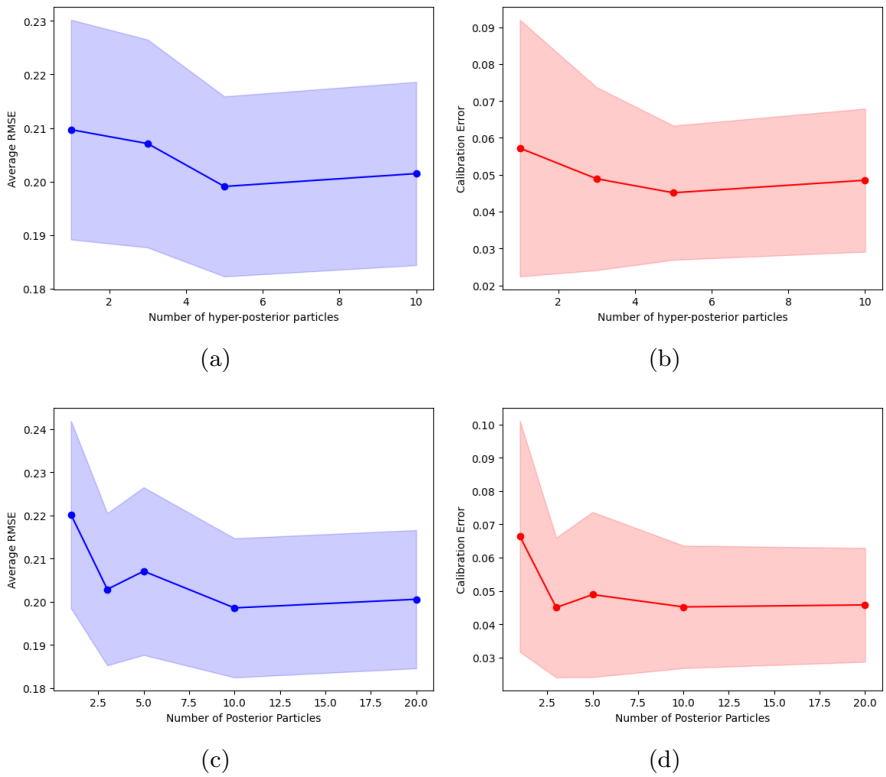


Figure 9: The shaded region in all the plots represents the mean with added/subtracted standard deviation of the obtained values. (a) and (b): average RMSE and calibration error respectively with respect to number of hyper-posterior particles. (c) and (d): average RMSE and calibration error respectively with respect to number of posterior particles

Again no significant deviation in the RMSE and calibration error can be noted from the previous results. This indicates that the PACOH-NN-SVGD is a rather robust algorithm with respect to the number of particles used to approximate the hyper-posterior $\mathcal{Q}^*$ and BNN posterior Q^*.

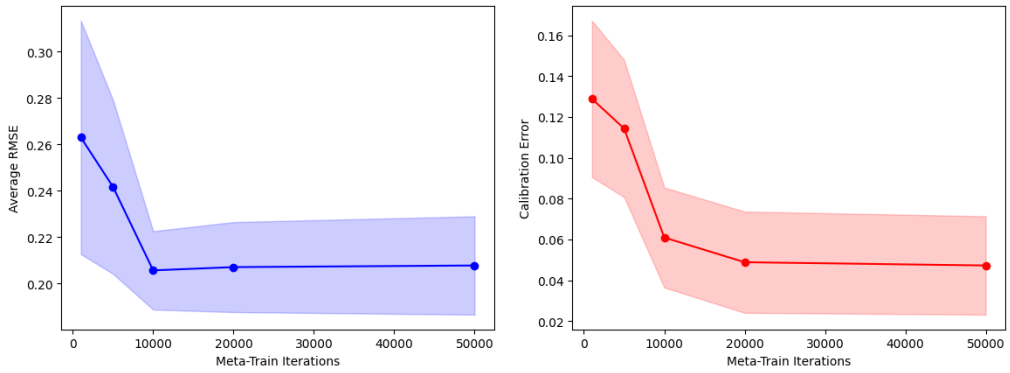
The next investigation is done with respect to the size of the hidden layers. Here the notation $(32, 32, 32, 32)$ indicates that 4 fully-connected layers with 32 nodes each are used as a hidden layer. The results are displays in the following Table.

Hidden Layer size	Average RMSE	Calibration Error	Average Log-Likelihood
(32, 32)	0.2052 ± 0.0156	0.0450 ± 0.0179	0.1602 ± 0.0851
(32, 32, 32, 32)	0.2071 ± 0.0194	0.0480 ± 0.0255	0.1642 ± 0.0847
(32, 32, 32, 32, 32, 32)	0.2012 ± 0.0196	0.0480 ± 0.0255	0.1859 ± 0.0839
(32, 32, 32, 32, 32, 32, 32, 32)	0.2048 ± 0.0187	0.0530 ± 0.0273	0.1707 ± 0.0908
(64, 64, 64, 64)	0.2090 ± 0.0188	0.0470 ± 0.0262	0.1515 ± 0.0981
(64, 64, 64, 64, 64, 64, 64, 64)	0.2125 ± 0.0219	0.0628 ± 0.0391	0.1266 ± 0.1330

It can be seen that the average RMSE error again does vary all too much. A small trend can be seen in the calibration error when the depth of the 32 node layers is increased. The worst calibration is

seen in the largest of the investigated hidden layers. This could be the meta-learning analogue to a phenomena in standard supervised learning pointed out by Minderer et al. [29], where in-distribution calibration worsens with increasing model size as. However, due to the increasing standard deviation in the calibration error for larger networks further investigation is necessary to confirm this.

Next the dependence on the number of meta-train iterations is investigated.



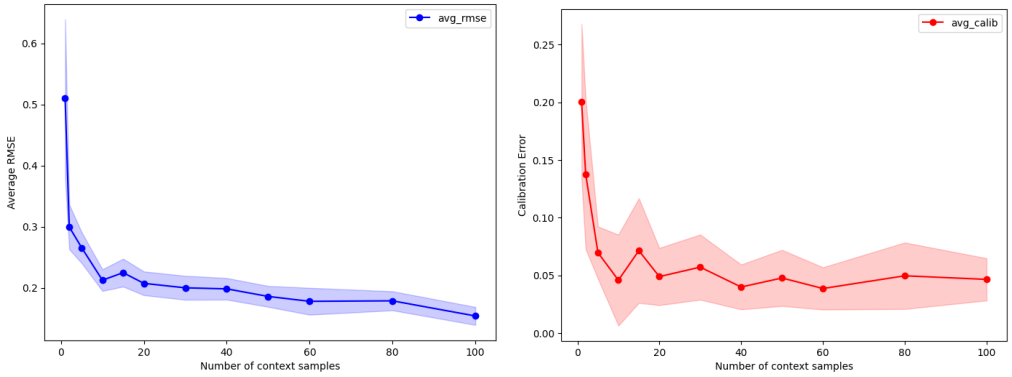
(a) (b)

Figure 10: The shaded region in all the plots represents the mean with added/subtracted standard deviation of the obtained values. (a) average RMSE with respect to number of meta-train iterations (b) calibration error with respect to number of meta-train iterations

Figure 10 shows how the average RMSE and calibration increases sharply for 1000 meta-train iterations. This is likely because the training process was stopped before the particles approximating the hyper-posterior converged. Both metrics appear to flatten for larger numbers of iterations, which is probably a result of the fact that the particles had converged already so SVGD does not update them further which makes additional training futile. Based on these findings we conclude that the number of meta-training iterations, if set 'high enough', is not an influential hyper-parameter. As to what constitutes 'high enough' exactly likely depends on the size of the hidden layers and therefore the size of parameter space $|\Theta|$. This could be interesting point of further investigation.

Lastly the significance of the number of context samples m is investigated. In Figure 11 it can be seen how the average RMSE and calibration error peak sharply for $m = 1$ or $m = 2$. This is because the meta-learner does not have enough data on the two dimensional function $f(x)$ to learn it accurately. It probably just learns the approximate range of the function and then randomly guesses. With increasing number of context samples the average RMSE decreases. This is expected as the learner can get more information from each task and also has more labelled examples from the unseen test tasks. Which of these effects contributes more cannot be inferred directly from the graphic above, however we suspect that the latter is more important. The calibration error also peaks sharply for smaller m and then remains roughly steady.

Experiment 4: 1D sinusoids regression in training-/ test-time distribution shift As mentioned in the beginning of Section 2.3 the PAC-Bayesian Meta-Learning theorems derived in this essay do not hold in the presence of a training-/ test-time distribution shift. In order to



(a) (b)

Figure 11: The shaded region in all the plots represents the mean with added/subtracted standard deviation of the obtained values. (a) average RMSE with respect to number of context samples (b) calibration error with respect to number of context samples

illustrate this numerically we will perform the following experiment on PACOH-NN-SVGD. Consider one-dimensional function regression in the domain $[-4, 4]$. The task are differentiated during training time through the random parameters $A \sim U[-4, 4]$, $\phi \sim U[-2, 2]$ and

$$f_{A,\phi}^{\text{training}}(x) = A \cdot \sin(x - \phi) + 5 + (x - \phi) \tag{152}$$

During test time the tasks are generated through sampling $A \sim U[-4, 4]$, $\phi \sim U[-2, 2]$ and

$$f_{A,\phi}^{\text{test}}(x) = A \cdot \cos(x - \phi) + 5 + (x - \phi) \tag{153}$$

The training dataset for each task in the created by sampling $x_i \sim U[-4, 4]$ and $y_i \sim f_{A,\phi}^{\text{training}}(x_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, 0.1^2)$. For the meta test data $x_i \sim U[-4, 4]$ and $y_i \sim f_{A,\phi}^{\text{training}}(x_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, 0.1^2)$. Here unlike in Experiment 1 the training data is sampled uniformly across the domain. The main difference to the sinusoids environment used in Rothfuss et al. [40] is that $f_{A,\phi}^{\text{training}}(x) \neq f_{A,\phi}^{\text{test}}(x)$ as the $\sin(x - \phi)$ term was replaced $\cos(x - \phi)$. Figure 12 depicts the posterior PACOH-NN-SVGD on three test tasks. It can be seen that the algorithm performs much worse than reported in Rothfuss et al. [40]. The posterior does not really line up with true shape of the curve. The average RMSE, as described in (146), is 1.7699 ± 0.5795 and the calibration error is 0.1836 ± 0.0745. Both of these are not very good values. However, it is worth pointing out that the confidence intervals in Figure 12 mainly capture the true shape. This is desirable as it indicates that these PAC-Bayesian meta-learners realise that the test data is not what they expect and widen the confidence intervals accordingly, which is a major advantage over point-based NN algorithms such as MAML or REPTILE. An interesting point of further research would be to attempt to change the algorithm in order to better accommodate situations like this either through adding additional terms into the PACOH model or to change the PAC-Bayesian bounds. Both of these methods would require further assumption on the test environment and the type of distribution shift.

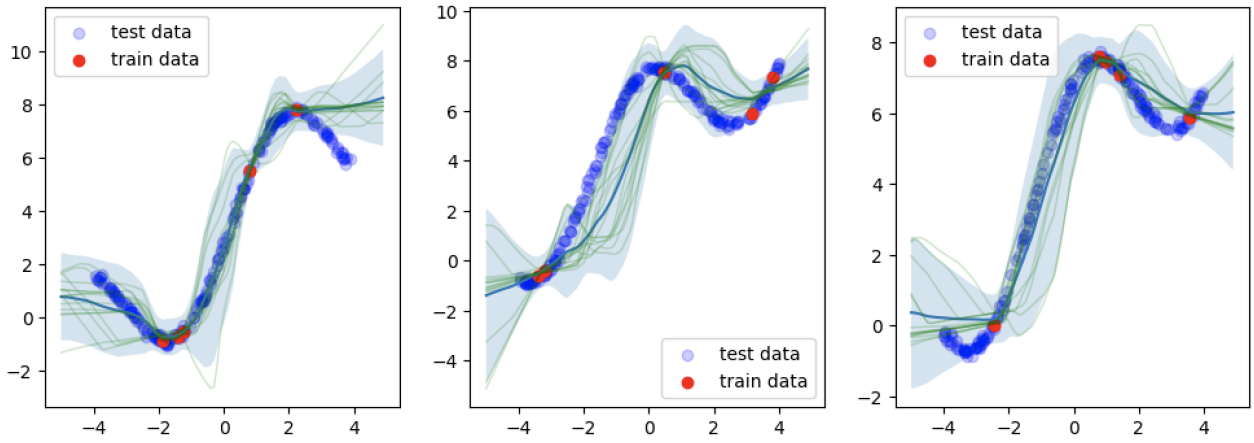


Figure 12: PACOH-NN-SVGD in distribution shift environment

5 Conclusion

In conclusion, meta-learning is a rapidly growing field with various architectures and approaches. This essay explored the field through a PAC-Bayesian lens, which provides probabilistic guarantees on the performance of certain algorithms, with a focus on supervised few-shot-learning. The essay initially examined different approaches to meta-learning, with a particular emphasis on optimisation-based algorithms like MAML and REPTILE. It then proceeded to develop the general supervised PAC-bounds and extend it to meta-learning. The developed techniques not only provide a rigorous theory but can also derive modern optimisation-based algorithms, showcasing the universality of the PAC-Bayesian meta-learning framework. The PACOH algorithm, a state-of-the-art meta-learning algorithm, was presented, and numerical experiments demonstrated its behaviour and effectiveness. Overall, the essay shows that PAC-Bayesian meta-learning provides a powerful and flexible framework for developing robust and efficient meta-learning algorithms.

References

[1] Pierre Alquier, James Ridgway, and Nicolas Chopin. *On the properties of variational approximations of Gibbs posteriors*. 2015. arXiv: 1506.04091 [stat.ML].

[2] Ron Amit and Ron Meir. *Meta-Learning by Adjusting Priors Based on Extended PAC-Bayes Theory*. 2019. arXiv: 1711.01244 [stat.ML].

[3] Antreas Antoniou, Harrison Edwards, and Amos Storkey. *How to train your MAML*. 2019. arXiv: 1810.09502 [cs.LG].

[4] Carlo Baldassi et al. "Learning through atypical phase transitions in overparameterized neural networks". In: *Physical Review E* 106.1 (July 2022). DOI: 10.1103/physreve.106.014116. URL: https://doi.org/10.11032Fphysreve.106.014116.

[5] Christopher M. Bishop and Michael E. Tipping. "Variational Relevance Vector Machines". In: *In Uncertainty in Artificial Intelligence 2000*. Boutilier and M. Goldszmidt, 2000, pp. 46–53. URL: https://www.miketipping.com/papers/Bishop-VRVM-UAI-00.pdf.

[6] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association* 112.518 (Apr. 2017), pp. 859–877. DOI: 10.1080/01621459.2017.1285773. URL: https://doi.org/10.10802F01621459.2017.1285773.

[7] Olivier Cantoni. "Pac-Bayesian Supervised Classification: The Thermodynamics of Statistical Learning". In: *IMS Lecture Notes Monograph Series* 56 (2007), pp. 1–163. DOI: 10.1214/074921707000000391. URL: https://arxiv.org/abs/0712.0248.

[8] Francesco D'Angelo, Vincent Fortuin, and Florian Wenzel. *On Stein Variational Neural Network Ensembles*. 2021. arXiv: 2106.10760 [cs.LG].

[9] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[10] Nan Ding et al. "Bridging the Gap Between Practice and PAC-Bayes Theory in Few-Shot Meta-Learning". In: *CoRR* abs/2105.14099 (2021). arXiv: 2105.14099. URL: https://arxiv.org/abs/2105.14099.

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *CoRR* abs/1703.03400 (2017). arXiv: 1703.03400. URL: http://arxiv.org/abs/1703.03400.

[12] Klemens Niklas Floege. *Meta-Learning: A PAC-Bayesian Perspective*. Version 1.0. May 2023. URL: https://github.com/kf416/Part-III-essay-meta-learning.git.

[13] Vincent Fortuin. *Priors in Bayesian Deep Learning: A Review*. 2022. arXiv: 2105.06868 [stat.ML].

[14] Vincent Fortuin, Heiko Strathmann, and Gunnar Rätsch. *Meta-Learning Mean Functions for Gaussian Processes*. 2020. arXiv: 1901.08098 [stat.ML].

[15] Marta Garnelo et al. *Neural Processes*. 2018. arXiv: 1807.01622 [cs.LG].

[16] Pascal Germain et al. *A New PAC-Bayesian Perspective on Domain Adaptation*. 2016. arXiv: 1506.04573 [stat.ML].

[17] Pascal Germain et al. *PAC-Bayesian Theory Meets Bayesian Inference*. 2017. arXiv: 1605.08636 [stat.ML].

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[19] Benjamin Guedj. *A Primer on PAC-Bayesian Learning*. 2019. arXiv: 1901.05353 [stat.ML].

[20] Timothy Hospedales et al. *Meta-Learning in Neural Networks: A Survey*. 2020. arXiv: 2004.05439 [cs.LG].

[21] Mike Huisman, Jan N. van Rijn, and Aske Plaat. "A survey of deep meta-learning". In: *Artificial Intelligence Review* 54.6 (Apr. 2021), pp. 4483–4541. DOI: 10.1007/s10462-021-10004-4. URL: https://doi.org/10.10072Fs10462-021-10004-4.

[22] Laurent Valentin Jospin et al. "Hands-On Bayesian Neural Networks—A Tutorial for Deep Learning Users". In: *IEEE Computational Intelligence Magazine* 17.2 (May 2022), pp. 29–48. DOI: 10.1109/mci.2022.3155327. URL: https://doi.org/10.11092Fmci.2022.3155327.

[23] Taesup Kim et al. *Bayesian Model-Agnostic Meta-Learning*. 2018. arXiv: 1806.03836 [cs.LG].

[24] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat.ML].

[25] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. *Accurate Uncertainties for Deep Learning Using Calibrated Regression*. 2018. arXiv: 1807.00263 [cs.LG].

[26] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. "Human-level concept learning through probabilistic program induction". In: *Science* 350.6266 (2015), pp. 1332–1338. DOI: 10.1126/science.aab3050. eprint: https://www.science.org/doi/pdf/10.1126/science.aab3050. URL: https://www.science.org/doi/abs/10.1126/science.aab3050.

[27] Qiang Liu and Dilin Wang. *Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm*. 2019. arXiv: 1608.04471 [stat.ML].

[28] Samuel Madden. *Intel Lab Data*. 2004. URL: http://db.csail.mit.edu/labdata/labdata.html.

[29] Matthias Minderer et al. *Revisiting the Calibration of Modern Neural Networks*. 2021. arXiv: 2106.07998 [cs.LG].

[30] Alex Nichol, Joshua Achiam, and John Schulman. "On First-Order Meta-Learning Algorithms". In: *CoRR* abs/1803.02999 (2018). arXiv: 1803.02999. URL: http://arxiv.org/abs/1803.02999.

[31] Anastasia Pentina and Christoph H. Lampert. *A PAC-Bayesian bound for Lifelong Learning*. 2014. arXiv: 1311.2838 [stat.ML].

[32] Antoine Picard-Weibel and Benjamin Guedj. *On change of measure inequalities for f-divergences*. 2022. arXiv: 2202.05568 [stat.ML].

[33] "Predicting In-Hospital Mortality of ICU Patients: The PhysioNet/Computing in Cardiology Challenge 2012". In: *Computing in Cardiology*. 2012. URL: https://physionet.org/content/challenge-2012/1.0.0/.

[34] Yunxiao Qin et al. "Rethink and Redesign Meta learning". In: *ArXiv* abs/1812.04955 (2018).

[35] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Boston: The MIT Press, 2006.

[36] Sachin Ravi and Hugo Larochelle. "Optimization as a Model for Few-Shot Learning". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=rJY0-Kcll.

[37] Danilo Jimenez Rezende and Shakir Mohamed. *Variational Inference with Normalizing Flows*. 2016. arXiv: 1505.05770 [stat.ML].

[38] Jonas Rothfuss et al. *Meta-Learning Reliable Priors in the Function Space*. 2022. arXiv: 2106.03195 [cs.LG].

[39] Jonas Rothfuss et al. *PAC-Bayesian Meta-Learning: From Theory to Practice*. 2022. DOI: 10.48550/ARXIV.2211.07206. URL: https://arxiv.org/abs/2211.07206.

[40] Jonas Rothfuss et al. *PACOH: Bayes-Optimal Meta-Learning with PAC-Guarantees*. 2021. arXiv: 2002.05551 [stat.ML].

[41] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *CoRR* abs/1409.0575 (2014). arXiv: 1409.0575. URL: http://arxiv.org/abs/1409.0575.

[42] "SwissFEL: the Swiss X-ray free electron laser". In: *Applied Sciences* 7.57 (2017), pp. 720–777. DOI: 10.3390/app7070720. URL: https://www.dora.lib4ri.ch/empa/islandora/object/empa:21160.

[43] S. Prince W. Zi L. S. Ghoraie. *Few-Shot Learning and Meta-Learning — Tutorial*. Accessed on May 3, 2023. 2023. URL: https://www.borealisai.com/research-blogs/tutorial-2-few-shot-learning-and-meta-learning-i/.

[44] Mingzhang Yin et al. *Meta-Learning without Memorization*. 2020. arXiv: 1912.03820 [cs.LG].

List of Figures

A Appendix

A.1 Relevant Definitions and Inequalities

Definition A.1 (sub-Gaussian random variables) *A random variable X with $\mathbb{E}(X) = 0$ is sub-Gaussian with parameter σ^2 if*

$$\mathbb{E}[e^{tX}] \leq e^{t^2\sigma^2/2} \tag{154}$$

for every $t \in \mathbb{R}$

Definition A.2 (sub-Gamma random variables) *A random variable X with $\mathbb{E}(X) = 0$ is sub-Gamma with variance factor $\sigma^2 > 0$ and scale parameter $c > 0$ if*

$$\ln \mathbb{E}[e^{tX}] \leq \frac{\sigma^2 t^2}{2(1-ct)} \tag{155}$$

for every $t \in [0, 1/c)$

Theorem A.1 (Markov's inequality) *Let X be a non-negative random variable. Then for all $t > 0$, we have*

$$\mathbb{P}(X \geq t) \leq \frac{\mathbb{E}[X]}{t} \tag{156}$$

This can be extended by considering a non-negative non-decreasing function $\phi : \mathbb{R} \to \mathbb{R}$ to the following

$$\mathbb{P}(X \geq t) = \mathbb{P}(\phi(X) \geq \phi(t)) \leq \frac{\mathbb{E}[\phi(X)]}{\phi(t)} \tag{157}$$

Theorem A.2 (Jensen's inequality) *Let X be a random variable and ψ a convex function, then*

$$\psi(\mathbb{E}[X]) \leq \mathbb{E}[\psi(X)] \tag{158}$$

if ψ is a concave function, then

$$\mathbb{E}[\psi(X)] \leq \psi(\mathbb{E}[X]) \tag{159}$$

Theorem A.3 (Hoeffding's inequality) *Let $X_1, ..., X_n$ be independent sub-Gaussian random variables, with X_i having sub-Gaussian parameter σ_i^2, for $i \in \{1, ..., n\}$. Then $\bar{X} = n^{-1}\sum_{i=1}^n X_i$ is sub-Gaussian with parameter $\bar{\sigma}^2/n$, where $\bar{\sigma}^2 = n^{-1}\sum_{i=1}^n$. In particular,*

$$\mathbb{P}(\bar{X} \geq x) \leq e^{-nx^2/(2\bar{\sigma}^2} \tag{160}$$

for every $x \geq 0$.

Lemma A.4 (Hoeffding's lemma) *Let X be a random variable taking values in a bounded interval $[a, b]$. Then X is sub-Gaussian with parameter $(b-a)^2/4$. This gives*

$$\mathbb{E}\left[e^{\lambda(X-\mathbb{E})}\right] \leq \exp\left(\frac{\lambda^2(b-a)^2}{8}\right) \tag{161}$$

A.2 PACOH Algorithm Details

The three inference methods MAP, SVGD and VI and their respective initialisation and update rules.

	Approximating Distribution	Init_Approx_Inference
MAP	$\tilde{\mathcal{Q}} = \delta(\tilde{\phi})$	$\tilde{\phi} \sim \mathcal{P}$
SVGD	$\tilde{\mathcal{Q}} = \frac{1}{K}\sum_{k=1}^{K} \delta(\tilde{\phi}), \tilde{\boldsymbol{\phi}} = [\tilde{\phi}_1, .., \tilde{\phi}_k]^T$	$\tilde{\phi} \sim \mathcal{P},\ k = 1, .., K$
VI	$\tilde{\mathcal{Q}} = \mathcal{N}(\mu_{\mathcal{Q}}, \sigma^2_{\mathcal{Q}},\ v = (\mu_{\mathcal{Q}}, \sigma^2_{\mathcal{Q}})$	$v = (\mu_{\mathcal{Q}}, \sigma^2_{\mathcal{Q}})$

	Sample_prior_Params	Init_Approx_Inference
MAP	$\phi \leftarrow \tilde{\phi}$	$\tilde{\phi} \leftarrow \tilde{\phi} + \eta \nabla_\phi \ln \mathcal{Q}^*(\tilde{\phi})$
SVGD	$\phi_k \leftarrow \tilde{\phi}_k$	$\tilde{\boldsymbol{\phi}} \leftarrow \tilde{\boldsymbol{\phi}} + \eta \boldsymbol{K} \nabla_\phi \ln \mathcal{Q}^* + \nabla_\phi \boldsymbol{K}$
VI	$\phi_k \leftarrow \mu_{\mathcal{Q}} + \sigma_{\mathcal{Q}} \circ \epsilon,\ \epsilon \sim \mathcal{N}(0, \boldsymbol{I})$	$v \leftarrow v + \frac{\eta}{K}\sum_{k=1}^{K} \nabla_v \left[\ln \mathcal{Q}^*(\phi_k) - \ln \tilde{\mathcal{Q}}_v(\phi_k)\right]$

A.3 Log-Sum-Exp Operator

The Log-Sum-Exp (LSE) is effective in normalising vectors of log probabilities. In order to see this consider the following.

$$
\begin{aligned}
y &= \ln\left(\sum_{n=1}^{N} \exp x_n\right) \\
e^y &= \sum_{n=1}^{N} \exp x_n \\
&= e^c \sum_{n=1}^{N} \exp(x_n - c) \\
y &= c + \ln\left(\sum_{n=1}^{N} \exp(x_n - c)\right)
\end{aligned}
$$

This holds for an arbitrary constant $c \in \mathbb{R}$. If $c = \max\{x_1, ..., x_n\}$ is chosen we can ensure that the largest positive exponentiated term is $e^0 = 1$.