Appendix

A. Proofs and Derivations

A.1. Proof of Theorem 2

Lemma 2. ***(Change of measure inequality)*** *Let f be a random variable taking values in a set A and let $X_1, ..., X_l$ be independent random variables, with $X_k \in A$ with 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 μ_k for any fixed $f \in A$. Then for any fixed distributions $\pi, \rho \in \mathcal{M}(A)$ and any $\lambda > 0$, we have that*

$$\mathbb{E}_{f\sim\rho}\left[\sum_{k=1}^{l} \xi_k(f) \; - g_k(f, X_k)\right] \leq \frac{1}{\lambda}\left(D_{KL}(\rho||\pi) + \ln \mathbb{E}_{f\sim\pi}\left[e^{\lambda\left(\sum_{k=1}^{l} \xi_k(f) - g_k(f,X_k)\right)}\right]\right). \tag{10}$$

To prove the Theorem 2, we need to bound the difference between *transfer error* $\mathcal{L}(\mathcal{Q}, \mathcal{T})$ and the *empirical multi-task error* $\hat{\mathcal{L}}(\mathcal{Q}, S_1, ..., S_n)$. To this end, we introduce an intermediate quantity, the *expected multi-task error*:

$$\tilde{\mathcal{L}}(\mathcal{Q}, \mathcal{D}_1, ..., \mathcal{D}_n) = \mathbb{E}_{P\sim\mathcal{Q}}\left[\frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{S\sim D_i^{m_i}}\left[\mathcal{L}(Q(S, P), \mathcal{D}_i)\right]\right] \tag{11}$$

In the following we invoke Lemma 2 twice. First, in step 1, we bound the difference between $\tilde{\mathcal{L}}(\mathcal{Q}, \mathcal{D}_1, ..., \mathcal{D}_n)$ and $\hat{\mathcal{L}}(\mathcal{Q}, S_1, ..., S_n)$, then, in step 2, the difference between $\mathcal{L}(\mathcal{Q}, \mathcal{T})$ and $\tilde{\mathcal{L}}(\mathcal{Q}, \mathcal{D}_1, ..., \mathcal{D}_n)$. Finally, in step 3, we use a union bound argument to combine both results.

Step 1 (Task specific generalization) First, we bound the generalization error of the observed tasks $\tau_i = (\mathcal{D}_i, m_i)$, $i = 1, ..., n$, when using a learning algorithm $Q : \mathcal{M} \times \mathcal{Z}^{m_i} \to \mathcal{M}$, which outputs a posterior distribution $Q_i = Q(S_i, P)$ over hypotheses h, given a prior distribution P and a dataset $S_i \sim \mathcal{D}_i^{m_i}$ of size m_i. In that, we define $\tilde{m} := (\sum_{i=1}^{n} m_i^{-1})^{-1}$ as the harmonic mean of sample sizes.

In particular, we apply Lemma 2 to the union of all training sets $S' = \bigcup_{i=1}^{n} S_i$ with $l = \sum_{i=1}^{n} m_i$. Hence, 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 be a tuple of one prior and n base hypotheses. This can be understood as a two-level hypothesis, wherein P constitutes a hypothesis of the meta-learning problem and h_i a hypothesis for solving the supervised task τ_i. Correspondingly, we take $\pi = (\mathcal{Q}, Q^n) = \mathcal{P} \cdot \prod_{i=1}^{n} P$ and $\rho = (\mathcal{Q}, Q^n) = \mathcal{Q} \cdot \prod_{i=1}^{n} Q_i$ as joint two-level distributions and $g_k(f, X_k) = \frac{1}{nm_i} l(h_i, z_{ij})$ as summand of the empirical multi-task error. We can now invoke Lemma 2 to obtain that (12) and (15)

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{P\sim\mathcal{Q}}\left[\mathcal{L}(Q_i, D_i)\right] \leq & \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{P\sim\mathcal{Q}}\left[\mathcal{L}(Q_i, S_i)\right] + \frac{1}{\gamma}\Bigg(D_{KL}\left[(\mathcal{Q}, Q^n)||(\mathcal{P}, P^n)\right] \\ & + \ln \mathbb{E}_{P\sim\mathcal{P}}\mathbb{E}_{h\sim P}\left[e^{\frac{\gamma}{n}\sum_{i=1}^{n}(\mathcal{L}(h,\mathcal{D}_i) - \hat{\mathcal{L}}(h,S_i))}\right]\Bigg)\end{aligned} \tag{12}$$

Using the above definitions, the KL-divergence term can be re-written in the following way:

$$D_{KL}\left[(\mathcal{Q}, Q^n)||(\mathcal{P}, P^n)\right] = \mathbb{E}_{P\sim\mathcal{Q}}\left[\mathbb{E}_{h\sim Q_i}\left[\ln \frac{\mathcal{Q}(P)\prod_{i=1}^{n}\prod Q_i(h)}{\mathcal{P}(P)\prod_{i=1}^{n} P(h)}\right]\right] \tag{13}$$

$$= \mathbb{E}_{P\sim\mathcal{Q}}\left[\ln \frac{\mathcal{Q}(P)}{\mathcal{P}(P)}\right] + \sum_{i=1}^{n} \mathbb{E}_{P\sim\mathcal{Q}}\left[\mathbb{E}_{h\sim Q_i}\left[\ln \frac{Q_i(h)}{P(h)}\right]\right] \tag{14}$$

$$= D_{KL}(\mathcal{Q}||\mathcal{P}) + \sum_{i=1}^{n} \mathbb{E}_{P\sim\mathcal{Q}}\left[D_{KL}(Q_i||P)\right] \tag{15}$$

Using (12) and (15) we can bound the expected multi-task error as follows:

$$\begin{aligned}\tilde{\mathcal{L}}(\mathcal{Q}, \mathcal{D}_1, ..., \mathcal{D}_n) \leq \hat{\mathcal{L}}(\mathcal{Q}, S_1, ..., S_n) + \frac{1}{\gamma} D_{KL}(\mathcal{Q}||\mathcal{P}) + \frac{1}{\gamma} \sum_{i=1}^{n} \mathbb{E}_{P\sim\mathcal{Q}} \left[D_{KL}(Q_i||P)\right] \\ + \underbrace{\frac{1}{\gamma} \ln \mathbb{E}_{P\sim\mathcal{P}} \mathbb{E}_{h\sim P} \left[e^{\frac{\gamma}{n} \sum_{i=1}^{n} (\mathcal{L}(h,\mathcal{D}_i) - \hat{\mathcal{L}}(h,S_i))}\right]}_{\Upsilon^{\mathrm{I}}(\gamma)}\end{aligned} \tag{16}$$

Step 2 (Task environment generalization) Now, we apply Lemma 2 on the meta-level. For that, we treat each task as random variable and instantiate the components as $X_k = \tau_i$, $l = n$ and $\mu_k = \mathcal{T}$. Furthermore, we set $\rho = \mathcal{Q}$, $\pi = \mathcal{P}$, $f = P$ and $g_k(f, X_k) = \frac{1}{n}\mathcal{L}(Q_i, \mathcal{D}_i)$. This allows us to bound the transfer error as

$$\mathcal{L}(\mathcal{Q}, \mathcal{T}) \leq \tilde{\mathcal{L}}(\mathcal{Q}, \mathcal{D}_1, ..., \mathcal{D}_n) + \frac{1}{\lambda} D_{KL}(\rho||\pi) + \Upsilon^{\mathrm{II}}(\lambda) \tag{17}$$

wherein $\Upsilon^{\mathrm{II}}(\lambda) = \frac{1}{\lambda} \ln \mathbb{E}_{P\sim\mathcal{P}} \left[e^{\frac{\lambda}{n} \sum_{i=1}^{n} \mathbb{E}_{(D,S)\sim\mathcal{T}} [\mathcal{L}(Q(P,S),\mathcal{D})] - \mathcal{L}(Q(P,S_i),\mathcal{D}_i)}\right]$.

Combining (16) with (17), we obtain

$$\begin{aligned}\mathcal{L}(\mathcal{Q}, \mathcal{T}) \leq \hat{\mathcal{L}}(\mathcal{Q}, S_1, ..., S_n) + \left(\frac{1}{\lambda} + \frac{1}{\gamma}\right) D_{KL}(\mathcal{Q}||\mathcal{P}) \\ + \frac{1}{\gamma} \sum_{i=1}^{n} \mathbb{E}_{P\sim\mathcal{Q}} \left[D_{KL}(Q_i||P)\right] + \Upsilon^{\mathrm{I}}(\gamma) + \Upsilon^{\mathrm{II}}(\lambda)\end{aligned} \tag{18}$$

Step 3 (Bounding the moment generating functions)

$$\begin{aligned}e^{(\Upsilon^{\mathrm{I}}(\gamma) + \Upsilon^{\mathrm{II}}(\lambda))} =& \mathbb{E}_{P\sim\mathcal{P}} \left[e^{\frac{\lambda}{n} \sum_{i=1}^{n} \mathbb{E}_{(D,S)\sim\mathcal{T}} [\mathcal{L}(Q(P,S),\mathcal{D})] - \mathcal{L}(Q(P,S_i),\mathcal{D}_i)}\right]^{1/\lambda} \cdot \\ & \mathbb{E}_{P\sim\mathcal{P}} \mathbb{E}_{h\sim P} \left[e^{\frac{\gamma}{n} \sum_{i=1}^{n} (\mathcal{L}(h,\mathcal{D}_i) - \hat{\mathcal{L}}(h,S_i)}\right]^{1/\gamma} \\ =& \mathbb{E}_{P\sim\mathcal{P}} \left[\prod_{i=1}^{n} e^{\left(\frac{\lambda}{n} \mathbb{E}_{(D,S)\sim\mathcal{T}} [\mathcal{L}(Q(P,S),\mathcal{D}))] - \mathcal{L}(Q(P,S_i),\mathcal{D}_i)\right)}\right]^{1/\lambda} \cdot \\ & \mathbb{E}_{P\sim\mathcal{P}} \mathbb{E}_{h\sim P} \left[\prod_{i}^{n} \prod_{i}^{m_i} e^{\frac{\gamma}{n m_i} (\mathcal{L}(h,\mathcal{D}_i) - l(h_i,z_{ij}))}\right]^{1/\gamma}\end{aligned} \tag{19}$$

Case I: bounded loss

If the loss function $l(h_i, z_{ij})$ is bounded in $[a, b]$, we can apply Hoeffding's lemma to each factor in (19), obtaining:

$$e^{\Upsilon^{\mathrm{I}}(\gamma) + \Upsilon^{\mathrm{II}}(\lambda)} \leq \mathbb{E}_{P\sim\mathcal{P}} \left[e^{\frac{\lambda^2}{8n}(b-a)^2}\right]^{1/\lambda} \cdot \mathbb{E}_{P\sim\mathcal{P}} \mathbb{E}_{h\sim P} \left[e^{\frac{\gamma^2}{8n\tilde{m}}(b-a)^2}\right]^{1/\gamma} \tag{20}$$

$$= e^{\left(\frac{\lambda}{8n} + \frac{\gamma}{8n\tilde{m}}\right)(b-a)^2} \tag{21}$$

Next, we factor out $\sqrt{n}$ from γ and λ, obtaining

$$e^{\Upsilon^{\mathrm{I}}(\gamma) + \Upsilon^{\mathrm{II}}(\lambda)} = \left(e^{\Upsilon^{\mathrm{I}}(\gamma\sqrt{n}) + \Upsilon^{\mathrm{II}}(\lambda\sqrt{n})}\right)^{\frac{1}{\sqrt{n}}} \tag{22}$$

Using

$$\mathbb{E}_{\mathcal{T}} \mathbb{E}_{\mathcal{D}_1} ... \mathbb{E}_{\mathcal{D}_n} \left[e^{\Upsilon^{\mathrm{I}}(\gamma\sqrt{n}) + \Upsilon^{\mathrm{II}}(\lambda\sqrt{n})}\right] \leq e^{\left(\frac{\lambda}{8\sqrt{n}} + \frac{\gamma}{8\sqrt{n}\tilde{m}}\right)(b-a)^2} \tag{23}$$

we can apply Markov's inequality w.r.t. the expectations over the task distribution $\mathcal{T}$ and data distributions $\mathcal{D}_i$ to obtain that

$$\Upsilon^{\text{I}}(\gamma) + \Upsilon^{\text{II}}(\lambda) \leq \underbrace{\frac{\lambda}{8n}(b-a)^2}_{\Psi^{\text{I}}(\lambda)} + \underbrace{\frac{\gamma}{8n\tilde{m}}(b-a)^2}_{\Psi^{\text{II}}(\gamma)} - \frac{1}{\sqrt{n}} \ln \delta \tag{24}$$

with probability at least $1-\delta$.

Case II: sub-gamma loss

First, we assume that, $\forall i = 1, ..., n$ the random variables $V_i^{\text{I}} := \mathcal{L}(h, \mathcal{D}_i) - l(h_i, z_{i,j})$ 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 $\mathcal{D}_i$. That is, their moment generating function can be bounded by that of a Gamma distribution $\Gamma(s_{\text{I}}^2, c_{\text{I}})$:

$$\mathbb{E}_{z\sim\mathcal{D}_i}\mathbb{E}_{P\sim\mathcal{P}}\mathbb{E}_{h\sim P}\left[e^{\gamma(\mathcal{L}(h,\mathcal{D}_i)-l(h,z))}\right] \leq \exp\left(\frac{\gamma^2 s_{\text{I}}^2}{2(1-c_{\text{I}}\gamma)}\right) \quad \forall \gamma \in (0, 1/c_{\text{I}}) \tag{25}$$

Second, we assume that, the random variable $V^{\text{II}} := \mathbb{E}_{(D,S)\sim\mathcal{T}}\left[\mathcal{L}(Q(P,S),\mathcal{D})\right] - \mathcal{L}(Q(P,S_i),\mathcal{D}_i)$ 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 $\Gamma(s_{\text{II}}^2, c_{\text{II}})$:

$$\mathbb{E}_{(\mathcal{D},S)\sim\mathcal{T}}\mathbb{E}_{P\sim\mathcal{P}}\left[e^{\lambda\,\mathbb{E}_{(D,S)\sim\mathcal{T}}[\mathcal{L}(Q(P,S),\mathcal{D})]-\mathcal{L}(Q(P,S),\mathcal{D})}\right] \leq \exp\left(\frac{\lambda^2 s_{\text{II}}^2}{2(1-c_{\text{II}}\lambda)}\right) \quad \forall \lambda \in (0, 1/c_{\text{II}}) \tag{26}$$

These two assumptions allow us to bound the expectation of (19) as follows:

$$\mathbb{E}\left[e^{\Upsilon^{\text{I}}(\gamma)+\Upsilon^{\text{II}}(\lambda)}\right] \leq \exp\left(\frac{\gamma s_{\text{I}}^2}{2n\tilde{m}(1-c_{\text{I}}\gamma/(n\tilde{m})}\right) \cdot \exp\left(\frac{\lambda s_{\text{II}}^2}{2n(1-c_{\text{II}}\lambda/n)}\right) \tag{27}$$

Next, we factor out $\sqrt{n}$ from γ and λ, obtaining

$$e^{\Upsilon^{\text{I}}(\gamma)+\Upsilon^{\text{II}}(\lambda)} = \left(e^{\Upsilon^{\text{I}}(\gamma\sqrt{n})+\Upsilon^{\text{II}}(\lambda\sqrt{n})}\right)^{\frac{1}{\sqrt{n}}} \tag{28}$$

Finally, by using Markov's inequality we obtain that

$$\Upsilon^{\text{I}}(\gamma) + \Upsilon^{\text{II}}(\lambda) \leq \underbrace{\frac{\gamma s_{\text{I}}^2}{2n\tilde{m}(1-c_{\text{I}}\gamma/(n\tilde{m})}}_{\Psi^{\text{I}}(\gamma)} + \underbrace{\frac{\lambda s_{\text{II}}^2}{2n(1-c_{\text{II}}\lambda/n)}}_{\Psi^{\text{II}}(\lambda)} - \frac{1}{\sqrt{n}} \ln \delta \tag{29}$$

with probability at least $1-\delta$.

To conclude the proof, we choose $\gamma = n\beta$ for $\beta > 0$.

A.2. Proof of Corollary 1

When we choose the posterior Q as the optimal Gibbs posterior $Q_i^* := Q^*(S_i, P)$, it follows that

$$\hat{\mathcal{L}}(\mathcal{Q}, S_1, ..., S_n) + \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\mathbb{E}_{P\sim\mathcal{Q}}\left[D_{KL}(Q_i^*||P)\right] \tag{30}$$

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

$$= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\left(\mathbb{E}_{P\sim\mathcal{Q}}\mathbb{E}_{h\sim Q_i^*}\left[\beta\hat{\mathcal{L}}(h,S_i) + \ln\frac{Q_i^*(h)}{P(h)}\right]\right) \tag{32}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\left(\mathbb{E}_{P\sim\mathcal{Q}}\mathbb{E}_{h\sim Q_i^*}\left[\beta\hat{\mathcal{L}}(h,S_i) + \ln\frac{P(h)e^{-\beta\hat{\mathcal{L}}(h,S_i)}}{P(h)Z_\beta(S_i,P)}\right]\right) \tag{33}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\left(-\mathbb{E}_{P\sim\mathcal{Q}}\left[\ln Z_\beta(S_i,P)\right]\right) \ . \tag{34}$$

This allows us to write the inequality in (4) as

$$\mathcal{L}(\mathcal{Q},\mathcal{T}) \leq \; -\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\mathbb{E}_{P\sim\mathcal{Q}}\left[\ln Z_\beta(S_i,P)\right] + \left(\frac{1}{\lambda}+\frac{1}{n\beta}\right) D_{KL}(\mathcal{Q}||\mathcal{P}) + C(\delta,n,\tilde{m}) \; . \tag{35}$$

According to Lemma 1, the Gibbs posterior $Q^*(S_i,P)$ is the minimizer of (32), in particular $\forall P \in \mathcal{M}(\mathcal{H}), \forall i = 1,...,n$:

$$Q^*(S_i,P) = \frac{P(h)e^{-\beta\hat{\mathcal{L}}(h,S_i)}}{Z_\beta(S_i,P)} = \underset{Q\in\mathcal{M}(\mathcal{H})}{\arg\min}\, \mathbb{E}_{h\sim Q}\left[\hat{\mathcal{L}}(h,S_i)\right] + \frac{1}{\beta}D_{KL}(Q||P) \; . \tag{36}$$

Hence, we can write

$$\begin{aligned}
\mathcal{L}(\mathcal{Q},\mathcal{T}) \leq & -\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\mathbb{E}_{P\sim\mathcal{Q}}\left[\ln Z_\beta(S_i,P)\right] + \left(\frac{1}{\lambda}+\frac{1}{n\beta}\right) D_{KL}(\mathcal{Q}||\mathcal{P}) + C(\delta,\lambda,\beta) && (37)\\
= & \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{P\sim\mathcal{Q}}\left[\min_{Q\in\mathcal{M}(\mathcal{H})}\hat{\mathcal{L}}(Q,S_i) + \frac{1}{\beta}D_{KL}(Q||P)\right] && (38)\\
& + \left(\frac{1}{\lambda}+\frac{1}{n\beta}\right) D_{KL}(\mathcal{Q}||\mathcal{P}) + C(\delta,n,\tilde{m}) && (39)\\
\leq & \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{P\sim\mathcal{Q}}\left[\hat{\mathcal{L}}(Q,S_i) + \frac{1}{\beta}D_{KL}(Q||P)\right] && (40)\\
& + \left(\frac{1}{\lambda}+\frac{1}{n\beta}\right) D_{KL}(\mathcal{Q}||\mathcal{P}) + C(\delta,\lambda,\beta) && (41)\\
= & \hat{\mathcal{L}}(\mathcal{Q},S_1,...,S_n) + \left(\frac{1}{\lambda}+\frac{1}{n\beta}\right) D_{KL}(\mathcal{Q}||\mathcal{P}) && (42)\\
& + \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\beta}\mathbb{E}_{P\sim\mathcal{Q}}\left[D_{KL}(Q_i||P)\right] + C(\delta,\lambda,\beta) \; , && (43)
\end{aligned}$$

which proves that the bound for Gibbs-optimal base learners in (35) and (5) is tighter than the bound in Theorem 2 which holds uniformly for all $Q \in \mathcal{M}(\mathcal{H})$.

A.3. Proof of Proposition 1: PAC-Optimal Hyper-Posterior

An objective function corresponding to (5) reads as

$$J(\mathcal{Q}) = -\mathbb{E}_{\mathcal{Q}}\left[\frac{\lambda}{n\beta+\lambda}\sum_{i=1}^{n}\ln Z(S_i,P)\right] + D_{KL}(\mathcal{Q}||\mathcal{P}) \; . \tag{44}$$

To obtain $J(\mathcal{Q})$, we omit all additive terms from (5) that do not depend on $\mathcal{Q}$ and multiply by the scaling factor $\frac{\lambda n\beta}{n\beta+\lambda}$. Since the described transformations are monotone, the minimizing distribution of $J(\mathcal{Q})$, that is,

$$\mathcal{Q}^* = \underset{\mathcal{Q}\in\mathcal{M}(\mathcal{M}(\mathcal{H}))}{\arg\min}\; J(\mathcal{Q}) \; , \tag{45}$$

is also the minimizer of (5). More importantly, $J(\mathcal{Q})$ is structurally similar to the generic minimization problem in (3). Hence, we can invoke Lemma 1 with $A = \mathcal{M}(\mathcal{H})$, $g(a) = -\sum_{i=1}^{n}\ln Z(S_i,P)$, $\beta = \frac{1}{\sqrt{n\tilde{m}}+1}$, to show that the optimal hyper-posterior is

$$\mathcal{Q}^*(P) = \frac{\mathcal{P}(P)\exp\left(\frac{\lambda}{n\beta+\lambda}\sum_{i=1}^{n}\ln Z_\beta(S_i,P)\right)}{Z^{\text{II}}(S_1,...,S_n,\mathcal{P})} \; , \tag{46}$$

wherein

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

□

Technically, this concludes the proof of Proposition 1. However, we want to remark the following result:

If we choose $\mathcal{Q} = \mathcal{Q}^*$, the PAC-Bayes bound in (5) can be expressed in terms of the meta-level partition function Z^{II}, that is,

$$\mathcal{L}(\mathcal{Q}, \mathcal{T}) \leq -\left(\frac{1}{\lambda} + \frac{1}{n\beta}\right) \ln Z^{\text{II}}(S_1, ..., S_n, \mathcal{P}) + C(\delta, \lambda, \beta) \ . \tag{47}$$

We omit a detailed derivation of (47) since it is similar to the one for Corollary 1.

B. Gaussian process regression

In GP regression, each data point corresponds to a feature-target tuple $z_{i,j} = (x_{i,j}, y_{i,j}) \in \mathbb{R}^d \times \mathbb{R}$. For the i-th dataset, we write $S_i = (\mathbf{X}_i, \mathbf{y}_i)$, where $\mathbf{X}_i = (x_{i,1}, ..., x_{i,m_i})^\top$ and $\mathbf{y}_i = (y_{i,1}, ..., y_{i,m_i})^\top$. GPs are a Bayesian method in which the prior $P_\phi(h) = \mathcal{GP}\left(h | m_\phi(x), k_\phi(x, x')\right)$ is specified by a positive definite kernel $k_\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and a mean function $m_\phi : \mathcal{X} \to \mathbb{R}$.

The empirical loss under the GP posterior Q^* coincides with the negative log-likelihood of regression targets $\mathbf{y}_i$, that is, $\hat{\mathcal{L}}(Q^*, S_i) = -\frac{1}{m_i} \ln p(\mathbf{y}_i | \mathbf{X}_i)$. Under a Gaussian likelihood $p(\mathbf{y}|\mathbf{h}) = \mathcal{N}(\mathbf{y}; h(\mathbf{x}), \sigma^2 I)$, the marginal log-likelihood $\ln Z(S_i, P_\phi) = \ln p(\mathbf{y}_i | \mathbf{X}_i, \phi)$ can be computed in closed form as

$$\ln p(\mathbf{y}|\mathbf{X}, \phi) = -\frac{1}{2}\left(\mathbf{y} - m_{\mathbf{X},\phi})\right)^\top \tilde{K}_{\mathbf{X},\phi}^{-1}\left(\mathbf{y} - m_{\mathbf{X},\phi}\right) - \frac{1}{2} \ln |\tilde{K}_{\mathbf{X},\phi}| - \frac{m_i}{2} \ln 2\pi \ , \tag{48}$$

where $\tilde{K}_{\mathbf{X},\phi} = K_{\mathbf{X},\phi} + \sigma^2 I$, with the kernel matrix $K_{\mathbf{X},\phi} = (k_\phi(x_l, x_k))_{l,k=1}^{m_i}$, observation noise variance σ^2, and mean vector $m_{\mathbf{X},\phi} = (m_\phi(x_1), ..., m_\phi(x_{m_i}))^\top$.

Previous work on Bayesian model selection in the context of GPs argues that the log-determinant $\frac{1}{2} \ln |\tilde{K}_{\mathbf{X},\phi}|$ in the marginal log-likelihood (48) acts as a complexity penalty (Rasmussen & Ghahramani, 2001; Rasmussen & Williams, 2006). However, we suspect that this complexity regularization is only effective if the class of considered priors is restrictive, for instance if we only optimize a small number of parameters such as the length- and output scale of a squared exponential kernel. If we consider expressive classes of GP priors (e.g., our setup where the mean and kernel function are neural networks), such a complexity penalty could be insufficient to avoid meta-overfitting. Indeed, this is what we also observe in our experiments (see Sec. 5.3).

C. PACOH-GP: Meta-Learning of GP priors

In this section, we provide further details on PACOH-GP, introduced in Section 5 of the paper and employed in our experiments. Following Section 5.3, we instantiate our framework with GP base learners. Since we are interested in meta-learning, we define the mean and kernel function both as parametric functions. Similar to Wilson et al. (2016) and Fortuin & Rätsch (2019), we instantiate m_ϕ and k_ϕ as neural networks, where the parameter vector ϕ can be meta-learned. To ensure the positive-definiteness of the kernel, we use the neural network as feature map $\Phi_\phi(x)$ on top of which we apply a squared exponential (SE) kernel. Accordingly, the parametric kernel reads as $k_\phi(x, x') = \frac{1}{2} \exp\left(-||\Phi_\phi(x) - \Phi_\phi(x')||_2^2\right)$. Both $m_\phi(x)$ and $\Phi_\phi(x)$ are fully-connected neural networks with 4 layers with each 32 neurons and $\tanh$ non-linearities. The parameter vector ϕ represents the weights and biases of both neural networks. As hyper-prior we choose a zero-mean isotropic Gaussian, that is, $\mathcal{P}(\phi) = \mathcal{N}(0, \sigma_{\mathcal{P}}^2 I)$. Further, we choose $\lambda = n$ and $\beta_i = m_i$

C.1. Meta-training with PACOH-GP

SVGD (Liu & Wang, 2016) approximates $\mathcal{Q}^*$ as a set of particles $\hat{\mathcal{Q}} = \{P_1, ..., P_K\}$. In our described setup, each particle corresponds to the parameters of the GP prior, that is, $\hat{\mathcal{Q}} = \{\phi_1, ..., \phi_K\}$. Initially, we sample random priors $\phi_k \sim \mathcal{P}$ from our hyper-posterior. Then, the SVGD iteratively transports the set of particles to match $\mathcal{Q}^*$, by applying a form of functional gradient descent that minimizes $D_{KL}(\hat{\mathcal{Q}} | \mathcal{Q}^*)$ in the reproducing kernel Hilbert space induced by a kernel function $k(\cdot, \cdot)$. We choose a squared exponential kernel with length scale (hyper-)parameter ℓ, that is, $k(\phi, \phi') = \exp\left(-\frac{||\phi - \phi'||_2^2}{2\ell}\right)$. In

each iteration, the particles are updated by

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

Here, we can again estimate $\nabla_{\phi_l} \ln \mathcal{Q}^*(\phi_l)$ with a mini-batch of H datasets $S_1, ..., S_H$:

$$\nabla_{\phi_l} \ln \mathcal{Q}^*(\phi_l) = \frac{n}{H} \cdot \sum_{h=1}^{H} \frac{1}{m_h + 1} \nabla_{\phi_l} \ln Z(S_h, P_{\phi_l}) + \nabla_{\phi_l} \ln \mathcal{P}(\phi_l) \,.$$

Importantly, $\nabla_{\phi_l} \ln \mathcal{Q}^*(\phi_l)$ does not depend on Z^{II} which makes SVGD tractable. Algorithm 2 summarizes the meta-training method for GP priors.

Algorithm 2 PACOH-GP: mini-batched meta-training

Input: hyper-prior $\mathcal{P}$, datasets $S_1, ..., S_n$
Input: SVGD kernel function $k(\cdot, \cdot)$, SVGD step size η, number of particles K
$\{\phi_1, ..., \phi_K\} \sim \mathcal{P}$ // Initialize prior particles
while not converged **do**
  $\{T_1, ..., T_{n_{bs}}\} \subseteq [n]$ // sample n_{bs} tasks uniformly at random
  for $k = 1, ..., K$ **do**
    for $i = 1, ..., n_{bs}$ **do**
      $\ln Z_{m_i}(S_i, P_{\phi_k}) \leftarrow -\frac{1}{2} \left(\mathbf{y}_i - m_{\mathbf{X}_i, \phi_k}\right)^\top \tilde{K}^{-1}_{\mathbf{X}_i, \phi_k} \left(\mathbf{y}_i - m_{\mathbf{X}_i, \phi_k}\right) - \frac{1}{2} \ln |\tilde{K}_{\mathbf{X}_i, \phi_k}| - \frac{m_i}{2} \ln 2\pi$ // compute MLL
    $\nabla_{\phi_k} \ln \tilde{\mathcal{Q}}^*(\phi_k) \leftarrow \nabla_{\phi_k} \ln \mathcal{P}(\phi_k) + \frac{n}{n_{bs}} \sum_{i=1}^{n_{bs}} \frac{1}{m_i + 1} \nabla_{\phi_k} \ln Z_{m_i}(S_i, P_{\phi_k})$ // compute score
  $\phi_k \leftarrow \phi_k + \frac{\eta}{K} \sum_{k'=1}^{K} \left[k(\phi_{k'}, \phi_k) \nabla_{\phi_{k'}} \ln \tilde{\mathcal{Q}}^*(\phi_{k'}) + \nabla_{\phi_{k'}} k(\phi_{k'}, \phi_k) \right] \forall k \in [K]$ // SVGD update
Output: set of GP priors $\{\mathcal{GP}\left(m_{\phi_1}(x), k_{\phi_1}(x, x')\right), ..., \mathcal{GP}\left(m_{\phi_K}(x), k_{\phi_K}(x, x')\right)\}$

C.2. Meta-Testing / Target-Training with PACOH-GP

Meta-learning with PACOH, as described above, gives us an approximation of $\mathcal{Q}^*$. In target-testing (see Figure 1), the base learner is instantiated with the meta-learned prior P_ϕ, receives a dataset $\tilde{S} = (\tilde{\mathbf{X}}, \tilde{\mathbf{y}})$ from an unseen task $\mathcal{D} \sim \mathcal{T}$ and outputs a posterior Q as product of its inference. In our GP setup, Q is the GP posterior and the predictive distribution $\hat{p}(y^*|x^*, \tilde{\mathbf{X}}, \tilde{\mathbf{y}}, \phi)$ is a Gaussian (for details see Rasmussen & Williams, 2006).

Since the meta-learner outputs a distribution over priors, that is, the hyper-posterior $\mathcal{Q}$, we may obtain different predictions for different priors $P_\phi \sim \mathcal{Q}$, sampled from the hyper-posterior. To obtain a predictive distribution under our meta-learned hyper-posterior, we empirically marginalize $\mathcal{Q}$. That is, we draw a set of prior parameters $\phi_1, ..., \phi_K \sim \mathcal{Q}$ from the hyper-posterior, compute their respective predictive distributions $\hat{p}(y^*|x^*, \tilde{\mathbf{X}}, \tilde{\mathbf{y}}, \phi_k)$ and form an equally weighted mixture:

$$\hat{p}(y^*|x^*, \tilde{\mathbf{X}}, \tilde{\mathbf{y}}, \mathcal{Q}) = \mathbb{E}_{\phi \sim \mathcal{Q}} \left[\hat{p}(y^*|x^*, \tilde{\mathbf{X}}, \tilde{\mathbf{y}}, \phi) \right] \approx \frac{1}{K} \sum_{k=0}^{K} \hat{p}(y^*|x^*, \tilde{\mathbf{X}}, \tilde{\mathbf{y}}, \phi_k) \,, \quad \phi_k \sim \mathcal{Q} \tag{49}$$

Since we are concerned with GPs, (49) coincides with a mixture of Gaussians. As one would expect, the mean prediction under $\mathcal{Q}$ (i.e., the expectation of (49)), is the average of the mean predictions corresponding to the sampled prior parameters $\phi_1, ..., \phi_K$. In case of PACOH-VI, we sample $K = 100$ priors from the variational hyper-posterior $\tilde{\mathcal{Q}}$. For PACOH-SVGD, samples from the hyper-posterior correspond to the $K = 10$ particles. PACOH-MAP can be viewed as a special case of SVGD with $K = 1$, that is, only one particle. Thus, $\hat{p}(y^*|x^*, \tilde{\mathbf{X}}, \tilde{\mathbf{y}}, \mathcal{Q}) \approx \hat{p}(y^*|x^*, \tilde{\mathbf{X}}, \tilde{\mathbf{y}}, \phi^{MAP})$ is a single Gaussian.

D. PACOH-NN: Meta-Learning BNN priors

In this section, we summarize and further discuss our proposed meta-learning algorithm *PACOH-NN*. An overview of our proposed framework is illustrated in Figure 1. Overall, it consists of two stages *meta-training* and *meta-testing* which we explain in more details in the following.

D.1. Meta-training

The hyper-posterior distribution $\mathcal{Q}$ that minimizes the upper bound on the transfer error is given by

$$\mathcal{Q}^*(P) = \frac{\mathcal{P}(P) \exp\left(\sum_{i=1}^n \frac{\lambda}{n\beta_i + \lambda} \ln \tilde{Z}(S_i, P)\right)}{Z^{\text{II}}(S_1, ..., S_n, \mathcal{P})} \tag{50}$$

In that, we no longer assume that $m = m_i \; \forall i = 1, ..., n$ which was done in the theory section to maintain notational brevity. Thus, we use a different β_i for each och the tasks as we want to set $\beta_i = m_i$ or $\beta_i = \sqrt{m_i}$. Provided with a set of datasets $S_1, ..., S_n$, the meta-learner minimizes the respective meta-objective, in the case of *PACOH-SVGD*, by performing SVGD on the $\mathcal{Q}^*$. Algorithm 3 outlines the required steps in more detail.

Algorithm 3 PACOH-NN: meta-training

Input: hyper-prior $\mathcal{P}$, datasets $S_1, ..., S_n$, kernel $k(\cdot, \cdot)$, step size η, number of particles K
$\{\phi_1, ..., \phi_K\} \sim \mathcal{P}$ // Initialize prior particles
while not converged **do**
  for $k = 1, ..., K$ **do**
    $\{\theta_1, ..., \theta_L\} \sim P_{\phi_k}$ // sample NN-parameters from prior
    for $i = 1, ..., n$ **do**
      $\ln \tilde{Z}(S_i, P_{\phi_k}) \leftarrow \text{LSE}_{l=1}^L \left(-\beta_i \hat{\mathcal{L}}(\theta_l, S_i)\right) - \ln L$ // estimate generalized MLL
    $\nabla_{\phi_k} \ln \tilde{\mathcal{Q}}^*(\phi_k) \leftarrow \nabla_{\phi_k} \ln \mathcal{P}(\phi_k) + \sum_{i=1}^n \frac{\lambda}{n\beta_i + \lambda} \nabla_{\phi_k} \ln \tilde{Z}(S_i, P_{\phi_k})$ // compute score
  $\forall k \in [K] : \phi_k \leftarrow \phi_k + \frac{\eta}{K} \sum_{k'=1}^K \left[k(\phi_{k'}, \phi_k) \nabla_{\phi_{k'}} \ln \tilde{\mathcal{Q}}^*(\phi_{k'}) + \nabla_{\phi_{k'}} k(\phi_{k'}, \phi_k)\right]$ // SVGD update
Output: set of priors $\{P_{\phi_1}, ..., P_{\phi_K}\}$

Alternatively, to estimate the score of $\nabla_{\phi_k} \tilde{\mathcal{Q}}^*(\phi_k)$ we can use mini-batching at both the task and the dataset level. Specifically, for a given meta-batch size of n_{bs} and a batch size of m_{bs}, we get Algorithm 4.

Algorithm 4 PACOH-NN-SVGD: mini-batched meta-training

Input: hyper-prior $\mathcal{P}$, datasets $S_1, ..., S_n$
Input: kernel function $k(\cdot, \cdot)$, SVGD step size η, number of particles K
$\{\phi_1, ..., \phi_K\} \sim \mathcal{P}$ // Initialize prior particles
while not converged **do**
  $\{T_1, ..., T_{n_{bs}}\} \subseteq [n]$ // sample n_{bs} tasks uniformly at random
  for $i = 1, ..., n_{bs}$ **do**
    $\tilde{S}_i \leftarrow \{z_1, ..., z_{m_{bs}}\} \subseteq S_{T_i}$ // sample m_{bs} datapoints from S_{T_i} uniformly at random
  for $k = 1, ..., K$ **do**
    $\{\theta_1, ..., \theta_L\} \sim P_{\phi_k}$ // sample NN-parameters from prior
    for $i = 1, ..., n_{bs}$ **do**
      $\ln \tilde{Z}(\tilde{S}_i, P_{\phi_k}) \leftarrow \text{LSE}_{l=1}^L \left(-\beta_i \hat{\mathcal{L}}(\theta_l, \tilde{S}_i)\right) - \ln L$ // estimate generalized MLL
    $\nabla_{\phi_k} \ln \tilde{\mathcal{Q}}^*(\phi_k) \leftarrow \nabla_{\phi_k} \ln \mathcal{P}(\phi_k) + \frac{n}{n_{bs}} \sum_{i=1}^{n_{bs}} \frac{\lambda}{n\beta_i + \lambda} \nabla_{\phi_k} \ln \tilde{Z}(S_i, P_{\phi_k})$ // compute score
  $\phi_k \leftarrow \phi_k + \frac{\eta}{K} \sum_{k'=1}^K \left[k(\phi_{k'}, \phi_k) \nabla_{\phi_{k'}} \ln \tilde{\mathcal{Q}}^*(\phi_{k'}) + \nabla_{\phi_{k'}} k(\phi_{k'}, \phi_k)\right] \forall k \in [K]$ // SVGD update
Output: set of priors $\{P_{\phi_1}, ..., P_{\phi_K}\}$

D.2. Meta-testing

The meta-learned prior knowledge is now deployed by a base learner. The base learner is given a training dataset $\tilde{S} \sim \mathcal{D}$ pertaining to an unseen task $\tau = (\mathcal{D}, m) \sim \mathcal{T}$. With the purpose of approximating the generalized Bayesian posterior $Q^*(S, P)$, the base learner performs (normal) posterior inference. Algorithm 5 details the steps of the approximating procedure referred to as *target training* when performed via SVGD. For a data point x^*, the respective predictor outputs a probability distribution given as $\tilde{p}(y^*|x^*, \tilde{S}) \leftarrow \frac{1}{K \cdot L} \sum_{k=1}^K \sum_{l=1}^L p(y^*|h_{\theta_l^k}(x^*))$. We evaluate the quality of the predictions on a held-out test dataset $\tilde{S}^* \sim \mathcal{D}$ from the same task, in a *target testing* phase (see Appendix E.2).

Algorithm 5 PACOH-NN: meta-testing

Input: set of priors $\{P_{\phi_1}, ..., P_{\phi_K}\}$, target training dataset $\tilde{S}$, evaluation point x^*
Input: kernel function $k(\cdot, \cdot)$, SVGD step size ν, number of particles L
for $k = 1, ..., K$ **do**
  $\{\theta_1^k, ..., \theta_L^k\} \sim P_{\phi_k}$ // initialize NN posterior particles from k-th prior
  while not converged **do**
    for $l = 1, ..., L$ **do**
      $\nabla_{\theta_l^k} Q^*(\theta_l^k)) \leftarrow \nabla_{\theta_l^k} \ln P_{\phi_k}(\theta_l^k)) + \beta \, \nabla_{\theta_l^k} \mathcal{L}(l, \tilde{S})$ // compute score
    $\theta_l^k \leftarrow \theta_l^k + \frac{\nu}{L} \sum_{l'=1}^{L} \left[k(\theta_{l'}^k, \theta_l^k) \nabla_{\theta_{l'}^k} \ln Q^*(\theta_{l'}^k) + \nabla_{\theta_{l'}^k} k(\theta_{l'}^k, \theta_l^k) \right] \forall l \in [L]$ // update particles
Output: a set of NN parameters $\bigcup_{k=1}^{K} \{\theta_1^k ..., \theta_L^k\}$

D.3. Properties of the score estimator

Since the marginal log-likelihood of BNNs is intractable, we have replaced it by a numerically stable Monte Carlo estimator $\ln \tilde{Z}_\beta(S_i, P_\phi)$ in (9), in particular

$$\ln \tilde{Z}_\beta(S_i, P_\phi) := \ln \frac{1}{L} \sum_{l=1}^{L} e^{-\beta \hat{\mathcal{L}}(\theta_l, S_i)} = \text{LSE}_{l=1}^{L} \left(-\beta \hat{\mathcal{L}}(\theta_l, S_i) \right) - \ln L \, , \;\; \theta_l \sim P_\phi \, . \tag{51}$$

Since the Monte Carlo estimator involves approximating an expectation of an exponential, it is not unbiased. However, we can show that replacing $\ln Z_\beta(S_i, P_\phi)$ by the estimator $\ln \tilde{Z}_\beta(S_i, P_\phi)$, we still minimize a valid upper bound on the transfer error (see Proposition 2).

Proposition 2. *In expectation, replacing* $\ln Z_\beta(S_i, P_\phi)$ *in (5) by the Monte Carlo estimate* $\ln \tilde{Z}_\beta(S_i, P) := \ln \frac{1}{L} \sum_{l=1}^{L} e^{-\beta \hat{\mathcal{L}}(\theta_l, S_i)}$, $\theta_l \sim P$ *still yields an valid upper bound of the transfer error. In particular, it holds that*

$$\mathcal{L}(\mathcal{Q}, \mathcal{T}) \leq -\frac{1}{n} \sum_{i=1}^{n} \frac{1}{\beta} \mathbb{E}_{P \sim \mathcal{Q}} \left[\ln Z(S_i, P) \right] + \left(\frac{1}{\lambda} + \frac{1}{n\beta} \right) D_{KL}(\mathcal{Q} || \mathcal{P}) + C(\delta, n, \tilde{m}) \tag{52}$$

$$\begin{aligned} &\leq -\frac{1}{n} \sum_{i=1}^{n} \frac{1}{\beta} \mathbb{E}_{P \sim \mathcal{Q}} \left[\mathbb{E}_{\theta_1, ..., \theta_L \sim P} \left[\ln \tilde{Z}(S_i, P) \right] \right] \\ &\quad + \left(\frac{1}{\lambda} + \frac{1}{n\beta} \right) D_{KL}(\mathcal{Q} || \mathcal{P}) + C(\delta, \lambda, \beta). \end{aligned} \tag{53}$$

Proof. Firsts, we show that:

$$\begin{aligned} \mathbb{E}_{\theta_1, ..., \theta_L \sim P} \left[\ln \tilde{Z}_\beta(S_i, P) \right] &= \mathbb{E}_{\theta_1, ..., \theta_L \sim P} \left[\ln \frac{1}{L} \sum_{l=1}^{L} e^{-\beta \hat{\mathcal{L}}(\theta_l, S_i)} \right] \\ &\leq \ln \frac{1}{L} \sum_{l=1}^{L} \mathbb{E}_{\theta_l \sim P} \left[e^{-\beta \hat{\mathcal{L}}(\theta_l, S_i)} \right] \\ &= \ln \mathbb{E}_{\theta \sim P} \left[e^{-\beta \hat{\mathcal{L}}(\theta, S_i)} \right] \\ &= \ln Z_\beta(S_i, P) \end{aligned} \tag{54}$$

which follows directly from Jensen's inequality and the concavity of the logarithm. Now, Proposition 2 follows directly from (54). □

In fact, by the law of large numbers, it is straightforward to show that as $L \to \infty$, the $\ln \tilde{Z}(S_i, P) \xrightarrow{\text{a.s.}} \ln Z(S_i, P)$, that is, the estimator becomes asymptotically unbiased and we recover the original PAC-Bayesian bound (i.e. (53) $\xrightarrow{\text{a.s.}}$ (52)). Also

it is noteworthy that the bound in (53) we get by our estimator is, in expectation, tighter than the upper bound when using the naive estimator

$$\ln \hat{Z}_\beta(S_i, P) := -\beta \, \frac{1}{L} \sum_{l=1}^{L} \hat{\mathcal{L}}(\theta_l, S_i) \quad \theta_l \sim P_\phi$$

which can be obtained by applying Jensen's inequality to $\ln \mathbb{E}_{\theta \sim P_\phi} \left[e^{-\beta \hat{\mathcal{L}}(\theta, S_i)} \right]$. In the edge case $L = 1$ our LSE estimator $\ln \tilde{Z}_\beta(S_i, P)$ falls back to this naive estimator and coincides in expectation with $\mathbb{E}[\ln \hat{Z}_\beta(S_i, P)] = -\beta \, \mathbb{E}_{\theta \sim P} \hat{\mathcal{L}}(\theta_l, S_i)$. As a result, we effectively minimize the looser upper bound

$$\mathcal{L}(\mathcal{Q}, \mathcal{T}) \leq \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\theta \sim P} \left[\hat{\mathcal{L}}(\theta, S_i) \right] + \left(\frac{1}{\lambda} + \frac{1}{n\beta} \right) D_{KL}(\mathcal{Q} || \mathcal{P}) + C(\delta, n, \tilde{m}). \tag{55}$$

$$= \mathbb{E}_{\theta \sim P} \left[\frac{1}{n} \sum_{i=1}^{n} \frac{1}{m_i} \sum_{j=1}^{m_i} - \ln p(y_{ij} | x_{ij}, \theta) \right] + \left(\frac{1}{\lambda} + \frac{1}{n\beta} \right) D_{KL}(\mathcal{Q} || \mathcal{P}) + C(\delta, n, \tilde{m}) \tag{56}$$

As we can see from (56), the boundaries between the tasks vanish in the edge case of $L = 1$, that is, all data-points are treated as if they would belong to one dataset. This suggests that L should be chosen greater than one. In our experiments, we used $L = 5$ and found the corresponding approximation to be sufficient.

E. Experiments

E.1. Meta-Learning Environments

In this section, we provide further details on the meta-learning environments used in Section 5.3. Information about the numbers of tasks and samples in the respective environments can be found in Table S1.

	Sinusoid	Cauchy	SwissFEL	Physionet	Berkeley
n	20	20	5	100	36
m_i	5	20	200	4 - 24	288

Table S1. Number of tasks n and samples per task m_i for the different meta-learning environments.

E.1.1. SINUSOIDS

Each task of the sinusoid environment corresponds to a parametric function

$$f_{a,b,c,\beta}(x) = \beta * x + a * \sin(1.5 * (x - b)) + c \, , \tag{57}$$

which, in essence, consists of an affine as well as a sinusoid function. Tasks differ in the function parameters (a, b, c, β) that are sampled from the task environment $\mathcal{T}$ as follows:

$$a \sim \mathcal{U}(0.7, 1.3), \quad b \sim \mathcal{N}(0, 0.1^2), \quad c \sim \mathcal{N}(5.0, 0.1^2), \quad \beta \sim \mathcal{N}(0.5, 0.2^2) \, . \tag{58}$$

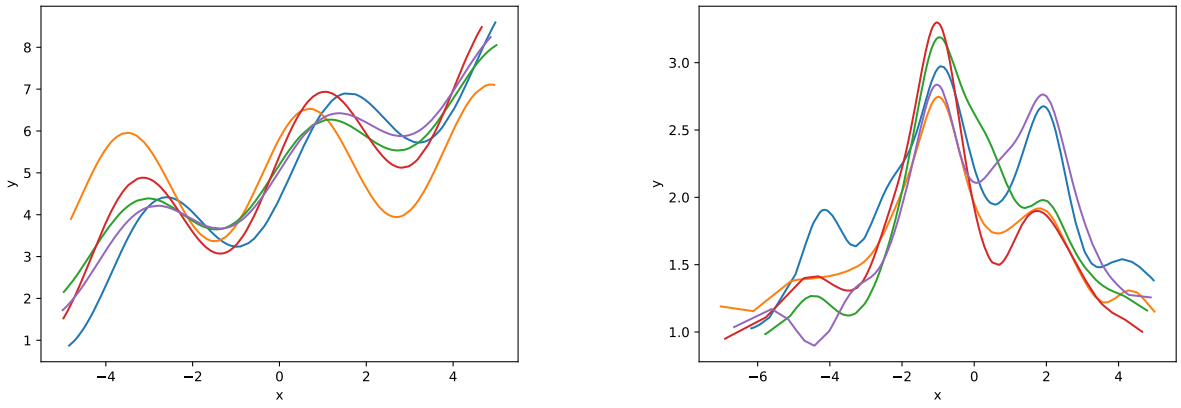
Figure S1a depicts functions $f_{a,b,c,\beta}$ with parameters sampled according to (58). To draw training samples from each task, we draw x uniformly from $\mathcal{U}(-5, 5)$ and add Gaussian noise with standard deviation 0.1 to the function values $f(x)$:

$$x \sim \mathcal{U}(-5, 5) \, , \qquad y \sim \mathcal{N}(f_{a,b,c,\beta}(x), 0.1^2) \, . \tag{59}$$

E.1.2. CAUCHY

Each task of the Cauchy environment can be interpreted as a two dimensional mixture of Cauchy distributions plus a function sampled from a Gaussian process prior with zero mean and SE kernel function $k(x, x') = \exp\left(\frac{||x - x'||_2^2}{2l} \right)$ with $l = 0.2$. The (unnormalized) mixture of Cauchy densities is defined as:

$$m(x) = \frac{6}{\pi \cdot (1 + ||x - \mu_1||_2^2)} + \frac{3}{\pi \cdot (1 + ||x - \mu_2||_2^2)} \, , \tag{60}$$



(a) Sinusoid tasks

(b) Cauchy tasks

Figure S1. Depiction of tasks (i.e., functions) sampled from the Sinusoid and Cauchy task environment, respectively. Note that the Cauchy task environment is two-dimensional ($\dim(\mathcal{X}) = 2$), while (b) displays a one-dimensional projection.

with $\mu_1 = (-1, -1)^\top$ and $\mu_2 = (2, 2)^\top$.

Functions from the task environments are sampled as follows:

$$f(x) = m(x) + g(x) \,, \qquad g \sim \mathcal{GP}(0, k(x, x')) \,. \tag{61}$$

Figure S1b depicts a one-dimensional projection of functions sampled according to (61). To draw training samples from each task, we draw x from a truncated normal distribution and add Gaussian noise with standard deviation 0.05 to the function values $f(x)$:

$$x := \min\{\max\{\tilde{x}, 2\}, -3\} \,, \;\; \tilde{x} \sim \mathcal{N}(0, 2.5^2) \,, \qquad y \sim \mathcal{N}(f(x), 0.05^2) \,. \tag{62}$$

E.1.3. SwissFEL

Free-electron lasers (FELs) accelerate electrons to very high speed in order to generate shortly pulsed laser beams with wavelengths in the X-ray spectrum. These X-ray pulses can be used to map nanometer scale structures, thus facilitating experiments in molecular biology and material science. The accelerator and the electron beam line of a FEL consist of multiple magnets and other adjustable components, each of which has several parameters that experts adjust to maximize the pulse energy (Kirschner et al., 2019a). Due do different operational modes, parameter drift, and changing (latent) conditions, the laser's pulse energy function, in response to its parameters, changes across time. As a result, optimizing the laser's parameters is a recurrent task.

Overall, our meta-learning environment consists of different parameter optimization runs (i.e., tasks) on the SwissFEL, an 800 meter long laser located in Switzerland (Milne et al., 2017). A picture of the SwissFEL is shown in Figure S2. The input space, corresponding to the laser's parameters, has 12 dimensions whereas the regression target is the pulse energy (1-dimensional). For details on the individual parameters, we refer to Kirschner et al. (2019b). For each run, we have around 2000 data points. Since these data-points are generated with online optimization methods, the data are non-i.i.d. and get successively less diverse throughout the optimization. As we are concerned with meta-learning with limited data and want to avoid issues with highly dependent data points, we only take the first 400 data points per run and split them into training and test subsets of size 200. Overall, we have 9 runs (tasks) available. 5 of those runs are used for meta-training and the remaining 4 runs are used for meta-testing.

E.1.4. PhysioNet

The 2012 Physionet competition (Silva et al., 2012) published an open-access dataset of patient stays on the intensive care unit (ICU). Each patient stay consists of a time series over 48 hours, where up to 37 clinical variables are measured. The

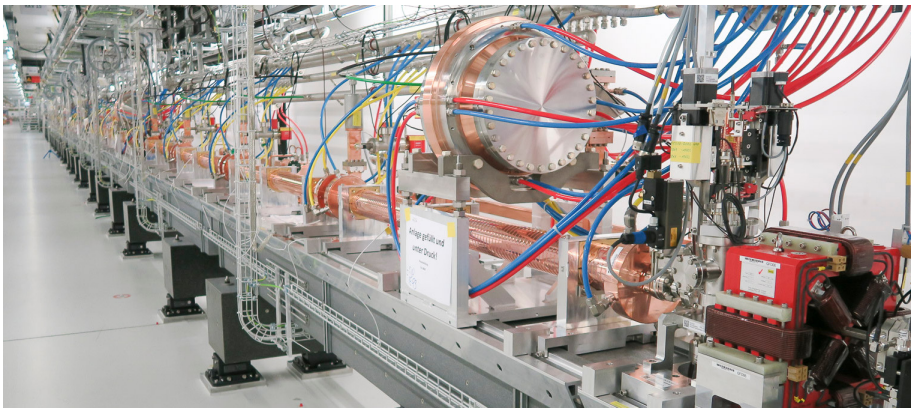


Figure S2. Accelerator of the Swiss Free-Electron Laser (SwissFEL).

original task in the competition was binary classification of patient mortality, but due to the large number of missing values (around 80 % across all features), the dataset is also popular as a test bed for time series prediction methods, especially using Gaussian processes (Fortuin et al., 2019).

In this work, we treat each patient as a separate task and the different clinical variables as different environments. We use the Glasgow coma scale (GCS) and hematocrit value (HCT) as environments for our study, since they are among the most frequently measured variables in this dataset. From the dataset, we remove all patients where less than four measurements of CGS (and HCT respectively) are available. From the remaining patients we use 100 patients for meta-training and 500 patients each for meta-validation and meta-testing. Here, each patient corresponds to a task. Since the number of available measurements differs across patients, the number of training points m_i ranges between 4 and 24.

E.1.5. BERKELEY-SENSOR

We use data from 46 sensors deployed in different locations at the Intel Research lab in Berkeley (Madden, 2004). The dataset contains 4 days of data, sampled at 10 minute intervals. Each task corresponds to one of the 46 sensors and requires auto-regressive prediction, in particular, predicting the next temperature measurement given the last 10 measurement values. In that, 36 sensors (tasks) with data for the first two days are use for meta-training and whereas the remaining 10 sensors with data for the last two days are employed for meta-testing. Note, that we separate meta-training and -testing data both temporally and spatially since the data is non-i.i.d. For the meta-testing, we use the 3rd day as context data, i.e. for target training and the remaining data for target testing.

E.2. Experimental Methodology

In the following, we describe our experimental methodology and provide details on how the empirical results reported in Section 5.3 were generated. Overall, evaluating a meta-learner consists of two phases, *meta-training* and *meta-testing*, outlined in Appendix D. The latter can be further sub-divided into *target training* and *target testing*. Figure 1 illustrates these different stages for our PAC-Bayesian meta-learning framework.

The outcome of the training procedure is an approximation for the generalized Bayesian posterior $Q^*(S, P)$ (see Appendix), pertaining to an unseen task $\tau = (\mathcal{D}, m) \sim \mathcal{T}$ from which we observe a dataset $\tilde{S} \sim \mathcal{D}$. In *target-testing*, we evaluate its predictions on a held-out test dataset $\tilde{S}^* \sim \mathcal{D}$ from the same task. For PACOH-NN, NPs and MLAP the respective predictor outputs a probability distribution $\hat{p}(y^*|x^*, \tilde{S})$ for the x^* in $\tilde{S}^*$. The respective mean prediction corresponds to the expectation of $\hat{p}$, that is $\hat{y} = \hat{\mathbb{E}}(y^*|x^*, \tilde{S})$. In the case of MAML, only a mean prediction is available. Based on the mean predictions, we compute the *root mean-squared error (RMSE)*:

$$\text{RMSE} = \sqrt{\frac{1}{|\tilde{S}^*|} \sum_{(x^*, y^*) \in S^*} (y^* - \hat{y})^2} \,. \tag{63}$$

and the *calibration error* (see Appendix E.2.1). Note that unlike e.g. Rothfuss et al. (2019a) who report the test log-likelihood,

we aim to measure the quality of mean predictions and the quality of uncertainty estimate separately, thus reporting both RMSE and calibration error.

The described meta-training and meta-testing procedure is repeated for five random seeds that influence both the initialization and gradient-estimates of the concerned algorithms. The reported averages and standard deviations are based on the results obtained for different seeds.

E.2.1. CALIBRATION ERROR

The concept of calibration applies to probabilistic predictors that, given a new target input x_i, produce a probability distribution $\hat{p}(y_i|x_i)$ over predicted target values y_i.

Calibration error for regression. Corresponding to the predictive density, we denote a predictor's cumulative density function (CDF) as $\hat{F}(y_j|x_j) = \int_{-\infty}^{y_j} \hat{p}(y|x_i)dy$. For confidence levels $0 \leq q_h < ... < q_H \leq 1$, we can compute the corresponding empirical frequency

$$\hat{q}_h = \frac{|\{y_j \mid \hat{F}(y_j|x_j) \leq q_h, j = 1, ..., m\}|}{m} , \tag{64}$$

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., 2018), we can define the calibration error as a function of residuals $\hat{q}_h - q_h$, in particular,

$$\text{calib-err} = \frac{1}{H} \sum_{h=1}^{H} |\hat{q}_h - q_h| . \tag{65}$$

Note that we while (Kuleshov et al., 2018) reports the average of squared residuals $|\hat{q}_h - q_h|^2$, 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. In our experiments, we compute (65) with $M = 20$ equally spaced confidence levels between 0 and 1.

Calibration error for classification. Our classifiers output a categorical probability distribution $\hat{p}(y = k|x)$ for $k = 1, ..., C$ where $\mathcal{Y} = \{1, ..., C\}$ with C denoting the number of classes. The prediction of the classifier is the most probable class label, i.e., $\hat{y}_j = \arg\max_k \hat{p}(y_j = k|x_j)$. Correspondingly, we denote the classifiers confidence in the prediction for the input x_j as $\hat{p}_j := \hat{p}(y_j = \hat{y}_j|x_j)$. Following, the calibraton error definition of Guo et al. (2017), we group the predictions into $H = 20$ interval bins of size $1/H$ depending on their prediction confidence. In particular, let $B_h = \{j \mid p_j \in \left(\frac{h-1}{H}, \frac{h}{H}\right]\}$ be the set of indices of test points $\{(x_j, y_j)\}_{j=1}^m$ whose prediction fall into the interval $\left(\frac{h-1}{H}, \frac{h}{H}\right] \subseteq (0, 1]$. Formally, we define the accuracy of within a bin B_h as

$$\text{acc}(B_h) = \frac{1}{|B_h|} \sum_{j \in B_h} \mathbf{1}(\hat{y}_i = y_j) \tag{66}$$

and the average confidence within a bin as

$$\text{conf}(B_h) = \frac{1}{|B_h|} \sum_{j \in B_h} \hat{p}_j . \tag{67}$$

If the classifier is calibrated, we expect that the confidence of the classifier reflects it's accuracy on unseen test data, that is, $\text{acc}(B_h) = \text{conf}(B_h)\ \forall h = 1, ..., H$. As proposed of Guo et al. (2017), we use the expected calibration error (ECE) to quantify how much the classifier deviates from this criterion: More precisely, in Table 2, we report the ECE with the following definition:

$$\text{calib-err} = \text{ECE} = \sum_{h=1}^{H} \frac{|B_h|}{m} \big|\text{acc}(B_h) - \text{conf}(B_h)\big| \tag{68}$$

with m denoting the overall number of test points.

E.3. Hyper-Parameter Selection

For each of the meta-environments and algorithms, we ran a separate hyper-parameter search to select the hyper-parameters. In particular, we use the `hyperopt`[4] package (Bergstra et al., 2013) which performs Bayesian optimization based on

[4] http://hyperopt.github.io/hyperopt/

Allele	A-0202	A-0203	A-0201	A-2301	A-2402
m_i	1446	1442	3088	103	196

Table S2. MHC-I alleles used for meta-training and their corresponding number of meta-training samples m_i.

regression trees. As optimization metric, we employ the average log-likelihood, evaluated on a separate validation set of tasks.

The scripts for reproducing the hyper-parameter search for PACOH-GP are included in our code repository[5]. For the reported results, we provide the selected hyper-parameters and detailed evaluation results under https://tinyurl.com/s48p76x.

E.4. Meta-Learning for Bandits - Vaccine Development

In this section, we provide additional details on the experiment in Section 6.3.

We use data from Widmer et al. (2010) which contains the binding affinities (IC_{50} values) of many peptide candidates to seven different MHC-I alleles. Peptides with $IC_{50} > 500$nM are considered non-binders, all others binders. Following Krause & Ong (2011), we convert the IC_{50} values into negative log-scale and normalize them such that 500nM corresponds to zero, i.e. $r := -\log_{10}(\text{IC}_{50}) + \log_{10}(500)$ with is used as reward signal of our bandit.

We use 5 alleles to meta-learn a BNN prior. The alleles and the corresponding number of data points, available for meta-training, are listed in Table S2. The most genetically dissimilar allele (A-6901) is used for our bandit task. In each iteration, the experimenter (i.e. bandit algorithm) chooses to test one peptide among the pool of 813 candidates and receives r as a reward feedback. Hence, we are concerned with a 813-arm bandit wherein the action $a_t \in \{1, ..., 813\} = \mathcal{A}$ in iteration t corresponds to testing a_t-th peptide candidate. In response, the algorithm receives the respective negative log-IC_{50} as reward $r(a_t)$.

As metrics, we report the *average regret*

$$R_T^{avg.} := \max_{a \in \mathcal{A}} r(a) - \frac{1}{T} \sum_{t=1}^{T} r(a_t)$$

and the *simple regret*

$$R_T^{simple} := \max_{a \in \mathcal{A}} r(a) - \max_{t=1,...,T} r(a_t)$$

To ensure a fair comparison, the prior parameters of the GP for GP-UCB and GP-TS are meta-learned by minimizing the GP's marginal log-likelihood on the five meta-training tasks. For the prior, we use a constant mean function and tried various kernel functions (linear, SE, Matern). Due to the 45-dimensional feature space, we found the linear kernel to work the best. So overall, the constant mean and the variance parameter of the linear kernel are meta-learned.

E.5. Further Experimental Results

Meta-overfitting In order to investigate whether the phenomenon of meta-overfitting, which we have observed consistently for PACOH-MAP and MLL, is also relevant to other meta-learning methods (MAML and NPs), we also report the meta-train test error and the meta-test test error across different numbers of tasks. The results, analogous to Figure 4, are plotted in Figure S3, showing a significant difference between the meta-train and meta-test error that vanishes as the number of tasks becomes larger. Once more, this supports our claim that meta-overfitting is a relevant issue and should be addressed in a principled manner.

[5] https://github.com/jonasrothfuss/meta_learning_pacoh

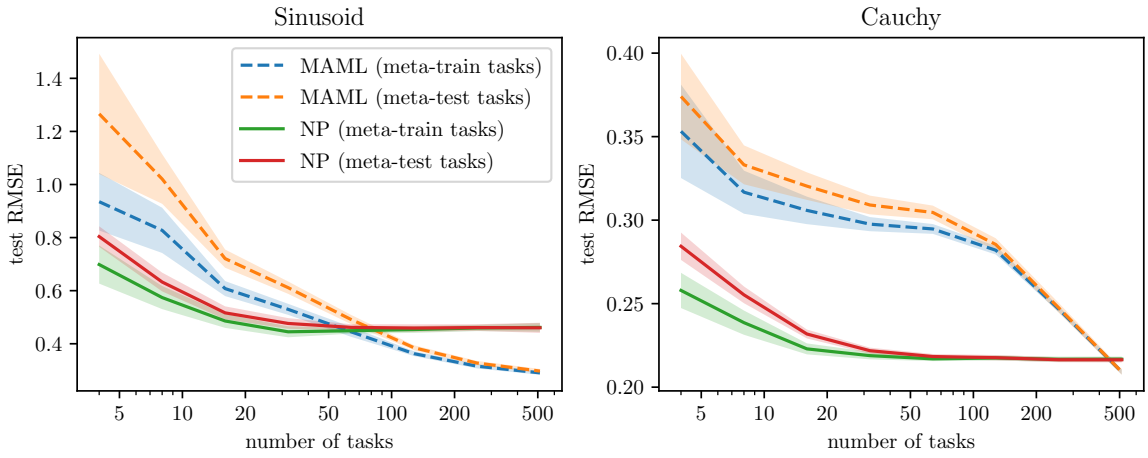


Figure S3. Test RMSE measured on the meta-training tasks and the meta-testing tasks as a function of the number of meta-training tasks for MAML and NPs. The performance gap between the meta-train and meta-test tasks clearly demonstrates overfitting on the meta-level for both methods.