Bayesian Uncertainty for Gradient Aggregation in Multi-Task Learning

Idan Achituve [1 2] Idit Diamant [2] Arnon Netzer [2] Gal Chechik [3] Ethan Fetaya [1]

Abstract

As machine learning becomes more prominent there is a growing demand to perform several inference tasks in parallel. Multi-task learning (MTL) addresses this challenge by learning a single model that solves several tasks simultaneously and efficiently. Often optimizing MTL models entails first computing the gradient of the loss for each task, and then aggregating all the gradients to obtain a combined update direction. However, common methods following this approach do not consider an important aspect, the sensitivity in the dimensions of the gradients. Some dimensions may be more lenient for changes while others may be more restrictive. Here, we introduce a novel gradient aggregation procedure using Bayesian inference. We place a probability distribution over the task-specific parameters, which in turn induce a *distribution* over the gradients of the tasks. This valuable information allows us to quantify the uncertainty associated with each of the gradients' dimensions which is factored in when aggregating them. We empirically demonstrate the benefits of our approach in a variety of datasets, achieving state-of-the-art performance.

1. Introduction

In many application domains, there is a need to perform several machine learning inference tasks simultaneously. For instance, an autonomous vehicle needs to identify and detect objects in its vicinity, perform lane detection, track the movements of other vehicles over time, and predict free space around it, all in parallel and in real-time. In deep Multi-Task Learning (MTL) the goal is to train a single neural network (NN) to solve several tasks simultaneously, thus avoiding the need to have one dedicated model for each task (Caruana, 1997). Besides reducing the computational demand at test time, MTL also has the potential to improve generalization (Baxter, 2000). It is therefore not surprising that applications of MTL are taking central roles in various domains, such as vision (Achituve et al., 2021a; Shamshad et al., 2023; Zheng et al., 2023), natural language processing (Liu et al., 2019b; Zhou et al., 2023), speech (Michelsanti et al., 2021), robotics (Devin et al., 2017; Shu et al., 2018), and general scientific problems (Wu et al., 2018) to name a few.

However, optimizing multiple tasks simultaneously is a challenging problem that may lead to degradation in performance compared to learning them individually (Standley et al., 2020; Yu et al., 2020). To address this issue, one basic formula that many MTL optimization algorithms follow is to first calculate the gradient of each task's loss, and then aggregate these gradients according to some specified scheme. For example, several studies focus on reducing conflicts between the gradients before averaging them (Yu et al., 2020; Wang et al., 2020), others find a convex combination with minimal norm (Sener & Koltun, 2018; Désidéri, 2012), and some use a game theoretical approach (Navon et al., 2022). However, by relying only on the gradient these methods miss an important aspect, the sensitivity of the gradient in each dimension.

Our approach builds on the following observation - for each task, there may be many "good" parameter configurations. Standard MTL optimization methods take only a single value into account, and as such lose information in the aggregation step. Hence, tracking all of the parameter configurations will yield a richer description of the gradient space that can be advantageous when finding an update direction. Specifically, to account for all parameter values, we propose to place a probability distribution over the task-specific parameters, which in turn induces a probability distribution over the gradients. As a result, we obtain uncertainty estimates for the gradients that reflect the sensitivity in each of their dimensions. High-uncertainty dimensions are more lenient for changes while dimensions with a lower uncertainty are more strict (see illustration in Figure 2).

To obtain a probability distribution over the task-specific parameters we take a Bayesian approach. According to the Bayesian view, a posterior distribution over parameters of

[1] Faculty of Engineering, Bar-Ilan University, Israel [2] Sony Semiconductor Israel [3] Department of Computer Science, Bar-Ilan University, Israel. Correspondence to: Idan Achituve <Idan.Achituve@Sony.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

interest can be derived through Bayes rule. In MTL, it is common to use a shared feature extractor network with linear task-specific layers (Ruder, 2017). Hence, if we assume a Bayesian model over the last task-specific layer weights during the back-propagation process, we obtain the posterior distributions over them. The posterior is then used to compute a Gaussian distribution over the gradients by means of moment matching. Then, to derive an update direction for the shared network, we design a novel aggregation scheme that considers the full distributions of the gradients. We name our method *BayesAgg-MTL*. An important implication of our approach is that BayesAgg-MTL assigns weights to the gradients at a higher resolution compared to existing methods, allocating a specific weight for each dimension and datum in the batch. We demonstrate our method effectiveness over baseline methods on the MTL benchmarks QM9 (Ramakrishnan et al., 2014), CIFAR-100 (Krizhevsky et al., 2009), ChestX-ray14 (Wang et al., 2017), and UTKFace (Zhang et al., 2017).

In summary, this paper makes the following novel contributions: (1) The first Bayesian formulation of gradient aggregation for Multi-Task Learning. (2) A novel posterior approximation based on a second-order Taylor expansion. (3) A new MTL optimization algorithm based on our posterior estimation. (4) New state-of-the-art results on several MTL benchmarks compared to leading methods. Our code is publicly available at https://github.com/ssi-research/BayesAgg_MTL.

2. Background

Notations. Scalars, vectors, and matrices are denoted with lower-case letters (e.g., x), bold lower-case letters (e.g., $\mathbf{x}$), and bold upper-case letters (e.g., $\mathbf{X}$) respectively. All vectors are treated as column vectors. Training samples are tuples consisting of shared features across all tasks and labels of K tasks, namely $(\mathbf{x}, \{\mathbf{y}^k\}_{k=1}^K) \sim \mathcal{D}$, where $\mathcal{D}$ denotes the training set. We denote the dimensionality of the input and the output of task k by $d_\mathbf{x}$ and o_k accordingly.

In this study, we focus on common NN architectures for MTL having a shared feature extractor and linear task-specific heads (Kendall et al., 2018; Sener & Koltun, 2018). The model parameters are denoted by $\{\boldsymbol{\theta}, \{\mathbf{w}^k\}_{k=1}^K\}$, where $\boldsymbol{\theta} \in \mathbb{R}^{d_{\boldsymbol{\theta}}}$ is the vector of shared parameters and $\{\mathbf{w}^k\}_{k=1}^K$ are task-specific parameter vectors, each lies in $\mathbb{R}^{d_k}$. The last shared feature representation is denoted by the vector $\mathbf{h}(\mathbf{x}; \boldsymbol{\theta}) \in \mathbb{R}^{d_\mathbf{h}}$. Hence, the output of the network for task k can be described as $\mathbf{f}^k(\mathbf{h}(\mathbf{x}; \boldsymbol{\theta}); \mathbf{w}^k)$. The loss of task $k \in [1, ..., K]$ is denoted by $\ell^k(\mathbf{x}, \mathbf{y}; \{\boldsymbol{\theta}, \mathbf{w}^k\})$. The gradient of loss ℓ^k w.r.t $\mathbf{h}(\mathbf{x}; \boldsymbol{\theta})$ is $\mathbf{g}^k := \frac{\partial \ell^k}{\partial \mathbf{h}(\mathbf{x}; \boldsymbol{\theta})}(\mathbf{x}, \mathbf{y}; \{\boldsymbol{\theta}, \mathbf{w}^k\}) \in \mathbb{R}^{d_h}$. For clarity of exposition, function dependence on input variables will be omitted from now on.

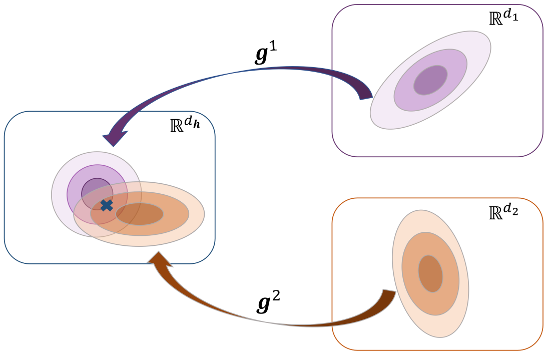


Figure 1. BayesAgg-MTL assumes a probability distribution over the last layer parameters of each task. It first maps these distributions to the space of the last shared representation. Then an update direction is found for the shared representation based on the mean and variance of all distributions (denoted by ✖).

2.1. Multi-Task Learning

A prevailing approach to optimize MTL models goes as follows. First, the gradient of each task loss is computed. Second, an aggregation rule is imposed to combine the gradients according to some algorithm. And lastly, perform an update step using the outcome of the aggregation step. Commonly the aggregation rule operates on the gradients of the loss w.r.t parameters, or only the shared parameters (e.g., Yu et al., 2020; Navon et al., 2022; Shamsian et al., 2023)). Alternatively, to avoid a costly full back-propagation process for each task, some methods suggest applying it on the last shared representation (e.g., Sener & Koltun, 2018; Liu et al., 2020; Senushkin et al., 2023). Here, to make our method fast and scalable, we take the latter approach and note that it could be extended to full gradient aggregation.

2.2. Bayesian Inference

We wish to incorporate uncertainty estimates for the gradients into the aggregation procedure. Doing so will allow us to find an update direction that takes into account the importance of each gradient dimension for each task. A natural choice to model uncertainty is using Bayesian inference. Since we would like to get uncertainty estimates w.r.t the last shared hidden layer, we treat only the last task-specific layer as a Bayesian layer. This "Bayesian last layer" approach is a common way to scale Bayesian inference to deep neural networks (Snoek et al., 2015; Calandra et al., 2016; Wilson et al., 2016a; Achituve et al., 2021c). We will now present some of the main concepts of Bayesian modeling that will be used as part of our method.

For simplicity, assume a single output variable. We also dropped the task notation for clarity. According to the Bayesian paradigm, instead of treating the parameters $\mathbf{w}$ as deterministic values that need to be optimized, they are treated as random variables, i.e. there is a distribution over the parameters. The posterior distribution for $\mathbf{w}$, after observing the data, is given using Bayes rule as

$$log\, p(\mathbf{w}|\mathcal{D}) \propto log\, p(\mathbf{y}|\mathbf{X}, \mathbf{w}) + log\, p(\mathbf{w}). \tag{1}$$

Predictions in Bayesian inference are given by taking the expected prediction with respect to the posterior distribution. In general, the Bayesian inference procedure for $\mathbf{w}$ is intractable. However, for some specific scenarios, there exists an analytic solution. For example, in linear regression, if we assume a Gaussian likelihood with a fixed independent scalar noise between the observations τ, $p(\mathbf{y}|\{\mathbf{x}_i\}_{i=1}^{|\mathcal{D}|}, \mathbf{w}) = \prod_{i=1}^{|\mathcal{D}|} \mathcal{N}(y_i|\mathbf{x}_i^T\mathbf{w}, \tau^2)$, and a Gaussian prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_p, \mathbf{S}_p)$ then,

$$\begin{aligned} p(\mathbf{w}|\mathcal{D}) &= \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S}) \\ \mathbf{m} &= \mathbf{S}((\mathbf{S}_p)^{-1}\mathbf{m}_p + \tau^{-2}\mathbf{X}\mathbf{y}) \\ \mathbf{S} &= ((\mathbf{S}_p)^{-1} + \tau^{-2}\mathbf{X}\mathbf{X}^T)^{-1}. \end{aligned} \tag{2}$$

Here $\mathbf{X} \in \mathbb{R}^{d_\mathbf{x} \times |\mathcal{D}|}$ is the matrix that results from stacking the vectors $\{\mathbf{x}_i\}_{i=1}^{|\mathcal{D}|}$. Similarly, we denote by $\mathbf{H} \in \mathbb{R}^{d_\mathbf{h} \times |\mathcal{D}|}$ the matrix that results from stacking the vectors of hidden representation. In the specific case of deep NNs with Bayesian last layer we get the same inference result only with $\mathbf{H}$ replacing $\mathbf{X}$. Going beyond a single output variable entails defining a covariance matrix for the noise model. However, in this study we assume independence between the output variables in these cases.

Unlike regression, in classification the likelihood is not a Gaussian, and the posterior can only be approximated. The common choice is to use variational inference (Wilson et al., 2016b; Achituve et al., 2021b; 2023), although there are other alternatives as well (Kristiadi et al., 2020).

3. Method

We start with an outline of the problem and our approach. Consider a deep network for multi-task learning that has a shared feature extractor part and task-specific linear layers. We propose to use Bayesian inference on the last layer as a means to train *deterministic* MTL models. For each task k, we define a Bayesian probabilistic model representing the uncertainty over the linear weights of the last, task-specific layer $\mathbf{w}^k$. The distribution over weights induces a distribution over gradients of the loss with respect to the last *shared* hidden layer. Given these per-task distributions on a joint space, we propose an aggregation rule for combining the gradients of the tasks to a shared update direction that takes into account the uncertainty in the gradients (see illustration in Figure 1). Then, the back-propagation process can proceed as usual.

Since regression and classification setups yield different inference procedures according to our approach, albeit having the same general framework, we discuss the two setups separately, starting with regression.

3.1. BayesAgg-MTL for Regression Tasks

Consider a standard square loss for task k, $\ell^k = (y^k - \hat{y}^k)^2$, between the label y^k and the network output $\hat{y}^k$. Given a random batch of example $\mathcal{B} \sim \mathcal{D}$, the gradient of the loss with respect to the hidden layer $\mathbf{h}$ for the i^{th} example is,

$$\mathbf{g}_i^k = \frac{\partial l_i^k}{\partial \hat{y}_i^k}\frac{\partial \hat{y}_i^k}{\partial \mathbf{h}_i} = 2\mathbf{w}^k(\mathbf{h}_i^T\mathbf{w}^k - y_i^k). \tag{3}$$

Our main observation is that $\mathbf{g}_i^k$ is a function of $\mathbf{w}^k$. Hence, if we view $\mathbf{w}^k$ in the back-propagation process as a random variable, then $\mathbf{g}_i^k$ will be a random variable as well. This view will allow us to capture the uncertainty in the task gradient. Since the dimension of the hidden layer is usually small compared to the dimension of all shared parameters, operations in this space, such as matrix inverse, should not be costly.

If we fix all the shared parameters, then the posterior over $\mathbf{w}^k$ has a Gaussian distribution with known parameters via Eq. 2. As $\mathbf{g}_i^k$ is quadratic in $\mathbf{w}^k$, it has a generalized chi-squared distribution (Davies, 1973). However, since this distribution does not admit a closed-form density function, and since the gradient aggregation needs to be efficient as we run it at each iteration, we approximate $\mathbf{g}_i^k$ as a Gaussian distribution. The optimal choice for the parameters of this Gaussian is given by matching its first two moments to those of the true density, as these parameters minimize the Kullback–Leibler divergence between the two distributions (Minka, 2001). Luckily, in the regression case, we can derive the first two moments from the posterior over $\mathbf{w}^k$,

$$\begin{aligned} &\mathbb{E}[\mathbf{g}_i^k] = 2[\mathbf{S}^k\mathbf{h}_i + \mathbf{m}^k(\mathbf{h}_i^T\mathbf{m}^k - y_i^k)], \\ &\mathbb{E}[\mathbf{g}_i^k(\mathbf{g}_i^k)^T] = 4[(y_i^k)^2(\mathbf{S}^k + \mathbf{M}^k) - 2y_i^k(\mathbf{m}^k\mathbf{h}_i^T(\mathbf{S}^k + \mathbf{M}^k) \\ &\quad + (\mathbf{S}^k + \mathbf{M}^k)\mathbf{h}_i(\mathbf{m}^k)^T + \mathbf{h}_i^T\mathbf{m}^k(\mathbf{S}^k - \mathbf{M}^k)) \\ &\quad + (\mathbf{S}^k + \mathbf{M}^k)(\mathbf{A}_i + \mathbf{A}_i^T)(\mathbf{S}^k + \mathbf{M}^k) \\ &\quad + Tr(\mathbf{A}_i\mathbf{S}^k)(\mathbf{S}^k + \mathbf{M}^k) + (\mathbf{m}^k)^T\mathbf{A}_i\mathbf{m}^k(\mathbf{S}^k - \mathbf{M}^k)], \end{aligned} \tag{4}$$

where $\mathbf{A}_i = \mathbf{h}_i\mathbf{h}_i^T$, $\mathbf{M}^k = \mathbf{m}^k(\mathbf{m}^k)^T$, we assumed $\tau = 1$, and $Tr(\cdot)$ is the matrix trace. We emphasize that the following approximation is for the gradient of a single data point and a single task, not for the gradient of the task with respect to the entire batch. The full derivation is presented in Appendix A.1.

Several points deserve attention here. First, note the similarity between the solution of the first moment and the gradient obtained via the standard back-propagation. The two differences are that the last layer parameters, $\mathbf{w}^k$, are replaced with the posterior mean, $\mathbf{m}^k$, and an uncertainty term was added. In the extreme case of $\mathbf{S}^k \rightarrow 0$ and $\mathbf{m}^k \rightarrow \mathbf{w}^k$, the mean coincides with that of the standard back-propagation. Second, in the case of a multi-output task, following our independence assumption between output variables, we can obtain the moments for each output dimension separately using the same procedure, so de facto we treat each output as a different task. Finally, during training, the shared parameters are constantly being updated. Hence, to compute the posterior distribution for $\mathbf{w}^k$ we need to iterate over the entire dataset at each update step. In practice, this can make our method computationally expensive. Therefore, we use the current batch data only to approximate the posterior over $\mathbf{w}^k$, and introduce information about the full dataset through the prior as described next.

Prior selection. A common choice in Bayesian deep learning is to choose uninformative priors, such as a standard Gaussian, to let the data be the main influence on the posterior (Wilson & Izmailov, 2020; Fortuin et al., 2021). However, in our case, we found this prior to be too weak. Since the posterior depends only on a single batch we opted to introduce information about the whole dataset through the prior. A natural choice is to use the posterior distribution of the previous batch as our prior (Särkkä, 2013, Chapter 3). However, this method did not work well in our experiments and we developed an alternative. During each epoch, we collect the feature representations and labels of all examples in the dataset. At the end of the epoch, we compute the posterior based on the full data (with an isotropic Gaussian prior) and use this posterior as the prior at each step in the subsequent epoch. Updating the full data prior more frequently is likely to have a beneficial effect on our overall model; however it will also probably make the training time longer. Hence, doing the update once an epoch strikes a good balance between performance and training time.

Aggregation step. Having an approximation for the gradient distribution of each task we need to combine them to find an update direction for the shared parameters. Denote the mean of the gradient of the loss for task k w.r.t the hidden layer for the i^{th} example by $\boldsymbol{\mu}_i^k := \mathbb{E}[\mathbf{g}_i^k]$, and similarly the covariance matrix $\boldsymbol{\Sigma}_i^k := (\boldsymbol{\Lambda}_i^k)^{-1} := \mathbb{E}[\mathbf{g}_i^k(\mathbf{g}_i^k)^T] - \mathbb{E}[\mathbf{g}_i^k]\mathbb{E}[\mathbf{g}_i^k]^T$. We strive to find an update direction for the last shared layer, $\mathbf{g}_i$, that lies in a high-density region for all tasks. Hence, we pick $\mathbf{g}_i$ that maximizes the following likelihood:

$$\begin{aligned}&\arg\max_{\mathbf{g}_i}\prod_{k=1}^{K}\mathcal{N}(\mathbf{g}_i|\boldsymbol{\mu}_i^k,\boldsymbol{\Sigma}_i^k) = \\ &\arg\min_{\mathbf{g}_i} -\sum_{k=1}^{K} log\,\mathcal{N}(\mathbf{g}_i|\boldsymbol{\mu}_i^k,\boldsymbol{\Sigma}_i^k).\end{aligned} \tag{5}$$

Thankfully, the above optimization problem can be solved

```
Algorithm 1 BayesAgg-MTL
Input: $\mathcal{B}$ - a random batch of examples; $p(\mathbf{w}^k|\mathcal{D})\ \ \forall k \in [1, ..., K]$ - posterior distributions over the task-specific parameters; $s$ - scaling hyper-parameter
For $i = 1, ..., |\mathcal{B}|$:
    For $k = 1, ..., K$:
        • Compute $\mathbb{E}[\mathbf{g}_i^k]$ and $\mathbb{E}[\mathbf{g}_i^k(\mathbf{g}_i^k)^T]$ as in Eq. 4 for regression or Eq. 11 for classification.
        • Set (operations are done element-wise),
          $\boldsymbol{\mu}_i^k := \mathbb{E}[\mathbf{g}_i^k]$,
          $\boldsymbol{\lambda}_i^k := (\mathbb{E}[(\mathbf{g}_i^k)^2] - \mathbb{E}[\mathbf{g}_i^k]\mathbb{E}[\mathbf{g}_i^k]))^{-1}$.
    End for
    Compute $\mathbf{g}_i = \sum_{k=1}^{K}\frac{(\boldsymbol{\lambda}_i^k)^s}{\sum_{k=1}^{K}(\boldsymbol{\lambda}_i^k)^s}\boldsymbol{\mu}_i^k$.
End for
Compute gradient via matrix multiplication w.r.t the shared parameters: $\frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\mathbf{g}_i\frac{\partial \mathbf{h}_i}{\partial \boldsymbol{\theta}}$
```

in closed-form, yielding the following solution:

$$\mathbf{g}_i = \left(\sum_{k=1}^{K}\boldsymbol{\Lambda}_i^k\right)^{-1}\left(\sum_{k=1}^{K}\boldsymbol{\Lambda}_i^k\boldsymbol{\mu}_i^k\right). \tag{6}$$

However, we found that modeling the full covariance matrix can be numerically unstable and sensitive to noise in the gradient. Instead, we assume independence between the dimensions of $\mathbf{g}_i^k$ for all tasks which results in diagonal covariance matrices having variance $(\boldsymbol{\sigma}_i^k)^2 := 1/\boldsymbol{\lambda}_i^k$. The update direction now becomes:

$$\mathbf{g}_i = \sum_{k=1}^{K}\frac{1/(\boldsymbol{\sigma}_i^k)^2}{\sum_{k=1}^{K}1/(\boldsymbol{\sigma}_i^k)^2}\boldsymbol{\mu}_i^k = \sum_{k=1}^{K}\overbrace{\frac{\boldsymbol{\lambda}_i^k}{\sum_{k=1}^{K}\boldsymbol{\lambda}_i^k}}^{\boldsymbol{\alpha}_i^k}\boldsymbol{\mu}_i^k, \tag{7}$$

where the division and multiplication are done element-wise. In Eq. 7 we intentionally denote by $\boldsymbol{\alpha}_i^k$ the vector of uncertainty-based weights that our method assigns to the mean gradient to highlight that the weights are unique per task, dimension, and datum. The final modification for the method involves down-scaling the impact of the precision by a hyper-parameter $s \in (0, 1]$, namely, we take $(\boldsymbol{\lambda}_i^k)^s$. Empirically, the scaling parameter helped to achieve better performance, perhaps due to misspecifications in the model (such as the diagonal Gaussian assumption over $\mathbf{g}_i^k$). With the aggregated gradient for each example, the back-propagation procedure proceeds as usual by averaging over all examples in the batch and then back-propagating this over to the shared parameters. We summarize our method in Algorithm 1.

To gain a better intuition about the update rule of BayesAgg-MTL , consider the illustration in Figure 2. In the figure, we plot the mean update direction of two tasks along with the uncertainty in them. The first task is more sensitive to

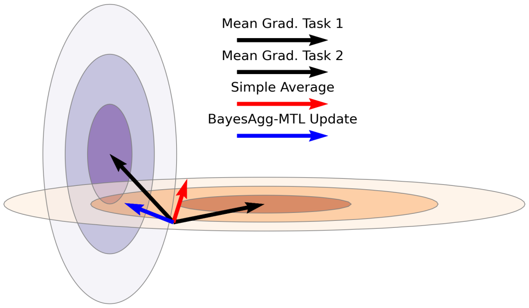


Figure 2. BayesAgg-MTL update for a two-dimensional feature representation. Black arrows indicate the mean update direction of each task; Red arrow is the update direction of a simple average; Blue arrow is the proposed update direction. Darker colors in the contours represent regions with higher density.

shifts in the vertical dimension and less so to shifts in the second (horizontal) dimension, while for the second task, it is the opposite. By taking the variance information into account, BayesAgg-MTL can find an update direction that works well for both, compared to a simple average of the gradient means.

Making predictions. Since we have a closed-form solution for the posterior of the task-specific parameters, BayesAgg-MTL does not learn this layer during training. Therefore, when making predictions we use the posterior mean, $\mathbf{m}^k$, computed on the full training set. We do so, instead of using a full Bayesian inference, for a fair comparison with alternative MTL approaches and to have an identical run-time and memory requirements when making predictions.

Connection to Nash-MTL. In (Navon et al., 2022) the authors proposed a cooperative bargaining game approach to the gradient aggregation step with the directional derivative as the utility of each player (task). They then proposed using the Nash bargaining solution, the direction that maximizes the product of all the utilities. One can consider Eq. 5 as the Nash bargaining solution with the utility of each task being its likelihood. However, unlike (Navon et al., 2022) we get an analytical formula for the bargaining solution since the Gaussian exponent and the logarithm cancel out.

3.2. BayesAgg-MTL for Classification Tasks

We now turn to present our approach for classification tasks. When dealing with classification there are two sources of intractability that we need to overcome. The first is the posterior of $\mathbf{w}^k$, and the second is estimating the moments of $\mathbf{g}_i^k$. We describe our solution to both challenges next.

Posterior approximation. In classification tasks the likelihood is not a Gaussian and in general, we cannot compute the posterior in closed-form. One common option is to approximate it using a Gaussian distribution and learn its parameters using a variational inference (VI) scheme (Saul et al., 1996; Neal & Hinton, 1998; Bishop, 2006). However, in our early experimentations, we didn't find it to work well without using a computationally expensive VI optimization at each update step. Alternatively to VI, the Laplace approximation (MacKay, 1992) approximates the posterior as a Gaussian using a second-order Taylor expansion. Since the expansion is done at the optimal parameter values that are learned point-wise, the Jacobean term in the expansion vanishes. Here, we follow a similar path; however, we cannot assume that the Jacobean is zero as we are not near a stationary point during most of the training. Nevertheless, we can still find a Gaussian approximation. A similar derivation was proposed in (Immer et al., 2021), yet they ignored the first order term eventually. Denote by $\hat{\mathbf{w}}^k$ the *learned* point estimate for the task parameters, and $\Delta\mathbf{w}^k := \mathbf{w}^k - \hat{\mathbf{w}}^k$. Then, at each step of the training by using Bayes rule we can obtain a posterior approximation for $\mathbf{w}^k$ using the following:

$$\begin{aligned}&log\ p(\mathbf{w}^k|\mathcal{B}) \approx log\ p(\hat{\mathbf{w}}^k|\mathcal{B}) + \\ &\left(-\frac{\partial log\ p(\mathbf{y}^k|\mathbf{X}, \mathbf{w}^k)}{\partial \mathbf{w}^k} - \frac{\partial log\ p(\mathbf{w}^k)}{\partial \mathbf{w}^k}\right)^T \Delta\mathbf{w}^k + \\ &\frac{1}{2}(\Delta\mathbf{w}^k)^T\left(-\frac{\partial^2 log\ p(\mathbf{y}^k|\mathbf{X}, \mathbf{w}^k)}{\partial (\mathbf{w}^k)^2} - \frac{\partial^2 log\ p(\mathbf{w}^k)}{\partial (\mathbf{w}^k)^2}\right)\Delta\mathbf{w}^k.\end{aligned} \tag{8}$$

The above takes the following form $c^k + (\mathbf{a}^k)^T(\mathbf{w}^k - \hat{\mathbf{w}}^k) + \frac{1}{2}(\mathbf{w}^k - \hat{\mathbf{w}}^k)^T\mathbf{B}^k(\mathbf{w}^k - \hat{\mathbf{w}}^k)$, where $\mathbf{a}^k \in \mathbb{R}^{d_k}, \mathbf{B}^k \in \mathbb{R}^{d_k \times d_k}, c^k \in \mathbb{R}$ are known constants. We stress here again, that since we apply Bayesian inference to the last layer parameters only, computing and inverting $\mathbf{B}^k$, typically does not incur a large computational overhead.

After rearranging and completing the square we obtain a quadratic form corresponding to the following Gaussian distribution (see full derivation in Appendix A.2):

$$p(\mathbf{w}^k|\mathcal{B}) \approx \mathcal{N}(\mathbf{w}^k|\hat{\mathbf{w}}^k - (\mathbf{B}^k)^{-1}\mathbf{a}^k, (\mathbf{B}^k)^{-1}). \tag{9}$$

Examining the above posterior reveals several insights. First, the posterior mean corresponds to the Newton method update step. Second, the covariance of this posterior is the same as that of the Laplace approximation. Third, at a stationary point the Laplace approximation is recovered if the gradient of the loss w.r.t the parameters approaches zero.

One limitation of the approximation in Eq. 9 is that the Hessian will not be positive-definite in most cases. Therefore, we replace it with the generalized Gauss-Newton (GGN) matrix (Schraudolph, 2002; Martens & Sutskever, 2011; Daxberger et al., 2021):

$$\tilde{\mathbf{B}}^k = \sum_{i=1}^{|\mathcal{B}|}(\mathbf{J}_i^k)^T\mathbf{H}_i^k\mathbf{J}_i^k + (\mathbf{S}_p^k)^{-1}. \tag{10}$$

Where, $\mathbf{J}_i^k = \partial \mathbf{f}^k(\mathbf{x}_i; \mathbf{w}^k)/\partial \mathbf{w}^k \in \mathbb{R}^{o_k \times d_k}$ is the Jacobean of the model output for task k w.r.t the last layer parameters of that task, $\mathbf{H}_i^k = -\partial^2 log\ p(\mathbf{y}_i^k|\mathbf{x}_i, \mathbf{w}^k)/\partial(\mathbf{f}^k(\mathbf{x}_i; \mathbf{w}^k))^2 \in \mathbb{R}^{o_k \times o_k}$ is the Hessian of the negative log-likelihood w.r.t the model outputs of task k, and $\mathbf{S}_p^k$ is the covariance of the Gaussian prior for $\mathbf{w}^k$. As in the regression case we use here an informative prior based on the posterior from the full dataset at each training step.

Moments estimation. Unlike the regression case, in classification $\mathbf{g}_i^k$ will depend on $\mathbf{w}^k$ through some non-linear function. Hence, obtaining the moments as in Eq. 4 in closed-form is more challenging. However, since we are estimating the parameters of the last layer only, which in many cases are relatively low-dimensional, we can efficiently approximate these moments with Monte-Carlo sampling:

$$\begin{aligned} \mathbb{E}[\mathbf{g}_i^k] &\approx \frac{1}{J}\sum_{j=1}^{J} \mathbf{g}_i^k(\mathbf{w}_j^k), \\ \mathbb{E}[\mathbf{g}_i^k(\mathbf{g}_i^k)^T] &\approx \frac{1}{J}\sum_{j=1}^{J} \mathbf{g}_i^k(\mathbf{w}_j^k)\mathbf{g}_i^k(\mathbf{w}_j^k)^T. \end{aligned} \tag{11}$$

Here, $\mathbf{w}_j^k$ are samples from $p(\mathbf{w}^k|\mathcal{B})$, and the total number of samples are J. Effectively this means that we need to back-propagate gradients w.r.t the shared hidden layer J times; however, since the task-specific layers are linear it can be done cheaply and in parallel. Having the moment estimation we proceed with the aggregation rule as described in Section 3.1.

Making predictions. Unlike the regression case, here we learn the parameters of the last layer as part of the posterior approximation. Therefore, making predictions is done as usual with a forward-pass through the network.

4. Related Work

Multi-task learning is an active research area that attempts to learn jointly multiple tasks, commonly using a shared representation (Ruder, 2017; Navon et al., 2022; Liu et al., 2023; Elich et al., 2023; Shi et al., 2023; Yun & Cho, 2023). Learning a shared representation for multiple tasks imposes some challenges. One challenge is trying to learn an architecture that can express both task-shared and task-specific features. Another challenge is to find the optimal balancing of the tasks and enable learning the different tasks with equal importance. One line of research in MTL suggests methods to introduce novel MTL-friendly architectures, such as task-specific modules (Misra et al., 2016), attention-based networks (Liu et al., 2019a), and an ensemble of single-task models (Dimitriadis et al., 2023). Yet, a more common line of research focuses on the MTL optimization process, trying to explain the difficulties in the process by e.g. conflicting gradients (Wang et al., 2020) or plateaus in the loss landscape (Schaul et al., 2019). Our method focuses on the latter, MTL optimization process improvement. We note though that there are several ways to formulate the MTL problem and refer to the survey of Zhang & Yang (2021) for an in-depth review.

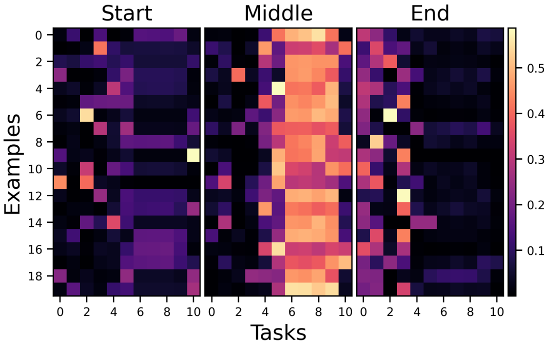


Figure 3. Mean weight over dimensions per-example for 20 random examples on the QM9 dataset at different training stages.

Different strategies were proposed to address the MTL optimization challenge to successfully balance the training of the different tasks and resolve their conflicts. The methods can broadly be categorized into two groups, loss-based and gradient-based (Dai et al., 2023). Loss-based approaches attempt to allocate weights for the tasks based on some criteria related to the loss, such as the difficulty of the task (Guo et al., 2018), random weights (Lin et al., 2022), geometric mean of the task losses (Chennupati et al., 2019; Yun & Cho, 2023), and task uncertainty (Kendall et al., 2018). Regarding the last one, to weigh the tasks it uses the uncertainty in the observations *only*. This is very different from our approach that weighs each dimension of the task *gradients* based on full Bayesian information.

Gradient-based methods attempt to balance the tasks by using the gradients information directly (Chen et al., 2018; 2020; Javaloy & Valera, 2022; Liu et al., 2020; Navon et al., 2022; Fernando et al., 2023; Senushkin et al., 2023). For example, GradNorm (Chen et al., 2018) dynamically tunes the gradient magnitudes to prevent imbalances between the tasks during training. PCGrad (Yu et al., 2020) identifies gradient conflicts as the main optimization issue in MTL, and attempts to reduce the conflicts by projecting each gradient to the other tasks' normal plane. Nash-MTL (Navon et al., 2022) suggests treating MTL as a bargaining game to find Pareto optimal solutions. Several studies suggested adaptations for the multiple-gradient descent algorithm (MGDA) (Désidéri, 2012; Sener & Koltun, 2018), such as CAGrad, (Liu et al., 2021), and MoCo (Fernando et al., 2023). As opposed to previous methods, our approach considers both the mean and the variance of the gradients to derive an update direction. It is worth noting that one possible limitation

Table 1. QM9. Test performance averaged over 3 random seeds.

	$\mathbf{\Delta_m}\%$ (↓)
LS	177.6 ± 3.4
SI	77.8 ± 9.2
RLW	203.8 ± 3.4
DWA	175.3 ± 6.3
UW	108.0 ± 22.5
MGDA	120.5 ± 2.0
PCGrad	125.7 ± 10.3
CAGrad	112.8 ± 4.0
IMTL-G	77.2 ± 9.3
Nash-MTL	62.0 ± 1.4
IGBv2	67.7 ± 8.1
Aligned-MTL-UB	71.0 ± 9.6
BayesAgg-MTL (Ours)	$\mathbf{53.2 \pm 7.1}$

Table 2. Test performance averaged over 3 random seeds on binary classification tasks from CIFAR-MTL & ChestX-ray14 datasets.

	CIFAR (Acc.) [↑]	**CX-ray** ($\mathbf{\Delta_m}\%$) [↓]
LS	$56.96 \pm .06$	-14.62 ± 0.2
SI	55.75 ± 0.3	-10.94 ± 0.4
RLW	$59.30 \pm .08$	-11.69 ± 0.1
DWA	58.44 ± 0.5	$\mathbf{-14.79 \pm .07}$
UW	56.63 ± 0.5	-13.95 ± 0.2
MGDA	$\mathbf{59.74 \pm .07}$	-14.44 ± 0.4
PCGrad	56.32 ± 0.2	-13.43 ± 0.5
CAGrad	56.59 ± 0.2	-14.49 ± 0.1
IMTL-G	57.09 ± 0.3	$-\ 8.23 \pm 1.8$
Nash-MTL	56.59 ± 0.2	-13.23 ± 0.5
IGBv2	56.61 ± 0.2	$-\ 2.82 \pm 0.6$
Aligned-MTL-UB	56.57 ± 0.7	-14.14 ± 0.2
BayesAgg-MTL (Ours)	$\mathbf{59.97 \pm 0.4}$	$\mathbf{-14.96 \pm 0.1}$

of BayesAgg-MTL , having in common with other popular MTL methods, is that it may fail on rare or atypical examples (Sagawa et al., 2019). But, we leave exploring this phenomenon for future works.

Lastly, some studies recently suggested performing model merging based on the uncertainty of the parameters (Matena & Raffel, 2022; Daheim et al., 2023). The goal there is usually to combine models for various tasks, such as model ensembling, federated learning, and robust fine-tuning. Unlike these methods, we assume a Bayesian model on the last layer only and propagate the uncertainty to the gradients for gradient aggregation.

5. Experiments

We evaluated BayesAgg-MTL on several MTL benchmarks differing in the number of tasks and their types. Unless specified otherwise, we report the average and standard deviation (std) of relevant metrics over 3 random seeds. In all datasets, we pre-allocated a validation set from the training set for hyper-parameter tuning and early stopping for all methods. Throughout our experiments, we used the ADAM optimizer (Kingma & Ba, 2015) which was found to be effective for MTL due to partial loss-scale invariance (Elich et al., 2023). Full experimental details are given in Appendix B.

Compared methods. We compare BayesAgg-MTL with the following baseline methods: **(1) Single Task Learning (STL)**, which learns each task independently under the same experimental setup as that of the MTL methods; **(2) Linear Scalarization (LS)**, which assigns a uniform weight to all tasks, namely $\sum_{k=1}^{K} \ell^k$; **(3) Scale-Invariant (SI)** (Navon et al., 2022), which assigns a uniform weight to the log of all tasks, namely $\sum_{k=1}^{K} log\ \ell^k$; **(4) Random Loss Weighting (RLW)** (Lin et al., 2022), which allocates random weights to the losses at each iteration; **(5) Dynamic Weight Average (DWA)** (Liu et al., 2019a), which allocates a weight based on the rate of change of the loss for each task; **(6) Uncertainty weighting (UW)** (Kendall et al., 2018), which minimize a scalar term corresponding to the *aleatoric* uncertainty for each task; **(7) Multiple-Gradient Descent Algorithm (MGDA)** (Désidéri, 2012; Sener & Koltun, 2018), which finds a minimum norm solution for a convex combination of the losses; **(8) Projecting Conflicting Gradients (PCGrad)** (Yu et al., 2020), which projects the gradient of each task onto the normal plane of tasks they are in conflict with; **(9) Conflict-Averse Grad (CAGrad)** (Liu et al., 2021), which searches an update direction centered at the LS solution while minimizing conflicts in gradients; **(10) Impartial MTL-Grad (IMTL-G)** (Liu et al., 2020), which finds an update vector such that the projection of it on each of the gradients of the tasks is equal; **(11) Nash-MTL (Navon et al., 2022)** that derives task weights based on the Nash bargaining solution; **(12) Improvable Gap Balancing (IGBv2)** (Dai et al., 2023), which suggests a Reinforcement learning procedure to balance the task losses; **(13) Aligned-MTL-UB (Senushkin et al., 2023)**, which aligns the principle components of a gradient matrix.

Evaluation metric. Unless specified otherwise, we report the $\Delta_m\%$ metric introduced in (Maninis et al., 2019). This metric measures the average relative difference between a method m compared to the STL baseline according to some criterion of interest M^k. Namely, $\Delta_m = \frac{1}{K}\sum_{k=1}^{K}(-1)^{\delta_k}(M_m^k - M_s^k)/M_s^k$. Where, M_m^k is the criterion value for task k under method m, M_s^k is the criterion value for task k under the STL baseline, and $\delta_k \in \{0, 1\}$. If $\delta_k = 0$ then a lower value for M^k is better (e.g., task loss), and if $\delta_k = 1$ then a higher value for M^k is preferred (e.g., task accuracy). Lower $\Delta_m\%$ indicates a better performance.

Pre-training stage. To obtain meaningful features for the Bayesian layer, it is a common practice to apply a pre-training step using standard NN training for several epochs

Table 3. UTKFace. Test performance averaged over 8 random seeds.

	Age $(\times 10^1)$ (↓)	**Gender** (↑)	**Ethnicity** (↑)	$\mathbf{\Delta_m}\%$ (↓)
STL	1.40 ± 0.03	92.32 ± 0.35	82.42 ± 0.42	–
LS	1.46 ± 0.02	92.92 ± 0.24	83.98 ± 0.43	0.69 ± 0.59
SI	1.42 ± 0.03	93.05 ± 0.29	83.40 ± 0.27	0.11 ± 0.89
RLW	1.44 ± 0.03	92.89 ± 0.25	83.70 ± 0.49	-0.31 ± 0.76
DWA	1.44 ± 0.02	92.90 ± 0.16	83.55 ± 0.33	0.35 ± 0.60
UW	1.43 ± 0.00	92.99 ± 0.24	83.09 ± 0.39	0.15 ± 0.24
MGDA	1.38 ± 0.02	$\mathbf{93.29 \pm 0.31}$	83.51 ± 0.30	-1.39 ± 0.50
PCGrad	1.47 ± 0.03	92.92 ± 0.28	83.28 ± 0.38	1.13 ± 0.57
CAGrad	1.40 ± 0.02	93.06 ± 0.26	83.28 ± 0.46	-0.58 ± 0.59
IMTL-G	1.41 ± 0.03	93.10 ± 0.16	83.78 ± 0.47	-0.50 ± 0.89
Nash-MTL	1.42 ± 0.02	92.89 ± 0.10	83.19 ± 0.50	-0.17 ± 0.71
IGBv2	1.42 ± 0.02	93.09 ± 0.22	83.34 ± 0.33	-0.21 ± 0.50
Aligned-MTL-UB	1.45 ± 0.02	93.00 ± 0.24	83.36 ± 0.43	0.66 ± 0.50
BayesAgg-MTL (Ours)	$\mathbf{1.35 \pm 0.03}$	93.01 ± 0.17	$\mathbf{84.25 \pm 0.35}$	$\mathbf{-2.23 \pm 0.76}$

(Wilson et al., 2016a;b). We follow the same path here and apply an initial pre-training step using linear scalarization. We would like to stress here that in all the experiments, the overall number of training steps for BayesAgg-MTL (including the pre-training) is the same as all methods.

5.1. BayesAgg-MTL for Regression

We first evaluated BayesAgg-MTL on an MTL problem with regression tasks only. We used the QM9 dataset which contains $\sim 130,000$ stable small organic molecules represented as graphs having node and edge features (Ramakrishnan et al., 2014; Wu et al., 2018). The goal here is to predict 11 chemical properties, such as geometric and energetic ones, that may vary in scale and difficulty of the tasks. We follow the experimental protocol of Navon et al. (2022). Specifically, we allocate approximately $110,000$ examples for training, with separate validation and testing sets with $10,000$ examples each. Additionally, we employ the message-passing neural network architecture (Gilmer et al., 2017) in conjunction with the pooling operator described in (Vinyals et al., 2016).

The test results for this dataset are presented in Table 1. Baseline method results were taken from (Dai et al., 2023), except for Aligned-MTL-UB, which is included here for the first time. The criterion used in Δ_m here is the mean absolute error (MAE) of the losses. From the table, BayesAgg-MTL achieves the best test performance, with a significant improvement compared to most of the baseline methods.

To gain a better intuition into the weights that BayesAgg-MTL assigns, we define here again the vector of weights per example and task from Eq. 7, $\boldsymbol{\alpha}_i^k := \boldsymbol{\lambda}_i^k / (\sum_{k=1}^K \boldsymbol{\lambda}_i^k)$. Figure 3 depicts for all tasks the average over dimensions of $\boldsymbol{\alpha}_i^k$ for 20 random examples at the start, middle, and end of training. The plot reveals an interesting pattern. Early in training, the average weights are distributed among the tasks without any specific pattern. As training progresses, larger weights are assigned for tasks $4 - 10$ in the middle of the training, while tasks $0 - 3$ receive smaller weights. At the end of the training, this pattern changes, and tasks $0 - 3$ are assigned with larger weights compared to tasks $4 - 10$.

5.2. BayesAgg-MTL for Binary Classification

Next, we evaluated BayesAgg-MTL on the MTL benchmarks CIFAR-MTL (Krizhevsky et al., 2009; Rosenbaum et al., 2018), and ChestX-ray14 (Wang et al., 2017). To the best of our knowledge, we are the first to evaluate MTL methods on the latter dataset. These datasets contain a large number of tasks, 20 and 14 respectively, with a high class-imbalance distribution. This poses a significant challenge for current MTL methods.

CIFAR-MTL uses the coarse labels of the CIFAR-100 dataset to create an MTL benchmark having 20 binary tasks. Classes from this dataset are grouped into super-classes (fish, flowers, trees, etc.), such that each example is given a one-hot encoding vector of labels indicating the super-class it belongs to. We use the official train-test split having $50,000$ examples and $10,000$ examples respectively. We allocate $5,000$ examples from the training set for a validation set. Our experiments on this dataset were conducted using a simple NN having 3 convolution layers.

ChestX-ray14 contains $\sim 112,000$ X-ray images of chests from $32,717$ patients. Each image has labels from 14 binary classes corresponding to the occurrence or absence of thoracic diseases. Multiple diseases can appear together in a patient. In our experiments, we mostly follow the training protocol suggested in (Taslimi et al., 2022) that used ResNet-34 for the shared parameters. we use the official split of $70\% - 10\% - 20\%$ for training, validation, and test.

We present the test results for these datasets in Table 2. On the CIFAR-MTL we report the accuracy in class assignment, and on the ChestX-ray14 we report the Δ_m based on the AUC-ROC values per task. From the table, BayesAgg-MTL performs best on both datasets. Interestingly, on the ChestX-ray14 dataset almost all methods, except for ours and DWA, under-perform the naive LS baseline. In Appendix C.2 we compare the run-time of all methods on this dataset and on the QM9. We show that BayesAgg-MTL is substantially faster than other baseline methods that use gradients w.r.t the shared parameters to weigh the tasks.

5.3. BayesAgg-MTL for Mixed Tasks

In the last set of experiments, we evaluated BayesAgg-MTL and baseline methods on the UTKFace dataset (Zhang et al., 2017). This dataset contains over $20,000$ face images with annotations of age, gender, and ethnicity. The age values range from 0 to 116, treated as a regression task. Gender is classified into binary categories, either male or female, while ethnicity is classified into five distinct categories, making it a multi-class classification task. We split the dataset according to $70\% - 10\% - 20\%$ to train, validation, and test datasets. Here, we use ResNet-18 for the shared network.

Results for this dataset based on 8 random seeds are presented in Table 3. Here as well BayesAgg-MTL outperforms all methods, having the best results on 2 out of 3 tasks. Interestingly, our approach and MGDA, were the only methods to improve upon the STL baseline on the regression task.

6. Conclusions

In this study, we present BayesAgg-MTL , a novel method for aggregating the task gradients in MTL. Instead of treating the gradient of each task as a deterministic quantity we advocate here to assign a probability distribution over them. The randomness in them arises by noticing that there are many possible configurations for the task-specific parameters that work well. Hence, by tracking all of them using Bayesian tools we can obtain a richer description of the gradient space. This in turn allows us to model the uncertainty in the gradients and derive an update direction for the shared parameters that takes it into account. We demonstrate our method's effectiveness on several benchmark datasets compared with leading baseline methods. For future work, we would like to extend BayesAgg-MTL beyond linear task heads. The challenge here would be to *efficiently* estimate the Bayesian posterior and the gradient moments.

Acknowledgements

This study was funded by a grant to GC from the Israel Science Foundation (ISF 737/2018), and by an equipment grant to GC and Bar-Ilan University from the Israel Science Foundation (ISF 2332/18). IA is supported by a PhD fellowship from Bar-Ilan data science institute (BIU DSI).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Achituve, I., Maron, H., and Chechik, G. Self-supervised learning for domain adaptation on point clouds. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 123–133, 2021a.

Achituve, I., Navon, A., Yemini, Y., Chechik, G., and Fetaya, E. GP-Tree: A Gaussian process classifier for few-shot incremental learning. In *International Conference on Machine Learning*, pp. 54–65. PMLR, 2021b.

Achituve, I., Shamsian, A., Navon, A., Chechik, G., and Fetaya, E. Personalized federated learning with Gaussian processes. *Advances in Neural Information Processing Systems*, 34:8392–8406, 2021c.

Achituve, I., Chechik, G., and Fetaya, E. Guided deep kernel learning. In *Uncertainty in Artificial Intelligence*. PMLR, 2023.

Baxter, J. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000.

Bishop, C. Pattern recognition and machine learning. *Springer google schola*, 2:531–537, 2006.

Brier, G. W. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

Brookes, M. The matrix reference manual. http://www.ee.imperial.ac.uk/hp/staff/dmb/matrix/intro.html, 2020.

Calandra, R., Peters, J., Rasmussen, C. E., and Deisenroth, M. P. Manifold Gaussian processes for regression. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 3338–3345. IEEE, 2016.

Caruana, R. Multitask learning. *Machine learning*, 28: 41–75, 1997.

Chen, Z., Badrinarayanan, V., Lee, C.-Y., and Rabinovich, A. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 794–803. PMLR, 10–15 Jul 2018.

Chen, Z., Ngiam, J., Huang, Y., Luong, T., Kretzschmar, H., Chai, Y., and Anguelov, D. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 2039–2050. Curran Associates, Inc., 2020.

Chennupati, S., Sistu, G., Yogamani, S., and Rawashdeh, S. Multinet++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

Daheim, N., Möllenhoff, T., Ponti, E., Gurevych, I., and Khan, M. E. Model merging by uncertainty-based gradient matching. In *The Twelfth International Conference on Learning Representations*, 2023.

Dai, Y., Fei, N., and Lu, Z. Improvable gap balancing for multi-task learning. In *Uncertainty in Artificial Intelligence*, pp. 496–506. PMLR, 2023.

D'Angelo, F. and Fortuin, V. Repulsive deep ensembles are Bayesian. *Advances in Neural Information Processing Systems*, 34:3451–3465, 2021.

Davies, R. B. Numerical inversion of a characteristic function. *Biometrika*, 60(2):415–417, 1973.

Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., and Hennig, P. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34:20089–20103, 2021.

Désidéri, J.-A. Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318, 2012.

Devin, C., Gupta, A., Darrell, T., Abbeel, P., and Levine, S. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2169–2176. IEEE, 2017.

Dimitriadis, N., Frossard, P., and Fleuret, F. Pareto manifold learning: Tackling multiple tasks via ensembles of single-task models. In *International Conference on Machine Learning*, pp. 8015–8052. PMLR, 2023.

Elich, C., Kirchdorfer, L., Köhler, J. M., and Schott, L. Challenging common assumptions in multi-task learning. *arXiv preprint arXiv:2311.04698*, 2023.

Fernando, H., Shen, H., Liu, M., Chaudhury, S., Murugesan, K., and Chen, T. Mitigating gradient bias in multi-objective learning: A provably convergent stochastic approach. In *International Conference on Learning Representations*, 2023.

Fey, M. and Lenssen, J. E. Fast graph representation learning with pytorch geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Fortuin, V., Garriga-Alonso, A., Ober, S. W., Wenzel, F., Ratsch, G., Turner, R. E., van der Wilk, M., and Aitchison, L. Bayesian neural network priors revisited. In *International Conference on Learning Representations*, 2021.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Guo, M., Haque, A., Huang, D.-A., Yeung, S., and Fei-Fei, L. Dynamic task prioritization for multitask learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Immer, A., Bauer, M., Fortuin, V., Rätsch, G., and Emtiyaz, K. M. Scalable marginal likelihood estimation for model selection in deep learning. In *International Conference on Machine Learning*, pp. 4563–4573. PMLR, 2021.

Javaloy, A. and Valera, I. Rotograd: Gradient homogenization in multitask learning. In *International Conference on Learning Representations*, 2022.

Kendall, A., Gal, Y., and Cipolla, R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7482–7491, 2018.

Kingma, D. P. and Ba, J. ADAM: A method for stochastic optimization. In *International Conference on Learning Representations*, 2014.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations*, 2015.

Kristiadi, A., Hein, M., and Hennig, P. Being Bayesian, even just a bit, fixes overconfidence in Relu networks. In *International conference on machine learning*, pp. 5436–5446. PMLR, 2020.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Kurin, V., De Palma, A., Kostrikov, I., Whiteson, S., and Mudigonda, P. K. In defense of the unitary scalarization for deep multi-task learning. *Advances in Neural Information Processing Systems*, 35:12169–12183, 2022.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Lin, B., Ye, F., Zhang, Y., and Tsang, I. W. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research*, 2022.

Liu, B., Liu, X., Jin, X., Stone, P., and Liu, Q. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890, 2021.

Liu, B., Feng, Y., Stone, P., and Liu, Q. Famo: Fast adaptive multitask optimization, 2023.

Liu, L., Li, Y., Kuang, Z., Xue, J.-H., Chen, Y., Yang, W., Liao, Q., and Zhang, W. Towards impartial multi-task learning. In *International Conference on Learning Representations*, 2020.

Liu, S., Johns, E., and Davison, A. J. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1871–1880, 2019a.

Liu, X., He, P., Chen, W., and Gao, J. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4487–4496, 2019b.

MacKay, D. J. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992.

Maninis, K.-K., Radosavovic, I., and Kokkinos, I. Attentive single-tasking of multiple tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1851–1860, 2019.

Martens, J. and Sutskever, I. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 1033–1040, 2011.

Matena, M. S. and Raffel, C. A. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

Michelsanti, D., Tan, Z.-H., Zhang, S.-X., Xu, Y., Yu, M., Yu, D., and Jensen, J. An overview of deep-learning-based audio-visual speech enhancement and separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1368–1396, 2021.

Minka, T. P. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pp. 362–369, 2001.

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3994–4003, 06 2016. doi: 10.1109/CVPR.2016.433.

Naeini, M. P., Cooper, G. F., and Hauskrecht, M. Obtaining well calibrated probabilities using Bayesian binning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pp. 2901–2907. AAAI Press, 2015.

Navon, A., Shamsian, A., Achituve, I., Maron, H., Kawaguchi, K., Chechik, G., and Fetaya, E. Multi-task learning as a bargaining game. In *International Conference on Machine Learning*, pp. 16428–16446. PMLR, 2022.

Neal, R. M. and Hinton, G. E. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pp. 355–368. Springer, 1998.

Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

Rosenbaum, C., Klinger, T., and Riemer, M. Routing networks: Adaptive selection of non-linear functions for multi-task learning. In *International Conference on Learning Representations*, 2018.

Ruder, S. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. ImageNet large scale visual recognition challenge. *International journal of computer vision*, 115: 211–252, 2015.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2019.

Särkkä, S. *Bayesian Filtering and Smoothing*, volume 3 of *Institute of Mathematical Statistics textbooks*. Cambridge University Press, 2013.

Saul, L. K., Jaakkola, T., and Jordan, M. I. Mean field theory for Sigmoid Belief Networks. *Journal of artificial intelligence research*, 4:61–76, 1996.

Schaul, T., Borsa, D., Modayil, J., and Pascanu, R. Ray interference: a source of plateaus in deep reinforcement learning, 2019.

Schraudolph, N. N. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738, 2002.

Sener, O. and Koltun, V. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

Senushkin, D., Patakin, N., Kuznetsov, A., and Konushin, A. Independent component alignment for multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20083–20093, 2023.

Shamshad, F., Khan, S., Zamir, S. W., Khan, M. H., Hayat, M., Khan, F. S., and Fu, H. Transformers in medical imaging: A survey. *Medical Image Analysis*, pp. 102802, 2023.

Shamsian, A., Navon, A., Glazer, N., Kawaguchi, K., Chechik, G., and Fetaya, E. Auxiliary learning as an asymmetric bargaining game. *arXiv preprint arXiv:2301.13501*, 2023.

Shi, H., Ren, S., Zhang, T., and Pan, S. J. Deep multitask learning with progressive parameter sharing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19924–19935, 2023.

Shu, T., Xiong, C., and Socher, R. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. In *International Conference on Learning Representations*, 2018.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Prabhat, M., and Adams, R. Scalable Bayesian optimization using deep neural networks. In *International conference on machine learning*, pp. 2171–2180. PMLR, 2015.

Standley, T., Zamir, A., Chen, D., Guibas, L., Malik, J., and Savarese, S. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pp. 9120–9132. PMLR, 2020.

Taslimi, S., Taslimi, S., Fathi, N., Salehi, M., and Rohban, M. H. SwincheX: Multi-label classification on chest X-ray images with transformers. *arXiv preprint arXiv:2206.04246*, 2022.

Vinyals, O., Bengio, S., and Kudlur, M. Order matters: Sequence to sequence for sets. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR*, 2016.

Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., and Summers, R. M. ChestX-ray8: Hospital-scale chest X-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2097–2106, 2017.

Wang, Z., Tsvetkov, Y., Firat, O., and Cao, Y. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. In *International Conference on Learning Representations*, 2020.

Wightman, R. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

Wild, V. D., Ghalebikesabi, S., Sejdinovic, D., and Knoblauch, J. A rigorous link between deep ensembles and (variational) Bayesian methods. *Advances in Neural Information Processing Systems*, 36, 2024.

Wilson, A. G. and Izmailov, P. Bayesian deep learning and a probabilistic perspective of generalization. *Advances in neural information processing systems*, 33:4697–4708, 2020.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In *Artificial intelligence and statistics*, pp. 370–378. PMLR, 2016a.

Wilson, A. G., Hu, Z., Salakhutdinov, R. R., and Xing, E. P. Stochastic variational deep kernel learning. *Advances in neural information processing systems*, 29, 2016b.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. MoleculeNet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Xin, D., Ghorbani, B., Gilmer, J., Garg, A., and Firat, O. Do current multi-task optimization methods in deep learning even help? *Advances in Neural Information Processing Systems*, 35:13597–13609, 2022.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33: 5824–5836, 2020.

Yun, H. and Cho, H. Achievement-based training progress balancing for multi-task learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16935–16944, October 2023.

Zhang, Y. and Yang, Q. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2021.

Zhang, Z., Song, Y., and Qi, H. Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5810–5818, 2017.

Zheng, C., Wu, W., Chen, C., Yang, T., Zhu, S., Shen, J., Kehtarnavaz, N., and Shah, M. Deep learning-based human pose estimation: A survey. *ACM Computing Surveys*, 56(1):1–37, 2023.

Zhou, C., Li, Q., Li, C., Yu, J., Liu, Y., Wang, G., Zhang, K., Ji, C., Yan, Q., He, L., et al. A comprehensive survey on pretrained foundation models: A history from BERT to ChatGPT. *arXiv preprint arXiv:2302.09419*, 2023.

A. Full Derivations

We now present the full derivation for Eq. 4 & Eq. 9 presented in the main text. For clarity, we drop here the superscript notation of the task.

A.1. Regression Moments

Starting with the first moment,

$$\begin{aligned}\mathbb{E}[\mathbf{g}_i] &= \int \mathbf{g}_i p(\mathbf{g}_i) d\mathbf{g}_i = \int \mathbf{g}_i(\mathbf{w}) p(\mathbf{w}|\mathcal{D}) d\mathbf{w} \\ &= 2\int \mathbf{w}(\mathbf{h}_i^T\mathbf{w} - y_i) p(\mathbf{w}|\mathcal{D}) d\mathbf{w} \\ &= 2\int \mathbf{w}\mathbf{w}^T\mathbf{h}_i - y_i\mathbf{w} p(\mathbf{w}|\mathcal{D}) d\mathbf{w} \\ &= 2([\mathbf{S} + \mathbf{m}\mathbf{m}^T]\mathbf{h}_i - y_i\mathbf{m}).\end{aligned} \tag{12}$$

Where we made explicit the dependence in $\mathbf{w}$ on the first step. For computing the second moment we aided by the matrix reference manual (Brookes, 2020),

$$\begin{aligned}\mathbb{E}[\mathbf{g}_i\mathbf{g}_i^T] &= \int \mathbf{g}_i\mathbf{g}_i^T p(\mathbf{g}_i) d\mathbf{g}_i = \int \mathbf{g}_i(\mathbf{w})\mathbf{g}_i^T(\mathbf{w}) p(\mathbf{w}|\mathcal{D}) d\mathbf{w} \\ &= 4\int \mathbf{w}(\mathbf{h}_i^T\mathbf{w} - y_i)(\mathbf{h}_i^T\mathbf{w} - y_i)\mathbf{w}^T p(\mathbf{w}|\mathcal{D}) d\mathbf{w} \\ &= 4\int (y_i^2\mathbf{w}\mathbf{w}^T\mathbf{h}_i - 2y_i\mathbf{w}\mathbf{h}_i^T\mathbf{w}\mathbf{w}^T + \mathbf{w}\mathbf{w}^T\mathbf{h}_i\mathbf{h}_i^T\mathbf{w}\mathbf{w}^T) p(\mathbf{w}|\mathcal{D}) d\mathbf{w}.\end{aligned} \tag{13}$$

We now solve each term separately and obtain the result,

$$\begin{aligned}\int \mathbf{w}\mathbf{w}^T p(\mathbf{w}|\mathcal{D}) d\mathbf{w} &= \mathbf{S} + \mathbf{m}\mathbf{m}^T, \\ \int \mathbf{w}\mathbf{h}_i^T\mathbf{w}\mathbf{w}^T p(\mathbf{w}|\mathcal{D}) d\mathbf{w} &= \mathbf{m}\mathbf{h}_i^T(\mathbf{S} + \mathbf{m}\mathbf{m}^T) + (\mathbf{S} + \mathbf{m}\mathbf{m}^T)\mathbf{h}_i\mathbf{m}^T + \mathbf{h}_i^T\mathbf{m}(\mathbf{S} - \mathbf{m}\mathbf{m}^T), \\ \int \mathbf{w}\mathbf{w}^T\mathbf{h}_i\mathbf{h}_i^T\mathbf{w}\mathbf{w}^T p(\mathbf{w}|\mathcal{D}) d\mathbf{w} &= (\mathbf{S} + \mathbf{m}\mathbf{m}^T)(\mathbf{A}_i + \mathbf{A}_i^T)(\mathbf{S} + \mathbf{m}\mathbf{m}^T) + \mathbf{m}^T\mathbf{A}_i\mathbf{m}(\mathbf{S} - \mathbf{m}\mathbf{m}^T) + Tr(\mathbf{A}_i\mathbf{S})(\mathbf{S} + \mathbf{m}\mathbf{m}^T).\end{aligned} \tag{14}$$

Where, $\mathbf{A}_i = \mathbf{h}_i\mathbf{h}_i^T$.

A.2. Second Order Posterior Approximation

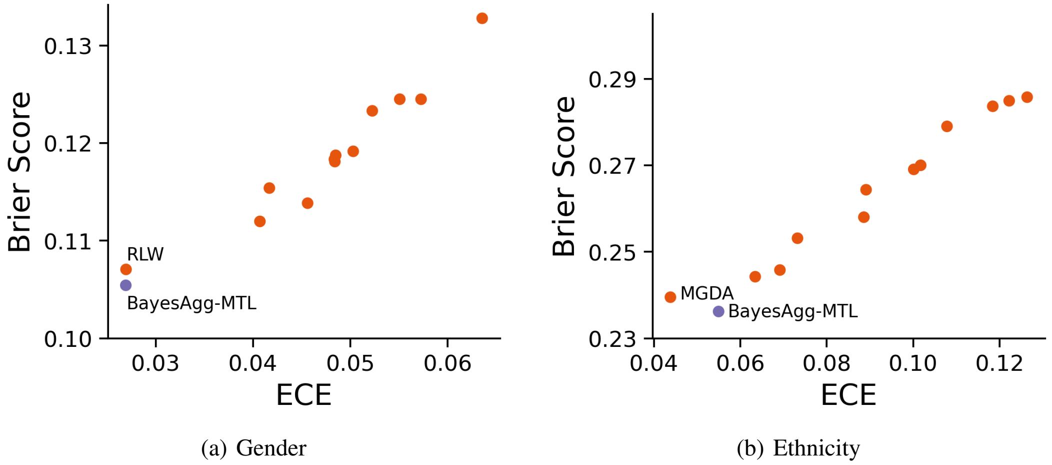
We now present the quadratic form of the log-posterior in Eq. 9. First we recap some of our notations here,

$$c = log\ p(\hat{\mathbf{w}}|\mathcal{B}); \quad \mathbf{a} = -\frac{\partial log\ p(\mathbf{y}|\mathbf{X},\mathbf{w})}{\partial \mathbf{w}} - \frac{\partial log\ p(\mathbf{w})}{\partial \mathbf{w}}\bigg|_{\mathbf{w}=\hat{\mathbf{w}}}; \quad \mathbf{B} = -\frac{\partial^2 log\ p(\mathbf{y}|\mathbf{X},\mathbf{w})}{\partial \mathbf{w}^2} - \frac{\partial^2 log\ p(\mathbf{w})}{\partial \mathbf{w}^2}\bigg|_{\mathbf{w}=\hat{\mathbf{w}}}. \tag{15}$$

Using these constants in Eq. 8 yields the following form:

$$\begin{aligned}&c + \mathbf{a}^T(\mathbf{w} - \hat{\mathbf{w}}) + \frac{1}{2}(\mathbf{w} - \hat{\mathbf{w}})^T\mathbf{B}(\mathbf{w} - \hat{\mathbf{w}}) \\ =&\underbrace{c - \mathbf{a}^T\hat{\mathbf{w}} + \frac{1}{2}(\hat{\mathbf{w}}^T\mathbf{B}\hat{\mathbf{w}}) - (\mathbf{B}^T\hat{\mathbf{w}} - \mathbf{a})^T\mathbf{B}^{-1}(\mathbf{B}^T\hat{\mathbf{w}} - \mathbf{a})}_{const.} + \frac{1}{2}(\mathbf{w} - (\hat{\mathbf{w}} - \mathbf{B}^{-1}\mathbf{a}))^T\mathbf{B}(\mathbf{w} - (\hat{\mathbf{w}} - \mathbf{B}^{-1}\mathbf{a})).\end{aligned} \tag{16}$$

The above takes the quadratic form of a Gaussian having mean $(\hat{\mathbf{w}} - \mathbf{B}^{-1}\mathbf{a})$ and covariance $\mathbf{B}^{-1}$.



(a) Gender (b) Ethnicity

Figure 4. Expected calibration error (ECE) vs Brier score for the Gender and Ethnicity tasks from the UTKFace dataset. In orange - baseline methods, and in purple our method. Lower values are better. We named our method and the top competitor on each plot.

B. Full Experimental Details

All the experiments were done using PyTorch on NVIDIA V100 and A100 GPUs having 32GB of memory.

QM9. On this dataset we followed the training protocol presented by Navon et al. (2022). Specifically, We allocated $110,000$ examples for training and $10,000$ examples for validation and testing. The task labels are normalized to have zero mean and unit std. We use the implementation of (Fey & Lenssen, 2019) for the message-passing NN presented in (Gilmer et al., 2017) as the base NN. Here, we trained only our method and the baseline method Aligned-MTL-UB. All the other results were taken from (Navon et al., 2022; Dai et al., 2023). We used the same random seeds as in those studies. Each method was trained for 300 epochs using the ADAM optimizer (Kingma & Ba, 2014) with an initial lr of $1e-3$. The batch size was set to 120. We use the ReduceOnPlate scheduler with the Δ_m metric, computed on the validation set. This metric was also used for early stopping and model selection. For BayesAgg-MTL we set the number of pre-training epochs using linear scalarization to 50. In initial experiments, we found that in regression tasks relatively higher values for the s hyper-parameter were preferred. Hence, we searched over $s \in \{0.75, 0.85, 0.95\}$. For the Aligned-MTL-UB we did a hyper-parameter search over the scale modes in {min, median, and rmse}, and whether to apply that scale to the task-specific parameters as well.

CIFAR-MTL. Similarly to (Rosenbaum et al., 2018), to form an MTL benchmark we used the coarse labels of CIFAR-100. Each example in the CIFAR-100 dataset belongs to one of 20 super-classes. We use these super-classes as separate binary MTL tasks, where the task value is 1 if the example indeed belongs to the super-class and 0 otherwise. We use the official CIFAR train-test split of $50,000$ and $10,000$ respectively. We allocated $5,000$ examples from the training set to validation. To train the models we use a CNN having 3 convolution layers with 160 channels and a kernel of size 3. Each convolution was followed by an Exponential Linear Unit (ELU) activation and max-pooling of 3×3. The final layer is a batch normalization layer. All methods were trained for 50 epochs using the ADAM optimizer, with an initial learning rate of $1e-3$ and a scheduler that drops the learning rate by a factor of 0.1 at 60% and 80% of the training. We set the batch size to 128 and used a weight decay of $1e-4$. For all baseline methods, we did a hyper-parameter grid search over the most important $2-3$ hyper-parameters. Specifically, we would like to highlight that we searched over additional weight decay values for the LS, SI, and RLW baselines as advocated by Kurin et al. (2022). As for BayesAgg-MTL , unlike the regression case, for classification smaller s values are preferred. We searched over $s \in \{5e^{-2}, 5e^{-3}, 5e^{-4}\}$. Also, we search over the number of pre-train epochs in $\{1, 3\}$. We set J, the number of Monte-Carlo samples, to 1024, although we could have used much less without performance degradation. We used the validation accuracy for early stopping and model selection.

ChestX-ray14. This dataset reports the absence or appearance of 14 types of chest diseases, which we view as an MTL problem. It contains approximately $112,000$ images from $32,717$ patients. We use the official data split presented in (Wang et al., 2017), having 70% training examples, 10% validation examples, and 20% test examples. We follow the experimental setup of (Taslimi et al., 2022) that uses PyTorch Image Models (Wightman, 2019) for data augmentations, a publicly

Table 4. Average run time (Sec. $\times 10^2$) of a training iteration on CIFAR-MTL and QM9 datasets.

	CIFAR-MTL	**QM9**
LS	1.551	3.932
SI	1.600	3.957
RLW	1.555	3.910
DWA	1.571	3.937
UW	1.740	4.033
MGDA	15.34	39.57
PCGrad	19.99	29.89
CAGrad	12.19	26.16
IMTL-G	10.12	27.19
Nash-MTL	22.47	45.32
IGBv2	3.651	4.723
Gradient w.r.t Representation		
MGDA-UB	5.522	9.352
IMTL-G	2.969	4.426
Aligned-MTL-UB	2.988	4.428
BayesAgg-MTL (Ours)	5.558	4.177

available repository. We resize each image to size 224×224 and use data augmentation such as color jitter having 0.4 intensity and random erase of pixels with a probability of 0.25. Images are normalized according to ImageNet statistics (Russakovsky et al., 2015). We use here ResNet-34 pre-trained on ImageNet as the shared feature extractor. We replaced the final classification layer with a fully connected layer of dimension 256 followed by an ELU activation. Experimental details and hyper-parameter searches are similar to those described for CIFAR-MTL, except for the following changes. Here we trained for 100 epochs, the batch size was set to 256, and we didn't use a weight decay. We use the Δ_m metric for early stopping and model selection.

UTKFace. This dataset contains approximately $23,700$ images of faces, each associated with the age, gender, and ethnicity of the person. We remove 3 examples from the dataset that have missing labels. We split the dataset to train/validation/test according to the $70-10-20$ scheme. The split was stratified by the age variable as it is the most diverse label. We treat the task of predicting the age as a regression task, and we normalize it to have zero mean and unit std. During training, images are resized to 140×140, randomly cropped to size 128, and undergo random horizontal flip. Test images are resized and centered cropped. Here, we used ResNet-18 with the final classification layer replaced by a fully connected layer of size 256 and an ELU activation. The experimental setup is similar to that described under the CIFAR-MTL, with the exception that here we trained for 100 epochs. We perform a hyper-parameter grid search for all methods on this dataset as well. For our method, we set the number of pre-training epochs to 10 and searched over $s \in \{0.3, 0.5, 0.8\}$ for the regression task and $s \in \{0.005, 0.05, 0.1\}$ for the classification tasks. We use the Δ_m metric for early stopping and model selection. Optimizing and evaluating the regression task is done using the MSE loss and the classification tasks using the standard cross-entropy loss.

C. Additional Experiments

C.1. Calibration

A possible benefit of using a Bayesian layer as the last layer is enhanced uncertainty estimation capabilities. Here we compare BayesAgg-MTL to baseline methods on that aspect. To do so we log the expected calibration error (ECE) (Naeini et al., 2015) and Brier score (Brier, 1950) for all methods on the classification tasks of the UTKFace dataset. In ECE we first discretize the $[0, 1]$ line segment and then measure a weighted average difference between the classifier confidence and accuracy. We use 15 interval bins in our comparison. Brier score measures the mean square error between the one-hot label encoding and the prediction probability vector. In both metrics, lower values are better. Results are presented in Figure 4. From the figure, BayesAgg-MTL is better calibrated than most methods on both datasets. On the gender task, it is best calibrated according to the two metrics. On the Ethnicity task, it has the best Brier score and second-best ECE score. We stress here that for a fair comparison with baseline methods, we did not use the Bayesian posterior of BayesAgg-MTL on the last layer to make test predictions, but rather the point estimate of it learned during training. Using the full posterior should yield even better results.

C.2. Training Time

Table 4 compares the run time of all methods on the CIFAR-MTL and QM9 datasets. We report the average processing time of a batch based on 10 epochs. To do the comparison, we use the best hyper-parameter configuration (in terms of performance) according to the CIFAR-MTL experiments. For MGDA and IMTL-G we present the run time under two settings, (1) when using in the aggregation scheme the full gradients w.r.t the shared parameters (top block); (2) when using in the aggregation scheme the gradients w.r.t the hidden layer (bottom block) as BayesAgg-MTL does. For BayesAgg-MTL we do not include the pre-training steps in the time measurements. From the table, methods that do not rely on the gradients for weighing the tasks are faster as outlined before in previous studies (Xin et al., 2022; Kurin et al., 2022); however, this often comes at a significant performance reduction. BayesAgg-MTL training time is almost as fast as those methods on regression problems, in which everything is done in closed-form, and slower on classification problems, partly due to the sampling process. Nevertheless, it is substantially faster than other gradient balancing methods that use gradients w.r.t the shared parameters.

C.3. Comparison to Bayesian Training

Table 5. Comparison of $\Delta_m\%$ values to Deep Ensembles averaged over 3 random seeds.

	QM9	**UTKFace**
Ensemble (1024 heads)	161.4 ± 13.1	0.99 ± 0.62
Ensemble (10 networks)	144.5 ± 0.3	-0.13 ± 0.39
BayesAgg-MTL (Ours)	$\mathbf{53.2 \pm 7.1}$	$\mathbf{-2.23 \pm 0.76}$

Given that we used a Bayesian inference procedure in our approach, a natural question one may ask is *how does standard approximate Bayesian training perform in MTL?*

Recall that the goal of this paper is to use Bayesian inference on the last layer as a means to train deterministic MTL models using the uncertainty estimates in the gradients of the tasks. We use these uncertainty estimates to come up with an aggregation rule for combining the gradients of the tasks to a shared update direction. More concisely, our aim is to *better learn a deterministic MTL model* while reducing as much as possible the computational overhead involved in training it. In standard approximate Bayesian training the gradient used in the backward process is considered as a deterministic quantity, similarly to non-Bayesian training. Hence, even when applying standard Bayesian inference to the task-specific parameters, the optimization issues regarding *how to combine* the gradients of the tasks effectively remain.

To showcase that we compare BayesAgg-MTL to deep ensembles (Lakshminarayanan et al., 2017) that have a strong link to approximate Bayesian methods (Wilson & Izmailov, 2020; D'Angelo & Fortuin, 2021; Wild et al., 2024). We chose deep ensembles because of their simplicity and predictive abilities. We show here results on QM9 and UTKFace for two baselines: (1) Using 1024 heads for each task and a shared backbone; (2) Using 10 networks, each with a different backbone and task heads. The latter is substantially computationally more demanding as it requires different copies of the backbone as well, which is usually large. We combine the tasks using linear scalarization (i.e., equal weighting of the tasks) and averaging over the ensemble members. We follow the same experimental protocol of the paper and report the Δ_m values for each method in Table 5. From the table, the ensemble model with the shared backbone performs roughly the same as standard linear scalarization, with a slight advantage on QM9. This result makes sense as the uncertainty information is not taken into account when aggregating the gradients (i.e., only the mean values are used). Full ensemble training improves upon the ensemble baseline having a shared feature extractor, but it comes with a substantial computational overhead. Finally, BayesAgg-MTL substantially outperforms both methods on both datasets.

C.4. Full Results

In Tables 6 and 7 we present the per-task results for all methods on the QM9 and ChestX-ray14 respectively. On QM9 we report the mean-absolute error of each task and on ChestX-ray14 the AUC-ROC of the tasks. Due to lack of space, we abbreviated several diseases names from the ChestX-ray14. We outline here the full names of all diseases: Atelectasis, Cardiomegaly, Consolidation, Edema, Effusion, Emphysema, Fibrosis, Hernia, Infiltration, Mass, Nodule, Pleural_Thickening, Pneumonia, Pneumothorax.

Table 6. QM9. Full test results averaged over 3 random seeds.

	μ	α	ϵ_{HOMO}	ϵ_{LUMO}	$\langle R^2 \rangle$	ZPVE	U_0	U	H	G	c_v	
						MAE ↓						$\mathbf{\Delta_m}\%$ (↓)
STL	0.067	0.181	60.57	53.91	0.503	4.53	58.80	64.20	63.80	66.20	0.072	
LS	0.106	0.326	73.57	89.67	5.197	14.06	143.4	144.2	144.6	140.3	0.129	177.6
SI	0.309	0.346	149.8	135.7	1.003	4.51	55.32	55.75	55.82	55.27	0.112	77.8
RLW	0.113	0.340	76.95	92.76	5.869	15.47	156.3	157.1	157.6	153.0	0.137	203.8
DWA	0.107	0.325	74.06	90.61	5.091	13.99	142.3	143.0	143.4	139.3	0.125	175.3
UW	0.387	0.425	166.2	155.8	1.065	5.00	66.42	66.78	66.80	66.24	0.123	108.0
MGDA	0.217	0.368	126.8	104.6	3.227	5.69	88.37	89.41	89.32	88.01	0.120	120.5
PCGrad	0.106	0.293	75.85	88.33	3.940	9.15	116.4	116.8	117.2	114.5	0.110	125.7
CAGrad	0.118	0.321	83.51	94.81	3.219	6.93	114.0	114.3	114.5	112.3	0.116	112.8
IMTL-G	0.136	0.288	98.31	93.96	1.753	5.70	101.4	102.4	102.0	100.1	0.097	77.2
Nash-MTL	0.103	0.249	82.95	81.89	2.426	5.38	74.52	75.02	75.10	74.16	0.093	62.0
IGBv2	0.251	0.333	149.1	130.2	0.956	4.39	56.75	57.19	57.25	56.73	0.110	67.7
Aligned-MTL-UB	0.172	0.350	117.3	109.0	1.520	5.23	76.13	76.58	76.62	75.71	0.980	71.0
BayesAgg-MTL (Ours)	0.122	0.280	87.78	90.44	1.776	5.31	63.33	64.91	66.71	81.91	0.093	**53.2**

Table 7. ChestX-ray14. Full test results averaged over 3 random seeds.

	Atel.	Card.	Cons.	Edema	Effu.	Emph.	Fibr.	Hernia	Infi.	Mass	Nodule	Pleu.	Pneumonia	Pneu.	
								AUC-ROC ↑							$\mathbf{\Delta_m}\%$ (↓)
STL	.7543	.8615	.7132	.8212	.8224	.6333	.7357	.7647	.6830	.6208	.5894	.6389	.5710	.7701	
LS	.7744	.8804	.7477	.8457	.8273	.8798	.8250	.9129	.7013	.8209	.7593	.7660	.7235	.8525	−14.62
SI	.7457	.8739	.7289	.8426	.8152	.8593	.7903	.8045	.6996	.7971	.7268	.7353	.6993	.8389	−1.94
RLW	.7596	.8704	.7389	.8385	.8218	.8390	.7956	.8646	.6991	.7933	.7340	.7362	.7101	.8345	−11.69
DWA	.7734	.8847	.7503	.8482	.8267	.8768	.8185	.9410	.6977	.8175	.7590	.7739	.7240	.8440	−14.79
UW	.7600	.8870	.7437	.8464	.8221	.8768	.8176	.9434	.7012	.8049	.7426	.7608	.7057	.8498	−13.95
MGDA	.7720	.8857	.7473	.8454	.8260	.8762	.8181	.9290	.6961	.8141	.7570	.7661	.7213	.8479	−14.44
PCGrad	.7678	.8793	.7461	.8432	.8266	.8721	.8165	.8565	.6991	.8123	.7499	.7629	.7203	.8451	−13.43
CAGrad	.7744	.8823	.7489	.8464	.8269	.8756	.8199	.9201	.6998	.8158	.7567	.7702	.7207	.8482	−14.50
IMTL-G	.7395	.8533	.7229	.8235	.8023	.7692	.7538	.8973	.6903	.7543	.7052	.7221	.6758	.8026	−8.24
Nash-MTL	.7623	.8774	.7445	.8420	.8206	.8627	.8214	.8997	.6999	.8035	.7412	.7553	.7117	.8447	−13.24
IGBv2	.7189	.8354	.7049	.8097	.7865	.7360	.7160	.7053	.6858	.6828	.6647	.7038	.6512	.7783	−2.83
Aligned-MTL-UB	.7689	.8801	.7491	.8456	.8245	.8772	.8221	.8992	.6997	.8115	.7543	.7674	.7208	.8497	−14.15
BayesAgg-MTL (Ours)	.7761	.8836	.7511	.8487	.8293	.8863	.8289	.9121	.6967	.8220	.7622	.7762	.7214	.8545	**−14.96**