Multimodal Probabilistic Inference for Robust
Uncertainty Quantification

by

Ghassen Jerfel

Department of Electrical and Computer Engineering
Duke University

Date: ________________

Approved:

_________________________
Katherine A. Heller, Advisor

_________________________
Cynthia Rudin

_________________________
Henry Pfister

_________________________
Hai Li

_________________________
Vahid Tarokh

Dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the Graduate School of
Duke University

2021
ABSTRACT

Multimodal Probabilistic Inference for Robust Uncertainty Quantification

by

Ghassen Jerfel

Department of Electrical and Computer Engineering
Duke University

Date: ____________________________

Approved:

______________________________
Katherine A. Heller, Advisor

______________________________
Cynthia Rudin

______________________________
Henry Pfister

______________________________
Hai Li

______________________________
Vahid Tarokh

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering in the Graduate School of Duke University

2021
Abstract

Deep learning models, which form the backbone of modern ML systems, generalize poorly to small changes to the data distribution. They are also bad at signalling failure, making predictions with high confidence when their training data or fragile assumptions make them unlikely to make reasonable decisions. This lack of robustness makes it difficult to trust their use in safety-critical settings. Accordingly, there is a pressing need to equip models with a notion of uncertainty to understand their failure modes and detect when their decisions cannot be used or require intervention. Uncertainty quantification is thus crucial for ML systems to work consistently on real-world data and fail loudly otherwise. One growing line of research on uncertainty quantification is probabilistic modelling which is concerned with capturing model uncertainty by placing a distribution over models hypotheses which can be marginalized at test-time. This is especially useful in underspecified models which can have diverse near-optimal settings in terms of population-level performance. However, probabilistic modelling approaches such as Bayesian Neural Networks (BNN) do not scale well in terms of memory and runtime and often underperform simple deterministic baselines in terms of accuracy. Furthermore, BNNs underperform deep ensembles as they fail to
explore multiple modes, in the loss space, while being effective at averaging within a single mode.

In this thesis, we develop multimodal representations of model uncertainty that can capture a diverse set of hypotheses. We first propose a scalable family of BNN priors (and corresponding approximate posteriors) that combine the local (i.e. within-mode) uncertainty with mode averaging to ensure robust and calibrated uncertainty estimates in addition to improving accuracy both in and out of distribution. We then develop a multimodal representation of inductive biases to modulate the amount of information transfer between tasks in meta-learning. Our proposed framework integrates Bayesian non-parametric mixtures with deep learning to enable NNs to adapt their capacity as more data is observed which is crucial for lifelong learning. Finally, we propose to replace the reverse Kullback-Leibler divergence (RKL), known for its mode-seeking behavior and for underestimating posterior covariance, with the forward KL (FKL) divergence in a novel theoretically-guided inference algorithm that efficiently combines variational boosting with adaptive importance sampling. The proposed algorithm offers a well-defined compute-accuracy trade-off and is guaranteed to converge to the optimal multimodal variational solution as well as the optimal importance sampling proposal distribution.
To my grandmothers and loving memory of my grandfathers.
Contents

Abstract iv

List of Figures xii

List of Tables xiv

Acknowledgements xvi

1 Introduction 1

1.1 Calibrated Predictive Uncertainty .......................... 2

1.2 Disentangling the Sources of Uncertainty ...................... 4

1.3 Multimodal Epistemic Uncertainty in Underspecified Models ........ 6

1.4 Multimodal Probabilistic Modeling ................................ 8

1.5 Thesis Structure .................................................. 10

1.5.1 Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors . 10

1.5.2 Nonparametric Mixtures for Lifelong Learning ................. 11

1.5.3 Multimodal Inference via Forward KL Optimization ............. 12

2 Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors 14

2.1 Introduction ...................................................... 13

2.2 Background ....................................................... 16

2.2.1 Variational inference for Bayesian Neural Networks .......... 16
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Illustration of the difference between the solutions captured by deep ensem-</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>bles and those captured by variational methods for BNNs</td>
<td></td>
</tr>
<tr>
<td>2.1</td>
<td>An illustration of how to generate the ensemble weights for two ensemble</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>members in BatchEnsembles</td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>Induced weight priors</td>
<td>20</td>
</tr>
<tr>
<td>2.3</td>
<td>Placing distributions over $r$ (output), $s$ (input), and both, evaluated on</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>the CIFAR-10 test set and CIFAR-10-C</td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td>Varying the number of mixture components in the rank-1 mixture of Gauss-</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>sians posteriors, evaluated on CIFAR-10 and CIFAR-10-C</td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>Training with a log-mixture likelihood vs an average per-component log-</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>likelihood</td>
<td></td>
</tr>
<tr>
<td>2.6</td>
<td>Disagreement versus accuracy and log-likelihood over consecutive model check-</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>points for rank-1 BNNs and BatchEnsemble on CIFAR-10/100</td>
<td></td>
</tr>
<tr>
<td>2.7</td>
<td>Out-of-distribution performance using ImageNet-C with ResNet-50</td>
<td>29</td>
</tr>
<tr>
<td>2.8</td>
<td>Out-of-distribution performance using CIFAR-10-C and CIFAR-100-C with WRN-</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>28-10</td>
<td></td>
</tr>
<tr>
<td>2.9</td>
<td>Dropout-parameterized initialization for the variational distribution’s stan-</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>dard deviations</td>
<td></td>
</tr>
<tr>
<td>2.10</td>
<td>Real-valued vs positive-valued priors</td>
<td>55</td>
</tr>
<tr>
<td>2.11</td>
<td>Median performance on CIFAR-10-C across corruption types</td>
<td>60</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Results for ResNet-50 on ImageNet ........................................ 37
2.2 Results for Wide ResNet-28-10 on CIFAR-10 .......................... 38
2.3 Results for Wide ResNet-28-10 on CIFAR-100 ....................... 39
2.4 Results for RNNs on the MIMIC-III EHR mortality task .......... 40
2.5 Hyperparameter values for Rank-1 BNNs with Wide ResNet-28-10 on CIFAR-10 and CIFAR-100 ........................................ 50
2.6 Hyperparameter values for Rank-1 BNNs with ResNet-50 on ImageNet ................................................................. 51
2.7 Hyperparameter values for Rank-1 Bayesian RNNs on MIMIC-III .... 52
2.8 Results across multiple weight samples (per mixture component, per example) at evaluation time for Wide ResNet-28-10 on CIFAR-10 ........................................ 56
3.1 Meta-test set accuracy on the miniImageNet 5-way, 1- and 5-shot classification benchmarks among methods using a comparable architecture .... 69
4.1 Comparing approximate inference techniques .......................... 97
4.2 Predictive log probabilities on test for BLR with Gaussian prior (mean ± standard error over 20 train/test splits) ................................ 108
4.3 Predictive log probabilities on test for BNNs with Gaussian prior (mean ± standard error over 20 train/test splits) ................................ 108
4.4 Predictive log probabilities on test for BLR with heavy tailed prior (mean ± standard error over 20 train/test splits) ........................ 110
4.5 Datasets used in experiments

4.6 Predictive log probabilities on test for BLR with Gaussian prior

4.7 Predictive log probabilities on test for BNNs with Gaussian prior

4.8 Predictive log probabilities on test for BLR with heavy tailed prior
My Ph.D. journey has been far from orthodox, but I have been extremely fortunate to encounter many amazing people without whom this dissertation would not be possible.

I would like to first express special thanks to my advisor, Katherine Heller, for her unwavering support and for offering me the flexibility to pursue my own research interests. Katherine has gone above and beyond to provide mentorship for both my professional and personal development. I am sincerely grateful for all the time and effort she has invested in my Ph.D.

I would also like to thank my host advisor at UC Berkeley, Michael Jordan, for welcoming me as one of his own and supporting me for the three wonderful years I spent at Berkeley.

I am grateful to my committee members Cynthia Rudin, Henry Pfister, David Carlson, Hai Li, and Vahid Tarokh for their guidance and support.

I am thankful to Yi-An Ma for his technical mentorship and advice as I was starting my research journey.

I am grateful for having met such an amazing group of collaborators and colleagues at Google Brain: Dustin Tran, Balaji Lakshminarayanan, Michael Dusenberry, Yeming Wen, Jeremy Nixon, Jasper Snoek, Zelda Mariet, Rodolphe Jenatton, and many others.

I have been lucky to meet excellent classmates and amazing friends during my first year at Duke including Federico Ferrari and Fan Bu.

I am thankful to several people I met over the course of my visiting position at Berkeley, whom I am fortunate to have as friends: Aldo Pacchiano, Nilesh Tripuraneni, Yeshwanth...
Cherapanamjeri, Eric Mazumdar, Armin Askari, Niladri Chatterjee, Serena Wang, Juanky Pedromo, and Tijana Zrnic.

I am grateful to old friends who supported me throughout my highs and lows at Berkeley such as Elizabeth Yang, Michael Chang, Kiran Vodrahalli, Angela Zhou, and Alexander Upjohn Beatson. I am grateful to my girlfriend, Komal, for her love, support, and patience.

Finally, I am thankful for my mom Monia and my dad Kamel for their academic inspiration, unconditional encouragement, and tasty care packages. I am also thankful to my sisters Ghada and Rima, aunts, uncles, and grandparents for their endless support throughout my journey in the United States.
Machine learning is increasingly deployed in high-impact decision-making applications such as self-driving cars [Caesar et al., 2020, Geiger et al., 2013, Sun et al., 2020], healthcare [Esteva et al., 2017, 2019, Gulshan et al., 2016, Henry et al., 2015, Yu et al., 2018, Tomasev et al., 2019, Schulam and Saria, 2017, Zech et al., 2018], and criminal justice [Lum and Isaac, 2016]. Without loss of generality, we limit our analysis in the following to deep neural networks which have formed the backbone of many recent machine learning algorithms for perception, classification, reinforcement learning, and action planning. These models are typically trained in sandbox environments limited to toy datasets or overused benchmarks which might systematically differ from the deployment environment [Stacke et al., 2019]. Therefore, it is not surprising that deep learning models generalize poorly to small changes to the data [Hendrycks and Dietterich, 2019]. For example, [Recht et al., 2019] create new test sets for the popular image recognition benchmarks of CIFAR-10 [Krizhevsky and Hinton, 2010] and ImageNet [Deng et al., 2009]. They observe that accuracies drop by between 3–15% on CIFAR and more than 11% on ImageNet (increases in error rate of 50–100%), even though the authors follow the original dataset creation processes. Furthermore, neural networks are bad at signalling failure [Hendrycks and Dietterich, 2019], and tend to make overconfident predictions [Guo et al., 2017, Ovadia et al., 2019] even when incorrect. Therefore, they fail in unexpected and poorly understood ways, hindering our ability to interpret and trust systems in which they are deployed [Azulay and Weiss, 2018].
To understand when one can rely on a model’s output, and ultimately build trust in ML systems, it is critical to qualify a model’s predictions with a notion of uncertainty or confidence in order to recognize its failure modes and detect when human intervention is necessary for making decisions based on such predictions. For example, machine learning is progressively deployed in pathology detection systems [Tomašev et al., 2019]. Knowing when the model is uncertain about its prediction allows practitioners to confidently let the machine handle the easier cases, in case of low uncertainty, and have more time to dedicate to the more complex cases if the model is conveying a high level of uncertainty.

Beside safety and reliability, there exist many applications which rely on a model’s uncertainty. These applications include choosing what data to learn from, or exploring an agent’s environment efficiently. Common to both these tasks is the use of uncertainty to learn from small amounts of data. This is often a necessity in settings in which data collection is expensive (such as the annotation of individual examples by an expert), or time consuming (such as the repetition of an experiment multiple times). Uncertainty is thus essential for exploration-exploitation trade-offs which characterize many tasks in active learning, black-box optimization, and reinforcement learning. Finally, uncertainty estimation is useful for going beyond i.i.d. assumptions and towards out-of-distribution (OOD) generalization. A measurement of high uncertainty should signal that an observed data point is significantly different from those observed during training. Furthermore, adapting to out-of-distribution examples could leverage uncertainty by re-weighting test-time predictions [Sugiyama et al., 2007].

1.1 Calibrated Predictive Uncertainty

Having established that model confidence is a critical quantity to possess, it is important to note that most deep learning models do not offer such information. For example, regression
models output a single vector regressing to the mean of the data. As for classification models, the probability vector obtained at the end of the pipeline (the softmax output) is often erroneously interpreted as model confidence [Guo et al., 2017]. In fact, softmax uncertainty is often overconfident [Guo et al., 2017].

A desired property of predictive uncertainty is calibration which ensures that a model’s confidence scores reflect the ground truth likelihood of the model’s correctness [Platt et al., 1999; Zadrozny and Elkan, 2001b]. In probabilistic terms, calibration certifies that the predicted probability of correctness (confidence) matches the observed frequency of correctness (accuracy). For example, in classification, calibration implies that if a model assigns a class with 90% probability, that class should appear 90% of the time. If a model is systematically over or under-confident, it can be difficult to reliably use its predicted probabilities for decision making.

The Expected Calibration Error (ECE) metric [Naeini et al., 2015] is the standard metric for approximating the calibration of a model given a finite dataset. Let \((\hat{Y}, \hat{P})\) denote the class prediction and associated confidence (predicted probability) of a classifier. We seek to compute the expected difference between confidence and accuracy [Naeini et al., 2015]:

\[
E_{\hat{P}}[|\mathbb{P}(\hat{Y} = Y|\hat{P} = p) - p|] \tag{1.1}
\]

ECE approximates this by binning the predictions in \([0, 1]\) under \(M\) equally-spaced intervals, and then taking a weighted average of each bins’ accuracy/confidence difference. Let \(B_m\) be the set of examples in the \(m^{th}\) bin whose predicted confidence falls into interval \((\frac{m-1}{M}, \frac{m}{M})\).

The bin \(B_m\)’s accuracy and confidence are:

\[
\text{Acc}(B_m) = \frac{1}{|B_m|} \sum_{x_i \in B_m} \mathbb{1}(\hat{y}_i = y_i), \quad \text{Conf}(B_m) = \frac{1}{|B_m|} \sum_{x_i \in B_m} \hat{p}_i, \tag{1.2}
\]

where \(\hat{y}_i\) and \(y_i\) are the predicted and true labels and \(\hat{p}_i\) is the confidence for example \(x_i\).
Given $n$ examples, ECE is

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} \left| \text{Acc}(B_m) - \text{Conf}(B_m) \right|.$$  \hfill (1.3)

Unfortunately, ground truth uncertainty is rarely available in practice. Furthermore, the cost of down-stream decisions may be difficult to model a-priori (e.g. in off-policy evaluation for sequential decision making). Therefore, to better understand how to achieve calibration, a decomposition of the sources of uncertainty is in order.

### 1.2 Disentangling the Sources of Uncertainty

In real-world deployment environments, uncertainty can originate from multiple sources:

- **Noisy data**: observed labels or features might be noisy, for example as a result of measurement imprecision.

- **Uncertainty in model parameters** that best explain the observed data: a large number of possible models might be able to explain a given dataset, which is known as underspecification. In this case, we might be uncertain which model parameters to predict with.

We can thus distinguish two sources of uncertainty: epistemic and aleatoric. Epistemic uncertainty is the uncertainty that arises because we have limited training data. The word comes from “episteme” which is Greek for “knowledge” (i.e. epistemic uncertainty is “knowledge uncertainty”). In essence, epistemic uncertainty arises when many model hypotheses are consistent with both the data we have observed. Given more training data, the epistemic uncertainty can be reduced, which improves the accuracy of our predictions. In modern literature, this has been also referred to as model uncertainty.

On the other hand, aleatoric uncertainty, which originates from the Latin word “aleator” or “dice player” is the uncertainty that arises from the inherent stochasticity within the
data generation process. Even if we repeat the same experiment under identical conditions, there is a chance that we may obtain different results. It is also called data uncertainty or irreducible uncertainty.

Probabilistic modelling provides a coherent framework for representing model uncertainty by placing prior distributions \( p(w) \) on model parameters \( w \) and inferring a posterior distribution \( p(w|x) \) that best explains the observed data. Combined with Bayesian decision theory, probabilistic modelling can form the foundation of rational decision making systems since it allows to integrate prior knowledge into learning systems and make sure that knowledge is updated in a coherent and robust way as more data is observed.

More specifically, let’s consider a probabilistic classification model \( p(y = f(x, w)|x) \) trained on a finite dataset \( D = \{x_j, y_j\}_{j=1}^N \sim p(x, y) \). The different sources of uncertainty are:

- **Input data uncertainty (aleatoric) \( p(x) \)** which is difficult to compute given a finite dataset but can be manipulated by data augmentation techniques.

- **Model uncertainty (epistemic)** which is represented by a distribution over functions \( f \sim G(f) \) often induced by a distribution over model parameters \( w \sim p(w) \).

- **Output uncertainty (aleatoric)**, also known as “noise” or “risk” \cite{Knight}, which is represented by the predictive distribution \( y \sim p(y = f(w, x)|x) \). For binary tasks, the predictive distribution equates to a Bernoulli distribution as parameterized by a single probability value \( \lambda = f(x, w) \) such that \( y \sim \text{Bernoulli}(\lambda) \).

Predictive uncertainty can then be written in the following equivalent forms:

\[
P(y|x) = \int p(y|\lambda)p(\lambda|x)d\lambda = \int p(y|x, w)p(w)dw
\] (1.4)

The first form of Eq.1.4 is commonly used for making predictions. However, for decision making purposes, we may be interested in separating the two types of uncertainty present as per the second formulation.
One popular way to measure the aleatoric uncertainty is thus the predictive entropy $E_{p(w)}[H(y = f(w, x)|w, x)]$. The epistemic uncertainty can then be quantified by computing the difference $H[y|x] - E_{p(w)}[H(y|w, x)] = I(y, w)$ which is the mutual information between $y$ and $w$.

### 1.3 Multimodal Epistemic Uncertainty in Under-specified Models

Underspecification is a phenomenon in which multiple model hypotheses can explain observed data equally well. In deep learning, underspecification is a direct consequence of having more degrees of freedom than data points. This leads to diffuse likelihoods in NNs, not strongly favoring any one setting of parameters. As such, a single problem might admit many near-optimal solutions, and these solutions might have very different properties when used for decision making which underscores the importance of multimodal epistemic uncertainty quantification. For example, in our initial exploration in [Dusenberry et al., 2019], we found that the individual models within an ensemble can collectively exhibit a wide variability in predicted probability for certain groups of data points despite being seemingly well-calibrated and having nearly identical dataset-level performances. We also found that this extends into the space of optimal decision: models with nearly equivalent metric performance can disagree significantly on the final decision, especially with non-linear utility.

The need for representing epistemic uncertainty is also evident from the shortcomings of popular methods for representing and calibrating predictive uncertainty such as maximum softmax probability [Hendrycks and Gimpel, 2016] and post-hoc calibration by temperature scaling (using i.i.d. validation set) [Guo et al., 2017; Platt et al., 1999], respectively. A recent benchmark [Ovadia et al., 2019] of state-of-the-art methods found that such methods are not robust under dataset shift (where the testing distribution differs from that of training) which is becoming commonplace in most applications. This can be explained by the fact
that these methods conflate data uncertainty with predictive uncertainty while ignoring the
model uncertainty component.

On the other hand deep ensembles [Lakshminarayanan et al., 2017] and Bayesian methods were found to perform well under distribution shift [Ovadia et al., 2019, Dusenberry et al., 2020]. Deep ensembles is a method for incorporating model uncertainty by collecting the predictions of M deterministic NNs trained independently with varying initialization [Lakshminarayanan et al., 2017] or hyper-parameters [Wenzel et al., 2020]. Bayesian methods typically marginalize over a distribution of models that can be represented by a distribution over model parameters in the simplest approaches. While these approaches suffer from a larger memory footprint and tend to be more resource intensive, they can capture multiple hypotheses, at training time, which enables a multimodal representation of epistemic uncertainty that is more robust to distribution shift.

It is also worth noting that such epistemic uncertainty methods have been fundamental to addressing important challenges in science:

- Diagnosing and indicating model misfit, e.g., indicating when a model does not know what to predict [Futoma et al., 2017, Kendall and Gal, 2017, Dusenberry et al., 2019, 2020]. This provides better understanding on how to revise the model including fairness, revise the data collecting, or how to include a human in the loop, especially for cost-sensitive or high-stakes decision-making.

- Improved detection, adaptation, and thus robustness to rare, anonymous, out of distribution, and adversarial examples [Lakshminarayanan et al., 2017, Hendrycks and Gimpel, 2016, Louizos and Welling, 2017, Malinin and Gales, 2018, Ovadia et al., 2019].

Figure 1.1: Illustration of the difference between the solutions captured by deep ensembles and those captured by variational methods for BNNs. x-axis indicates parameter values and y-axis plots the negative loss on train and validation data (taken from Fort et al., 2019).

- Transferring knowledge (sharing statistical strength) across a collection of datasets, and from arbitrary dataset sizes: continual learning, meta learning, few-shot learning [Garnelo et al., 2018, Titsias et al., 2019, Wen et al., 2020].

1.4 Multimodal Probabilistic Modeling

Having established the need for (multimodal) epistemic uncertainty, we turn our attention to probabilistic modeling with Bayesian Neural Networks (BNN) [MacKay, 1992b, Blundell et al., 2015]. BNNs marginalize over a distribution of neural network models, allowing for uncertainty quantification and improved robustness in deep learning. BNNs were extensively studied by [MacKay, 1992b], whose work was extended by [Neal, 1993]. More recently these ideas have been resurrected under different names with variational techniques by [Blundell et al., 2015, Graves, 2011, and Kingma and Ba, 2014b] although such techniques
can be traced back as far as Hinton and Van Camp [1993] and Bishop [1998]. Note that deep ensembles can be considered as a special type of BNNs.

In principle, BNNs can permit graceful failure, signalling when a model does not know what to predict [Kendall and Gal, 2017] and can also generalize better to out-of-distribution examples [Louizos and Welling, 2017, Malinin and Gales, 2018]. However, BNNs can be difficult to work with—often requiring many more parameters to be optimised—and haven’t really caught-on within the deep learning community, perhaps because of their limited practicality. More specifically, BNNs often underperform on metrics such as accuracy and do not scale as well as simpler baselines [Gal, 2016, Lakshminarayanan et al., 2017, Maddox et al., 2019].

A possible reason is that the best configurations for BNNs remain unknown: What is the optimal parameterization, weight prior, approximate posterior, or optimization strategy? The flexibility that accompanies these choices makes BNNs broadly applicable, but adds a high degree of complexity. Furthermore, maintaining a distribution over weights incurs a significant cost both in additional parameters and runtime complexity. For example, mean-field variational inference [Blundell et al., 2015] requires doubling the existing millions or billions of network weights (i.e., mean and variance for each weight) yet often underperforms the simple deterministic model.

Nonetheless, neural networks are typically underspecified by the data. Therefore, by selecting only a single set of weights, at the end of training, we lose uncertainty when the models could disagree for a test point. This is exactly when marginalization in BNNs can make the biggest difference, not only for calibration, but also for generalization. However, computing the exact posterior distribution over $w$ is computationally expensive (if not impossible) when $p(y|x, w)$ is a deep neural network (NN). A variety of approximations have been developed for Bayesian neural networks, including Laplace approximation [MacKay, 1992b], Markov chain Monte Carlo methods [Springenberg et al., 2016, Neal, 1995, Welling and Teh, 2011], variational inference (VI) [Graves, 2011, Blundell et al., 2015, Louizos and Welling, 2017, Wen et al., 2018] and Monte Carlo dropout [Gal, 2016].
Unfortunately, approximate inference typically converges to unimodal posteriors in practice (see Figure 1.1). While MCMC is known to struggle with multimodal targets, optimization-based approaches such as VI are vulnerable to pathologies in the divergence metric. Reverse KL ($KL(q||p) = E_q[\log q/p]$), for example, is known for being mode-seeking and underestimating posterior covariance.

On the other hand, deep ensembles capture different modes with independent random initializations and demonstrate strong performance both in terms of accuracy and calibrated uncertainty. However, deep ensembles are parameter inefficient—often capturing similar if not identical model hypotheses—and do not capture the same local uncertainty that BNNs do. Ovadia et al. [2019] and Gustafsson et al. [2019] independently benchmarked existing methods for uncertainty quantification on a variety of datasets and architectures, and observed that ensembles tend to outperform approximate Bayesian neural networks in terms of both accuracy and uncertainty, particularly under dataset shift. This can be explained by the fact that ensembles tend to sample from different modes in function space, whereas approximate Bayesian methods (such as VI which minimizes the mode-seeking reverse KL divergence) might fail to explore multiple modes even though they are effective at capturing uncertainty within a single mode (see Figure 1.1 for a cartoon illustration).

There is thus a clear need for BNNs which can combine the multimodal representation of deep ensembles with the local uncertainty of Bayesian methods in underspecified deep learning models.

### 1.5 Thesis Structure

#### 1.5.1 Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors

We seek to develop models that recognize when they do not know how to predict by capturing their epistemic uncertainty through the framework of Bayesian neural networks (BNN). BNNs have shown promise for improved uncertainty estimates, but generally struggle with
(1) underfitting at scale, and (2) parameter inefficiency. As an alternative, Deep (deterministic) Ensembles [Lakshminarayanan et al., 2017] demonstrate strong performance, but are still parameter inefficient and capture epistemic uncertainty in a possibly different manner. How can we address these practical challenges and combine the strengths of both approaches?

In Chapter 2 we propose a rank-1 parameterization of Bayesian neural nets, where each weight matrix involves only a distribution on a rank-1 subspace. Rank-1 BNNs involve sampling rank-1 multiplicative perturbations (or a mixture thereof) that can be applied to a shared weight matrix (which is learned as a point-estimate). This enables the use of mixture approximate posterior distributions in order to benefit from a multimodal representation with local uncertainty around each mode. However, unlike typical ensembles, mixtures on the rank-1 subspace involve a significantly reduced dimensionality. This low-dimensional parameterization also allows us to more efficiently leverage heavy-tailed distributions such as Cauchy [Louizos et al., 2017] without sacrificing predictive performance.

1.5.2 Nonparametric Mixtures for Lifelong Learning

One significant challenge for BNNs is the choice of prior. Meta-learning offers a set of tools for automatically learning priors from data in order to generalize to new tasks with few observations. In fact, gradient-based meta-learning can be considered as an empirical Bayes procedure [Grant et al., 2018]. However, one shortcoming of meta-learning algorithms is the assumption that all tasks are equally related. The real world often presents scenarios in which an agent must decide what degree of transfer is appropriate. In some cases, a subset of tasks are more strongly related to each other, and so non-uniform transfer provides a strategic advantage. On the other hand, transfer in the presence of dissimilar or outlier tasks worsens generalization performance [Rosenstein et al., 2005]. Moreover, when the underlying task distribution is non-stationary, inductive transfer to previously observed tasks should exhibit graceful degradation to address the catastrophic forgetting problem [Kirkpatrick et al., 2017]. In this scenario, multimodal epistemic uncertainty is crucial to
avoiding the consolidation of all inductive biases into a single set of parameters that are not well suited to deal with evolving task distributions.

In Chapter 3 we propose to leverage nonparametric priors over neural network parameters to automatically detect and adapt to task distribution shift in a non-stationary training environment; addressing the non-trivial setting of task-agnostic continual learning in which the task change is unobserved [Kirkpatrick et al., 2017]. This allows the neural network model to adjust its capacity as the observed data grows in complexity. We propose a stochastic and augmented Expectation-Maximization algorithm for approximating a Dirichlet process mixture model over the parameters. We then translate stochastic point estimation in an infinite mixture [Rasmussen, 2000] over model parameters into a gradient-based algorithm that is compatible with any differentiable likelihood model and requires no distributional assumptions.

1.5.3 Multimodal Inference via Forward KL Optimization

Variational inference, the most popular method for approximate Bayesian inference, posits a family \( Q \) of distributions that are easy to evaluate or sample from, and defines the variational approximation \( q^* \in Q \) as the distribution that minimizes the reverse Kullback-Leibler (RKL) divergence \( (E_q[\log q/p]) \) to the posterior \( p \). Regardless of the choice of family \( Q \), the RKL divergence favors a single mode (mode seeking) and is biased towards avoiding false positives, which can cause severe under-estimation of the covariance [Blei et al., 2016; Murphy, 2012] and poses difficulties when approximating heavy-tailed [Guo et al., 2016; Li and Turner, 2016; Dieng et al., 2017] or multimodal targets [Miller et al., 2017]. Furthermore, since the variational family \( Q \) rarely includes the true posterior, refining posterior approximations through importance weighting is often recommended to de-bias posterior expectations of interest [Yao et al., 2018; Vehtari et al., 2015]. However, the light tails of VI approximations lead to high or even infinite variance for the IS estimator. As for multimodal targets, variational boosting (VB) [Jerfel, 2017; Miller et al., 2017; Guo et al., 2016; Locatello et al., 2018] has been suggested in various forms to address to iteratively
construct a variational mixture distribution. However, reverse KL-based boosting is known to struggle with degeneracy where the optimization at certain boosting iterations can lead to point-mass components [Campbell and Li 2019].

In Chapter 4 we propose to substitute the reverse KL divergence with the forward KL divergence (\(E_p[\log p/q]\)). This paves the path for combining of Variational Inference and Importance Sampling by constructing an IS proposal distribution through the unifying lens of forward KL (FKL) minimization which has been proven to control the downstream error of importance sampling (IS) [Chatterjee and Diaconis 2018]. This approach guarantees asymptotic consistency and a fast convergence towards both the optimal IS estimator and the optimal variational approximation. We can straightforwardly extend this approach to perform a sequential greedy construction of mixture proposal distributions and prove that the resulting FKL-based boosting algorithm is guaranteed to converge on multimodal targets without the degeneracy concerns of RKL boosting.
Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors

2.1 Introduction

Bayesian neural networks (BNNs) marginalize over a distribution of neural network models for prediction, allowing for uncertainty quantification and improved robustness in deep learning. In principle, BNNs can permit graceful failure, signalling when a model does not know what to predict \cite{KendallGal2017,Dusenberry2019}, and can also generalize better to out-of-distribution examples \cite{Louizos2017,MalininGales2018}. However, there are two important challenges prohibiting their use in practice.

First, Bayesian neural networks often underperform on metrics such as accuracy and do not scale as well as simpler baselines \cite{Gal2016,Lakshminarayanan2017,Maddox2019}. A possible reason is that the best configurations for BNNs remain unknown. What is the best parameterization, weight prior, approximate posterior, or optimization strategy? The flexibility that accompanies these choices makes BNNs broadly applicable, but adds a high degree of complexity.

Second, maintaining a distribution over weights incurs a significant cost both in additional parameters and runtime complexity. Mean-field variational inference \cite{Blundell2015}, for example, requires doubling the existing millions or billions of network weights.
Using an ensemble of size 5, or 5 MCMC samples, requires 5x the number of weights. In contrast, simply scaling up a deterministic model to match this parameter count can lead to much better predictive performance on both in- and out-of-distribution data [Recht et al., 2019].

In this chapter, we develop a flexible distribution over neural network weights that achieves state-of-the-art accuracy and uncertainty while being highly parameter-efficient. We address the first challenge by building on ideas from deep ensembles [Lakshminarayanan et al., 2017], which work by aggregating predictions from multiple randomly initialized, stochastic gradient descent (SGD)-trained models. Fort et al. [2019] identified that deep ensembles’ multimodal solutions provide uncertainty benefits that are distinct and complementary to distributions centered around a single mode.

We address the second challenge by leveraging recent work that has identified neural network weights as having low effective dimensionality for sufficiently diverse and accurate predictions. For example, Li et al. [2018] find that the “intrinsic” dimensionality of popular architectures can be on the order of hundreds to a few thousand. Izmailov et al. [2019] perform Bayesian inference on a learned 5-dimensional subspace. Wen et al. [2020] apply ensembling on a rank-1 perturbation of each weight matrix and obtain strong empirical success without needing to learn the subspace. Swiatkowski et al. [2019] apply singular value decomposition post-training and observe that a rank of 1-3 captures most of the variational posterior’s variance.

**Contributions.** We propose a rank-1 parameterization of Bayesian neural nets, where each weight matrix involves only a distribution on a rank-1 subspace. This parameterization addresses the above two challenges. It also allows us to more efficiently leverage heavy-tailed distributions [Louizos and Welling, 2017], such as Cauchy, without sacrificing predictive performance. Finally, we revisit the use of mixture approximate posteriors as a simple strategy for aggregating multimodal weight solutions, similar to deep ensembles. Unlike typical ensembles, however, mixtures on the rank-1 subspace involve a significantly reduced dimensionality (for a mixture of size 10 on ResNet-50, it is only 0.4% more parameters.
instead of 900%). Rank-1 BNNs are thus not only parameter-efficient but also scalable, as Bayesian inference is only done over thousands of dimensions.

Section 2.3 performs an empirical study on the choice of prior, variational posterior, and likelihood formulation. Section 2.3 also presents a theoretical analysis of the expressiveness of rank-1 distributions. Section 2.4 shows that, on ImageNet with ResNet-50, rank-1 BNNs outperform the original network and BatchEnsemble [Wen et al., 2020] on log-likelihood, accuracy, and calibration on both the test set and ImageNet-C. On CIFAR-10 and 100 with Wide ResNet 28-10, rank-1 BNNs outperform the original model, Monte Carlo dropout, BatchEnsemble, and original BNNs across log-likelihood, accuracy, and calibration on both the test sets and the corrupted versions, CIFAR-10-C and CIFAR-100-C [Hendrycks and Dietterich, 2019]. Finally, on the MIMIC-III electronic health record (EHR) dataset [Johnson et al., 2016] with LSTMs, rank-1 BNNs outperform deterministic and stochastic baselines from Dusenberry et al. [2019].

2.2 Background

2.2.1 Variational inference for Bayesian Neural Networks

Bayesian neural networks posit a prior distribution over weights \( p(W) \) of a network architecture. Given a dataset \((X, y)\) of \( N \) input-output pairs, we perform approximate Bayesian inference using variational inference: we select a family of variational distributions \( q(W) \) with free parameters and then minimize the Kullback-Leibler (KL) divergence from \( q(W) \) to the true posterior \( p(W | X, y) \) [Jordan et al., 1999]. Taking a minibatch of size \( B \), this is equivalent to minimizing the loss function,

\[
- \frac{N}{B} \sum_{b=1}^{B} \mathbb{E}_{q(W)}[\log p(y_b | x_b, W)] + KL(q(W) || p(W)),
\]

with respect to the parameters of \( q(W) \). This loss function is an upper bound on the negative log-marginal likelihood \(- \log p(y | X)\) and can be interpreted as the model’s ap-
proximate description length [Hinton and Van Camp, 1993].

In practice, Bayesian neural nets often underfit, mired by complexities in both the choice of prior and approximate posterior, and in stabilizing the training dynamics involved by the loss function (e.g., posterior collapse [Bowman et al., 2016]) and the additional variance from sampling weights to estimate the expected log-likelihood. In addition, note even the simplest solution of a fully-factorized normal approximation incurs a 2x cost in the typical number of parameters.

2.2.2 Ensemble & BatchEnsemble

Deep ensembles [Lakshminarayanan et al., 2017] are a simple and effective method for ensembling, where one trains multiple copies of a network and then makes predictions by
aggregating the individual models to form a mixture distribution. However, this comes at the cost of training and predicting with multiple copies of network parameters.

BatchEnsemble [Wen et al., 2020] is a parameter-efficient extension that ensembles over a low-rank subspace. Let the ensemble size be $K$ and, for each layer, denote the original weight matrix $W \in \mathbb{R}^{m \times d}$, which will be shared across ensemble members. Each ensemble member $k$ owns a tuple of trainable vectors $r_k$ and $s_k$ of size $m$ and $d$ respectively. BatchEnsemble defines $K$ ensemble weights: each is

$$W_k' = W \circ F_k, \text{ where } F_k = r_k s_k^\top \in \mathbb{R}^{m \times d},$$

and $\circ$ denotes element-wise product. BatchEnsemble’s forward pass can be rewritten, where for a given layer,

$$y = \phi \left( W_k' x \right) = \phi \left( \left( W \circ r_k s_k^\top \right) x \right)$$

$$= \phi \left( \left( W (x \circ s_k) \right) \circ r_k \right),$$

where $\phi$ is the activation function, and $x \in \mathbb{R}^d, y \in \mathbb{R}^m$ is a single example. In other words, the rank-1 vectors $r_k$ and $s_k$ correspond to elementwise multiplication of input neurons and pre-activations (see Figure 2.1 for an illustration). This admits efficient vectorization as we can replace the vectors $x$, $r_k$, and $s_k$ with matrices where each row of $X \in \mathbb{R}^{B \times d}$ is a batch element and each row of $R \in \mathbb{R}^{B \times m}$ and $S \in \mathbb{R}^{B \times d}$ is a choice of ensemble member: $\phi \left( \left( (X \circ S)W^\top \right) \circ R \right)$. This vectorization extends to other linear operators such as convolution and recurrence.

### 2.3 Rank-1 Bayesian Neural Nets

Building on Equation 2.1, we introduce a rank-1 parameterization of Bayesian neural nets. We then empirically study choices such as the prior and variational posterior.
2.3.1 Rank-1 Weight Distributions

Consider a Bayesian neural net with rank-1 factors: parameterize every $m \times d$ weight matrix $W' = W \circ rs^T$, where the factors $r$ and $s$ are $m$ and $d$-vectors respectively. We place priors on $W'$ by placing priors on $r$, $s$, and $W$. Upon observing data, we compute for $r$ and $s$ (the rank-1 weight distributions), while treating $W$ as deterministic.

Variational Inference. For training, we apply variational EM where we perform approximate posterior inference over $r$ and $s$, and point-estimate the weights $W$ with maximum likelihood. The loss function is

$$
L = -\frac{N}{B} \sum_{b=1}^{B} \mathbb{E}_{q(r)q(s)}[\log p(y_b \mid x_b, W, r, s)] 
+ \text{KL}(q(r) \parallel p(r)) + \text{KL}(q(s) \parallel p(s)) - \log p(W),
$$

where the parameters are $W$ and the variational parameters of $q(r)$ and $q(s)$. In all experiments, we set the prior $p(W)$ to a zero-mean normal with fixed standard deviation, which is equivalent to an L2 penalty for deterministic models.

Using rank-1 distributions enables significant variance reduction: weight sampling only comes from the rank-1 variational distributions rather than over the full weight matrices (tens of thousands compared to millions). In addition, Equation 2.1 holds, enabling sampling of new $r$ and $s$ vectors for each example and for arbitrary distributions $q(r)$ and $q(s)$.

Multiplicative or Additive Perturbation? A natural question is whether to use a multiplicative or additive update. For location-scale family distributions, multiplication and addition only differ in the location parameter and are invariant under a scale reparameterization. For example: let $r_i \sim \text{Normal}(\mu, \sigma^2)$ and for simplicity, ignore $s$; then

$$
w_{ij}r_i = w_{ij}(\mu_i + \sigma_i \epsilon_i) = w_{ij}\mu_i + r_i',
$$

where $r_i' \sim \text{Normal}(0, \sigma_i'^2)$ and $\sigma_i' = w_{ij}\sigma_i$. Therefore additive perturbations only differ in
Figure 2.2: Induced weight priors. The distribution of a weight element is \( w'_{ij} = w_{ij}r_is_j \), where \( w_{ij} \sim N(0, \cdot) \), \( s_j \) is fixed at 1, and \( r_i \) is varied. Normal and Cauchy priors on \( r_i \) both encourage sparse weight posteriors: Cauchy has less mass around 0 and heavier tails. Inverse-Gamma \( r_i^2 \) induces a Student-T weight prior unlike a normal weight prior.

an additive location parameter \((+x \circ s \circ r)\). An additive location is often redundant as, when vectorized under Equation 2.1 it’s subsumed by any biases and skip connections.

2.3.2 Rank-1 Priors Are Hierarchical Priors

Priors over the rank-1 factors can be viewed as hierarchical priors on the weights in a noncentered parameterization, that is, where the distributions on the weights and scale factors are independent. This removes posterior correlations between the weights which can be otherwise difficult to approximate [Ingraham and Marks 2017, Louizos and Welling 2017]. We examine choices for priors based on this connection.

Hierarchy across both input and output neurons. Typical hierarchical priors for
Figure 2.3: Placing distributions over $\mathbf{r}$ (output), $\mathbf{s}$ (input), and both, evaluated over three runs on the CIFAR-10 test set and CIFAR-10-C. The best setup differs on the test set, while priors over both vectors generalize better on corruptions.

BNNs are Gaussian-scale mixtures, which take the form

$$p(\mathbf{W'}) = \int \mathcal{N}(\mathbf{W'} | 0, r^2 \sigma^2)p(r)p(\sigma^2) \, dr \, d\sigma^2,$$

where $\mathbf{r}$ is a vector shared across rows or columns and $\sigma$ is a global scale across all elements. Settings of $\mathbf{r}$ and $\sigma$ lead to well-known distributions (Figure 2.2): Inverse-Gamma variance induces a Student-t distribution on $\mathbf{W'}$; half-Cauchy scale induces a horseshoe distribution [Carvalho et al., 2009]. For rank-1 priors, the induced weight distribution is

$$p(\mathbf{W'}) = \iint \mathcal{N}(\mathbf{W'} | 0, (\mathbf{rs}^T \sigma)^2)p(\mathbf{r})p(\mathbf{s}) \, dr \, ds,$$

where $\mathbf{r}$ is a vector shared across columns; $\mathbf{s}$ is a vector shared across rows; and $\sigma$ is a scalar hyperparameter.

To better understand the importance of hierarchy, Figure 2.3 examines three settings under the best model on CIFAR-10 (Section 2.4.2): priors (paired with non-degenerate
posterior) on (1) only the vector $s$ that is applied to the layer’s inputs, (2) only the vector $r$ that is applied to the outputs, and (3) the default of both $s$ and $r$. The presence of a prior corresponds to a mixture of Gaussians with tuned, shared mean and standard deviation, and the corresponding approximate posterior is a mixture of Gaussians with learnable parameters; the absence of a prior indicates point-wise estimation. L2 regularization on the point-estimated $W$ is also tuned.

Looking at test performance, we find that the settings perform comparably on accuracy and differ slightly on test NLL and ECE. More interestingly, when we look at the corruptions task, the hierarchy of priors across both vectors outperforms the others on all three metrics, suggesting improved generalization. We hypothesize that the ability to modulate the uncertainty of both the inputs and outputs of each layer assists in handling distribution shift.

**Cauchy priors:** **Heavy-tailed real-valued priors.** Weakly informative priors such as the Cauchy are often preferred for robustness as they concentrate less probability at the mean thanks to heavier tails [Gelman et al., 2006]. The heavy tails encourage the activation distributions to be farther apart at training time, reducing the mismatch when passed out-of-distribution inputs. However, the exploration of heavy-tailed priors has been mostly limited to half-Cauchy [Carvalho et al., 2010] and log-uniform priors [Kingma et al., 2015] on the scale parameters, and there has been a lack of empirical success beyond compression tasks. These priors are often justified by the assumption of a positive support for scale distributions. However, in a non-centered parametrization, such restriction on the support is unnecessary and we find that real-valued scale priors typically outperform positive-valued ones (Section 2.7.3.2). Motivated by this, we explore in Section 2.4 the improved generalization and uncertainty calibration provided by Cauchy rank-1 priors.

2.3.3 Choice of Variational Posterior

**Role of Mixture Distributions.** Rank-1 BNNs admit few stochastic dimensions, making mixture distributions over weights more feasible to scale. For example, a mixture approx-
imate posterior with $K = 10$ components for ResNet-50 results in an 0.4% increase in parameters, compared with a 900% increase in deep ensembles. A natural question is: to what extent can we scale $K$ before there are diminishing returns? Figure 2.4 examines the best-performing rank-1 model under our CIFAR-10 setup, varying the mixture size $K \in \{1, 2, 4, 8, 16\}$. For each, we tune over the total number of training epochs, and measure NLL, accuracy, and ECE on both the test set and CIFAR-10-C corruptions dataset. As the number of mixture components increases from 1 to 8, the performance across all metrics increases. At $K = 16$, however, there is a decline in performance. Based on our findings, all experiments in Section 2.4 use $K = 4$.

For mixture size $K = 16$, we suspect the performance is a result of the training method and hardware memory constraints. Namely, we start with a batch of $B$ examples and duplicate it $K$ times so that each mixture component applies a forward pass for each example; the total batch size supplied to the model is $B \cdot K$. We keep this total batch size constant as we increase $K$ in order to maintain constant memory. This implies that as the number of mixture components increases, the batch size $B$ of new data points decreases. We suspect alternative implementations such as sampling mixture components may enable further scaling.

**Role of Non-Degenerate Components.** To understand the role of non-degenerate distributions (i.e., distributions that do not have all probability mass at a single point), note that BatchEnsemble can be interpreted as using a mixture of Dirac delta components. Section 2.4 compares to BatchEnsemble in depth, providing broad evidence that mixtures consistently improve results (particularly accuracy), and using non-degenerate components further lowers probabilistic metrics (NLL and ECE) as well as improves generalization to out-of-distribution examples.

### 2.3.4 Log-likelihood: Mixture or Average?

When using mixture distributions as the approximate posterior, the expected log-likelihood in Equation 2.2 involves an average over all mixture components. By Jensen’s inequality,
Figure 2.4: Varying the number of mixture components in the rank-1 mixture of Gaussians posteriors, evaluated over five runs on the CIFAR-10 test set and CIFAR-10-C corrupted dataset. Increasing the number of components yields improved performance up to a limit.

one can get a tighter bound on the log-marginal likelihood by using the log-mixture density,

\[
\log \frac{1}{K} \sum_{k=1}^{K} p(y_n \mid x_n, \theta_k) \geq \frac{1}{K} \sum_{k=1}^{K} \log p(y_n \mid x_n, \theta_k),
\]

where \( \theta_k \) are per-component parameters. The log-mixture likelihood is typically preferred over the average as it is guaranteed to provide at least as good a bound on the log-marginal. Appendix 2.7.4 contains a further derivation of the various choices of log-likelihood losses for such models.

However, deep ensembles when interpreted as a mixture distribution correspond to using the average as the loss function: for the gradient of parameters \( \theta_k \) in mixture component
\[ \nabla_{\theta_{k'}} \log \frac{1}{K} \sum_{k=1}^{K} p(y \mid x, \theta_k) = \frac{\nabla p(y \mid x, \theta_{k'})}{K^{-1} \sum_{k=1}^{K} p(y \mid x, \theta_k)} \]
\[ \nabla_{\theta_{k'}} \frac{1}{K} \sum_{k=1}^{K} \log p(y \mid x, \theta_k) = \frac{1}{K} \nabla p(y \mid x, \theta_{k'}). \]

Therefore, while the log-mixture likelihood is an upper bound, it incurs a communication cost where each mixture component’s gradients are a function of how well the other mixture components fit the data. This communication cost prohibits the use of log-mixture likelihood as a loss function for deep ensembles, where randomly initialized ensemble members are trained independently.

We wonder whether deep ensembles’ lack of communication across mixture components and relying purely on random seeds for diverse solutions is in fact better. With rank-1 priors, we can do either with no extra cost: Figure 2.5 compares the two using the best rank-1 BNN hyperparameters on CIFAR-10. Note that we always use the log-mixture likelihood for evaluation. While the training metrics in Figure 2.5 are comparable, the log-mixture likelihood generalizes worse than the average log-likelihood, and the individual mixture components also generalize worse. It seems that, at least for misspecified models such as overparametrized neural networks, training a looser bound on the log-likelihood leads to improved predictive performance. We conjecture that this might simply be a case of ease of optimization allowing the model to explore more distinct modes throughout the training procedure.

### 2.3.5 Ensemble Diversity

The diversity of predictions returned by different members of an ensemble is an important indicator of the quality of uncertainty quantification [Fort et al., 2019] and of the robustness of the ensemble [Pang et al., 2019]. Following [Fort et al., 2019], Figure 2.6 examines the disagreement of rank-1 BNNs and BatchEnsemble members against accuracy and log-
Figure 2.5: Training with a log-mixture likelihood vs an average per-component log-likelihood. Blue is averaged (test) performance; colors are individual components; **black** is averaged (train) performance. Training metrics are identical but the average consistently outperforms on the test set.

We quantify diversity by the fraction of points where discrete predictions differ between two members, averaged over all pairs. This disagreement measure is normalized by \((1 - \text{acc})\) to account for the fact that the lower the accuracy of a member, the more random its predictions can be. Unsurprisingly, Figure 2.6 demonstrates a negative correlation between accuracy and diversity for both methods. For the same or higher predictive performance, rank-1 BNNs achieve a higher degree of ensemble diversity than BatchEnsemble on both CIFAR-10 and CIFAR-100.

This can be attributed to the non-degenerate posterior distribution around each mode of the mixture, which can better handle modes that are closest together. In fact, a deterministic mixture model could place multiple modes within a single valley in the loss landscape parametrized by weights. Accordingly, the ensemble members are likely to collapse on near-identical modes in the function space. On the other hand, a mixture model that can capture the uncertainty around each mode might be able to detect a single ‘wide’ mode, as characterized by large variance around the mean. Overall, the improved diversity
Figure 2.6: Disagreement versus accuracy and log-likelihood over consecutive model checkpoints, at the end of training, for rank-1 BNNs and BatchEnsemble on CIFAR-10/100. Rank-1 BNNs demonstrate a higher diversity while achieving better predictive performance than BatchEnsemble.

result confirms our intuition about the necessity of combining local (near-mode) uncertainty with a multimodal representation in order to improve the predictive performance of mode averaging.

2.3.6 Expressiveness of Rank-1 Distribution

A natural question is how expressive a rank-1 distribution is. Theorem 1 below demonstrates that the rank-1 perturbation encodes a wide range of perturbations in the original weight matrix $W$. We prove that, for a fully connected neural network, the rank-1 parameterization has the same local variance structure in the score function as a full-rank’s.

Theorem 1 (Informal). In a fully connected neural network of any width and depth, let $W_*$ denote a local minimum associated with a score function over a dataset. Assume that
the full-rank perturbation on the weight matrix in layer $h$ has the multiplicative covariance structure that

$$
\mathbb{E}_{\mathbf{W}^{(h)}} \left[ \left( \mathbf{W}^{(h)} - \mathbf{W}_*^{(h)} \right)_{i,j} \left( \mathbf{W}^{(h)} - \mathbf{W}_*^{(h)} \right)_{k,l} \right] = \mathbf{W}_*^{(h)} \Sigma_{i,j,k,l} \mathbf{W}_*^{(h)}_{k,l},
$$

for some symmetric positive semi-definite matrix $\Sigma$. Let $\mathbf{s}_*^{(h)}$ denote a column vector of ones. Then if the rank-1 perturbation has covariance

$$
\mathbb{E}_{\mathbf{s}^{(h)}} \left[ \left( \mathbf{s}^{(h)} - \mathbf{s}_*^{(h)} \right) \left( \mathbf{s}^{(h)} - \mathbf{s}_*^{(h)} \right) \right]^T = \Sigma,
$$

the score function has the same variance around the local minimum.

**Theorem 1** demonstrates a correspondence between the covariance structure in the perturbation of $\mathbf{W}$ and that of $\mathbf{s}$. Since $\Sigma$ can be any symmetric positive semi-definite matrix, our rank-1 parameterization can efficiently encode a wide range of fluctuations in $\mathbf{W}$. In particular, it is especially suited for multiplicative noise as advertised. If the covariance of $(\mathbf{W} - \mathbf{W}_*)$ is proportional to $\mathbf{W}_* \otimes \mathbf{W}_*^T$ itself, then we can simply take the covariance of $(\mathbf{s} - \mathbf{s}_*)$ to be identity. See Section 2.7.1 for a formal version of **Theorem 1**.

### 2.4 Experiments

In this section, we show results on image classification and electronic health record classification tasks: ImageNet, CIFAR-10, CIFAR-100, their corrupted variants [Hendrycks and Dietterich, 2019], and binary mortality prediction with the MIMIC-III EHR dataset [Johnson et al., 2016]. For ImageNet, we use a ResNet-50 baseline as it’s the most commonly benchmarked model [He et al., 2016]. For CIFAR, we use a Wide ResNet 28-10 baseline as it’s a simple architecture that achieves 95%+ test accuracy on CIFAR-10 with little data augmentation [Zagoruyko and Komodakis, 2016]. For MIMIC-III, we use recurrent neural networks (RNNs) based on the setup in [Dusenberry et al., 2019].

**Baselines.** For the image classification tasks, we reproduce and compare to baselines
Figure 2.7: Out-of-distribution performance using ImageNet-C with ResNet-50. We plot NLL, accuracy, and ECE for varying corruption intensities; each result is the mean performance over 10 runs and over 15 corruption types. The error bars represent the standard deviation across corruption types. Figure 2.13 elaborates on these results in the Appendix. Rank-1 BNNs (red) perform best across all metrics.

with equal parameter count: deterministic (original network); Monte Carlo dropout [Gal, 2016]; and BatchEnsemble [Wen et al., 2020]. Although 2x the parameter count of other methods, we also tune a vanilla BNN baseline for CIFAR that uses Gaussian priors and approximate posteriors over the full set of weights with Flipout [Wen et al., 2018] for estimating expectations. We additionally include reproduced results for two deep ensemble [Lakshminarayanan et al., 2017] setups: one with an equal parameter count for the entire ensemble, and one with $K$ times more parameters for an ensemble of $K$ members.

For the EHR task, we reproduce and compare to the LSTM-based RNN baselines from [Dusenberry et al., 2019]: deterministic; Bayesian Embeddings (distributions over the embeddings); and Fully Bayesian (distributions over all parameters). We additionally compare against BatchEnsemble, and include reproduced results for deep ensembles.
2.4.1 ImageNet and ImageNet-C

ImageNet-C [Hendrycks and Dietterich, 2019] applies a set of 15 common visual corruptions to ImageNet [Deng et al., 2009] with varying intensity values (1-5). It was designed to benchmark the robustness to image corruptions. Table 2.1 presents results for negative log-likelihood (NLL), accuracy, and expected calibration error (ECE) on the standard ImageNet test set, as well as on ImageNet-C. We also include mean corruption error (mCE) [Hendrycks and Dietterich, 2019]. Figure 2.7 examines out-of-distribution performance in more detail by plotting the mean result across corruption types for each corruption intensity.

BatchEnsemble improves accuracy (but not NLL or ECE) over the deterministic baseline. Rank-1 BNNs, which involve non-degenerate mixture distributions over BatchEnsemble, further improve results across all metrics.

Rank-1 BNN’s results are comparable in terms of test NLL and accuracy to previous works which scaled up BNNs to ResNet-50. Zhang et al. [2019] use 9 MCMC samples and report 77.1% accuracy and 0.888 NLL; and Heek and Kalchbrenner [2019] use 30 MCMC samples and report 77.5% accuracy and 0.883 NLL. Rank-1 BNNs have a similar parameter count to deterministic ResNet-50, instead of incurring a 9-30x memory cost, and use a single MC sample from each mixture component by default. Rank-1 BNNs also do not use techniques such as tempering, which trades off uncertainty in favor of predictive performance. We predict rank-1 BNNs may outperform these methods if measured by ECE or out-of-distribution performance.

2.4.2 CIFAR-10 and CIFAR-10-C

Table 2.2 demonstrates results with respect to NLL, accuracy, and ECE on the CIFAR-10 test set, and the same three metrics on CIFAR-10-C. Figure 2.8 examines out-of-distribution performance as the skew intensity (severity of corruption) increases. Section 2.7.5.1 contains a clearer comparison.

Heek and Kalchbrenner [2019] also report results using a single sample: 74.2% accuracy, 1.08 NLL. Rank-1 BNNs outperform.
Figure 2.8: Out-of-distribution performance using CIFAR-10-C (top) and CIFAR-100-C (bottom) with WRN-28-10. We plot NLL, accuracy, and ECE for varying corruption intensities; each result is the mean performance over 10 runs and 15 corruption types. The error bars represent a fraction of the standard deviation across corruption types. Rank-1 BNNs (red) perform best across all metrics.

On CIFAR-10, both Gaussian and Cauchy rank-1 BNNs outperform similarly-sized baselines in terms of NLL, accuracy, and ECE. The improvement on NLL and ECE is more significant than that on accuracy, which highlights the improved uncertainty measurement. An even more significant improvement is observed on CIFAR-10-C: the NLL improvement from BatchEnsemble is 1.02 to 0.74; accuracy increases by 3.7%; and calibration decreases by 0.05. This, in addition to Figure 2.11 in the Appendix, is clear evidence of improved generalization and uncertainty calibration for rank-1 BNNs, even under distribution shift.

The vanilla BNN baseline underfits compared to the deterministic baseline, despite
an extensive search over hyperparameters. We suspect this is a result of the difficulty of optimization given weight variance and overregularization due to priors over all weights. Rank-1 BNNs do not face these issues and consistently outperform vanilla BNNs.

In comparison to deep ensembles [Lakshminarayanan et al., 2017], rank-1 BNNs outperform the similarly-sized ensembles on accuracy, while only underperforming deep ensembles that have 4 times the number of parameters. Rank-1 BNNs still perform better on in-distribution ECE, as well as on accuracy and NLL under distribution shift.

Rank-1 BNN’s results are similar to SWAG [Maddox et al., 2019] and Subspace Inference [Izmailov et al., 2019] despite those having a significantly stronger deterministic baseline and 5-25x parameters: SWAG gets 96.4% accuracy, 0.112 NLL, 0.009 ECE; Subspace Inference gets 96.3% accuracy, 0.108 NLL, and does not report ECE; their deterministic baseline gets 96.4% accuracy, 0.129 NLL, 0.017 ECE (vs. our 96.0%, 0.159, 0.023). They don’t report out-of-distribution performance. Rank-1 outperforms on accuracy and underperforms on NLL.

2.4.3 CIFAR-100 and CIFAR-100-C

Table 2.3 contains NLL, accuracy, and ECE on both CIFAR-100 and CIFAR-100-C. Rank-1 BNNs with mixture of Cauchy priors and variational posteriors outperform BatchEnsemble and similarly-sized deep ensembles by a significant margin across all metrics. To the best of our knowledge, this is the first convincing empirical success of Cauchy priors in BNNs, as it significantly improves on predictive performance, robustness, and uncertainty calibration, as observed in Figure 2.8 and Section 2.7.5.2. On the other hand, the Gaussian rank-1 BNNs have a slightly worse accuracy than BatchEnsemble, but outperform all baselines on NLL and ECE while generalizing better on CIFAR-100-C.

This is an exciting result for heavy-tailed priors in Bayesian deep learning. It has long been conjectured that such priors can be more robust to out-of-distribution data while inducing sparsity [Louizos and Welling, 2017] at the expense of accuracy. However, in both experiments summarized in Table 2.3 and Table 2.2 we can see significant improvements,
without a compromise, on modern Wide ResNet architectures.

Rank-1 BNNs also outperform deep ensembles of WRN-28-10 models on uncertainty calibration and robustness while having 4 times fewer parameters. Rank-1 BNNs also significantly close the gap between BatchEnsemble and deep ensembles on in-distribution accuracy. Holding the number of parameters constant, rank-1 BNNs outperform deep ensembles by a significant margin across all metrics. Conclusions compared to SWAG and Subspace Inference are consistent with CIFAR-10's.

2.4.4 MIMIC-III Mortality Prediction From EHRs

Extending beyond image classification tasks, we also show results using rank-1 sequential models. Following Dusenberry et al. [2019], we experiment with RNN models for predicting medical outcomes for patients given their de-identified electronic medical records. More specifically, we replicate their setup for the MIMIC-III [Johnson et al., 2016] binary mortality task. In our case, we replace the existing variational LSTM Schmidhuber and Hochreiter [1997] and affine layers with their rank-1 counterparts, and keep the variational embedding vectors. We use global mixture distributions for the rank-1 layers, and the resulting model is a mixture model with shared stochastic embeddings.

Table 2.4 shows results for NLL, AUC-PR, and ECE on the validation and test sets. We evaluate on 25 Monte Carlo samples at evaluation time versus 200 samples in the previous work, and report mean results over 25 random seeds. Our rank-1 Bayesian RNN outperforms all other baselines, including the fully-Bayesian RNN, across all metrics. These results demonstrate that our rank-1 BNN methodology can be easily adapted to different types of tasks, different data modalities, and different architectures.

While Gaussian rank-1 RNNs outperform all baselines, the Cauchy variant does not perform as well in terms of AUC-PR, while still improving on NLL and ECE. This result, in addition to that of the ImageNet experiments, indicates the need for further inspection of heavy-tailed distributions in deep or recurrent architectures. In fact, ResNet-50 is a deeper architecture than WRN-28-10, while MIMIC-III RNNs can be unrolled over hundreds of
time steps. Given that heavy-tailed posteriors lead to more frequent samples further away from the mode, we hypothesize that instability in the training dynamics is the main reason for underfitting.

2.5 Related Work

**Hierarchical priors and variational approximations.** Rank-1 factors can be interpreted as scale factors that are shared across weight elements. Section 2.3.2 details this and differences from other hierarchical priors [Louizos and Welling, 2017, Ghosh and Doshi-Velez, 2017]. The outer product of rank-1 vectors resembles matrixvariate Gaussians [Louizos and Welling, 2016]: the major difference is that rank-1 priors are uncertain about the scale factors shared across rows and columns rather than fixing a covariance. Rank-1 BNNs’ variational approximation can be seen as a form of hierarchical variational model [Ranganath et al., 2016] similar to multiplicative normalizing flows, which posit an auxiliary distribution on the hidden units [Louizos and Welling, 2017]. In terms of the specific distribution, instead of normalizing flows we focus on mixtures, a well-known approach for expressive variational inference [Jaakkola and Jordan, 1998, Lawrence, 2001]. Building on these classic works, we examine mixtures in ways that bridge algorithmic differences from deep ensembles and using modern model architectures.

**Variance reduction techniques for variational BNNs.** Sampling with rank-1 factors (Equation 2.1) is closely related to Gaussian local reparameterization [Kingma et al., 2015, Molchanov et al., 2017], where noise is reparameterized to act on the hidden units to enable weight sampling per-example, providing significant variance reduction over naively sampling a single set of weights and sharing it across the minibatch. Unlike Gaussian local reparameterization, rank-1 factors are not limited to feedforward layers and location-scale distributions: it is exact for convolutions and recurrence and for arbitrary distributions. This is similar to “correlated weight noise,” which [Kingma et al., 2015] also studies and finds performs better than being fully Bayesian. Enabling weight sampling to these settings
otherwise necessitates techniques such as Flipout \cite{Wen2018}.

**Parameter-efficient ensembles.** Monte Carlo Dropout is arguably the most popular efficient ensembling technique, based on Bernoulli noise that deactivates hidden units during training and testing \cite{Srivastava2014,Gal2016}. More recently, BatchEnsemble has emerged as an effective technique that is algorithmically similar to deep ensembles, but on rank-1 factors \cite{Wen2020}. We compare to both MC-dropout and BatchEnsemble as our primary baselines. If a single set of weights is sufficient (as opposed to a distribution for model uncertainty), there are also empirically successful averaging techniques such as Polyak-Ruppert \cite{Ruppert1988}, checkpointing, and stochastic weight averaging \cite{Izmailov2018}.

**Scaling up BNNs.** We are aware of three previous works scaling up BNNs to ImageNet. Variational Online Gauss Newton reports results on ResNet-18, outperforming a deterministic baseline in terms of NLL but not accuracy, and using 2x the number of neural network weights \cite{Osawa2019}. Cyclical SGMCMC \cite{Zhang2019} and adaptive thermostat MC \cite{Heek2019} report results on ResNet-50, outperforming a deterministic baseline in terms of NLL and accuracy, using at least 9 samples (i.e., 9x cost). In our experiments, we use ResNet-50 with comparable parameter count for all methods; we examine not only NLL and accuracy, but also uncertainties via calibration and out-of-distribution evaluation; and rank-1 BNNs do not apply strategies such as fixed KL scaling or tempering, which complicate the Bayesian interpretation.

Like rank-1 BNNs, \cite{Izmailov2019} perform Bayesian inference in a low-dimensional space. Instead of end-to-end training like rank-1 BNNs, it uses two stages where one first performs stochastic weight averaging and then applies PCA to form a projection matrix from the set of weights to, e.g., 5 dimensions, over which one can then perform inference. This projection matrix requires 5x the number of weights.
2.6 Discussion

We described rank-1 BNNs, which posit a prior distribution over a rank-1 factor of each weight matrix and are trained with mixture variational distributions. Rank-1 BNNs are parameter-efficient and scalable as Bayesian inference is done over a much smaller dimensionality. Across ImageNet, CIFAR-10, CIFAR-100, and MIMIC-III, rank-1 BNNs achieve the best results on predictive and uncertainty metrics across in- and out-of-distribution data.

For future work, we’d like to push further on our results by scaling to larger ImageNet models to achieve state-of-the-art in test accuracy alongside other metrics. Although we focus on variational inference in this chapter, applying this parameterization in MCMC is a promising parameter-efficient strategy for scalable BNNs. As an alternative to using mixtures trained with the average per-component log-likelihood, one can use multiple independent chains over the rank-1 factors. Another direction for future work is the straightforward extension to higher rank factors. However, prior work [Swiatkowski et al., 2019, Izmailov et al., 2019] has demonstrated diminishing returns that practically stop at ranks 3 or 5.

One surprising finding in our experimental results is that heavy-tailed priors, on a low-dimensional subspace, can significantly improve robustness and uncertainty calibration while maintaining or improving accuracy. This is likely due to the heavier tails allowing for more points in loss landscape valleys to be covered, whereas a mixture of lighter tails could place multiple modes that are nearly identical. However, with deeper or recurrent architectures, samples from the heavy-tailed posteriors seem to affect the stability of the training dynamics, leading to slightly worse predictive performance. One additional direction for future work is to explore ways to stabilize automatic differentiation through such approximate posteriors or to pair heavy-tailed priors with sub-Gaussian posteriors.
Table 2.1: Results for ResNet-50 on ImageNet: negative log-likelihood (lower is better), accuracy (higher is better), and expected calibration error (lower is better). cNLL, cA, and cECE are NLL, accuracy, and ECE averaged over ImageNet-C’s corruption types and intensities. mCE is mean corruption error. Results are averaged over 10 seeds, and over 1 weight sample (per mixture component, per seed) for the Gaussian rank-1 BNN, and 4 samples for Cauchy. We include results for Zhang et al. [2019] and Heek and Kalchbrenner [2019]. Rank-1 BNNs consistently outperform baselines across all metrics.

<table>
<thead>
<tr>
<th>Method</th>
<th>NLL(↓)</th>
<th>Accuracy(↑)</th>
<th>ECE(↓)</th>
<th>cNLL / cA / cECE</th>
<th>mCE(↓)</th>
<th># Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>0.943</td>
<td>76.1</td>
<td>0.0392</td>
<td>3.20 / 40.5 / 0.105</td>
<td>75.34</td>
<td>25.6M</td>
</tr>
<tr>
<td>BatchEnsemble</td>
<td>0.951</td>
<td>76.5</td>
<td>0.0532</td>
<td>3.23 / 41.4 / 0.120</td>
<td>74.14</td>
<td>25.8M</td>
</tr>
<tr>
<td><strong>Rank-1 BNN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gaussian</td>
<td>0.886</td>
<td><strong>77.3</strong></td>
<td>0.0166</td>
<td><strong>2.95 / 42.9 / 0.054</strong></td>
<td><strong>72.12</strong></td>
<td>26.0M</td>
</tr>
<tr>
<td>Cauchy (4 samples)</td>
<td>0.897</td>
<td>77.2</td>
<td>0.0192</td>
<td>2.98 / 42.5 / 0.059</td>
<td>72.66</td>
<td>26.0M</td>
</tr>
<tr>
<td>Deep Ensembles</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ResNet-50</td>
<td><strong>0.877</strong></td>
<td><strong>77.5</strong></td>
<td>0.0305</td>
<td>2.98 / 42.1 / 0.050</td>
<td>73.25</td>
<td>146.7M</td>
</tr>
<tr>
<td>MCMC BNN(^1)</td>
<td>0.888</td>
<td>77.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>230.4</td>
</tr>
<tr>
<td>MCMC BNN(^2)</td>
<td>0.883</td>
<td><strong>77.5</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>768M</td>
</tr>
</tbody>
</table>
Table 2.2: Results for Wide ResNet-28-10 on CIFAR-10, averaged over 10 seeds. Gaussian rank-1 BNNs with 1 sample reach top accuracy with BatchEnsemble and otherwise outperform baselines with comparable parameter count across all metrics.

<table>
<thead>
<tr>
<th>Method</th>
<th>NLL(↓)</th>
<th>Accuracy(↑)</th>
<th>ECE(↓)</th>
<th>cNLL / cA / cECE</th>
<th># Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>0.159</td>
<td>96.0</td>
<td>0.023</td>
<td>1.05 / 76.1 / 0.153</td>
<td>36.5M</td>
</tr>
<tr>
<td>BatchEnsemble</td>
<td>0.143</td>
<td>96.2</td>
<td>0.020</td>
<td>1.02 / 77.5 / 0.129</td>
<td>36.6M</td>
</tr>
<tr>
<td>MC Dropout</td>
<td>0.160</td>
<td>95.9</td>
<td>0.024</td>
<td>1.27 / 68.8 / 0.166</td>
<td>36.5M</td>
</tr>
<tr>
<td>MFVI BNN</td>
<td>0.214</td>
<td>94.7</td>
<td>0.029</td>
<td>1.46 / 71.3 / 0.181</td>
<td>73M</td>
</tr>
<tr>
<td><strong>Rank-1 BNN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gaussian</td>
<td>0.128</td>
<td>96.3</td>
<td>0.008</td>
<td>0.84 / 76.7 / 0.080</td>
<td>36.6M</td>
</tr>
<tr>
<td>Cauchy(4 samples)</td>
<td><strong>0.120</strong></td>
<td><strong>96.5</strong></td>
<td>0.009</td>
<td><strong>0.74 / 80.5 / 0.090</strong></td>
<td>36.6M</td>
</tr>
<tr>
<td><strong>Deep Ensembles</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRN-28-5</td>
<td>0.115</td>
<td>96.3</td>
<td>0.008</td>
<td>0.84 / 77.2 / 0.089</td>
<td>36.68M</td>
</tr>
<tr>
<td>WRN-28-10</td>
<td><strong>0.114</strong></td>
<td><strong>96.6</strong></td>
<td>0.010</td>
<td>0.81 / 77.9 / 0.087</td>
<td>146M</td>
</tr>
</tbody>
</table>
Table 2.3: Results for Wide ResNet-28-10 on CIFAR-100, averaged over 10 seeds. Gaussian rank-1 BNNs with 1 sample reach slightly worse accuracy than BatchEnsemble and otherwise outperform baselines with comparable parameter count.

<table>
<thead>
<tr>
<th>Method</th>
<th>NLL(↓)</th>
<th>Accuracy(↑)</th>
<th>ECE(↓)</th>
<th>cNLL / cA / cECE</th>
<th># Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>0.875</td>
<td>79.8</td>
<td>0.085</td>
<td>2.70 / 51.3 / 0.239</td>
<td>36.5M</td>
</tr>
<tr>
<td>BatchEnsemble</td>
<td>0.734</td>
<td>81.5</td>
<td>0.033</td>
<td>2.49 / 54.1 / 0.191</td>
<td>36.6M</td>
</tr>
<tr>
<td>MC Dropout</td>
<td>0.830</td>
<td>79.6</td>
<td>0.050</td>
<td>2.33 / 51.5 / 0.148</td>
<td>36.5M</td>
</tr>
<tr>
<td>MFVI BNN</td>
<td>1.030</td>
<td>77.3</td>
<td>0.111</td>
<td>3.48 / 48.0 / 0.299</td>
<td>73M</td>
</tr>
<tr>
<td><strong>Rank-1 BNN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gaussian</td>
<td>0.692</td>
<td>81.3</td>
<td>0.018</td>
<td>2.24 / 53.8 / 0.117</td>
<td>36.6M</td>
</tr>
<tr>
<td>Cauchy (4 samples)</td>
<td><strong>0.689</strong></td>
<td><strong>82.4</strong></td>
<td><strong>0.012</strong></td>
<td><strong>2.04 / 57.8 / 0.142</strong></td>
<td>36.6M</td>
</tr>
<tr>
<td><strong>Deep Ensembles</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRN-28-5</td>
<td>0.694</td>
<td>81.5</td>
<td>0.017</td>
<td>2.19 / 53.7 / 0.111</td>
<td>36.68M</td>
</tr>
<tr>
<td>WRN-28-10</td>
<td><strong>0.666</strong></td>
<td><strong>82.7</strong></td>
<td>0.021</td>
<td>2.27 / 54.1 / 0.138</td>
<td>146M</td>
</tr>
</tbody>
</table>
Table 2.4: Results for RNNs on the MIMIC-III EHR mortality task, averaged over 25 seeds, and over 25 weight samples per seed for all Bayesian models. Rank-1 Bayesian RNNs achieve the best metric performance compared to baselines.

<table>
<thead>
<tr>
<th>Method</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NLL(↓)</td>
<td>AUC-PR(↑)</td>
</tr>
<tr>
<td>Deterministic</td>
<td>0.211</td>
<td>0.446</td>
</tr>
<tr>
<td>BatchEnsemble</td>
<td>0.215</td>
<td>0.447</td>
</tr>
<tr>
<td>Bayesian Embeddings</td>
<td>0.213</td>
<td>0.449</td>
</tr>
<tr>
<td>Fully-Bayesian</td>
<td>0.220</td>
<td>0.424</td>
</tr>
<tr>
<td><strong>Rank-1 BNN</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gaussian</td>
<td>0.209</td>
<td><strong>0.451</strong></td>
</tr>
<tr>
<td>Cauchy</td>
<td><strong>0.207</strong></td>
<td>0.446</td>
</tr>
<tr>
<td>Deep Ensembles</td>
<td>Deterministic</td>
<td>0.202</td>
</tr>
</tbody>
</table>
2.7 Appendix

2.7.1 Variance Structure of the Rank-1 Perturbations

We hereby study how variance in the score function is captured by the full-rank weight matrix \( W \) parameterization versus the rank-1 \( W_\star \circ rs^T \) parameterization. We first note that around a local optimum \( W_\star \), the score function \( \sum_{n=1}^N f(x_n|W) \) can be approximated using the Hessian \( \sum_{n=1}^N \nabla^2_W f(x_n|W) \):

\[
\sum_{n=1}^N (f(x_n|W) - f(x_n|W_\star)) \approx \frac{1}{2} \sum_{n=1}^N \sum_{h=1}^H \langle W^{(h)} - W^{(h)}_\star, \nabla^2_{W^{(h)}} f(x_n|W_\star) (W^{(h)} - W^{(h)}_\star) \rangle_F.
\]

We can therefore characterize variance around a local optimum via expected fluctuation in the score function, \( \sum_{n=1}^N E_W f(x_n|W) - f(x_n|W_\star) \). We compare here the effect of the two parameterizations: \( \sum_{n=1}^N E_W [f(x_n|W) - f(x_n|W_\star)] \) versus \( \sum_{n=1}^N E_{W_\star} [f(x_n|W_\star \circ rs^T) - f(x_n|W_\star)] \).

In what follows, we take fully connected networks to demonstrate that the rank-1 parameterization can have the same local variance structure as the full-rank parameterization.

We first formulate the fully connected neural network in the following recursive relation. For fully connected network of width \( M \) and depth \( H \), the score function \( f(x|W) \) can be recursively defined as:

\[
x^{(0)} = x, \\
x^{(h)} = \sqrt{\frac{c_\sigma}{M}} \sigma \left( W^{(h)} x^{(h-1)} \right), \quad 1 \leq h \leq H \\
f(x|W) = a^T x^{(H)}.
\]

**Theorem 1** (Formal). For a fully connected network of width \( M \) and depth \( H \) learned over \( N \) data points, let \( W_\star \) denote local minimum of \( \sum_{n=1}^N f(x_n|W) \) in the space of weight matrices. Consider both full-rank perturbation \( (W - W_\star) \) and rank-1 perturbation \( (W_\star \circ rs^T - W_\star) \). Assume that the full-rank perturbation has the multiplicative covariance
structure that

\[ \mathbb{E}_{W^h} \left[ (W^{(h)} - W_*^{(h)})_{i,j} (W^{(h)} - W_*^{(h)})_{k,l} \right] = W_*^{(h)} \Sigma_{j,k} W_*^{(h)}_{j,k}, \tag{2.4} \]

for some symmetric positive semi-definite matrix \( \Sigma \). Let \( s_*^{(h)} \) denote a column vector of ones. Then if the rank-1 perturbation has covariance

\[ \mathbb{E}_{s^{(h)}} \left[ \left( (s^{(h)} - s_*^{(h)}) (s^{(h)} - s_*^{(h)}) \right)^T \right] = \Sigma \]

\[ \sum_{n=1}^{N} \sum_{h=1}^{H} \mathbb{E}_{W^h} \left[ \left\langle (W^{(h)} - W_*^{(h)}), \nabla^2_{W^{(h)}} f(x_n | W_*^{(h)}) \right\rangle \right] \]

\[ \sum_{n=1}^{N} \sum_{h=1}^{H} \mathbb{E}_{s^{(h)}} \left[ \left\langle (s^{(h)} - s_*^{(h)}), \nabla^2_{s^{(h)}} f(x_n | W) \right\rangle \right] . \tag{2.5} \]

Theorem 1 demonstrates a correspondence between the covariance structure in the perturbation of \( W \) and that of \( s \). Since \( \Sigma \) can be any symmetric positive semi-definite matrix, we have demonstrated here that our rank-1 parameterization can efficiently encode a wide range of fluctuations in \( W \). In particular, it is especially suited for multiplicative noise as advertised. If the covariance of \( (W^{(h)} - W_*^{(h)}) \) is proportional to \( W_* \otimes W_*^T \) itself, then we can simply take the covariance of \( (s - s_*) \) to be identity.

We devote the rest of this section to prove Theorem 1.

Proof of Theorem 1. We first state the following lemma for the fluctuations of the score function \( f \) in \( W \) and \( s \) spaces.

Lemma 1. For a fully connected network of width \( M \) and depth \( H \) learned over \( N \) data points, let \( W_* \) denote local minimum of \( \sum_{n=1}^{N} f(x_n | W) \) in the space of weight matrices.
Then the local fluctuations of the score function in the space of the weight matrix $W$ is:

$$
\mathbb{E}_{W^{(h)}} \left[ \left\langle W^{(h)} - W^{(h)}_*, \nabla^2 f(x_n | W) \left( W^{(h)} - W^{(h)}_* \right) \right\rangle_F \right] = \left( \frac{c_\sigma}{M} \right) \frac{H-h+1}{2} \text{trace} \left( \mathbb{E}_{W^{(h)}} \left[ \left( W^{(h)} - W^{(h)}_* \right) x_n^{(h-1)} x_n^{(h-1)} \left( W^{(h)} - W^{(h)}_* \right)^T \right] \right)
\cdot \text{diag} \left( \prod_{h=h+1}^H \text{diag} \left( \sigma' \left( W^{(h)} x_n^{(h-1)} \right) \right) \cdot \text{diag} \left( \sigma'' \left( W^{(h)} x_n^{(h-1)} \right) \right) \right).
$$

(2.6)

and in the space of the low rank representation $s$,

$$
\mathbb{E}_{s^{(h)}} \left[ \left\langle \left( s^{(h)} - s^{(h)}_* \right) , \nabla^2_{s^{(h)}} f(x_n | W) \left( s^{(h)} - s^{(h)}_* \right) \right\rangle \right] = \left( \frac{c_\sigma}{M} \right) \frac{H-h+1}{2} \text{trace} \left( \mathbb{E}_{s^{(h)}} \left[ \left( s^{(h)} - s^{(h)}_* \right) x_n^{(h-1)} x_n^{(h-1)} \left( s^{(h)} - s^{(h)}_* \right)^T \right] \right)
\cdot \text{diag} \left( \prod_{h=h+1}^H \text{diag} \left( \sigma' \left( W^{(h)} x_n^{(h-1)} \right) \right) \cdot \text{diag} \left( \sigma'' \left( W^{(h)} x_n^{(h-1)} \right) \right) \right).
$$

(2.7)

For perturbations $(W - W_*)$ with a multiplicative structure, we can write that

$$
\mathbb{E}_{W^{(h)}} \left[ \left( W^{(h)} - W^{(h)}_* \right)_{i,j} \left( W^{(h)} - W^{(h)}_* \right)_{k,l} \right] = W_{*,i,j} \Sigma_{j,k,l} W_{*,k,l},
$$

for some matrix $\Sigma$.

Note that in the simplest case where $\Sigma = \epsilon \cdot I$, this corresponds to the covariance of $(W - W_*)$ being a decomposable tensor:

$$
\mathbb{E}_{W^{(h)}} \left[ \left( W^{(h)} - W^{(h)}_* \right) \left( W^{(h)} - W^{(h)}_* \right)^T \right] = \epsilon \cdot W_* \otimes W_*^T.
$$

In this multiplicative perturbation case, we can show that if

$$
\mathbb{E}_{s^{(h)}} \left[ \left( s^{(h)} - s_* \right) \left( s^{(h)} - s_* \right)^T \right] = \Sigma
$$

43
then

\[
\mathbb{E}_{W^{(h)}} \left[ (W^{(h)} - W_*^{(h)}) x_n^{(h-1)} (x_n^{(h-1)})^T (W^{(h)} - W_*^{(h)})^T \right]
\]

\[
= W_*^{(h)} \text{diag} \left( x_n^{(h-1)} \right) \Sigma \text{diag} \left( x_n^{(h-1)} \right) (W_*^{(h)})^T
\]

\[
= W_*^{(h)} \text{diag} \left( x_n^{(h-1)} \right) \mathbb{E}_{s^{(h)}} \left[ (s^{(h)} - s_*) (s^{(h)} - s_*)^T \right] \text{diag} \left( x_n^{(h-1)} \right) (W_*^{(h)})^T
\]

\[
= W_*^{(h)} \mathbb{E}_{s^{(h)}} \left[ \text{diag} \left( s^{(h)} - s_* \right) (x_n^{(h-1)}) (x_n^{(h-1)})^T \text{diag} \left( s^{(h)} - s_* \right) \right] (W_*^{(h)})^T.
\]

Plugging this result into equations 2.6 and 2.7, we know that for any \( n \) and \( h \),

\[
\mathbb{E}_{W^{(h)}} \left[ \langle W^{(h)} - W_*^{(h)}, \nabla^2_{W^{(h)}} f(x_n|W) (W^{(h)} - W_*^{(h)}) \rangle_F \right]
\]

\[
= \mathbb{E}_{s^{(h)}} \left[ \langle (s^{(h)} - s_*), \nabla^2_{s^{(h)}} f(x_n|W) (s^{(h)} - s_*) \rangle \right].
\]

Therefore,

\[
\sum_{n=1}^N \sum_{h=1}^H \mathbb{E}_{W^{(h)}} \left[ \langle W^{(h)} - W_*^{(h)}, \nabla^2_{W^{(h)}} f(x_n|W) (W^{(h)} - W_*^{(h)}) \rangle_F \right]
\]

\[
= \sum_{n=1}^N \sum_{h=1}^H \mathbb{E}_{s^{(h)}} \left[ \langle (s^{(h)} - s_*), \nabla^2_{s^{(h)}} f(x_n|W) (s^{(h)} - s_*) \rangle \right]. \quad (2.8)
\]

**Proof of Lemma 1** We first analyze the local geometric structures of the score function in the space of the full-rank weight matrix \( W \) and the low rank vector \( s \), respectively. We then leverage this Hessian information to finish our proof.

**Local Geometry of the score function** \( f(x_n|W_* \circ rs^T) \): We can first compute the gradient of weight \( W \) at \( h \)-th layer for the predictive score function \( f \) of an \( H \) layer.
fully connected neural network taken at data point $x_n$:

\[
\nabla_{w^{(h)}} f(x_n | W) = \frac{\partial x_n^{(h)}}{\partial W^{(h)}} \nabla x_n^{(h)} f(x | W)
\]

\[
= \sqrt{\frac{C_\sigma}{M}} \text{diag} \left( \sigma' \left( W^{(h)} x_n^{(h-1)} \right) \right) \cdot \frac{\partial \nabla x_n^{(h)}}{\partial W^{(h)}} f(x_n | W) \cdot (x_n^{(h-1)})^T
\]

\[
= \left( \frac{C_\sigma}{M} \right)^{\frac{H-h+1}{2}} \text{diag} \left( \sigma' \left( W^{(h)} x_n^{(h-1)} \right) \right) \cdot \prod_{h=h+1}^H \text{diag} \left( \sigma' \left( W^{(h)} x_n^{(h-1)} \right) \right) \cdot W^{h} a \cdot (x_n^{(h-1)})^T.
\]

If we instead take the gradient over the vector $s$, we obtain that

\[
\nabla_{s^{(h)}} f(x_n | W_* \circ rs^T) = \left[ \frac{\partial}{\partial W^{(h)}} f(x_n | W), \frac{\partial W^{(h)}}{\partial s^{(h)}} \right]_F
\]

\[
= \left( \frac{\partial}{\partial W^{(h)}} f(x_n | W) \right)^T \cdot \left( W_*^{(h)} \right)^T r^{(h)}
\]

\[
= \left( \frac{C_\sigma}{M} \right)^{\frac{H-h+1}{2}} \left( W_*^{(h)} \right)^T \circ x_n^{(h-1)} \cdot (v_n^{(h)})^T r^{(h)}
\]

\[
= \left( \frac{C_\sigma}{M} \right)^{\frac{H-h+1}{2}} \text{diag} \left( x_n^{(h-1)} \right) \left( W_*^{(h)} \right)^T \text{diag} \left( r^{(h)} \right) v_n^{(h)}.
\]

We can further analyze the Hessian of $f$:

\[
\nabla^2_{w^{(h)}} f(x_n | W) = \left( \frac{C_\sigma}{M} \right)^{\frac{H-h+1}{2}} \text{diag} \left( \prod_{h=h+1}^H \text{diag} \left( \sigma' \left( W^{(h)} x_n^{(h-1)} \right) \right) W^{h} a \right)
\]

\[
\cdot \text{diag} \left( \sigma'' \left( W^{(h)} x_n^{(h-1)} \right) \right) \circ x_n^{(h-1)} (x_n^{(h-1)})^T.
\]

(2.9)
Whereas for $s$,

$$
\nabla_{s_{(h)}}^2 f(x_n|W_* \circ rs^T) = \left( \frac{c_\sigma}{M} \right)^{\frac{h-h+1}{2}} \text{diag} \left( x_{n_{(h-1)}} \right) \left( W_{*_{(h)}} \right)^T \text{diag} \left( r_{(h)} \right) \\
\cdot \text{diag} \left( \prod_{b=h+1}^H \text{diag} \left( \sigma' \left( W_{(h)}x_{(b-1)} \right) \right) \cdot W_{(h)a} \right) \\
\cdot \text{diag} \left( \sigma'' \left( W_{(h)}x_{(h-1)} \right) \right) \text{diag} \left( r_{(h)} \right) W_{*_{(h)}} \text{diag} \left( x_{n_{(h-1)}} \right),
$$

(2.10)

Variance Structures in the Score Function: Applying the results in equations 2.9 and 2.10 we obtain that

$$
\mathbb{E}_{W_{(h)}} \left[ \left\langle W_{(h)} - W_{*_{(h)}}, \nabla_{W_{(h)}}^2 f(x_n|W) \left( W_{(h)} - W_{*_{(h)}} \right) \right\rangle_F \right] \\
= \left( \frac{c_\sigma}{M} \right)^{\frac{h-h+1}{2}} \mathbb{E}_{W_{(h)}} \left[ \left( x_{n_{(h-1)}} \right)^T \left( W_{(h)} - W_{*_{(h)}} \right)^T \right. \\
\text{diag} \left( \prod_{b=h+1}^H \text{diag} \left( \sigma' \left( W_{(h)}x_{(b-1)} \right) \right) W_{(h)a} \right) \text{diag} \left( \sigma'' \left( W_{(h)}x_{(h-1)} \right) \right) \left( W_{(h)} - W_{*_{(h)}} \right) x_{n_{(h-1)}} \left] \\
= \left( \frac{c_\sigma}{M} \right)^{\frac{h-h+1}{2}} \text{trace} \left( \mathbb{E}_{W_{(h)}} \left[ \left( W_{(h)} - W_{*_{(h)}} \right) x_{n_{(h-1)}}^T \left( W_{(h)} - W_{*_{(h)}} \right)^T \right] \\
\text{diag} \left( \prod_{b=h+1}^H \text{diag} \left( \sigma' \left( W_{(h)}x_{(b-1)} \right) \right) W_{(h)a} \right) \text{diag} \left( \sigma'' \left( W_{(h)}x_{(h-1)} \right) \right) \right).}

46
and that

\[
\mathbb{E}_{s^{(h)}} \left[ \left( \left( s^{(h)} - s^{(h)}_* \right) \cdot \nabla^2_{s^{(h)}} f(x_n | W) \left( s^{(h)} - s^{(h)}_* \right) \right) \right] \\
= \left( \frac{c_{s}}{M} \right) \frac{H - h + 1}{2} \mathbb{E} \left( W^{(h)}_* \left( x^{(h-1), n} \circ \left( s^{(h)} - s^{(h)}_* \right) \right) \cdot r^{(h)}_* \right)^T \\
\cdot \text{diag} \left( \prod_{b=h+1}^{H} \text{diag} \left( \sigma' \left( W^{(h)} x^{(b-1)} \right) \cdot W^h a \right) \right) \\
\cdot \text{diag} \left( \sigma'' \left( W^{(h)} x^{(h-1)} \right) \right) \cdot W^{(h)}_* \left( x^{(h-1), n} \circ \left( s^{(h)} - s^{(h)}_* \right) \right) \cdot r^{(h)}_* \\
= \left( \frac{c_{s}}{M} \right) \frac{H - h + 1}{2} \text{trace} \left( W^{(h)}_* \mathbb{E} \text{diag} \left( s^{(h)} - s^{(h)}_* \right) \left( x^{(h-1), n} \right)^T \text{diag} \left( s^{(h)} - s^{(h)}_* \right) \\
\cdot \left( W^{(h)}_* \right)^T \text{diag} \left( \prod_{b=h+1}^{H} \text{diag} \left( \sigma' \left( W^{(h)} x^{(b-1)} \right) \right) \cdot W^h a \right) \cdot \text{diag} \left( \sigma'' \left( W^{(h)} x^{(h-1)} \right) \right) \right).
\]

2.7.2 Additional Experimental Details and Hyperparameters

We experiment with both mixture of Gaussian and mixture of Cauchy priors (and variational posteriors) for the rank-1 factors. All reported results are averages over 10 runs for the image classification tasks and 25 runs for the EHR task. For Gaussian distributions in the image tasks, we achieve superior metric performance using only 1 Monte Carlo sample for each of 4 components to estimate the integral in Equation 2.2 for both training and evaluation, unlike much of the BNN literature, and we show further gains from using larger numbers of samples (4 and 25; see section 2.7.3.3). For Cauchy distributions on those image tasks, we use 1 Monte Carlo sample for each of 4 components for training, and use 4 samples per component during evaluation. For the EHR task, we also use only 1 sample during training, but use 25 samples during evaluation (down from 200 samples for the Bayesian models in Dusenberry et al. [2019]). See Section 2.7.2 for details on hyperparameters. Our code uses TensorFlow and Edward2’s Bayesian Layers [Tran et al., 2018]; all experiments

47
are available at https://github.com/google/edward2.

For rank-1 BNNs, there are three hyperparameters in addition to the deterministic baseline’s: the number of mixture components (we fix it at 4); prior standard deviation (we vary among 0.05, 0.1, and 1); and the mean initialization for variational posteriors (either random sign flips with probability random_sign_init or a random normal with mean 1 and standard deviation random_sign_init). All hyperparameters for our rank-1 BNNs can be found in Tables 2.5, 2.6, and 2.7.

Following Section 2.3’s ablations, we always (with one exception) use a prior with mean at 1, the average per-component log-likelihood, and initialize variational posterior standard deviations under the dropout parameterization as $10^{-3}$ for Gaussian priors and 10. The one exception is the Cauchy rank-1 Bayesian RNN on MIMIC-III, where we use a prior with mean 0.5.

Rank-1 BNNs apply rank-1 factors to all layers in the network except for normalization layers and the embedding layers in the MIMIC-III models. We are not Bayesian about the biases, but we do not find it made a difference.

We use a linear KL annealing schedule for 2/3 of the total number of training epochs (we also tried 1/3 and 1/4 and did not find the setting sensitive). Rank-1 BNNs use 250 training epochs for CIFAR-10/100 (deterministic uses 200); 135 epochs for ImageNet (deterministic uses 90); and 12000 to 25000 steps for MIMIC-III.

All methods use the largest batch size before we see a generalization gap in any method. For ImageNet, this is 32 TPUv2 cores with a per-core batch size of 128; for CIFAR-10/100, this is 8 TPUv2 cores with a per-core batch size of 64; for MIMIC-III this differs depending on the architecture. All CIFAR-10/100 and ImageNet methods use SGD with momentum with the same step-wise learning rate decay schedule, built on the deterministic baseline. For MIMIC-III, we use Adam Kingma and Ba [2014a] with no decay schedule.

For MIMIC-III, all hyperparameters for the baselines match those of Dusenberry et al. [2019], except we used a batch size of 128 for the deterministic and Bayesian Embeddings models. Since Dusenberry et al. [2019] tuned each model separately, including the architec-
ture sizes, we also tuned our rank-1 Bayesian RNN architecture sizes (for performance and memory constraints). Of note, the Gaussian rank-1 RNN has a slightly smaller architecture (rnn_dim=512 vs. 1024).
Table 2.5: Hyperparameter values for Rank-1 BNNs with Wide ResNet-28-10 on CIFAR-10 and CIFAR-100. Alpha and Gamma refer to the $r$ and $s$ vectors in the main text. The initializer determines the form of the variational posterior whereas the regularizer dictates the choice of priors. Note that all priors and approximate posteriors are mixtures.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>ensemble_size</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>base_learning_rate</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>prior_mean</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>per_core_batch_size</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>num_cores</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>lr_decay_ratio</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>train_epochs</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>lr_decay_epochs</td>
<td>[80, 160, 180]</td>
<td></td>
</tr>
<tr>
<td>kl_annealing_epochs</td>
<td>200</td>
<td></td>
</tr>
</tbody>
</table>

| l2                    | 0.0001   | 0.0003    |

<table>
<thead>
<tr>
<th>Method</th>
<th>Normal</th>
<th>Cauchy</th>
<th>Normal</th>
<th>Cauchy</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha_initializer</td>
<td>trainable_normal</td>
<td>trainable_cauchy</td>
<td>trainable_normal</td>
<td>trainable_cauchy</td>
</tr>
<tr>
<td>alpha_regularizer</td>
<td>normal_kl_divergence</td>
<td>cauchy_kl_divergence</td>
<td>normal_kl_divergence</td>
<td>cauchy_kl_divergence</td>
</tr>
<tr>
<td>gamma_initializer</td>
<td>trainable_normal</td>
<td>trainable_cauchy</td>
<td>trainable_normal</td>
<td>trainable_cauchy</td>
</tr>
<tr>
<td>gamma_regularizer</td>
<td>normal_kl_divergence</td>
<td>cauchy_kl_divergence</td>
<td>normal_kl_divergence</td>
<td>cauchy_kl_divergence</td>
</tr>
<tr>
<td>prior_stddev</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.01</td>
</tr>
<tr>
<td>dropout_rate (init)</td>
<td>0.001</td>
<td>$10^{-6}$</td>
<td>0.001</td>
<td>$10^{-6}$</td>
</tr>
<tr>
<td>random_sign_init</td>
<td>−0.5</td>
<td>−0.5</td>
<td>−1.0</td>
<td>−1.0</td>
</tr>
</tbody>
</table>
Table 2.6: Hyperparameter values for Rank-1 BNNs with ResNet-50 on ImageNet.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ImageNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>ensemble_size</td>
<td>4</td>
</tr>
<tr>
<td>base_learning_rate</td>
<td>0.1</td>
</tr>
<tr>
<td>prior_mean</td>
<td>1.0</td>
</tr>
<tr>
<td>per_core_batch_size</td>
<td>128</td>
</tr>
<tr>
<td>num_cores</td>
<td>32</td>
</tr>
<tr>
<td>lr_decay_ratio</td>
<td>0.1</td>
</tr>
<tr>
<td>train_epochs</td>
<td>135</td>
</tr>
<tr>
<td>lr_decay_epochs</td>
<td>[45, 90, 120]</td>
</tr>
<tr>
<td>kl_annealing_epochs</td>
<td>90</td>
</tr>
<tr>
<td>l2</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Normal</th>
<th>Cauchy</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha_initializer</td>
<td>trainable_normal</td>
<td>trainable_cauchy</td>
</tr>
<tr>
<td>alpha_regularizer</td>
<td>normal_kl_divergence</td>
<td>cauchy_kl_divergence</td>
</tr>
<tr>
<td>gamma_initializer</td>
<td>trainable_normal</td>
<td>trainable_cauchy</td>
</tr>
<tr>
<td>gamma_regularizer</td>
<td>normal_kl_divergence</td>
<td>cauchy_kl_divergence</td>
</tr>
<tr>
<td>prior_stddev</td>
<td>0.05</td>
<td>0.005</td>
</tr>
<tr>
<td>dropout_rate (init)</td>
<td>0.001</td>
<td>10^-6</td>
</tr>
<tr>
<td>random_sign_init</td>
<td>-0.75</td>
<td>-0.5</td>
</tr>
</tbody>
</table>
Table 2.7: Hyperparameter values for Rank-1 Bayesian RNNs on MIMIC-III.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MIMIC-III</th>
</tr>
</thead>
<tbody>
<tr>
<td>ensemble_size</td>
<td>4</td>
</tr>
<tr>
<td>embeddings_initializer</td>
<td>trainable_normal</td>
</tr>
<tr>
<td>embeddings_regularizer</td>
<td>normal_kl_divergence</td>
</tr>
<tr>
<td>random_sign_init</td>
<td>0.5</td>
</tr>
<tr>
<td>rnn_dim</td>
<td>512</td>
</tr>
<tr>
<td>hidden_layer_dim</td>
<td>512</td>
</tr>
<tr>
<td>l2</td>
<td>1e−4</td>
</tr>
<tr>
<td>bagging_time_precision</td>
<td>86400</td>
</tr>
<tr>
<td>num_ece_bins</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Normal</th>
<th>Cauchy</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha_initializer</td>
<td>trainable_normal</td>
<td>trainable_cauchy</td>
</tr>
<tr>
<td>alpha_regularizer</td>
<td>normal_kl_divergence</td>
<td>cauchy_kl_divergence</td>
</tr>
<tr>
<td>gamma_initializer</td>
<td>trainable_normal</td>
<td>trainable_cauchy</td>
</tr>
<tr>
<td>gamma_regularizer</td>
<td>normal_kl_divergence</td>
<td>cauchy_kl_divergence</td>
</tr>
<tr>
<td>prior_mean</td>
<td>1.0</td>
<td>0.5</td>
</tr>
<tr>
<td>prior_stddev</td>
<td>0.1</td>
<td>0.0001</td>
</tr>
<tr>
<td>dropout_rate (init)</td>
<td>0.001</td>
<td>5e−7</td>
</tr>
<tr>
<td>dense_embedding_dimension</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>embedding_dimension_multiplier</td>
<td>0.85827</td>
<td>0.984215</td>
</tr>
<tr>
<td>batch_size</td>
<td>128</td>
<td>32</td>
</tr>
<tr>
<td>learning_rate</td>
<td>0.00030352</td>
<td>0.001</td>
</tr>
<tr>
<td>fast_weight_lr_multiplier</td>
<td>1.0</td>
<td>0.575</td>
</tr>
<tr>
<td>kl_annealing_steps</td>
<td>20000</td>
<td>694216</td>
</tr>
<tr>
<td>max_steps</td>
<td>25000</td>
<td>12000</td>
</tr>
<tr>
<td>bagging_aggregate Older_than</td>
<td>−1</td>
<td>60 * 60 * 24 * 90</td>
</tr>
<tr>
<td>clip_norm</td>
<td>7.29199</td>
<td>1.83987</td>
</tr>
</tbody>
</table>
2.7.3 Further Ablation Studies

2.7.3.1 Initialization

There are two sets of parameters to initialize: the set of weights $W$ and the variational parameters of the rank-1 distributions $q(r)$ and $q(s)$. The weights are initialized just as in deterministic networks. For the variational posterior distributions, we initialize the mean following BatchEnsemble: random sign flips of $\pm 1$ or a draw from a normal centered at 1. This encourages each sampled vector to be roughly orthogonal from one another (thus inducing different directions for diverse solutions as one takes gradient steps); unit mean encourages the identity.

For the variational standard deviation parameters $\sigma$, we explore two approaches (Figure 2.9). The first is a “deterministic initialization,” where $\sigma$ is set close to zero such that—when combined with KL annealing—the initial optimization trajectory resembles a deterministic network’s. This is commonly used for variational inference (e.g., Kucukelbir et al. 2017). Though this aids optimization and aims to prevent underfitting, one

Figure 2.9: Dropout-parameterized initialization for the variational distribution’s standard deviations. Each boxplot is over 96 runs from a hyperparameter sweep. Using a dropout rate (and therefore standard deviation) close to zero gets much better accuracy at a slight cost of calibration error.
potential reason for why BNNs still underperform is that a deterministic initialization encourages poorly estimated uncertainties: the distribution of weights may be less prone to expand as the annealed KL penalizes deviations from the prior (the cost tradeoff under the likelihood may be too high). Alternatively, we also try a “dropout initialization”, where standard deviations are reparameterized with a dropout rate: $\sigma = \sqrt{p/(1-p)}$ where $p$ is the binary dropout probability. Dropout rates between 0.1 and 0.3 (common in modern architectures) imply a standard deviation of 0.3-0.65. Figure 2.9 shows accuracy and calibration both decrease as a function of initialized dropout rate; NLL stays roughly the same. We recommend deterministic initialization as the accuracy gains justify the minor cost in calibration.

2.7.3.2 Real-valued Scale Parameterization

As shown in Equation 2.3, the hierarchical prior over $r$ and $s$ induces a prior over the scale parameters of the layer’s weights. A natural question that arises is: should the $r$ and $s$ priors be constrained to be positive-valued, or left unconstrained as real-valued priors? Intuitively, real-valued priors are preferable because they can modulate the sign of the layer’s inputs and outputs. To determine whether this is beneficial and necessary, we perform an ablation under our CIFAR-10 setup (Section 2.4). In this experiment, we compare a global mixture of Gaussians for the real-valued prior, and a global mixture of log-Gaussian distributions for the positive-valued prior. For each, we tune over the initialization of the prior’s standard deviation, and the L2 regularisation for the point-wise estimated $W$. For the Gaussians, we also tune over the initialization of the prior’s mean.

Figure 2.10 displays our findings. Similar to study of priors over $r$, $s$, or both, we compare results across NLL, accuracy, and ECE on the test set and CIFAR-10-C corruptions dataset. We find that both setups are comparable on test accuracy, and that the real-valued setup outperforms the other on test NLL and ECE. For the corruptions task, the two setups

2To derive this, observe that dropout’s Bernoulli noise, which takes the value 0 with probability $p$ and $1/(1-p)$ otherwise, has mean 1 and variance $p/(1-p)$ [Srivastava et al. 2014].
compare equally on NLL, and differ on accuracy and ECE.

2.7.3.3 Number of Evaluation Samples

In Table 2.8 we experiment with using multiple weight samples, per mixture component, per example, at evaluation time for our Wide ResNet-28-10 model trained on CIFAR-10. In all cases, we use the same model that was trained using only a single weight sample (per mixture component, per example). As expected, an increased number of samples improves metric performance, with a significant improvement across all corrupted metrics. This demonstrates one of the benefits to incorporating local distributions over each mixture component, namely that given an increased computational budget, one can improve upon the metric performance at prediction time.
Table 2.8: Results across multiple weight samples (per mixture component, per example) at evaluation time for Wide ResNet-28-10 on CIFAR-10. Greater than 1 sample with Gaussian distributions yields a marginal improvement on in-distribution NLL and ECE, while yielding a significant improvement on all corrupted metrics. Cauchy rank-1 BNNs with 4 weight samples outperform Gaussians on all metrics except ECE. Note that training still uses a single weight sample (per mixture component, per example) for both Gaussian and Cauchy rank-1 BNNs. We include the deep ensembles results again to show that with an increased number of samples, a rank-1 WRN-28-10 can exceed an ensemble of WRN-28-5 models, which collectively have a comparable parameter count.

<table>
<thead>
<tr>
<th>Method</th>
<th>NLL(↓)</th>
<th>Accuracy(↑)</th>
<th>ECE(↓)</th>
<th>cNLL / cA / cECE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 sample</td>
<td>0.128</td>
<td>96.3</td>
<td>0.008</td>
<td>0.84 / 76.7 / 0.080</td>
</tr>
<tr>
<td>Rank-1 BNN - Gaussian 4 samples</td>
<td>0.126</td>
<td>96.3</td>
<td>0.008</td>
<td>0.80 / 77.3 / 0.074</td>
</tr>
<tr>
<td>25 samples</td>
<td>0.125</td>
<td><strong>96.3</strong></td>
<td><strong>0.007</strong></td>
<td><strong>0.77 / 77.8 / 0.070</strong></td>
</tr>
<tr>
<td>Rank-1 BNN - Cauchy 4 samples</td>
<td><strong>0.120</strong></td>
<td>96.5</td>
<td>0.009</td>
<td><strong>0.74 / 80.5 / 0.090</strong></td>
</tr>
<tr>
<td>Deep Ensembles</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRN-28-5</td>
<td>0.115</td>
<td>96.3</td>
<td>0.008</td>
<td>0.84 / 77.2 / 0.089</td>
</tr>
<tr>
<td>WRN-28-10</td>
<td><strong>0.114</strong></td>
<td><strong>96.6</strong></td>
<td>0.010</td>
<td>0.81 / 77.9 / 0.087</td>
</tr>
</tbody>
</table>
2.7.4 Choices of Loss Functions

2.7.4.1 Definitions

\[ \mathbf{x} \in \mathbb{R}^d, \quad y_c \in \{0, 1\}, \sum_{c=1}^{C} y_c = 1 \]

logits = \( f(x, \theta) \)

probs = \( \text{softmax}(\text{logits}) \)

\[ \text{softmax}(\lambda) = \frac{e^{\lambda}}{\sum_{i=1}^{d} e^{\lambda_i}} \]

\[ p(y|x, \theta) = \text{Categorical}(y; \text{probs}) \]

\[ = \prod_{c=1}^{C} (\text{softmax}(f(x, \theta))_c)^{y_c} \]

\[- \log p(y|x, \theta) = - \sum_{c=1}^{C} y_c \log \text{softmax}(f(x, \theta))_c \]

\[ = -y^\top \log \text{softmax}(f(x, \theta)) \]

\[ M = \text{num\_weight\_samples} \]

\[ C = \text{num\_classes} \]

2.7.4.2 Negative log-likelihood of marginalized logits

\[ = -y^\top \log \text{softmax} \left( \int f(x, \theta)p(\theta)d\theta \right) \]

\[ \approx -y^\top \log \text{softmax} \left( \frac{1}{M} \sum_{m=1}^{M} f(x, \theta^{(m)}) \right) \] (2.11)

2.7.4.3 Negative log-likelihood of marginalized_probs

\[ = -y^\top \log \left\{ \int \text{softmax}(f(x, \theta))p(\theta)d\theta \right\} \]

\[ \approx -y^\top \log \left\{ \frac{1}{M} \sum_{m=1}^{M} \text{softmax}(f(x, \theta^{(m)})) \right\} \] (2.12)
2.7.4.4 Marginal Negative log-likelihood (i.e., average NLL or Gibbs cross-entropy)

\[
\begin{align*}
&= \mathbb{E}_{p(\theta)}[- \log p(y|x, \theta)] \\
&= \int - \log \{p(y|x, \theta)p(\theta)d\theta \\
&\approx \frac{1}{M} \sum_{m=1}^{M} \{- \log p(y|x, \theta^{(m)})\}
\end{align*}
\]  

(2.13)

2.7.4.5 Negative log marginal likelihood (i.e., mixture NLL)

\[
\begin{align*}
&= - \log p(y|x) \\
&= - \log \left\{ \int p(y|x, \theta)p(\theta)d\theta \right\} \\
&\approx - \log \left\{ \frac{1}{M} \sum_{m=1}^{M} p(y|x, \theta^{(m)}) \right\} \\
&= - \log \left\{ \sum_{m=1}^{M} p(y|x, \theta^{(m)}) \right\} + \log M \\
&= - \log \left\{ \sum_{m=1}^{M} \exp \log p(y|x, \theta^{(m)}) \right\} + \log M \\
&= - \log \text{sumexp} \left\{ \log p(y|x, \theta^{(m)}) \right\} + \log M
\end{align*}
\]  

(2.14)

As we saw in Section 2.3 due to Jensen’s inequality, (2.14) \leq (2.13). However, we find that minimizing the upper bound (i.e. Eq. 2.13) to be easier while allowing for improved generalization performance. Note that for classification problems (i.e., Bernoulli or Categorical predictive distributions), Eq. 2.12 is equivalent to Eq. 2.14 though more generally, marginalizing the parameters of the predictive distribution before computing the negative log likelihood (Eq. 2.12) is different from marginalizing the likelihood before taking the negative log (Eq. 2.14), and from marginalizing the negative log likelihood (Eq. 2.13). Also note that though they are mathematically equivalent for classification, the formulation of Eq. 2.14 is more numerically stable than Eq. 2.12.
2.7.5 Out-of-distribution Performance

2.7.5.1 CIFAR-10-C Results
Figure 2.11: Results on CIFAR-10-C showing median performance across corruption types, and for increasing settings of the skew intensity.
2.7.5.2 CIFAR-100-C Results

![Graphs showing median performance across corruption types and for increasing settings of the skew intensity.](image)

(a) Accuracy (higher is better).

(b) Negative log-likelihood (lower is better).

(c) Expected calibration error (lower is better).

**Figure 2.12**: Results on CIFAR-100-C showing median performance across corruption types, and for increasing settings of the skew intensity.
2.7.5.3 ImageNet-C Results

(a) Accuracy (higher is better).

(b) Negative log-likelihood (lower is better).

(c) Expected calibration error (lower is better).

Figure 2.13: Results on ImageNet-C showing median performance across corruption types, and for increasing settings of the skew intensity.
Reconciling Meta-learning and Continual Learning with Nonparametric Mixtures

3.1 Introduction

Meta-learning algorithms aim to increase the efficiency of learning by treating task-specific learning episodes as examples from which to generalize [Schmidhuber, 1987]. The central assumption of a meta-learning algorithm is that some tasks are inherently related and so inductive transfer can improve sample efficiency and generalization [Caruana, 1993, 1998, Baxter, 2000]. In learning a single set of domain-general hyperparameters that parameterize a metric space [Vinyals et al., 2016] or an optimizer [Ravi and Larochelle, 2017, Finn et al., 2017], recent meta-learning algorithms make the assumption that tasks are equally related, and therefore non-adaptive, mutual transfer is appropriate. This assumption has been cemented in recent few-shot learning benchmarks, which comprise a set of tasks generated in a uniform manner [e.g., Vinyals et al., 2016, Finn et al., 2017].

However, the real world often presents scenarios in which an agent must decide what degree of transfer is appropriate. In some cases, a subset of tasks are more strongly related to each other, and so non-uniform transfer provides a strategic advantage. On the other hand, transfer in the presence of dissimilar or outlier tasks worsens generalization performance [Rosenstein et al., 2005, Deleu and Bengio, 2018]. Moreover, when the underlying
task distribution is non-stationary, inductive transfer to previously observed tasks should exhibit graceful degradation to address the catastrophic forgetting problem [Kirkpatrick et al., 2017]. In these settings, the consolidation of all inductive biases into a single set of hyperparameters is not well-posed to deal with changing or diverse tasks. In contrast, in order to account for this degree of task heterogeneity, people detect and adapt to novel contexts by attending to relationships between tasks [Collins and Frank, 2013].

In this work, we learn a mixture of hierarchical models that allows a meta-learner to adaptively select over a set of learned parameter initializations for gradient-based adaptation to a new task. The method is equivalent to clustering task-specific parameters in the hierarchical model induced by recasting gradient-based meta-learning as hierarchical Bayes [Grant et al., 2018] and generalizes the model-agnostic meta-learning (MAML) algorithm introduced in [Finn et al., 2017]. By treating the assignment of task-specific parameters to clusters as latent variables, we can directly detect similarities between tasks on the basis of the task-specific likelihood, which may be parameterized by a complex model such as a neural network. Our approach therefore alleviates the need for explicit geometric or probabilistic modeling assumptions about the weights of a complex parametric model and provides a scalable method to regulate information transfer between episodes.

We additionally consider the setting of a non-stationary or evolving task distribution, which necessitates a meta-learning method that possesses adaptive complexity. We extend our latent variable model to the non-parametric setting and leverage stochastic point estimation in an infinite mixture [Rasmussen, 2000] over model parameters; point estimation is scalable and requires no distributional assumptions, and is so the online gradient-based mixture approach is compatible with any meta-learning algorithm or neural network architecture that admits gradient-based optimization. We demonstrate the unexplored ability of this combination of non-parametric parameter priors with neural network models to automatically detect and adapt to task distribution shift in a naturalistic image dataset. Our work tackles the non-trivial setting of task-agnostic continual learning—where the task change is unobserved—thus addressing an unresolved challenge in task-aware continual learn-
Gradient-based Meta-learning as Hierarchical Bayes

Since our approach is grounded in the probabilistic formulation of meta-learning as hierarchical Bayes [Baxter, 1997], our approach can be applied to any probabilistic meta-learner. In this work, we focus on model-agnostic meta-learning (MAML) [Finn et al., 2017], a gradient-based meta-learning approach that estimates global parameters to be shared among task-specific models as an initialization for a few steps of gradient descent. MAML admits a natural interpretation as parameter estimation in a hierarchical probabilistic model, where the learned initialization acts as data-driven regularization for the estimation of task-specific parameters $\hat{\phi}_j$.

In particular, Grant et al. [2018] cast MAML as posterior inference for task-specific parameters $\phi_j$ given some samples of task-specific data $x_{j,1:N}$ and a prior over $\phi_j$ that is induced by early stopping of an iterative optimization procedure; truncation at $K$ steps of gradient descent on the negative log-likelihood $-\log p(x_{j,1:N} \mid \phi_j)$ starting from $\phi_{j(0)} = \theta$ can be then understood as mode estimation of the posterior $p(\phi_j \mid x_{j,1:N}, \theta)$. See Figure 3.1 for an illustration of the corresponding graphical model.

The mode estimates $\hat{\phi}_j = \phi_{j(0)} + \alpha \sum_{k=1}^{K} \nabla_{\phi} \log p(x_{j,1:N} \mid \phi_{j(k-1)})$ are then combined to evaluate the marginal likelihood for each task as

$$p(x_{j,N+1:N+M} \mid \theta) = \int p(x_{j,N+1:N+M} \mid \phi_j) p(\phi_j \mid \theta) d\phi_j \approx p(x_{j,N+1:N+M} \mid \hat{\phi}_j), \quad (3.1)$$

where $x_{j,N+1:N+M}$ is another set of samples from the $j$th task. A training dataset can then be summarized in an empirical Bayes point estimate of $\theta$ computed by gradient-based optimization of the joint marginal likelihood in (3.1) in across tasks, so that the likelihood of a datapoint sampled from a new task can be computed using only $\theta$ and without storing the task-specific parameters.
Figure 3.1: (a) The standard hierarchical Bayesian model for multi-task learning. A set of global parameters $\theta$ provides an inductive bias for the estimation of task-specific parameters $\phi_j$. (b) In a mixture of hierarchical Bayesian models, the cluster assignment of each task-specific parameter set $\phi_j$ is represented with a latent Categorical variable $z_j$. (c) Allowing an unbounded number of mixture components instantiates a non-parametric model that has the potential to grow with the data in capacity.

3.3 Improving Meta-learning by Modeling Latent Task Structure

If the task distribution is heterogeneous, assuming a single parameter initialization $\theta$ for gradient-based meta-learning is not suitable because it is unlikely that the point estimate computed by a few steps of gradient descent will sufficiently adapt the task-specific parameters $\phi$ to a diversity of tasks. Moreover, explicitly estimating relatedness between tasks has the potential to aid the efficacy of a meta-learning algorithm by modulating both positive and negative transfer [Thrun and Pratt, 1998, Zhang and Schneider, 2010, Rothman et al., 2010, Zhang and Yeung, 2014], and by identifying outlier tasks that require a more significant degree of adaptation [Xue et al., 2007, Gupta et al., 2013]. Nonetheless, defining an appropriate notion of task relatedness is a difficult problem in the high-dimensional parameter or activation space of models such as neural networks.

Using the probabilistic interpretation of Section 3.2, we deal with the variability in the tasks by assuming that each set of task-specific parameters $\phi_j$ is drawn from a mixture of base distributions each of which is parameterized by a hyperparameter $\theta^{(l)}$ (as in Figure 3.1). Accordingly, we capture task relatedness by estimating the likelihood of assigning each task to a mixture component based simply on the task negative log likelihood after
Algorithm 1: Stochastic gradient-based EM for finite and infinite mixtures

dataset $\mathcal{D}$, meta-learning rate $\beta$, adaptation rate $\alpha$, temperature $\tau$, initial cluster count $L_0$, meta-batch size $J$, training batch size $N$, validation batch size $M$, adaptation iteration count $K$, global prior $G_0$

Initialize cluster count $L \leftarrow L_0$ and meta-level parameters $\theta^{(1)}, \ldots, \theta^{(L)} \sim G_0$

while not converged do
    Draw tasks $T_1, \ldots, T_J \sim p(\mathcal{T})$
    for $j$ in $1, \ldots, J$ do
        Draw task-specific datapoints, $x_{j1}, \ldots, x_{jN+M} \sim p_{T_j}(x)$
        Draw a parameter initialization for a new cluster from the global prior, $\theta^{(L+1)} \sim G_0$
        for $\ell$ in $1, \ldots, L, L+1$ do
            Initialize $\hat{\phi}^{(\ell)}_j \leftarrow \theta^{(\ell)}$
            Compute task-specific mode estimate, $\hat{\phi}^{(\ell)}_j \leftarrow \hat{\phi}^{(\ell)}_j + \alpha \sum_k \nabla_{\phi} \log p(x_{jN} | \hat{\phi}^{(\ell)}_j)$
            Compute assignment of tasks to clusters, $\gamma_j \leftarrow \text{E-STEP}(x_{jN+M} | \hat{\phi}^{(1:L)}_j)$
            Update each component $\ell$ in $1, \ldots, L$, $\theta^{(\ell)} \leftarrow \text{M-STEP}(\{x_{jN+M} | \hat{\phi}^{(\ell)}_j, \gamma_j\}_{j=1}^J)$
        Summarize $\{\theta_1, \ldots\}$ to update global prior $G_0$
    return $\{\theta^{(1)}, \ldots\}$

E-STEP($\{x_{ji}\}_{i=1}^N, \{\hat{\phi}^{(\ell)}_j\}_{\ell=1}^L$)

return $\tau$-softmax($\sum_i \log p(x_{ji} | \hat{\phi}^{(\ell)}_j)$)

M-STEP($\{x_{ji}\}_{i=1}^N, \hat{\phi}^{(\ell)}_j, \gamma_j$)

return $\beta \nabla_{\theta} \sum_j \gamma_j \log p(x_{ji} | \hat{\phi}^{(\ell)}_j)$

Top: Algorithm 3.2: Stochastic gradient-based expectation maximization (EM) for probabilistic clustering of task-specific parameters in a meta-learning setting.
Bottom: Subroutine 3.3: The E-STEP and M-STEP for a finite mixture of hierarchical Bayesian models implemented as gradient-based meta-learners.

A few steps of gradient-based adaptation. The result is a scalable meta-learning algorithm that jointly learns task-specific cluster assignments and model parameters, and is capable of modulating the transfer of information across tasks by clustering together related task-specific parameter settings.

Formally, let $z_j$ be the categorical latent variable indicating the cluster assignment of each task-specific parameter $\phi_j$. A direct maximization of the mixture model likelihood is a gubernatorial optimization problem that can grow intractable. This intractability is equally problematic for the posterior distribution over the cluster assignment variables $z_j$ and the task-specific parameters $\phi_j$, which are both treated as latent variables in the
probabilistic formulation of meta-learning. A scalable approximation involves representing the conditional distribution for each latent variable with a maximum a posteriori (MAP) estimate. In our meta-learning setting of a mixture of hierarchical Bayes (HB) models, this suggests an augmented expectation maximization (EM) procedure [Dempster et al., 1977] alternating between an E-STEP that computes an expectation of the task-to-cluster assignments $z_j$, which itself involves the computation of a conditional mode estimate for the task-specific parameters $\phi_j$, and an M-STEP that updates the hyperparameters $\theta^{(1:L)}$ (see Subroutine 3.3).

To ensure scalability, we use the minibatch variant of stochastic optimization [Robbins and Monro, 1951] in both the E-STEP and the M-STEP; such approaches to EM are motivated by a view of the algorithm as optimizing a single free energy at both the E-STEP and the M-STEP [Neal and Hinton, 1998]. In particular, for each task $j$ and cluster $\ell$, we follow the gradients to minimize the negative log-likelihood on the training data points $x_{j1:N}$, using the cluster parameters $\theta^{(\ell)}$ as initialization. This allows us to obtain a modal point estimate of the task-specific parameters $\hat{\phi}_j^{(\ell)}$. The E-STEP in Subroutine 3.3 leverages the connection between gradient-based meta-learning and HB [Grant et al., 2018] and the differentiability of our clustering procedure to employ the task-specific parameters to compute the posterior probability of cluster assignment. Accordingly, based on the likelihood of the same training data points under the model parameterized by $\hat{\phi}_j^{(\ell)}$, we compute the cluster assignment probabilities as

$$
\gamma_{j}^{(\ell)} := p(z_j = \ell \mid x_{j1:N}, \theta^{(1:L)}) \propto \int p(x_{j1:N} \mid \phi_j) p(\phi_j \mid \theta^{(\ell)}) \, d\phi_j \approx p(x_{j1:N} \mid \hat{\phi}_j^{(\ell)}) . \quad (3.2)
$$

The cluster means $\theta^{(\ell)}$ are then updated by gradient descent on the validation loss in the M-STEP in Subroutine 3.3; this M-STEP is analogous to the MAML algorithm in [Finn et al., 2017] with the addition of mixing weights $\gamma_{j}^{(\ell)}$.

Note that, unlike other recent approaches to probabilistic clustering [e.g., Bauer et al., 2017] we adhere to the episodic meta-learning setup for both training and testing since only the task support set $x_{j1:N}$ is used to compute both the point estimate $\hat{\phi}_j^{(\ell)}$ and the
Table 3.1: Meta-test set accuracy on the miniImageNet 5-way, 1- and 5-shot classification benchmarks from Vinyals et al. [2016] among methods using a comparable architecture (the 4-layer convolutional network from Vinyals et al. [2016]). For methods on which we report results in later experiments, we additionally report the total number of parameters optimized by the meta-learning algorithm.  

a Results reported by Ravi and Larochelle [2017].  
b We report test accuracy for models matching train and test “shot” and “way”.  
c We report test accuracy for a comparable base (task-specific network) architecture.

<table>
<thead>
<tr>
<th>Model</th>
<th>Num. param.</th>
<th>1-shot (%)</th>
<th>5-shot (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>matching network [Vinyals et al., 2016]</td>
<td>43.56± 0.84</td>
<td>55.31± 0.73</td>
<td></td>
</tr>
<tr>
<td>meta-learner LSTM [Ravi and Larochelle, 2017]</td>
<td>43.44± 0.77</td>
<td>60.60± 0.71</td>
<td></td>
</tr>
<tr>
<td>prototypical networks [Snell et al., 2017]</td>
<td>46.61± 0.78</td>
<td>65.77± 0.70</td>
<td></td>
</tr>
<tr>
<td>MAML [Finn et al., 2017]</td>
<td>48.70± 1.84</td>
<td>63.11± 0.92</td>
<td></td>
</tr>
<tr>
<td>MT-net [Lee and Choi, 2018]</td>
<td>38,907</td>
<td>51.70± 1.84</td>
<td></td>
</tr>
<tr>
<td>PLATIPUS [Finn et al., 2018]</td>
<td>65,546</td>
<td>50.13± 1.86</td>
<td></td>
</tr>
<tr>
<td>VERSA [Gordon et al., 2019]</td>
<td>807,938</td>
<td>48.53± 1.84</td>
<td></td>
</tr>
<tr>
<td>Our method: 2 components</td>
<td>65,546</td>
<td>49.60± 1.50</td>
<td>64.60± 0.92</td>
</tr>
<tr>
<td>3 components</td>
<td>98,319</td>
<td>51.20± 1.52</td>
<td>65.00± 0.96</td>
</tr>
<tr>
<td>4 components</td>
<td>131,092</td>
<td>50.49± 1.46</td>
<td>64.78± 1.43</td>
</tr>
<tr>
<td>5 components</td>
<td>163,865</td>
<td>51.46± 1.68</td>
<td></td>
</tr>
</tbody>
</table>

Cluster responsibilities \( \gamma_j^{(t)} \). See Algorithm 3.2 for the full algorithm, whose high-level structure is shared with the non-parametric variant of our method detailed in Section 3.5.

A comparison of the graphical models is in Figure 3.1.

### 3.4 Experiment: miniImageNet Few-shot Classification

Clustering task-specific parameters provides a way for a meta-learner to deal with task heterogeneity as each cluster can be associated with a subset of the tasks that would benefit most from mutual transfer. While we do not expect existing tasks to present a significant degree of heterogeneity given the uniform sampling assumptions behind their design, we nevertheless conduct an experiment to validate that our method gives an improvement on a standard benchmark for few-shot learning.

We apply Algorithm 3.2 with Subroutine 3.4 and \( L \in \{2, 3, 4, 5\} \) components to the
1-shot and 5-shot, 5-way, miniImageNet few-shot classification benchmarks [Vinyals et al., 2016]; Appendix 3.9.2.2 contains additional experimental details. We demonstrate in Table 1 that a mixture of meta-learners improves the performance of gradient-based meta-learning on this task for any number of components. However, the performance of the parametric mixture does not improve monotonically with the number of components \( L \). This leads us to the development of non-parametric clustering for continual meta-learning, where enforcing specialization to subgroups of tasks and increasing model complexity is in fact necessary to preserve performance on prior tasks due to significant heterogeneity.

### 3.5 Scalable Online Mixtures for Task-agnostic Continual Learning

The mixture of meta-learners developed in Section 3.3 addresses a drawback of meta-learning approaches such as MAML that consolidate task-general information into a single set of hyperparameters. However, the method adds another dimension to model selection in the form of identifying the correct number of mixture components. While this may be resolved by cross-validation if the dataset is static and therefore the number of components can remain fixed, adhering to a fixed number of components throughout training is not appropriate in the non-stationary regime, where the underlying task distribution changes as different types of tasks are presented sequentially in a continual learning setting. In this regime, it is important to incrementally introduce more components that can each specialize to the distribution of tasks observed at the time of spawning.

To address this, we derive a scalable stochastic estimation procedure to compute the expectation of task-to-cluster assignments (E–STEP) for a growing number of task clusters in a non-parametric mixture model [Rasmussen, 2000] called the Dirichlet process mixture model (DPMM). The formulation of the Dirichlet process mixture model (DPMM) that is most appropriate for incremental learning is the sequential draws formulation that corresponds to an instantiation of the Chinese restaurant process (CRP) [Rasmussen, 2000].
A CRP prior over $z_j$ allows some probability to be assigned to a new mixture component while the task identities are inferred in a sequential manner, and has therefore been key to recent online and stochastic learning of the DPMM [Lin 2013]. A draw from a CRP proceeds as follows: For a sequence of tasks, the first task is assigned to the first cluster and the $j$th subsequent task is then assigned to the $\ell$th cluster with probability

$$p(z_j = \ell \mid z_{1:j-1}, \zeta) = \begin{cases} 
n^{(\ell)}/n + \zeta & \text{for } \ell \leq L \\
\zeta/n + \zeta & \text{for } \ell = L + 1,
\end{cases} \quad (3.3)$$

where $L$ is the number of non-empty clusters, $n^{(\ell)}$ is the number of tasks already occupying a cluster $\ell$, and $\zeta$ is a fixed positive concentration parameter. The prior probability associated with a new mixture component is therefore $p(z_j = L + 1 \mid z_{1:j-1}, \zeta)$. And the joint log-likelihood of the data and parameters for a specific task $j$ in the mixture of hierarchical models can be written as

$$\log p(x_j, \phi_j \mid \theta^{(1:L)}) = \sum_{\ell} \gamma_j^{(\ell)} \log p(x_j, \phi_j \mid \theta^{(\ell)}) + L \log \zeta + \sum_{\ell} \log \Gamma(n^{(\ell)}) , \quad (3.4)$$

where $\gamma_j^{(\ell)}$ is the responsibility of cluster $\ell$ for task-specific parameters $\phi_j$, and where we assume a single task-specific datapoint $x_j$ for notational convenience; in general, the likelihood factorizes across datapoints specific to a task as in Section 3. We refer the reader to [Rasmussen 2000] for more details on the likelihood function associated with the DPMM.

In a similar spirit to Section 3.3, we develop a stochastic EM procedure for the estimation of the latent task-specific parameters $\phi_{1:J}$ and the meta-level parameters $\theta^{(1:L)}$ in the DPMM, which allows the number of observed task clusters to grow in an online manner with the diversity of the task distribution. While computation of the mode estimate of the task-specific parameters $\phi_j$ is mostly unchanged from the finite variant, the estimation of the cluster assignment variables $z$ in the E-STEP requires revisiting the Gibbs conditional distributions due to the potential addition of a new cluster at each step. For a DPMM, the
E-STEP ( \( x_{j1:N}, \hat{\phi}_j^{(1:L)}, \text{concentration } \zeta, \text{threshold } \epsilon \))

1. DPMM log-likelihood for all \( \ell \) in 1, \ldots, \( L \),
   \[ \rho_j^{(\ell)} \leftarrow \sum_i \log p(x_j | \hat{\phi}_j^{(\ell)}) + \log n^{(\ell)} \]
2. DPMM log-likelihood for new component, \( \rho_j^{(L+1)} \leftarrow \sum_i \log p(x_j | \hat{\phi}_j^{(L+1)}) + \log \zeta \)
3. DPMM assignments, \( \gamma_j \leftarrow \tau\text{-softmax}(\rho_j^{(1)}, \ldots, \rho_j^{(L+1)}) \)
   
   if \( \gamma_j^{(L+1)} > \epsilon \) then
   1. Expand the model by incrementing \( L \leftarrow L + 1 \)
   else
   1. Renormalize \( \gamma_j \leftarrow \tau\text{-softmax}(\rho_j^{(1)}, \ldots, \rho_j^{(L)}) \)

return \( \gamma_j \)

M-STEP ( \( \{ \{ x_j \}_{j=1}^M, \hat{\phi}_j^{(\ell)}, \gamma_j \}, \text{concentration } \zeta \))

return \( \beta \nabla_\theta [\sum_j \gamma_j \log p(x_j | \hat{\phi}_j^{(\ell)}) + \log n^{(\ell)}] \)

**Subroutine 3.4:** The E-STEP and M-STEP for an infinite mixture of hierarchical Bayesian models.

Conditional distributions for \( z_j \) are

\[
p(z_j = \ell | x_{j1:M}, z_{1:j-1}) \propto \begin{cases} 
  n^{(\ell)} \int p(x_{j1:M} | \phi_j^{(\ell)}) p(\phi_j^{(\ell)} | \theta) \, d\phi_j \, dG_\ell(\theta) & \text{for } \ell \leq L \\
  \zeta \int p(x_{j1:M} | \phi_j^{(0)}) p(\phi_j^{(0)} | \theta) \, d\phi_j \, dG_0(\theta) & \text{for } \ell = L + 1
\end{cases}
\]

(3.5)

with \( G_0 \) as the base measure or global prior over the components of the CRP, \( G_\ell \) is the prior over each cluster’s parameters, initialized with a draw from a Gaussian centered at \( G_0 \) with a fixed variance and updated over time using summary statistics from the set of active components \( \{ \theta^{(0)}, \ldots, \theta^{(L)} \} \).

Taking the logarithm of the posterior over task-to-cluster assignments \( z_j \) in (3.5) and using a mode estimate \( \hat{\phi}_j^{(\ell)} \) for task-specific parameters \( \phi_j \) as drawn from the \( \ell \)th cluster gives the E-STEP in Subroutine 3.4. We may also omit the prior term \( \log p(\hat{\phi}_j^{(\ell)} | \theta^{(\ell)}) \) as it arises as an implicit prior resulting from truncated gradient descent, as explained in Section 3.3 of Grant et al. [2018].
3.6 Experiments: Task-agnostic Continual Few-shot Regression & Classification

By treating the assignment of tasks to clusters as latent variables, the algorithm of Section 3.5 can adapt to a changing distribution of tasks, without any external information to signal distribution shift (i.e., in a task-agnostic manner). Here, we present our main experimental results on both a novel synthetic regression benchmark as well as a novel evolving variant of miniImageNet, and confirm the algorithm’s ability to adapt to distribution shift by spawning a newly specialized cluster.

High-capacity baselines. As an ablation, we compare to the non-uniform parametric mixture proposed in Section 3.3 with the number of components fixed at the total number of task distributions in the dataset (3). We also consider a uniform parametric mixture in which each component receives equal assignments; this can also be seen as the non-uniform mixture in the infinite temperature ($\tau$) limit. Note that our meta-learner has a lower capacity than these two baselines for most of the training procedure, as it may decide to expand its capacity past one component only when the task distribution changes. Finally, for the large-scale experiment in Section 3.6.2 we compare with three recent meta-learning algorithms that report improved performance on the standard miniImageNet benchmark of Section 3.3, but are not explicitly posed to address the continual learning setting of evolving tasks: MT-net [Lee and Choi, 2018], PLATIPUS [Finn et al., 2018], and VERSA [Gordon et al., 2019].
Figure 3.7: Results on the evolving dataset of few-shot regression tasks (lower is better). Each panel (row) presents, for a specific task type (polynomial, sinusoid or sawtooth), the average meta-test set accuracy of each method over cumulative number of few-shot episodes. We additionally report the degree of loss in backward transfer (i.e., catastrophic forgetting) to the tasks in each meta-test set in the legend; all methods but the non-parametric method experience a large degree of catastrophic forgetting during an inactive phase.

3.6.1 Continual Few-shot Regression

We first consider an explanatory experiment in which three regression tasks are presented sequentially with no overlap. For input $x$ sampled uniformly from $[-5, 5]$, each regression task is generated, in a similar spirit to the sinusoidal regression setup in Finn et al. [2017], from one of a set of simple but distinct one-dimensional functions (polynomial Figure 3.5a, sinusoid wave Figure 3.5b and sawtooth wave Figure 3.5c). For the experiment in Figure 3.7 and Figure 3.8 we presented the polynomial tasks for 4000 iterations, followed by sinusoid tasks for 3000 iterations, and finally sawtooth tasks. Additional details on the experimental setup can be found in Appendix 3.9.2.2.

Results: Distribution shift detection. The cluster responsibilities in Figure 3.8 on the meta-test dataset of tasks, from each of the three regression types in Figure 3.5
Figure 3.8: Each panel (row) presents task-specific per-cluster meta-test responsibilities $\gamma^{(t)}$ over time. A higher responsibility entails a higher degree of specialization of a particular cluster (color) to a particular task (row).

indicates that the non-parametric algorithm recognizes a change in the task distribution and spawns a new cluster at iterations 4000 and a bit after 7000. Each newly created cluster is specialized to the task distribution observed at the time of spawning and remains as such throughout training, since the majority of assignments for each type of regression remains under a given cluster from the time of its introduction.

Results: Improved generalization and slower degradation of performance.

We investigate the progression of the meta-test mean-squared error (MSE) for the three regression task distributions in Figure 3.7. We first note the clear advantage of non-uniform assignment both in improved generalization, when testing on the active task distribution, and in slower degradation, when testing on previous distributions. This is due to the ability of these methods to modulate the transfer of information in order to limit negative transfer. In contrast, the uniform method cannot selectively adapt specific clusters to be responsible for any given task, and thus inevitably suffers from catastrophic forgetting.

The adaptive capacity of our non-parametric method allows it to spawn clusters that specialize to newly observed tasks. Accordingly, even if the overall capacity is lower than
Figure 3.9: Results on the evolving dataset of filtered miniImageNet few-shot classification tasks (higher is better). Each panel (row) presents, for a specific task type (filter), the average meta-test set accuracy over cumulative number of few-shot episodes. We additionally report the degree of loss in backward transfer (catastrophic forgetting, CF) in the legend. This is calculated for each method as the average drop in accuracy on the first two tasks at the end of training (lower is better).

that of the comparable non-uniform parametric method, our method achieves similar or better generalization, at any given training iteration. More importantly, specialization allows our method to better modulate information transfer as the clusters are better differentiated. Consequently, each cluster does not account for many assignments from more than a single task distribution, throughout training. Therefore, we observed a significantly slower rate of degradation of the MSE on previous task distributions as new tasks are introduced. This is especially evident from the performance on the first task in Figure 3.7.
3.6.2 Continual Few-shot Classification

Next, we consider an evolving variant of the miniImageNet few-shot classification task. In this variant, one of a set of artistic filters are applied to the images during the meta-training procedure to simulate a changing distribution of few-shot classification tasks. For the experiment in Figure 8 and Figure 9 we first train using images with a “blur” filter (Figure 3.6b) for 7500 iterations, then with a “night” filter (Figure 3.6c) for another 7500 iterations, and finally with a “pencil” filter (Figure 3.6d). Additional details on the experimental setup can be found in Appendix 3.9.2.2.

Results: Meta-test accuracy. In Figure 9, we report the evolution of the meta-test accuracy for two variants of our non-parametric meta-learner in comparison to the parametric baselines introduced in Section 3.6, high-capacity baselines. The task-agnostic variant is the core algorithm described in previous sections, as used for the regression tasks. The task-aware variant augments the core algorithm with a cool-down period that prevents overspawning for the duration of a training phase. This requires some knowledge
of the duration which is external to the meta-learner, thus the task-aware nomenclature (see Appendix 3.9.3.1 for further details).

It is clear from Figure 8 that neither of our algorithms suffer from catastrophic forgetting to the same degree as the parametric baselines. In fact, at the end of training, both of our methods outperform all the parametric baselines on the first and second task.

**Results: Specialization.** Given the higher capacity of the parametric baselines, and the inherent degree of similarity between the filtered miniImageNet task distributions (unlike the regression tasks in the previous section), the parametric baselines perform better on each task distribution while during its active phase. However, they quickly suffer from degradation once the task distribution shifts. Our approach does not suffer from this phenomenon and can handle non-stationarity owing to the credit assignment of a single task distribution to a specialized cluster. This specialization is illustrated in Figure 9, where we track the evolution of the average cluster responsibilities on the meta-test dataset from each of the three miniImageNet few-shot classification tasks. Each cluster is specialized so as to acquire the majority of a single task distribution’s test set assignments, despite the degree of similarity between tasks originating from the same source (miniImageNet). We observed this difficulty with the non-monotone improvement of parametric clustering, as a function of components, in Section 4.

### 3.7 Related Work

**Meta-learning.** In this work, we show how changes to the hierarchical Bayesian model assumed in meta-learning [Grant et al., 2018, Fig. 1(a)] can be realized as changes to a meta-learning algorithm. In contrast, follow-up approaches to improving the performance of meta-learning algorithms [e.g., Lee and Choi, 2018, Finn et al., 2018, Gordon et al., 2019] do not change the underlying probabilistic model; what differs is the inference procedure to infer values of the global (shared across tasks) and local (task-specific) parameters; for example, [Gordon et al., 2019] consider feedforward conditioning while [Finn et al., 2018]
employ variational inference. Due to consolidation into one set of global parameters shared uniformly across tasks, none of these methods inherently accommodate heterogeneity or non-stationarity.

**Continual learning.** Techniques developed to address the catastrophic forgetting problem in continual learning, such as elastic weight consolidation (EWC) \[Kirkpatrick et al., 2017\], synaptic intelligence (SI) \[Zenke et al., 2017\], variational continual learning (VCL) \[Nguyen et al., 2017\], and online Laplace approximation \[Ritter et al., 2018\] require access to an explicit delineation between tasks that acts as a catalyst to grow model size, which we refer to as task-aware. In contrast, our non-parametric algorithm tackles the task-agnostic setting in which the meta-learner recognizes a latent shift in the task distribution and adapts accordingly.

**Multi-task learning.** \[Rosenstein et al., 2005\] demonstrated that negative transfer can worsen generalization performance, and avoidance of negative transfer has motivated much work on hierarchical Bayes in transfer learning and domain adaptation \[e.g., Lawrence and Platt, 2004; Yu et al., 2005; Gao et al., 2008; Daumé III, 2009; Wan et al., 2012\]. Closest to our proposed approach is early work on hierarchical Bayesian multi-task learning with neural networks that places a prior only on the output layer \[Heskes, 1998; Bakker and Heskes, 2003; Salakhutdinov et al., 2013; Srivastava and Salakhutdinov, 2013\]. In contrast, we place a non-parametric prior on the full set of neural network weights. Furthermore, none of these approaches were applied to the episodic training setting of meta-learning. Similar to our point estimation procedure, \[Heskes, 1998; Srivastava and Salakhutdinov, 2013\] propose training a mixture model over the output layer weights of a neural network using MAP inference. However, these approaches do not scale well to all the layers in a network as performing full passes on the dataset for inference of the full set of weights is computationally intractable in general.
Clustering. Incremental or stochastic clustering was considered in the EM setting in [Neal and Hinton 1998], and in the K-means setting in [Sculley 2010]. Lin [2013] conducted online learning of a non-parametric mixture model using sequential variational inference. A key distinction between our work and these approaches is that we leverage the connection between empirical Bayes in a hierarchical model and gradient-based meta-learning [Grant et al., 2018] to use a MAML-like [Finn et al., 2017] objective as a log posterior surrogate. This allows our algorithm to make use of a scalable stochastic gradient descent optimizer instead of alternating a maximization step with an inference pass over the full dataset [c.f., Srivastava and Salakhutdinov 2013, Bauer et al., 2017].

Our approach is also distinct from recent work on gradient-based clustering [Greff et al., 2017] since we employ the episodic batching of [Vinyals et al., 2016]. This can be a challenging setting for a clustering algorithm, as the assignments need to be computed using, for example, $K = 1$ examples per class in the 1-shot setting.

Contrasting the batch and stochastic settings. In the stochastic setting, access to past data is unavailable, and so none of the standard algorithms and heuristics for inference in non-parametric models are applicable [e.g., Jain and Neal, 2004, Hughes et al., 2012]. In particular, our proposed algorithm does not refine the cluster assignments of previously observed points by way of multiple expensive passes over the whole dataset.

In contrast, we incrementally infer model parameters and add components during episodic training based on noisy estimates of the gradients of the marginal log-likelihood. Moreover, we avoid the need to preserve task assignments, which is potentially harmful due to stale parameter values, since the task assignments in our framework are meant to be easily reconstructed on-the-fly using the E–STEP with updated parameters $\theta^{(0)}, \ldots, \theta^{(L)}, G$.

Maximum a posteriori estimation as iterated conditional modes. Due to the high-dimensionality of the parameter set of neural networks, we consider a mode estimation procedure based on iterated conditional modes (ICM) [Besag, 1986, Zhang et al., 2001, Welling and Kurihara, 2006, Raykov et al., 2016] that can leverage gradient computa-
tion instead of the expensive process of Gibbs sampling. Iterated conditional modes (ICM) is a greedy strategy that iteratively maximizes the full conditional distribution for each variable (i.e., computes the MAP estimate), instead of sampling from the conditional as is done in Gibbs sampling [Welling and Kurihara 2006]. This leads to a fast point-estimation of the DPMM parameters in which we only need to track the means of the cluster priors.

Alternative inference procedures in probabilistic mixtures. A standard approach for estimation in latent variable models, such as probabilistic mixtures, is to represent the distribution using samples produced via some sampling algorithm. The most widely used is the Gibbs sampler [Neal 2000, Gershman and Blei 2012], which draws from the conditional distribution of each latent variable, given the others, until convergence to the posterior distribution over all the latents. However, in the setting of latent variables defined over high-dimensional parameter spaces such as those of neural network models, using a sampling algorithm such as the Gibbs sampler is prohibitively expensive [Neal 2012, MacEachern and Müller 1998]. Instead of sampling, one can fit factorized variational distributions to the exact distribution \( p(\phi, z|x) \approx q(\phi)q(z) \) [Ghahramani and Beal 2000, Blei et al. 2006]. It should be noted that we do not claim that our method of point estimation in the DPMM is the most accurate method for posterior inference but we leave improved approximate inference extensions to future work.

The main drawback of using point estimates for a non-parametric mixture estimation is the inability to leverage the diffusion of the global prior \( G_0 \) when computing the likelihood of a new cluster. Highly concentrated parameter estimates for non-empty clusters should lead to low likelihoods for outlier tasks, whereas the diffused global prior should be better at capturing a wider variety of tasks. Nonetheless, point estimation is a necessary trade-off between computation and accuracy. To allow for a more accurate estimate of the likelihood, we experimented with simulating a normal centered at the global prior mean with a variance hyperparameter that can be annealed over time to account for increased certainty about the prior choice. We can then compare the average cluster responsibility to the threshold. Another interesting extension we experimented with was to compute the gradient for each
of the samples and average over the number of samples as to approximate the expectation of the gradient under the global prior. However, we found this to be less stable than simply comparing the cluster responsibilities to the threshold.

### 3.8 Conclusion

Meta-learning is a source of learned inductive bias. Occasionally, this inductive bias is harmful because the experience gained from solving a task does not transfer. Here, we present an approach that allows a probabilistic meta-learner to explicitly modulate the amount of transfer between tasks, as well as to adapt its parameter dimensionality when the underlying task distribution evolves. We formulate this as probabilistic inference in a mixture model that defines a clustering of task-specific parameters. To ensure scalability, we make use of the recent connection between gradient-based meta-learning and hierarchical Bayes [Grant et al., 2018] to perform approximate *maximum a posteriori* (MAP) inference in both a finite and an infinite mixture model. Our work is a first step towards more realistic settings of diverse task distributions, and crucially, *task-agnostic* continual meta-learning. The approach stands to benefit from orthogonal improvements in posterior inference beyond MAP estimation (*e.g.*, variational inference [Jordan et al., 1999], Laplace approximation [MacKay, 1992a], or stochastic gradient Markov chain Monte Carlo [Metropolis and Ulam, 1949]), as well as scaling up the base model (*e.g.*, trading the four-layer convolutional network for a more complex architecture).
3.9 Appendix

3.9.1 Maximum a Posteriori Estimation in the Dirichlet Process Mixture Model

From Equation [3.5] and using a conditional mode estimate for task-specific parameters \( \phi_j \),

\[
\log p \left( z_j = \ell \mid x_{j_1:M}, z_{1:j-1}, \theta^{(\ell)} \right) \approx \begin{cases} 
\log n^{(\ell)} + \log p(x_{j_1:M} | \hat{\phi}_j^{(\ell)}) + \\
\log p(\hat{\phi}_j^{(\ell)} | \theta^{(\ell)}) \\
\log \zeta + \log p(x_{j_1:M} | \hat{\phi}_j^{(\ell)}) + \\
\log(\hat{\phi}_j^{(\ell)} | \theta^{(0)}) 
\end{cases} \text{ for } \ell \leq L
\]

\[
\log n^{((L+1)} \text{ for } \ell = L + 1.
\]

(3.6)

3.9.2 Experimental Setup

3.9.2.1 Dataset Details

Few-shot Regression

- Polynomial wave [Figure 3.5a]:

\[ y = \sum_i a_i x^{p_i} \]

and \( a \sim U(-5.0, 5.0) \).

- Sinusoid wave [Figure 3.5b]:

\[ y = a \sin(x - \phi) \]

where \( \phi \sim U(0, \pi) \) and \( a \sim U(0.1, 5.0) \).
- Sawtooth wave (Figure 3.5c):

\[
y = -\frac{2a}{\pi} \arctan(\cot(\frac{x\pi}{\phi}))
\]

where \( \phi \sim U(0, \pi) \), \( a \sim U(0.1, 5.0) \).

### 3.9.2.2 Hyperparameter Choices

**MiniImageNet Few-shot Classification.** We use the same data split, neural network architecture, and hyperparameter values as in Finn et al. [2017] for common components. We use \( \tau = 1 \) for the softmax temperature and the same initialization as Finn et al. [2017] for the global prior \( G_0 \). We determine an iteration number for early stopping using the validation set.

**Continual few-shot regression.** Our architecture is a feedforward neural network with 2 hidden layers with ReLU nonlinearities, each of size 40. We use a meta-batch size of 10 tasks (both for the inner updates and the meta-gradient updates) for 5-shot regression. Our non-parametric algorithm starts with a single cluster (\( L_0 = 1 \) in Algorithm 3.4). In these experiments, we set the spawning threshold \( \epsilon = 0.95T/(L+1) \), with \( L \) the number of non-empty clusters and \( T \) the size of the meta-batch. We use the mean-squared error for each task as the inner loop and meta-level objectives.

**Continual few-shot miniImageNet classification.** We use the same data split, neural network architecture, and hyperparameter values as in Finn et al. [2017] for common components. We use a meta-batch size of 4 tasks, start with a single cluster, and set the spawning threshold to the same formula as in Section 3.9.2.2. We use the multi-class cross entropy error for each task as the inner loop and meta-level objectives. More details on the practical implementation for image datasets of the non-parametric algorithm can found in Section 3.9.3.
3.9.3 Practical and Implementational Details

3.9.3.1 Task-aware vs. task-agnostic

Since a cluster is not well-tuned immediately after its creation, we consider a cool-down period after the spawning of each new cluster where we do not consider the creation of new clusters for a fixed number of iterations, and we freeze the updating of existing clusters for a same number of iterations. This allows the newly-created cluster to take enough gradient updates in order to move from its global prior initialization, allowing it to sufficiently differentiate from the global prior.

This experimental paradigm also allows us to approximate the task-aware algorithms of prior work [e.g., Kirkpatrick et al, 2017, Zenke et al, 2017, Nguyen et al, 2017, Ritter et al, 2018] which require access to an explicit delineation between tasks that acts as a catalyst to grow model size. For the task-aware non-parametric mixture results reported in the experiments, we set this cool-down period to be exactly the length of the training phase for the appropriate dataset; therefore, clusters which are not meant to be specialized for the active dataset are not updated. In contrast, the task-aware results consider a cool-down period of 1k iterations, which is less than 15% of the active period for each dataset. Extensions to this fixed cool-down period could consider the rate of learning in the active cluster in order to detect when the new component has been sufficiently fit to the new task.

3.9.3.2 Practical extensions to the non-parametric algorithm

The penalty term of $\log n^{(t)}$ or $\log \zeta$ is necessary to regularize the likelihood of a potential new cluster in order to limit overspawning. However, in the setting where the likelihood is approximated by the loss function of a complex neural network, as in the case for most meta-learning applications, there is a large difference in orders of magnitude between the loss value (especially for the cross-entropy function) and the penalty term, even after a single batch of assignments. Furthermore, the classical log observation count $\log n$ term is misaligned with our stochastic setting for two reasons. First, since we do not re-evaluate
over the whole dataset for every meta-learning episode, we are thus more concerned with the relative number of task assignments over recent iterations than the total number of assignments over the duration of training. Second, the number of tasks to be assigned can grow too large in the stochastic setting (e.g. 60k for miniImageNet) which exacerbates the already large difference in orders of magnitudes between the loss function and the penalty term.

Accordingly, we propose two changes; First, we compute the observation based on a moving window of fixed size (5 in the experiments). Second, we apply a coefficient, which can be tuned, to the log observation count in (3.5). This provides more flexibility to our meta-learner as it allows it to apply to any black-box function approximator which might exhibit losses of orders of magnitudes smaller than those expected of classical probabilistic models. While the moving window size and CRP penalty coefficient terms are somewhat interdependent, we propose them as a simple starting point to tune this non-parametric meta-learner beyond what is empirically explored in this chapter.

Note that without such changes in the stochastic setting of meta-learning, a nonparametric algorithm would be unable to spawn a new cluster after the first handful of iterations. Even if we were to lower the threshold \( \epsilon \), multiple almost identical clusters would be spawned in the first few iterations before it would be impossible to spawn anymore. Furthermore, the clusters would be nearly identical given the small step size of a gradient update for each meta-learning episode. Finally, this would be computationally intensive since unlike the typical applications of non-parametric mixture learning where one can afford to spawn hundreds of components then prune them over the training procedure.

### 3.9.3.3 Thresholding

A marked difference that is not immediate from the Gibbs conditionals is the use of a threshold on the cluster responsibilities, detailed in the E-STEP in Subroutine 4, to account for noise from stochastic optimization when spawning a cluster on the basis of a single batch. This threshold is necessary for the stochastic mode estimation procedure of Algorithm 3.4.
as it ensures that a new cluster’s responsibility needs to exceed a certain value before being permanently added to the set of components.

If a cluster has close to an equal share of responsibilities as compared to existing clusters after accounting for the CRP penalty $\log n^{(i)}$ or $\log \zeta$, it is spawned. Accordingly, this approximate inference routine still preserves the preferential attachment (“rich-get-richer”) dynamics of Bayesian nonparametrics [Raykov et al., 2016]. A sequential approximation for non-parametric mixtures with a similar threshold was proposed in Lin [2013] and Tank et al. [2015], in which variational Bayes was used instead of point estimation in a DPMM.

3.9.3.4 Pruning heuristics

None of the results reported in our experiments used a pruning heuristic as we used a rather conservative hyper parameter setting that deters overspawning. We did however explore different heuristics which could work in more general settings, especially in the presence of many more latent clusters of tasks than considered in the experimental settings in this work. One such heuristic is to prune small clusters that have received disproportionately few assignments over a certain number of past iterations. Another is to evaluate the functional similarity of two clusters by computing an odds-ratio statistic for the assignment probabilities to each cluster over a set of validation tasks. If the odds-ratio statistic is below a certain threshold, the smaller cluster can be pruned.

3.9.3.5 Estimating the CRP hyperparameters

We fixed $\alpha$ at the size of the meta-batch. An alternative is to place a $\Gamma(1,1)$ on the concentration parameter. Based on the likelihood, the posterior is then proportional to $p(\alpha|N,K) \propto \frac{\Gamma(\alpha)}{\Gamma(\alpha+N)} \alpha^K e^{-\alpha}$ This is not a standard distribution but Rasmussen [2000] have shown that $\log p(\alpha|N,K)$ is log-concave and methods such as L-BFGS have been used successfully in prior works. Alternatively, if we have some prior knowledge about the expected number of clusters, we can compute $\alpha$ based on $E[K] = \alpha \log N$. For the window-size, we
considered an initial size of 20 iterations that can grow as more cluster are considered.

3.9.3.6 Implementation details

We implemented both of our parametric and non-parametric meta-learners in TensorFlow (TF) \cite{abadi2016tensorflow}. We considered 2 different settings for the M-STEP optimization:

- Train each cluster’s parameters separately based on its corresponding loss function in an alternating manner closest to the classic EM algorithm.
- Train all cluster weights simultaneously using a surrogate loss over all validation batches.

Since the latter better leverages the differentiability of softmax-clustering and performed better empirically, we used it to report all experimental results.

**Nonparametric Implementation** For the nonparametric algorithm, we chose the first approach to the M-STEP by constructing separate optimizers for each cluster’s parameters. We pre-allocate a set of weights and use a mask during training to discard the parameters of empty clusters due to the static nature of TF graphs. When the algorithm exhausts the set of pre-allocated weights, we simply construct more network weight and reinitialize our optimizers.

**CRP global prior** The likelihood of a new cluster is sensitive to the choice of a base measure or prior prior, \( G_0 \) on the cluster hyperparameters. Our gradient-based point estimation does not make any modeling assumption on the distribution of the weights, rendering the problem of principally updating the base measure, after or during training, non-trivial. We chose to initialize all weights with zero-mean normals in the fully-connected layers. For the convolutional layers, we leveraged Xavier initialization \cite{glorot2010understanding} similarly to prior work \cite{finn2017model} in meta-learning.
However, such initialization is poor in the non-parametric for most non-trivial regression or classification tasks. Therefore, in the nonparametric setting, we start with a single cluster for a fixed number of iterations. We then initialize all clusters with the weights of the first clusters. This set of weights can be considered as the mean of the base measure or global prior in our setting.

We periodically update the global prior using a uniform average of the parameters of the existing clusters. This can be done by simply averaging over the parameter of the non-empty clusters as weighted by their sizes. Note that, we found that performing weighted KDE smoothing with a small bandwidth hyperparameter to perform slightly better than the average which is to be expected for neural network parameters. The number of iterations between updates of the global prior is a hyperparameter that we tune on the validation set. It is also possible to continuously, but less frequently over time, update this global prior as more data is encountered.
Variational Refinement for Importance Sampling via Forward KL Optimization

4.1 Introduction

Bayesian analysis provides a principled framework to encode complex hierarchical structures and prior beliefs in order to capture posterior uncertainty about latent variables $\theta$ given observed data $x$ via the posterior $p(\theta|x)$. The inferential goal often involves computing expectations over this posterior distribution, $\mathbb{E}_{\theta \sim p(\theta|x)}[f(\theta)]$, which is typically accomplished by sampling. Unfortunately, sampling directly from the posterior is usually intractable. Computing posterior functionals thus requires approximate inference methods such as variational inference (VI) [Jordan et al., 1999, Wainwright et al., 2008], Markov Chain Monte Carlo (MCMC) [Brooks et al., 2011, Andrieu et al., 2003], and importance sampling (IS) [Gelman and Meng, 1998], among others.

Recently, variational inference has grown in popularity because it recasts the inference problem as an optimization problem that can leverage recent advances in stochastic optimization [Bottou, 2010, Hoffman et al., 2013] and automatic differentiation [Maclaurin et al., 2015]. Specifically, VI poses a tractable family of distributions $Q$ and minimizes the reverse Kullback-Leibler divergence (RKL) between $q$ and $p$, i.e. $\text{KL}(q||p)$. However, $Q$ is often misspecified leading to unknown bias in VI solutions [Blei et al., 2016]. It is also
generally difficult to assess the quality of a VI approximation on downstream tasks based on the value of the RKL divergence \cite{YaoEtAl2018, CampbellAndLi2019, RainforthEtAl2018, HugginsEtAl2020}. This motivates the use of importance sampling (IS) to de-bias posterior summaries regardless of the misspecification of $Q$ \cite{GelmanMeng1998, Owen2013}. Unlike the RKL, the performance of $q$ as an IS proposal is indicative of its quality as an approximation of $p$ on downstream tasks \cite{YaoEtAl2018}.

This workflow capitalizes on complementary strengths: the computational efficiency of VI can sidestep the challenge of selecting a proposal distribution for IS which, in turn, ensures the consistency of the refined posterior expectation. However, RKL minimization typically results in light tails which can cause instability for importance sampling \cite{DiengEtAl2017, YaoEtAl2018}. On the other hand, the forward KL divergence (FKL or $KL(p||q)$) is known to control the estimation error of importance sampling \cite{ChatterjeeDiaconis2018} but is rarely used due to its intractability.

In this chapter, we propose to replace reverse KL with forward KL as the variational objective whose minimization yields an optimal IS proposal distribution. We make four distinct contributions in this vein:

- We derive a self-normalized importance sampling estimate for the intractable FKL divergence.

- We demonstrate how FKL-based boosting can combine IS and VI for multimodal target distributions.

- We show that FKL boosting is guaranteed to converge at a rate of $O\left(\frac{1}{K}\right)$, where $K$ is the number of boosting iterations, to the best approximation from a family of mixture distributions. This immediately guarantees convergence to the optimal proposal distribution as per the results of \cite{ChatterjeeDiaconis2018}.

- We demonstrate empirically that our approach is competitive with state-of-the-art VI and Hamiltonian Monte Carlo on regression tasks over real datasets using Bayesian neural networks (BNNs) and Bayesian linear regression (BLR).
Our proposed algorithm is thus a principled inference technique, with a well-defined computation-quality trade-off, that can be used independently or as a refining step to correct for the error in a given approximation.

\[
\text{Minimize Reverse KL: } KL(q\|p) = E_q[\log \frac{q}{p}]
\]

\[
\text{Minimize Forward KL: } KL(p\|q) = E_p[\log \frac{p}{q}]
\]

**Figure 4.1**: Results of minimizing RKL vs. FKL for a Gaussian variational approximation on a bimodal target.

### 4.2 Background

Let \( \theta \) denote the variable of interest with probability density \( p \), and let \( f \) denote a function of \( \theta \). Our goal is to estimate an expectation

\[
E_{\theta \sim p(\theta | x)} [f(\theta)].
\]
In Bayesian inference, $\theta$ generally represents a latent variable to be integrated over in the posterior distribution, $p(\theta|x)$, conditioned on the observed data $x$.

If one can draw $S$ samples $\{\theta_s\}_{s=1}^S$ from the target distribution $p(\theta|x)$ then Eq. Equation (4.1) can be estimated by simple Monte Carlo integration. However, $p(\theta|x)$ is typically only known up to a normalization constant and thus cannot be readily sampled from. Instead, given samples from an approximation $q$ of the posterior, we can estimate the expectation as:

$$\mathbb{E}_{p(\theta|x)}[f(\theta)] \approx \frac{\sum_{i=1}^S f(\theta_s)w_s}{\sum_{i=1}^S w_s}.$$  

(4.2)

If the samples are weighted equally (i.e., $w_s = 1$), Eq. Equation (4.2) is equivalent to the VI estimate, which has low variance but can be biased and inconsistent [Owen, 2013].

If instead we weigh by the importance ratios, $w_s = \frac{p(\theta_s|x)}{q(\theta_s)}$, we recover the IS estimate:

$$\mathbb{E}_{q \sim p(\theta|x)}[f(\theta)] = \mathbb{E}_{q \sim q(\theta)} \left[ \frac{p(\theta|x)}{q(\theta)} f(\theta) \right],$$  

(4.3)

which is consistent (bias=$O(1/S)$) but with potentially large or infinite variance [Owen, 2013].

### 4.2.1 VI with Reverse KL Minimization

Variational inference posits a family $Q$ of distributions that are easy to evaluate or sample from, and defines the variational approximation $q^* \in Q$ as the distribution that minimizes the reverse Kullback-Leibler (RKL) divergence to the posterior $p$: $q^* = \text{argmin}_{q \in Q} \text{KL}(q||p)$.

Unless the choice of $Q$ specifically includes the target distribution, minimizing the reverse KL divergences leads to a $q^*$ that underestimates the target covariance. The decomposition of this divergence sheds light on the cause of covariance underestimation when minimizing the RKL, regardless of the choice of family $Q$:

$$\text{KL}(q||p) = \mathbb{E}_q[\log q] - \mathbb{E}_q[\log p]$$  

(4.4)
The first term in Eq. (4.4) is the entropy of $q$ whose penalization is known to cause light tails. Furthermore, the second term is minimized when $q = 0$ for $p > 0$, leading to zero forcing or over-pruning [Higgins et al., 2016]. The RKL divergence thus favors a single mode (mode seeking), is biased towards avoiding false positives, and poses difficulties when approximating heavy-tailed [Guo et al., 2016] [Li and Turner, 2016] [Dieng et al., 2017] or multimodal targets [Miller et al., 2017]. This effect is illustrated in Figure 4.1.

### 4.2.2 Importance Sampling and FKL

Due to the unknown bias of an RKL-minimizing $q^*$, refining through importance weighting is recommended to de-bias the estimate of Eq. (4.1) [Yao et al., 2018] [Vehtari et al., 2015]. However, the light tails of this $q^*$ can lead to high or even infinite variance for an IS estimator which limits the efficiency of $q^*$ as an IS proposal distribution.

As an alternative to RKL, minimizing the forward KL divergence can mitigate the issue of covariance underestimation and tail undersampling. Consider the decomposition of the forward KL divergence:

$$KL(p\|q) = E_p[\log p] - E_p[\log q].$$

(4.5)

In Eq. (4.5), the cross-entropy is minimized by setting $q > 0$ whenever $p > 0$, which leads to mass covering as illustrated in Figure 4.1. With better tail coverage, an FKL-minimizing $q^*$ can yield an IS estimate (Eq. (4.3)) with lower variance than an approximation generated by RKL. Indeed, Chatterjee and Diaconis [2018] demonstrate that the variance of an importance sampling estimate scales as $O\left(\frac{\text{KL}(p\|q)^2}{\sqrt{S}}\right)$, where the number of samples $S$ required for IS to provide accurate mean estimates scales exponentially with FKL.
4.3 Related work

We present related work on improving estimates of the expectation over intractable target distributions (e.g. high dimensional, heavy tailed, or multimodal).

4.3.1 Variational Inference Divergences

Prior work has addressed the covariance underestimation and light tails pathologies of reverse KL minimization while seeking to improve the quality of the approximation as an IS proposal through the minimization of alternative divergences such as (reversed) Renyi-$\alpha$ [Li and Turner, 2016, Hernández-González and Cerquides, 2020], Chi-square ($\alpha = 2$) [Dieng et al., 2017], or Hellinger ($\alpha = 1/2$) [Campbell and Li, 2019] divergences. FKL can be seen as a special case of $\alpha$ divergences when $\alpha \rightarrow 1$ which is not considered in any of these prior VI works.

4.3.2 FKL for Approximate Inference

While the forward KL’s computational inconvenience has limited its use for variational inference, inference techniques such as Belief Propagation (BP) and Expectation Propagation (EP) can be regarded as performing FKL minimization locally [Minka et al., 2005]: KL is minimized one data partition at a time instead of globally as in VI. Minka [2001] demonstrates that this local minimization procedure is not guaranteed to converge and may not result in representative posteriors. Another set of techniques that utilize variants of the FKL divergence includes reweighted wake-sleep [Bornschein and Bengio, 2014] which alternates minimizing an approximation of FKL during the sleep phase while minimizing an approximation of RKL during the wake phase. However, this is known to lead to a biased estimator [Bornschein and Bengio, 2014].
4.3.3 Variational Boosting

Variational boosting (VB) \cite{Miller2017,Guo2016,Locatello2018,Jerfel2017} has been suggested in various forms to address the multimodality challenge for variational inference. Variational boosting posits a family of mixture distributions $Q_k$:

$$Q_k = \left\{ q : q(\theta) = \sum_{i=1}^{k} \lambda_i f_i(\theta), \lambda \in \Delta_k \right\}, \quad (4.6)$$

and sequentially constructs a variational mixture approximation by adding and re-weighting one (typically Gaussian) mixture component at a time to minimize the KL objective:

$$\{\mu_i\}_{i=1}^{k}, \{\Sigma_i\}_{i=1}^{k}, \{\lambda_i\}_{i=1}^{k} \leftarrow \arg\min_{\mu, \Sigma, \lambda} \text{KL} \left( \sum_{i=1}^{k} \lambda_i f_i(\theta; \mu_i, \Sigma_i) \| p(\theta|\mathbf{x}) \right). \quad (4.7)$$

This form of reverse KL-based boosting is known to struggle with degeneracy where the optimization at certain boosting iterations can lead to point-mass mixture components \cite{Campbell2019}. Ad-hoc regularization techniques are often needed in practice \cite{Locatello2018}, but are not necessarily sufficient \cite{Campbell2019}. To address this pathology, \cite{Campbell2019} proposed a boosting algorithm, based on the Hellinger divergence, which does not not guarantee scalability with dimensions.

4.3.4 Adaptive Importance Sampling

Adaptive IS (AIS) methods such as Adaptive multiple IS \cite{Cornuet2012} and incremental mixture IS \cite{Raftery2010} are designed for multimodal targets. However, none of the existing works directly optimize for the FKL divergence which controls the worst case IS estimation error. For example, \cite{Capp2008} minimize an entropy criterion whereas \cite{Douc2007} minimize the empirical variance of the importance weights, which does not necessarily correlate with the quality of IS estimation \cite{Vehtari2015}.
4.3.5 Combining IS and VI

As outlined in Table 4.1, VI and VB suffer from covariance underestimation, and can struggle to approximate heavy-tailed distributions [Blei et al., 2016]. AIS, on the other hand, can approximate heavy-tailed targets but cannot scale efficiently in dimensionality [Owen, 2013]. A combination of these two lines of research may benefit from their complementary strengths while sidestepping shared weaknesses (e.g., multimodality). However, it is often difficult to combine optimization-based and sampling-based inference techniques. This is because sampling methods such as MCMC define the approximate distribution implicitly such that its density cannot be evaluated. This has driven the development of alternatives to the KL divergence such as the variational contrastive divergence [Ruiz and Titsias, 2019]. However, we are not aware of similar work for IS that leverages the computational efficiency of VI through a unifying loss such as the FKL divergence.

Other prior proposals combining IS and VI [Domke and Sheldon, 2018] have focused on minimizing the RKL, and thus do not inherently capture heavier tails of the target distribution. [Prangle, 2019] recently presented concurrent work on combining IS and VI. However, their method relies on normalizing flows for constructing the proposal distribution such that it does not guarantee a multimodal approximation. [Ramos-López et al., 2018] uses a stream of IS weights to fit parameters for Gaussian mixture posteriors to minimize the FKL divergence. However, unlike our work, they assume access to normalized importance weights instead of samples.

Table 4.1: Comparing approximate inference techniques.

<table>
<thead>
<tr>
<th>Failure Mode</th>
<th>VI</th>
<th>IS</th>
<th>VB</th>
<th>AIS</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multimodality</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Heavy tails</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cov estimation</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>High dimensions</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗*</td>
<td>✓</td>
</tr>
</tbody>
</table>
4.4 Methodology

We develop our novel approach to integrate variational inference and importance sampling using the forward KL divergence with a focus on multimodal targets.

Note that we assume the target density \( p \) and the approximation \( q \) share the same support which can be \( \mathbb{R}^d \) or a subset thereof. This guarantees that \( p \) is absolutely-continuous with respect to \( q \) (noted as \( p \ll q \)) and vice-versa which is necessary for the definition of the KL divergence.

4.4.1 Forward KL Variational Approximation for IS

A theoretically reasonable desire is for the proposal distribution to minimize the forward KL divergence. However, we cannot compute FKL exactly for unnormalized target distributions. This stems from the expectation under the target \( p \) in Eq. [Equation (4.5)].

By contrast, the reverse KL takes the expectation in Eq. [Equation (4.4)] under a normalized approximation \( q \). This can be seen as a tractable approximation to FKL with unknown bias [Yao et al., 2018]. In fact, for misspecified choices of \( Q \) (i.e. when \( \text{KL}(q^*||p) \neq 0 \)), [Campbell and Li, 2019] demonstrate that minimizing the RKL is not guaranteed to minimize the FKL divergence. Furthermore, the light tails of the RKL minimization solution renders it inadequate for IS.

Alternatively, towards deriving a consistent approximation of the forward KL divergence, we rearrange densities inside the expectation as follows:

\[
\text{KL}(p||q) = \mathbb{E}_p \left[ \log \left( \frac{p}{q} \right) \right] = \mathbb{E}_q \left[ \frac{p}{q} \log \left( \frac{p}{q} \right) \right]. \tag{4.8}
\]

FKL can then be approximated through self-normalized importance sampling (SNIS) [Pran-].
which is known to be consistent:

\[
\theta_s \sim q(\theta) \quad r_s = \frac{p(\theta_s|x)}{q(\theta_s)} \quad w_s = \frac{r_s}{\sum_{s=1}^{S} r_s} \quad KL(p||q) = \sum_{s=1}^{S} w_s \cdot \log \left( \frac{p(\theta_s|x)}{q(\theta_s)} \right).
\] (4.9)

The SNIS estimation can have arbitrarily high variance depending on \( q \). Gradients of Eq. [Equation 4.8] with respect to the parameters of \( q_i \) can also have high variance since we optimize over the same distribution from which samples are drawn. For certain distributions, the reparametrization trick can reduce this variance [Kingma and Welling 2014]. Furthermore, the sequential setting described in section 4.4.3 contributes to reducing the gradient variance.

### 4.4.2 Forward KL Boosting

We sequentially construct a proposal mixture distribution \( q \) that is both easy to sample from (lower SNIS bias) and minimizes the FKL objective for efficient IS estimation.

Given the computational convenience of Gaussian distributions, we set \( Q \) to be the family of Gaussian mixtures in Section 4.7 such that the proposal at the \( K \)th boosting iteration is

\[
q_K(\theta) := \sum_{i=1}^{K} \lambda_i f_i(\theta; \phi_i) \quad \text{where} \quad f_i = \mathcal{N}(\mu_i, \Sigma_i).
\]

Our framework also supports mixtures of heavy-tailed distributions which are desirable for adaptive IS [Geweke 1989].

At each boosting iteration, we minimize the FKL between the mixture \( q_i \) and the target \( p \) by fitting a new component \( f_i \) and a mixture weight \( \lambda_i = \gamma \) while holding the parameters of previously-learned mixture components fixed:

\[
\arg\min_{q_i \in Q} KL(p||q_i) = \arg\min_{f_i, \gamma} KL(p||\gamma f_i + (1 - \gamma)q_{i-1})
\]

This is known to be more efficient and stable than the joint optimization of all mixture components at each iteration [Locatello et al. 2018]. The mixture weights for the fixed
mixture components \(\{\lambda_j\}_{j=1}^{i-1}\) are re-scaled by \((1 - \gamma)\) and can be further adjusted with a fully-corrective weight search \cite{Locatello2018} using the gradient derived in Eq. \text{Equation [4.23]}. Fitting the mixture weights is a convex problem that can benefit from higher-order optimizers such as Newton’s. We provide an alternative approach to building \(q_K\) using the remainder distribution in Appendix 4.10.3.

4.4.3 Lower Variance SNIS with Boosting

The main computational concern in estimating the FKL divergence with SNIS is the variance of importance sampling in Eq. \text{Equation (4.8)}. However, the sequential construction described above enables further reformulations of the SNIS approximation which can lead to a lower variance than the naive approximation of Eq. \text{Equation (4.8)}. In fact, the global objective can be re-written as:

\[
\arg\min_{q_i \in \mathcal{Q}} \text{KL}(p\|q_i) = \arg\min_{q_i \in \mathcal{Q}} \mathbb{E}_p \left[ \log \frac{p}{q_i} \right] = \arg\min_{f_i, \gamma} \mathbb{E}_{q_{i-1}} \left[ \frac{p}{q_{i-1}} \log \frac{p}{\gamma f_i + (1 - \gamma)q_{i-1}} \right]
\]

\text{Equation (4.12)} which is only possible in a sequential setting such as ours, reduces the gradient variance since the component being estimated, \(f_i\), is independent of the distribution \(q_{i-1}\) from which samples are drawn to approximate the FKL. Furthermore, we can draw a large number of samples from \(q_{i-1}\) a single time at the start of each boosting iteration.

The SNIS approximation of Eq. \text{Equation (4.12)} given samples \(\theta_s\) from \(q_{i-1}\) can then be computed as:

\[
\sum_{s=1}^{S} w_s [\log p(\theta_s) - \log (\lambda f_i(\theta_s) + (1 - \lambda)q_{i-1}(\theta_s))]
\]
where $w_s$ are computed as in Eq. [Equation 4.9].

**Trade-off of sampling $q_i$ or $q_{i-1}$:** While $q_i$ is expected to handle well-separated modes better, in our experiments [Equation 4.7] we found that using $q_{i-1}$ is sufficient for capturing distant modes (e.g. Fig. 4.4) while reducing the SNIS variance. Variance reduction was especially crucial on higher dimensional applications such as Bayesian NNs in Table. [4.3]

This objective also enjoys lower gradient variance since the distribution being sampled from is a single Gaussian component and not a multimodal proposal. However, the light tails of a single Gaussian component can lead to high variance SNIS estimates in the case of severe mismatch with the target distribution. Therefore, the choice of either should be guided by assumptions about the tails of the target. However, given comparable empirical results of the two approaches, we limit the analysis of Section [4.5] and the experiments of Section [4.7] to the first and more straightforward approach.

### 4.4.4 Initialize with RKL, Refine with FKL

For the first boosting iteration, we do not have an existing approximation to sample from. Instead, Eq. [Equation 4.12] can be re-written as follows:

$$
\text{argmin}_{f_i, \gamma} \mathbb{E}_{f_i} \left[ \frac{p}{f_i} \log \frac{p}{\gamma f_i + (1 - \gamma)q_{i-1}} \right].
$$

This exacerbates the sensitivity to the initialization of $f_i$. In fact, a sharply peaked initialization centered could limit the SNIS estimate from properly capturing the tail behavior. Therefore, a diffuse initialization is likely to be beneficial.

An even more practical initialization would use RKL-based VI to identify the mode of the target and provide a computationally efficient approximation that can be refined by FKL boosting in later iterations. As such, this FKL-boosting workflow can be considered as general framework for the iterative refinement of any given posterior approximation to be used for the estimation of expectations of interest. This best combines the strengths of
VI and IS as the first iteration of RKL minimization can reduce the variance of the SNIS approximation of FKL for the following iterations.

4.5 Analysis

We provide theoretical analysis of the proposed method of performing importance sampling with a proposal distribution constructed from FKL-based boosting.

4.5.1 FKL Controls Moment Estimation Error

Minimizing the FKL implicitly minimizes the error in posterior probabilities and moments via its control on total variational (through Pinsker’s inequality\cite{Tsybakov2008} and l-Wasserstein as follows\cite{Bolley2005}: Assume that the probability density $q$ is $n$-exponentially integrable. Then for target distribution $p$ such that $p \ll q$:

$$W_n(q,p) \leq C_n^{EI}(q) \left[ \text{KL}(p\|q)^{\frac{1}{n}} + \frac{\text{KL}(p\|q)^{\frac{1}{2n}}}{2} \right],$$

(4.13)

where

$$C_n^{EI}(q) = \inf_{x_0 \in \mathbb{R}^d, \alpha > 0} \left( \frac{3}{\alpha} + \frac{2}{\alpha} \log \int e^{\alpha \|x-x_0\|^n} q(x)dx \right).$$

This implies the convergence of the first $n$ moments.

Note that because of the symmetry of Wasserstein distances, we can switch probability densities $p$ and $q$ in the above inequalities and obtain bounds in terms of RKL. However, that would incur the $n$-exponential integrability condition on the target probability $p$, which boils down to generalized sub-gaussianity of its tail.

The $n$-exponential integrability assumption is not required of the target density $p$ in the case of Eq.\cite{Equation4.13}. Instead, it is only required of the family of variational approximations $q$ which is easier to enforce and verify (and is automatically satisfied by the mixture of Gaussians). A smaller constant $C_n^{EI}$ can also be achieved by the same reasoning.
4.5.2 FKL Boosting Converges at $O(1/K)$

Assuming $p \ll q$, which can be verified by the design of $Q$, the functional gradient of the forward KL divergence is derived in Appendix 4.10.2.5 as $\frac{\delta \text{KL}(p||q)}{\delta q} = -\frac{p}{q}$. The convexity of $\text{KL}(p||q)$ in $q$ is well established in the literature (proven with the log-sum inequality).

Furthermore, we show in Appendix 4.5.2 that the FKL functional is also $\beta$-smooth in $q$ where $\beta$ depends on the range of the values that the density $q$ can take. This requires bounding $q$ away from zero and from above which is typical in prior theoretical work [Guo et al., 2016] and aligns with practice as it can translate to a bounded parameter space for a given family of distributions.

For a convex and strongly smooth functional, the greedy sequential approximation framework of [Zhang, 2003] provides an asymptotic guarantee for the convergence to a target distribution in the convex hull of a given base family such that the approximation error at the $K^{th}$ iteration is $\text{KL}(p||q_K) = \text{KL} \left( p \parallel \sum_{i=1}^{K} \lambda_i f_i \right) = O(1/K)$. This framework does not require each iteration to exactly solve for the optimal mixture component which can be difficult for the non-convex optimization sub-problems.

4.5.3 Computation-Quality Trade-Off

While our iterative algorithm is guaranteed to converge asymptotically to the optimal proposal distribution, we can identify three sources of approximation error: the variational inference error, the SNIS approximation bias, and the greedy sequential approximation error which depend on the number of VI iterations, IS samples, and boosting iterations, respectively. These three errors thus finely control the compute-quality trade-off of our framework. We analyze these tradeoffs in more depth through simulation experiments below.
4.5.4 The Dependence of the IS Proposal on the Integrand

Similarly to prior empirical [Owen, 2013] and theoretical [Chatterjee and Diaconis, 2018] works, we do not address any assumptions about the function $f$ being integrated. However, the optimal proposal $q^* \propto |f| \cdot p$ when $p$ is normalizable and $q^* \propto |f - I| \cdot p$ otherwise [Kahn and Marshall, 1953]. Nonetheless, we only require that the integrand $f$ does not contain any singularities and shares the same support as $p$ and $q$. In this case, a simple rearrangement of the terms inside the expectation (4.1) would imply that $q$ should approximate $f \cdot p$ instead of $p$ where $f$ is the integrand. Therefore, characteristics of $f$ have no effect on our asymptotic guarantees or general methodology.

4.6 Simulation Experiments

Using two illustrative simulation experiments, we provide further intuition for the behavior and performance of the proposed methodology using the FKL.

4.6.1 Simulation 1: Cauchy

First, we demonstrate the aforementioned computation-quality trade-offs using a standard Cauchy target distribution. For intuition on the behavior of boosting using the FKL, Figure 4.3 illustrates the density plots of the target $p$ and the boosting approximation $q$ after various iterations. From previous sections, we observe a trade-off between sample complexity of SNIS and the optimization complexity of variational boosting. Inclusion of more mixture components $K$ and more accurate optimization in the variational boosting steps can save exponentially many samples in SNIS. However, there is a diminishing gain in increasing $K$. We demonstrate in Fig. 4.2 this effect: both FKL boosting and RKL boosting decreases forward KL divergence as more variational components are added. The decrease slows down significantly after inclusion of 3 mixture components. We therefore introduce in the experiments
up to 3 mixture components and select the best performance on validation data set. From Fig. 4.2 and the experimental results, we observe uniform improvements of FKL over RKL methods.

**Figure 4.2:** A comparison in terms of the FKL divergence to a Cauchy target distribution over the course of variational boosting using the FKL and RKL divergences.

**Figure 4.3:** Log density plots of the result \( \log q_k \) (blue, solid) of FKL boosting on the Cauchy target \( \log p \) (red, solid). The log-residual \( \log p/q \) (green, dashed) is indicative of the IS variance (from left to right: boosting iterations 1, 5, and 10).

### 4.6.2 Simulation 2: Well-separated Modes

We next demonstrate the performance of boosting with the FKL on a distribution with a large number of well-separated modes. We set as the target distribution a 2-dimensional Gaussian mixture model (GMM) with 20 components, previously used by [Ma et al. 2019](#).
The log-residual in Fig. 4.4 demonstrates the sequential improvement of our approximation. Moment estimation results for this experiment can be found in Fig. 4.5 in the Appendix. Overall, boosting with the FKL effectively improves both moment estimation and the actual FKL, and outperforms the RKL as the number of boosting iterations increases.

Figure 4.4: Log residual (log \( p/q_k \)) plots of for FKL boosting on a 2-dimensional GMM of 20 components [Ma et al., 2019].

4.7 Real Data Experiments

We evaluate the performance of the proposed method when applied to Bayesian linear regression (BLR) and Bayesian neural networks (BNNs) using a Gaussian prior and a heavy tailed prior. We use four datasets from UCI [Dua and Graff, 2017] (Table 4.5 in Appendix 4.10.5). We split each dataset into twenty randomly drawn 90%/10% train/test splits, which we denote \( D_{\text{train}} = \{x_i, y_i\}_{i=1}^{N_{\text{train}}} \) and \( D_{\text{test}} = \{x_i, y_i\}_{i=1}^{N_{\text{test}}} \), with input \( x_i \) and output \( y_i \). We report the mean and std. dev. of results over all splits.

We demonstrate our proposed method with \( K = 1, 2, \) and \( 3 \). We refer to fitting a single Gaussian, or \( K = 1 \), as FKL VI. For runs with more than one boosting iteration, we initialize the first component using the RKL, and optimize subsequent iterations using the FKL (as described in Section 4.4.4).

**Optimization details:** We use the ADAM optimizer [Kingma and Ba, 2014a] for each boosting iteration with a fixed learning rate and compute gradients based on a fixed number
of samples using Autograd [Maclaurin et al., 2015]. At the end of each boosting iteration, mixture weights are fully re-optimized using simplex-projected gradient descent [Bubeck, 2014] based on the analytical gradient in Eq. Equation (4.23). Details about practical considerations and hyperparameters can be found in Appendix 4.10.5, and we include our code with the submission.

**Parameter initialization:** We follow standard practice with similar experiments (see e.g. [Miller et al., 2017]) where the means of each component are initialized at zero and the diagonal elements of the initial covariance matrix are drawn from $\mathcal{N}(0,\sigma^2)$ ($\sigma$ is tuned as a hyperparameter, see Appendix 4.10.5.2). At the start of each boosting iteration, we further apply an initialization heuristic which approximates the mode of the residual by gradient descent on the remainder density. This requires a single sample per gradient step and is run for 400 steps. Note that this initialization is significantly simpler and less compute-intensive than prior approaches which include a weighted EM algorithm [Miller et al., 2017] and a random search over 10,000 samples or more [Campbell and Li, 2019].

**Comparisons:** We compare our approach to variational inference (RKL VI) and variational boosting with 2 and 3 boosting iterations (RKL VB) [Miller et al., 2017]. For the comparison to RKL VI and RKL VB, we use the same parametrization, initialization, and optimization techniques as for FKL VI and FKL VB. This might lead to discrepancies compared to the published results [Miller et al., 2017]; however, keeping these details consistent better disentangles the effect of the RKL vs. FKL optimization.

We also compare to directly sampling from the target distribution using Hamiltonian Monte Carlo (HMC) [Neal, 2011], implemented using the TensorFlow Probability library [Dillon et al., 2017]. We additionally ran 3 HMC chains in parallel and averaged the results, similar to [Hoffman and Ma, 2020]. Results were comparable between 1 and 3 HMC chains, and we include the results for 3 chains in Appendix 4.10.5.
Table 4.2: Predictive log probabilities on test for BLR with Gaussian prior (mean ± standard error over 20 train/test splits).

<table>
<thead>
<tr>
<th>Method</th>
<th>Wine ($d = 14$)</th>
<th>Boston ($d = 16$)</th>
<th>Concrete ($d = 11$)</th>
<th>Power ($d = 7$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMC</td>
<td>-1.002 (± 0.012)</td>
<td>-2.923 (± 0.035)</td>
<td>-3.780 (± 0.013)</td>
<td>-2.923 (± 0.006)</td>
</tr>
<tr>
<td>RKL VI</td>
<td>-1.013 (± 0.013)</td>
<td>-2.947 (± 0.035)</td>
<td>-3.798 (± 0.013)</td>
<td>-2.921 (± 0.006)</td>
</tr>
<tr>
<td>RKL VB 2</td>
<td>-1.014 (± 0.011)</td>
<td>-2.924 (± 0.033)</td>
<td>-3.878 (± 0.014)</td>
<td>-3.067 (± 0.003)</td>
</tr>
<tr>
<td>RKL VB 3</td>
<td>-1.007 (± 0.012)</td>
<td>-2.945 (± 0.039)</td>
<td>-3.800 (± 0.015)</td>
<td>-3.073 (± 0.003)</td>
</tr>
<tr>
<td>FKL VI</td>
<td>-0.998 (± 0.012)</td>
<td>-2.905 (± 0.036)</td>
<td>-3.775 (± 0.014)</td>
<td>-2.920 (± 0.005)</td>
</tr>
<tr>
<td>FKL VB 2</td>
<td>-0.998 (± 0.012)</td>
<td>-2.906 (± 0.036)</td>
<td>-3.762 (± 0.013)</td>
<td>-2.921 (± 0.006)</td>
</tr>
<tr>
<td>FKL VB 3</td>
<td>-0.998 (± 0.013)</td>
<td>-2.904 (± 0.036)</td>
<td>-3.762 (± 0.013)</td>
<td>-2.921 (± 0.006)</td>
</tr>
</tbody>
</table>

Table 4.3: Predictive log probabilities on test for BNNs with Gaussian prior (mean ± standard error over 20 train/test splits).

<table>
<thead>
<tr>
<th>Method</th>
<th>Wine ($d = 653$)</th>
<th>Boston ($d = 753$)</th>
<th>Concrete ($d = 503$)</th>
<th>Power ($d = 303$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMC</td>
<td>-0.990 (± 0.014)</td>
<td>-2.709 (± 0.101)</td>
<td>-3.281 (± 0.017)</td>
<td>-2.817 (± 0.007)</td>
</tr>
<tr>
<td>RKL VI</td>
<td>-0.993 (± 0.014)</td>
<td>-2.858 (± 0.020)</td>
<td>-3.230 (± 0.015)</td>
<td>-2.851 (± 0.008)</td>
</tr>
<tr>
<td>RKL VB 2</td>
<td>-0.990 (± 0.015)</td>
<td>-2.832 (± 0.018)</td>
<td>-3.253 (± 0.015)</td>
<td>-2.945 (± 0.009)</td>
</tr>
<tr>
<td>RKL VB 3</td>
<td>-0.981 (± 0.012)</td>
<td>-2.744 (± 0.011)</td>
<td>-3.255 (± 0.017)</td>
<td>-3.002 (± 0.012)</td>
</tr>
<tr>
<td>FKL VI</td>
<td>-0.991 (± 0.015)</td>
<td>-2.677 (± 0.011)</td>
<td>-3.328 (± 0.019)</td>
<td>-2.872 (± 0.012)</td>
</tr>
<tr>
<td>FKL VB 2</td>
<td>-0.979 (± 0.017)</td>
<td>-2.779 (± 0.012)</td>
<td>-3.193 (± 0.016)</td>
<td>-2.870 (± 0.008)</td>
</tr>
<tr>
<td>FKL VB 3</td>
<td>-0.967 (± 0.014)</td>
<td>-2.801 (± 0.012)</td>
<td>-3.192 (± 0.016)</td>
<td>-2.851 (± 0.009)</td>
</tr>
</tbody>
</table>

4.7.1 BLR and BNNs with Gaussian Priors

We follow the experimental setup of [Miller et al., 2017] for both BNNs and BLR. We place a Gaussian prior over each weight in the model, and an inverse Gamma prior on the variances:

\[
\alpha \sim \text{Gamma}(1, 0.1); \quad \tau \sim \text{Gamma}(1, 0.1);
\]

\[
w_i \sim \mathcal{N}(0, 1/\alpha); \quad y|x, w, \tau \sim \mathcal{N}(\phi(x, w), 1/\tau),
\]

where $w$ is the set of weights, and $\phi(x, w)$ is either a linear function of $x$ (BLR) or a multi-layer perception (BNN). For our BNNs, we set $\phi$ to be a one-hidden layer neural network with 50 hidden units and ReLU activation function, as done by [Miller et al., 2017] [Hernández-Lobato and Adams, 2015]. The full set of parameters that we sample is
\( \theta = (w, \alpha, \tau) \). We use the posterior predictive distribution to compute the distribution for a given new input \( x \):

\[
p(y|x, D_{\text{train}}) = \mathcal{L} \int p(y|x, \theta)p(\theta|D_{\text{train}})d\theta.
\] (4.14)

We use importance sampling to estimate this posterior predictive distribution given \( S \) samples \( \theta_s \sim q(\theta) \):

\[
p(y|x, D_{\text{train}}) \approx \frac{1}{S} \sum_{i=1}^{S} \frac{p(\theta_s|D_{\text{train}})}{q(\theta_s)} p(y|x, \theta_s),
\] (4.15)

where \( q(\theta) \) is the proposal distribution fit to \( p(\theta|D_{\text{train}}) \) using either forward KL refinement (our method, FKL VB) or reverse KL refinement (RKL VB, [Miller et al., 2017]).

Note that Miller et al. [2017] does not use importance sampling to estimate the posterior predictive distribution. We add importance weights here as an ablation to limit our analysis to the difference between FKL and RKL optimization. For completeness, we report the estimates without IS using RKL VB in Appendix 4.10.5. For comparing to HMC, we do not use importance sampling due to lack of an explicit density function, and instead compute (Eq. 4.14) by averaging over direct HMC samples from the posterior distribution \( p(\theta|D_{\text{train}}) \) (Eq. Equation 4.31 in Appendix 4.10.5).

As our final evaluation metric, we report the average predictive log probabilities on held-out test data:

\[
\frac{1}{|D_{\text{test}}|} \sum_{x,y \in D_{\text{test}}} \log p(y|x, D_{\text{train}}).
\] (4.16)

### 4.7.2 BLR with Heavy Tailed Priors

In addition to the Gaussian prior, we also perform Bayesian linear regression with a heavy tailed prior. Following Campbell and Li [2019] we place a \( T_2 \) prior on the weights. We use
Table 4.4: Predictive log probabilities on test for BLR with heavy tailed prior (mean ± standard error over 20 train/test splits).

<table>
<thead>
<tr>
<th>Method</th>
<th>Wine ($d = 13$)</th>
<th>Boston ($d = 15$)</th>
<th>Concrete ($d = 10$)</th>
<th>Power ($d = 6$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMC</td>
<td>-1.004 (± 0.012)</td>
<td>-2.962 (± 0.033)</td>
<td>-3.808 (± 0.026)</td>
<td><strong>-2.916 (± 0.005)</strong></td>
</tr>
<tr>
<td>RKL VI</td>
<td>-1.011 (± 0.013)</td>
<td>-2.924 (± 0.037)</td>
<td>-3.789 (± 0.013)</td>
<td>-2.924 (± 0.006)</td>
</tr>
<tr>
<td>RKL VB 2</td>
<td>-1.007 (± 0.014)</td>
<td>-2.944 (± 0.031)</td>
<td>-3.788 (± 0.012)</td>
<td>-3.028 (± 0.003)</td>
</tr>
<tr>
<td>RKL VB 3</td>
<td>-1.008 (± 0.012)</td>
<td>-2.940 (± 0.035)</td>
<td>-3.796 (± 0.014)</td>
<td>-3.005 (± 0.004)</td>
</tr>
<tr>
<td>FKL VI</td>
<td>-0.993 (± 0.013)</td>
<td><strong>-2.904 (± 0.036)</strong></td>
<td>-3.775 (± 0.014)</td>
<td>-2.940 (± 0.005)</td>
</tr>
<tr>
<td>FKL VB 2</td>
<td><strong>-0.973 (± 0.009)</strong></td>
<td>-2.907 (± 0.035)</td>
<td><strong>-3.773 (± 0.015)</strong></td>
<td>-2.921 (± 0.006)</td>
</tr>
<tr>
<td>FKL VB 3</td>
<td>-0.975 (± 0.009)</td>
<td>-2.906 (± 0.036)</td>
<td>-3.774 (± 0.015)</td>
<td>-2.922 (± 0.006)</td>
</tr>
</tbody>
</table>

the same inverse Gamma prior on the variance:

$$
\tau \sim \text{Gamma}(1, 0.1); \quad w \sim \mathcal{T}_2(0, A^T A);
$$

$$
y | x, w, \tau \sim \mathcal{N}(\phi(x, w), 1/\tau),
$$

where $A$ is fixed, and each entry is drawn i.i.d. before the optimization process: $A_{ij} \sim \mathcal{N}(0, 1)$. For these BLR experiments, $\phi(x, w)$ is a linear function of $x$ with weight parameters $w$. The full set of parameters that we sample is $\theta = (w, \tau)$. We estimate the same posterior predictive distribution in Eq. [Equation (4.14)] using IS in Eq. [Equation (4.15)] and report the average predictive log probabilities from Eq. [Equation (4.16)].

### 4.8 Discussion

Tables 4.2, 4.3, and 4.4 present the results on the UCI datasets for BLR with a Gaussian prior, BNNs with a Gaussian prior, and BLR with a heavy tailed prior, respectively. The lowest mean predictive log probability is highlighted in bold.

Minimizing the FKL divergence outperforms RKL across all four datasets and three experimental settings. This demonstrates the inadequacy of RKL-based VI for the construction of IS proposals. This should incentivize wider adoption of the FKL divergence, especially when the downstream task extends beyond simple prediction and requires a cal-
ibrated estimation of the posterior predictive distribution. FKL outperforming HMC, the gold standard of Bayesian inference, on three of the four datasets across all settings is another promising result.

However, we observe a decay in the held-out log-likelihood as more components are added for certain datasets (e.g. FKL VB with BNN on Boston). This is consistent with prior variational boosting results on the same datasets [Miller et al., 2017] and typically signals an over-fitting problem. Therefore, it is worth emphasizing that the convergence analysis of Section 4.5 is limited to optimization guarantees and does not extend to learning or generalization guarantees.

**Computational considerations** One main advantage of RKL methods is the low computational overhead, especially as compared to MCMC methods. In our experiments we observed that, even in high dimensions, there do not seem to be significant computational differences between optimizing the FKL objective and optimizing the RKL objective. In fact, our reported results compare RKL and FKL methods for the same number of IS samples and optimization iterations. See Appendix 4.10.5 for exact hyperparameter values and wall clock times.

### 4.9 Conclusion

Overall, we propose a principled algorithm that combines the strengths of importance sampling and variational inference to efficiently approximate multimodal and possibly heavy-tailed targets. Unlike prior work that relies on RKL, our minimization of FKL aligns with the analysis of the variance of IS [Chatterjee and Diaconis, 2018] which guarantees an optimal proposal distribution asymptotically. One challenge for this approach is the variance of the SNIS estimate of the forward KL divergence. Developing variance-reduction schemes for these types of objectives is an open research problem. Nonetheless, existing techniques for re-sampling or smoothing the importance weights can immediately apply to our proposed method.
4.10 Appendix

4.10.1 Further Methodology Discussion

4.10.1.1 Combining HMC and FKL VB

Using an MCMC method such as HMC can exploit the fact that the unnormlized density \( r_i \) gets shallower and less multimodal over iterations which makes it increasingly easy to sample from once, at the beginning of each boosting iteration, via techniques such as Hamiltonian Monte Carlo [Neal, 2011]. While we saw some promising preliminary performance with this method on real data experiments using Bayesian logistic regression (BLR), the higher dimensional experiments with Bayesian neural networks (BNN) struggled with numerical instability that would require further tuning of the HMC hyperparameters which include the number of burn-in steps, learning rate, and number of leapfrog steps, among others.

4.10.1.2 Stabilization of Likelihood Ratios

To avoid high-variance gradient estimates due to a mismatch between the target and the proposal, especially at the beginning of inference, we stabilize the importance weights in an unbiased way that parallels the log-sum-exp trick [Nielsen and Sun, 2016] in order to handle potential under- or overflow when exponentiating large values:

\[
\begin{aligned}
    r_s &= \frac{p(\theta_s|x)}{q_i(\theta_s)}, \\
    d_{\text{max}} &= \max_s (\log p(\theta_s|x) - \log q_i(\theta_s)) \\
    d_s &= \exp (\log p(\theta_s|x) - \log q_i(\theta_s) - d_{\text{max}}) \\
    w_s &= \frac{d_s}{\sum_{s=1}^{S} d_s} = \frac{r_s}{\sum_{s=1}^{S} r_s}
\end{aligned}
\]

As for the log residual \( \log p/q \) we introduce a biased stabilization heuristic that is typical

112
in variational boosting \cite{Guo2016, Campbell2019} with an \( \epsilon = e^{-10} \):

\[
\log \left( \frac{p(\theta|x) + \epsilon}{q_i(\theta_s) + \epsilon} \right) \approx \log \left( \frac{p(\theta|x)}{q_i(\theta_s)} \right)
\]

A range of variance reduction schemes is applicable to our method such as weight clipping, re-sampling and re-weighting. However, we leave that for future work.

### 4.10.2 Derivations

#### 4.10.2.1 Connecting Forward KL to Other Metrics Used for VI

By the monotonicity of Renyi-\( \alpha \) divergences, given that \( \lim_{\alpha \to 1} D_\alpha(p,q) = KL(p\|q) \), and from \cite{Dieng2017}:

\[
KL(p\|q) \leq D_2(p,q) \leq \chi^2(p,q)
\]

#### 4.10.2.2 Reverse KL Remainder: Intrinsic Entropy Regularization

Computing the remainder-reverse KL objective using our approach leads to the well-known although usually ad-hoc entropy regularization (e.g. \cite{Locatello2018}).

\[
KL(f_i\|r_i) = KL(f_i\|\frac{p}{q_{i-1}})
\]

\[
= \mathbb{E}_{f_i}[\log \frac{f_i q_{i-1}}{p}]
\]

\[
= \mathbb{E}_{f_i}[\log \frac{q_{i-1}}{p}] + \mathbb{E}_{f_i}[\log f_i]
\]

As we can see, while the first term is the mean of the log-residual under the new component \( f_i \), the typical objective for gradient boosting, the second term is the entropy of \( f_i \).
4.10.2.3 SNIS Derivation

Since we do not assume to know the normalization constant of \( p \), we shall approximate the above quantities by self-normalized importance sampling while making the distinction between the normalized \( p \) and the un-normalized \( \hat{p} \):

\[
\theta_s \sim q_{i-1}, \quad w^s = \frac{p(\theta_s)}{q_{i-1}(\theta_s)}, \quad w^s_{\text{norm}} = \frac{w^s}{\sum_s w^s} \tag{4.22}
\]

\[
E_{q_{i-1}} \left[ \frac{p}{q_{i-1}} \log \frac{p}{\lambda f_i + (1 - \lambda) q_{i-1}} \right] = E_{q_{i-1}} \left[ \frac{\hat{p}}{q_{i-1}} \log \frac{p}{\lambda f_i + (1 - \lambda) q_{i-1}} \right] \approx \sum_s \frac{p(\theta_s)}{q_{i-1}(\theta_s)} \left[ \log \left( \lambda f_i(\theta_s) + (1 - \lambda) q_{i-1}(\theta_s) \right) \right] = \sum_s w^s_{\text{norm}} \left[ \log p(\theta_s) - \log (\lambda f_i(\theta_s) + (1 - \lambda) q_{i-1}(\theta_s)) \right]
\]

4.10.2.4 Gradients of Mixture Weights

For forward KL:

\[
\nabla_{\lambda_i} E_p \left[ \log p - \log \sum_j^K \lambda_j q_j \right] = -E_p \left[ \nabla_{\lambda_i} \log \sum_j^K \lambda_j q_j \right] = -E_p \left[ \frac{q_i}{\sum_j^K \lambda_j q_j} \right] = -E_{q_i} \left[ \frac{p}{q} \right] \tag{4.23}
\]

For reverse KL:

\[
\nabla_{\lambda_i} E_q [\log q - \log p] = E_{\sum_j^K \lambda_j q_j} [\nabla_{\lambda_i} \log (\sum_j^K \lambda_j q_j)] = E_{q_i} [\log q - \log p]
\]

114
4.10.2.5 The Functional Gradient of the Forward KL Divergence

We assume \( \text{supp } p \subseteq \text{supp } q \): that is, \( p \) is absolutely continuous with respect to the variational approximation \( q_i \) which can be ensured by the design of the variational family \( Q \).

Let \( D(q) = \text{KL}(p\|q) \). Functional gradient \( \frac{\delta D}{\delta q} \) can be computed from the Taylor expansion of the KL functional [Friedman, 2001] as follows:

\[
\lim_{\epsilon \to 0} \frac{D(q + \epsilon \cdot h) - D(q)}{\epsilon} = \int \frac{\partial D}{\partial q} h dx
\]

\[
\frac{D(q + \epsilon \cdot h) - D(q)}{\epsilon} = \frac{1}{\epsilon} \int p \log p - p \log (q + \epsilon h) + p \log p - p \log q
\]

\[
= -\frac{1}{\epsilon} \int p \log (q + \epsilon h) - p \log q
\]

\[
= -\frac{1}{\epsilon} \int p \log (1 + \frac{\epsilon h}{q})
\]

We have the logarithmic inequality \( \frac{x}{x+1} \leq \log (1 + x) \leq x \forall x > -1 \) where we can substitute \( \epsilon \frac{h}{q} > 0 \) for \( x \) and arrive at

\[
-\frac{h}{q} \leq -\frac{1}{\epsilon} \log (1 + \frac{\epsilon h}{q}) \leq -\frac{h}{q} \frac{1 + \epsilon \frac{h}{q}}{1 + \epsilon \frac{h}{q}}
\]

By the monotone convergence theorem we can take the limit inside the integral and arrive at

\[
\lim_{\epsilon \to 0} \int -\frac{1}{\epsilon} \log (1 + \frac{\epsilon h}{q}) = \int -p \frac{h}{q}
\]

\[
\frac{\delta D(q)}{\delta q} = -\frac{p}{q}
\]
4.10.2.6 Boosting Convergence Analysis

For a convex and strongly smooth functional, the greedy sequential approximation framework of [Zhang, 2003] provides an asymptotic guarantee for the convergence to a target distribution in the convex hull of the base family at a rate of $O(1/K)$ where $K$ is the number of boosting iterations. This framework does not require each iteration to exactly solve for the optimal mixture component which can be difficult in variational inference.

While the convexity of $KL(p\|q)$ in $q$ is well established in the literature (proven with the log-sum inequality) for the forward KL divergence functional, we can show that FKL is also $\beta$-smooth in $q$ where $\beta$ depends on the maximum and minimum values that the density $q$ can take. To establish strong smoothness, on the other hand, stricter assumptions about the densities are necessary. If we assume that all densities are bounded away from 0 and from above $q_1$ then for any pair of densities $q_1$ and $q_2$ there exists a $\beta = \sup_{q_1,q_2} \frac{p}{q_1+q_2} \geq 0$ such that the functional gradient $\frac{\delta D}{\delta q}$ is $\beta$-Lipschitz, that is $\left|\frac{\delta D}{\delta q}(q_2) - \frac{\delta D}{\delta q}(q_1)\right| \leq \beta |q_2 - q_1|$. We can verify this choice of $\beta$:

\[
\left|\frac{\delta D}{\delta q}(q_2) - \frac{\delta D}{\delta q}(q_1)\right| = \left|\frac{-p}{q_2} - \frac{-p}{q_1}\right| = \left|\frac{p}{q_2} - \frac{p}{q_1}\right| = \left|\frac{p}{q_2}q_1 - q_1\right| = \frac{p}{q_2q_1}|q_2 - q_1| \leq \beta |q_2 - q_1|
\]

Note that the boundedness assumptions are not unrealistic in practice and can translate to a bounded parameter space for a given family of distributions.

\[
KL(p\|q_i) = KL(p\|\sum_{i=1}^{k} \lambda_i f_i) = O(1/k)
\]
4.10.3 An Alternative Approach to FKL-Based Boosting: Minimizing the Remainder

As we seek to construct an optimal proposal through the minimization of an SNIS approximation of FKL, a trade-off arises: “should we make the distribution easier to sample from in order to minimize the SNIS variance or should we bring it closer to the target in order to improve the worst-case IS estimation error?” In particular, the closer the proposal gets to a multimodal target, the harder it may be to sample from. Therefore, this trade-off translates to two distinct approaches for the greedy additive construction of an optimal proposal mixture distribution.

The first approach described in Section 4.4.2 is the most straightforward as it minimizes the forward KL between the mixture \( q_i \) and the target \( p \) while holding the parameters of previously-learned mixture components fixed.

Alternatively, define the remainder distribution at iteration \( i \) as \( r_i(\theta) = \frac{p(\theta|x)}{q_i(\theta)} \). A second approach is to minimize the FKL between each new component and \( r_i \), which may be simpler with fewer modes than \( p \):

\[
\arg\min_{f_i} \text{KL}(r_i\|f_i) = \arg\min_{f_i} \text{KL} \left( \frac{p_i}{q_{i-1}} \bigg\| f_i \right).
\]

The mixture weight can be estimated in this scenario for each mixture component by gradient descent using the gradient with respect to FKL (see Appendix 4.10.2.4).

This second approach is appealing because, at each boosting iteration, \( r_i \) becomes shallower with fewer modes which makes it easier to sample from than the multimodal proposal \( q_i \). This approach can also be motivated by gradient boosting [Friedman, 2001] or matching pursuit [Mallat and Zhang, 1993] where one seeks to identify the mixture component that best fits the functional residual. Furthermore, this approach might be less prone to degeneracy. In fact, a derivation of this approach for the reverse KL, in Appendix 4.10.2.2, identifies intrinsic entropy regularization which is often incorporated ad-hoc in similar objectives.
4.10.4 Additional Simulation Experiment Results

Fig. 4.5 provides moment estimation results for the simulation with well-separated modes on a mixture of 20 2-dimensional Gaussians.

![Image](image.png)

**Figure 4.5:** Evolution of (exact) FKL divergence and the mean-squared error of moment estimation (using samples from the FKL solution) on the task of estimating a 2-dimensional GMM of 20 components [Ma et al., 2019].

4.10.5 Additional Real Data experiment details and results

We provide additional details and results for the experiments on real data.

4.10.5.1 Parameter Transformations

For the covariance matrices of each component, we optimize over the square root of diagonal matrices to ensure the non-negativity of the final diagonal covariance estimate.

To ensure non-negative or zero mixture weights, we optimize over the logits of the weights from which we recover the final weights by logistic transformation.

4.10.5.2 Hyperparameters

Hyperparameters for both the RKL and FKL boosting methods include the learning rate for the mean and learning rate for the covariance matrix when optimizing each boosting component. These were each tuned between \{0.0001, 0.001, 0.01, 0.1\}. The number
of steps for each boosting component was tuned between 200 and 1000, and the number of samples for each gradient computation was tuned between \( \{25, 50, 100, 200\} \). The variance \( \sigma \) when initializing the covariance matrix for each component was tuned between \( \{0.0001, 0.0005, 0.001, 0.005, 0.01\} \).

For evaluation, we draw 6000 parameter samples from the final mixture of Gaussians from which we compute the final metrics. Increasing this number to 50000 did not lead to a significant change and posed a strain on computational resources.

For the comparison to HMC, each HMC chain was initialized with a sample drawn from \( \mathcal{N}(0, \sigma^2 I) \), with \( \sigma = 0.01 \). For all BLR tasks, the HMC comparison was run using an adaptive step size schedule with a starting step size of 1.0, 1000 burn-in steps, and 800 adaptation steps. For the BNN tasks, the HMC comparison was run using a fixed step size, tuned between \( \{0.001, 0.01\} \).

### 4.10.5.3 Datasets

We use four datasets from UCI, listed in Table 4.5. Table 4.5 reports the number of attributes in the dataset, or the dimensionality of the input \( x \), but the actual dimensionality of the sampling problem in each experiment is higher than the number of attributes depending on the size of the weight vector \( w \), and the additional variance parameters given in Sections 4.7.1 and 4.7.2. All tasks are regression tasks.

<table>
<thead>
<tr>
<th>Dataset name</th>
<th># attributes</th>
<th># examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>wine</td>
<td>11</td>
<td>4898</td>
</tr>
<tr>
<td>boston</td>
<td>13</td>
<td>506</td>
</tr>
<tr>
<td>concrete</td>
<td>8</td>
<td>1030</td>
</tr>
<tr>
<td>power</td>
<td>4</td>
<td>9568</td>
</tr>
</tbody>
</table>

**Table 4.5**: Datasets used in experiments.
4.10.5.4 Computational Considerations

In our experiments we observed that even in high dimensions, there do not seem to be significant computational differences between our method for optimizing the FKL and optimizing the RKL. The reported results use the same number of IS samples and optimization iterations for both RKL-VI and FKL-VI. In terms of wall clock time, we evaluate the highest dimensional experiment using BNNs on the Boston dataset \((d = 753)\). FKL VI had a wall clock time of 783.10 seconds and RKL VI had a wall clock time of 862.93 seconds after optimizing a single boosting component for 200 gradient steps when run on a single 8-core machine with an Intel Xeon CPU @ 2.20GHz.

4.10.5.5 Additional Experiment Results

In [Miller et al., 2017], the posterior predictive distribution is simply estimated as an average over samples from the posterior \(p(\theta|\mathcal{D}_{\text{train}})\) using variational boosting. In Tables 4.6, 4.7 and 4.8 we report the results for the RKL boosting methods where the posterior predictive distribution is computed without importance sampling (as in Eq. (4.15)), and is instead computed by directly averaging over samples from the variational distribution, as in Eq. (4.31).

\[
p(y|x^*, \mathcal{D}_{\text{train}}) \approx \frac{1}{L} \sum_{l=1}^{L} p(y|x^*, \theta^{(l)}), \quad \theta^{(l)} \sim p(\theta|\mathcal{D}_{\text{train}}) \tag{4.31}
\]

We also report results for HMC with 3 chains run in parallel. To compute the final predictive log probabilities, 2000 samples were drawn from each chain, and the predictive log probability was averaged over all 6000 combined samples using Eq. (4.31).
### Table 4.6: Predictive log probabilities on test for BLR with Gaussian prior (mean ± standard error over 20 train/test splits). (*Results from only 5 train/test splits due to computational constraints.)*

<table>
<thead>
<tr>
<th>Method</th>
<th>Wine $(d = 14)$</th>
<th>Boston $(d = 16)$</th>
<th>Concrete $(d = 11)$</th>
<th>Power $(d = 7)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMC (3 chains)</td>
<td>-1.003 (± 0.012)</td>
<td>-2.923 (± 0.035)</td>
<td>-3.781 (± 0.013)</td>
<td>-2.942* (± 0.017)</td>
</tr>
<tr>
<td>RKL VI (no IS)</td>
<td>-1.003 (± 0.012)</td>
<td>-2.924 (± 0.035)</td>
<td>-3.780 (± 0.013)</td>
<td>-2.921 (± 0.006)</td>
</tr>
<tr>
<td>RKL VB 2 (no IS)</td>
<td>-1.003 (± 0.012)</td>
<td>-2.923 (± 0.035)</td>
<td>-3.781 (± 0.013)</td>
<td>-2.994 (± 0.004)</td>
</tr>
<tr>
<td>RKL VB 3 (no IS)</td>
<td>-1.003 (± 0.012)</td>
<td>-2.924 (± 0.035)</td>
<td>-3.781 (± 0.013)</td>
<td>-2.972 (± 0.005)</td>
</tr>
</tbody>
</table>

### Table 4.7: Predictive log probabilities on test for BNNs with Gaussian prior (mean ± standard error over 20 train/test splits). (*Results from only 5 train/test splits due to computational constraints.)*

<table>
<thead>
<tr>
<th>Method</th>
<th>Wine $(d = 653)$</th>
<th>Boston $(d = 753)$</th>
<th>Concrete $(d = 503)$</th>
<th>Power $(d = 303)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMC (3 chains)</td>
<td>-0.988 (± 0.014)</td>
<td>-2.706 (± 0.093)</td>
<td>-3.279 (± 0.019)</td>
<td>-2.824* (± 0.017)</td>
</tr>
<tr>
<td>RKL VI (no IS)</td>
<td>-0.991 (± 0.015)</td>
<td>-2.858 (± 0.019)</td>
<td>-3.230 (± 0.015)</td>
<td>-2.850 (± 0.009)</td>
</tr>
<tr>
<td>RKL VB 2 (no IS)</td>
<td>-0.990 (± 0.015)</td>
<td>-2.835 (± 0.020)</td>
<td>-3.231 (± 0.015)</td>
<td>-2.943 (± 0.011)</td>
</tr>
<tr>
<td>RKL VB 3 (no IS)</td>
<td>-0.983 (± 0.014)</td>
<td>-2.753 (± 0.015)</td>
<td>-3.232 (± 0.015)</td>
<td>-2.997 (± 0.011)</td>
</tr>
</tbody>
</table>

### Table 4.8: Predictive log probabilities on test for BLR with heavy tailed prior (mean ± standard error over 20 train/test splits). (*Results from only 5 train/test splits due to computational constraints.)*

<table>
<thead>
<tr>
<th>Method</th>
<th>Wine $(d = 13)$</th>
<th>Boston $(d = 15)$</th>
<th>Concrete $(d = 10)$</th>
<th>Power $(d = 6)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMC (3 chains)</td>
<td>-1.008 (± 0.012)</td>
<td>-3.085 (± 0.056)</td>
<td>-3.824 (± 0.023)</td>
<td>-2.942* (± 0.017)</td>
</tr>
<tr>
<td>RKL VI (no IS)</td>
<td>-1.002 (± 0.012)</td>
<td>-2.915 (± 0.034)</td>
<td>-3.780 (± 0.013)</td>
<td>-2.923 (± 0.006)</td>
</tr>
<tr>
<td>RKL VB 2 (no IS)</td>
<td>-1.001 (± 0.013)</td>
<td>-2.920 (± 0.034)</td>
<td>-3.780 (± 0.013)</td>
<td>-2.982 (± 0.003)</td>
</tr>
<tr>
<td>RKL VB 3 (no IS)</td>
<td>-1.001 (± 0.013)</td>
<td>-2.920 (± 0.034)</td>
<td>-3.781 (± 0.013)</td>
<td>-2.961 (± 0.005)</td>
</tr>
</tbody>
</table>
Conclusions

By leveraging probabilistic modelling for uncertainty quantification we can develop principled and practical algorithms for "knowing what or when a model doesn’t know". This is crucial for the increased deployment of machine learning in real-world decision-making systems, especially for those characterized by asymmetric or nonlinear risks and benefits. However, current approaches for integrating probabilistic modelling in deep learning often suffer from lack of scalability and generally underperform simpler baselines, especially in terms of predictive performance.

In this thesis, we propose a scalable family of mixture priors for neural networks that combine mode averaging with local (within-mode) uncertainty in order to guarantee calibration and robustness under distribution shift. We propose a low-rank reparametrization of model weights in order to alleviate over-regularization. This approach also allows us to incorporate heavy-tailed distributions, such as Cauchy, without sacrificing predictive performance. Our empirical studies across various benchmarks and network architectures show that our proposed approach outperforms standard baselines across log-likelihood, accuracy, and calibration on both the test sets and their corrupted versions.

We then build on the empirical Bayes interpretation of meta-learning by proposing a nonparametric mixture of hierarchical Bayesian models over the parameters of meta-learners. In contrast to consolidating inductive biases into a single set of hyperparameters, our approach of task-dependent parameter selection better handles distribution shift, as
demonstrated on a set of evolving, image-based, few-shot learning benchmarks. Such multimodal representation can have a costly memory footprint when applied to the millions (or billions) of parameters in a neural network. We thus propose a stochastic and augmented Expectation-Maximization algorithm for approximating a Dirichlet process mixture model over the parameters.

Finally, we turn our attention to the most popular inference algorithm, variational inference (VI), and its guiding divergence, the reverse Kullback-Leibler divergence (RKL), which typically underestimates the tail of the posterior leading to miscalibration and potential degeneracy. While Importance sampling (IS), can be used to fine-tune and de-bias the estimates of a VI procedures, its success requires the VI solution to have heavier tails than the target, which is rarely achievable by minimizing the RKL. We thus propose a novel combination of optimization and sampling techniques for approximate Bayesian inference by constructing an IS proposal distribution through the minimization of a forward KL (FKL) divergence. This approach guarantees asymptotic consistency and a fast convergence towards both the optimal IS estimator and the optimal multimodal variational approximation. We empirically demonstrate on real data that our method is competitive with variational boosting and MCMC.

**Potential Extensions**

Going forward, we would like to explore new architectures specifically designed for distributions over models. This is a huge performance bottleneck, with deterministic nets already seeing major benefits from architecture redesign. Alternatively, we would like to explore data-dependent mixture weights as well as different consensus algorithms from the standard unweighted mixture of experts approach.

Our approach for continual meta-learning with nonparametric mixtures also stands to benefit from orthogonal improvements in posterior inference beyond MAP estimation (e.g., variational inference, Laplace approximation, or stochastic gradient Markov chain Monte Carlo) in order to incorporate local uncertainty.
Finally, one bottleneck of using the forward KL divergence is the potentially high variance of the self-normalized importance sampling estimate. Developing variance reduction schemes for likelihood ratios is an open research problem. In practice, we could also explore the use of re-sampling and re-weighting as well as some computationally intensive smoothing schemes.
Bibliography


Carlos M Carvalho, Nicholas G Polson, and James G Scott. Handling sparsity via the horseshoe. In Artificial Intelligence and Statistics, 2009.


Jing Gao, Wei Fan, Jing Jiang, and Jiawei Han. Knowledge transfer via multiple model local structure mapping. In Proceedings of the 14th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pages 283–291. ACM, 2008.


129


Biography

Ghassen Jerfel graduated *summa cum laude* from Princeton University with a B.S.E. in Computer Science in 2016 and an M.S.E. in Computer Science in 2017.

Ghassen entered the Duke Electrical and Computer Engineering Ph.D. program in August 2017 with a Pratt-Gartner Graduate Fellowship and will obtain his doctoral degree under the direction of Katherine A. Heller in July 2021. He was awarded the Facebook Emerging Scholar fellowship in January 2019.

Ghassen spent 3 years (2018-2021) at UC Berkeley as a visiting student researcher with Michael I. Jordan. During this period of time, he also worked at Google Brain as a student researcher.

Upon graduating, Ghassen will join Waymo as a research scientist working on robust planning under uncertainty for autonomous agents.