Implementation of a probabilistic Softmax for BNNs

Aeiau Zzzz [*1] **Bauiu C. Yyyy** [*1 2] **Cieua Vvvvv** [2] **Iaesut Saoeu** [3] **Fiuea Rrrr** [1] **Tateu H. Yasehe** [3 1 2]
Aaoeu Iasoh [2] **Buiui Eueu** [3] **Aeuia Zzzz** [3] **Bieea C. Yyyy** [1 2] **Teoau Xxxx** [3] **Eee Pppp** [3]

Abstract

This document provides a basic paper template and submission guidelines. Abstracts must be a single paragraph, ideally between 4–6 sentences long. Gross violations will trigger corrections at the camera-ready phase.

1. Introduction

In recent years, deep learning has achieved remarkable success across a wide range of classification tasks, including image recognition, natural language processing, and medical diagnosis (**?**). Despite their high predictive accuracy, traditional neural networks often operate as black boxes, providing point estimates without quantifying the uncertainty associated with their predictions. This limitation poses significant challenges, especially in safety-critical applications such as autonomous driving, healthcare, and financial decision-making, where understanding the confidence of predictions is crucial for reliable and trustworthy outcomes (**?**). Uncertainty estimation in neural networks addresses this challenge by providing measures of confidence alongside predictions. Bayesian Neural Networks (BNNs) offer a principled framework for uncertainty estimation by treating model parameters as random variables and capturing their posterior distribution (Blundell et al., 2015). By integrating over the uncertainty in the model parameters, BNNs can provide more robust predictions and better generalize to unseen data. However, exact Bayesian inference in neural networks is intractable due to the high dimensionality and complexity of the parameter space, necessitating the development of approximate inference methods. Several approaches have been proposed to approximate the posterior distribution in BNNs, each with its own set of advantages and limitations. Sampling-based methods like Stochastic Gradient Langevin Dynamics (SGLD) () and Hamiltonian Monte Carlo (HMC) (Neal, 2012) provide accurate posterior approximations but are often computationally expensive and challenging to scale to large networks. On the other hand, approximation techniques such as Monte Carlo Dropout (MC Dropout) (Gal & Ghahramani, 2016) and Deep Ensembles (**?**) offer more computationally feasible alternatives but may lack the theoretical rigor and flexibility to capture complex posterior distributions fully. Variational Inference (VI) methods, including Bayesian Backpropagation (BBP) (Blundell et al., 2015), strike a balance between scalability and accuracy but are constrained by the choice of variational families, which may oversimplify the true posterior. Despite these advancements, existing methods often face trade-offs between computational efficiency, scalability, and the fidelity of uncertainty estimates. In particular, many approaches rely on assumptions such as Gaussianity in the parameter or output space, which may not hold in practice, especially for complex and high-dimensional data distributions. Additionally, the computational overhead associated with sampling-based methods limits their applicability to large-scale classification tasks, prompting the need for more efficient inference techniques.

In this work, we implement a novel method that leverages Tractable Approximate Gaussian Inference (TAGI) to provide efficient and scalable uncertainty estimation in Bayesian Deep Networks. TAGI offers an to sampling-based approaches by maintaining and updating approximate Gaussian distributions over network parameters, enabling the propagation of uncertainty through deep learning models seamlessly. Furthermore, we propose a probabilistic softmax formulation that translates the uncertainty from the logit space to the output space, allowing for accurate and computationally efficient uncertainty estimates in classification tasks.

The focus of our project can be summarized as follows:

- **Tractable Approximate Gaussian Inference (TAGI):** We re-implement the toy example proposed in the original TAGI paper (Goulet et al., 2021).
- **Probabilistic Softmax for Uncertainty Estimation:** We provide an implementation of the probabilistic softmax formulation porposed by the authors of TAGI on

[*]Equal contribution [1]Department of Computation, University of Torontoland, Torontoland, Canada [2]Googol ShallowMind, New London, Michigan, USA [3]School of Computation, University of Edenborrow, Edenborrow, United Kingdom. Correspondence to: Cieua Vvvvv <c.vvvvv@googol.com>, Eee Pppp <ep@eden.co.uk>.

IFT 6269 project report - Université de Montréal

C++ and CUDA for the cutagi library (), along with an evaluation on datasets such as MNIST and CIFAR-10.

- **Ablation Study:** We conduct an ablation study to evaluate the effectiveness of TAGI-based inference and the probabilistic softmax formulation in uncertainty estimation for classification tasks.

2. Related Work

Bayesian Neural Networks (BNNs) provide a principled framework for uncertainty estimation by capturing a posterior distribution over model parameters. Various methods, including sampling-based techniques and approximations, have been proposed to estimate uncertainty in classification tasks efficiently. Below, we review prominent methods alongside a recent advance using analytical approximations.

2.1. Monte Carlo Dropout (MC Dropout)

MC Dropout, introduced by (Gal & Ghahramani, 2016), approximates Bayesian inference by using dropout during both training and inference. Multiple stochastic forward passes are performed at inference by sampling different dropout masks, and the outputs are aggregated to estimate the predictive distribution. The predictive mean is computed as:

$$\hat{\mathbf{y}} = \frac{1}{T}\sum_{t=1}^{T} f(\mathbf{x}, \mathbf{w}^{(t)}),$$

where $\mathbf{w}^{(t)}$ are the sampled weights with dropout and T is the number of stochastic passes. This approach is computationally efficient and easy to implement but is limited by the reliance on dropout regularization, which might not fully capture posterior uncertainty.

2.2. Stochastic Gradient Langevin Dynamics (SGLD)

SGLD, proposed by (Welling & Teh, 2011), combines stochastic gradient descent (SGD) with Langevin dynamics to sample from the posterior distribution of weights. The weight updates include Gaussian noise:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla \mathcal{L}(\mathbf{w}_t) + \sqrt{2\eta}\epsilon_t,$$

where η is the learning rate, $\mathcal{L}$ is the loss function, and $\epsilon_t \sim \mathcal{N}(0, I)$ is Gaussian noise. By incorporating noise into the gradient updates, SGLD provides samples that approximate the posterior. However, the method is more computationally intensive compared to approximation techniques like MC Dropout and can be sensitive to hyperparameters such as the step size.

2.3. Hamiltonian Monte Carlo (HMC)

HMC (Neal, 2012) extends traditional Markov Chain Monte Carlo (MCMC) by using Hamiltonian dynamics to sample efficiently from the posterior. It introduces momentum variables and explores the posterior landscape using the Hamiltonian:

$$\mathcal{H}(\mathbf{w}, \mathbf{p}) = -\log p(\mathbf{w}|\mathcal{D}) + \frac{\|\mathbf{p}\|^2}{2},$$

where $\mathbf{p}$ is the momentum. HMC provides accurate posterior samples, but its computational cost is prohibitive for high-dimensional neural network parameters. Additionally, it requires careful tuning of the step size and number of leapfrog steps.

2.4. Bayesian Backpropagation (BBP)

Bayesian Backpropagation (BBP), also known as Bayes by Backprop (Blundell et al., 2015), uses variational inference to approximate the posterior distribution over weights. A variational distribution $q(\mathbf{w})$, typically Gaussian, is optimized to minimize the Kullback-Leibler (KL) divergence from the true posterior. The variational posterior is parameterized as:

$$q(\mathbf{w}|\boldsymbol{\theta}) = \prod_i \mathcal{N}(w_i; \mu_i, \sigma_i^2),$$

where $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$ are learnable parameters. Inference involves sampling weights from $q(\mathbf{w})$ and averaging multiple forward passes. While BBP provides a principled approach to Bayesian inference, the quality of the approximation depends on the chosen variational family, which might not fully capture the true posterior.

2.5. Deep Ensembles

Deep Ensembles (Lakshminarayanan et al., 2017) estimate uncertainty by training multiple neural networks with different random initializations. The ensemble predictions are averaged to compute the predictive distribution:

$$\hat{\mathbf{y}} = \frac{1}{M}\sum_{m=1}^{M} f(\mathbf{x}; \mathbf{w}_m),$$

where $\mathbf{w}_m$ are the weights of the m^{th} ensemble member. The variance across the ensemble outputs represents model uncertainty. This method is robust and often outperforms other techniques in practice. However, it is computationally expensive due to the need to train multiple models independently.

2.6. Variational Inference (VI)

Variational Inference (Graves, 2011) approximates the posterior distribution $p(\mathbf{w}|\mathcal{D})$ with a simpler variational distribution $q(\mathbf{w})$ by minimizing the KL divergence:

$$\mathrm{KL}(q(\mathbf{w})\|p(\mathbf{w}|\mathcal{D})) = \int q(\mathbf{w}) \log \frac{q(\mathbf{w})}{p(\mathbf{w}|\mathcal{D})}\, d\mathbf{w}.$$

Common variational families include mean-field Gaussians, which are computationally efficient but may not fully capture the complexity of the true posterior. Variational inference is scalable and adaptable but often trades accuracy for efficiency.

2.7. Fast Predictive Uncertainty for Classification with Bayesian Deep Networks

(Hobbhahn et al., 2022) propose a method that revisits the Laplace Bridge technique to approximate the softmax output distribution using a Dirichlet distribution. The method maps Gaussian distributions in logit space $\mathcal{N}(\mu, \Sigma)$ to Dirichlet distributions in output space $\text{Dir}(\boldsymbol{\alpha})$ via the analytical mapping:

$$\alpha_k = \exp\left(\mu_k + \frac{\sigma_k^2}{2}\right), \quad \text{for } k = 1, \ldots, K,$$

where μ_k and σ_k^2 are the mean and variance of the logit for class k , and α_k is the parameter of the resulting Dirichlet distribution. This approach avoids sampling and significantly reduces computational costs. It scales effectively to large datasets like ImageNet and complex architectures like DenseNet. While the method is efficient, its limitations include reliance on Gaussian assumptions in logit space and potential inaccuracies in modeling highly non-Gaussian posteriors. Additionally, its performance on small or imbalanced datasets is underexplored.

Each method offers unique strengths and weaknesses. MC Dropout and Deep Ensembles are practical and straightforward but may lack the theoretical rigor of sampling-based methods like SGLD and HMC. Variational inference approaches, such as BBP, balance scalability and accuracy but depend heavily on the choice of variational family. Sampling-based methods like SGLD and HMC provide accurate posterior approximations but are computationally expensive. The Dirichlet-based approach by Hobbhahn et al. introduces computational efficiency and scalability but trades off some flexibility in capturing complex posterior distributions.

In the next section, we introduce a novel method that combines the efficiency of analytical approximations with the theoretical grounding of Bayesian inference. The method relies on Tractable Approximate Gaussian Inference (TAGI) as a learning paradigm that allows passing uncertainty through deep learning models, and the formulation of a probabilistic softmax that further allows passing that uncertainty to the output layer.

3. Methodology

3.1. Tractable approximate gaussian inference (TAGI)

Goulet et al. (Goulet et al., 2021) proposed the Tractable Approximate Gaussian Inference (TAGI) method, which is a Bayesian approach to neural networks. Compared to traditional NNs, this method allows to (1) optimize parameters analytically, (2) treat uncertainties from input to output layers and provide density forecasts.
Consider a fully connected NN presented in Figure 1, the formulations of a TAGI-NN, which is similar to that of a traditional NN except that parameters and variables are assumed to be normally distributed, are defined by the following deterministic equations:

$$\begin{aligned}
\boldsymbol{z}^{(1)} &= \boldsymbol{w}^{(0)}\boldsymbol{x} + \boldsymbol{b}^{(0)}, \\
\boldsymbol{a}^{(i)} &= \tilde{\sigma}(\boldsymbol{z}^{(i)}) \quad (i = 1 : \mathtt{L}), \\
\boldsymbol{z}^{(i)} &= \boldsymbol{w}^{(i)}\boldsymbol{a}^{(i-1)} + \boldsymbol{b}^{(i)} \quad (i = 2 : \mathtt{L}), \\
\boldsymbol{z}^{(\mathtt{O})} &= \boldsymbol{w}^{(\mathtt{L})}\boldsymbol{a}^{(\mathtt{L})} + \boldsymbol{w}^{(\mathtt{L})} \\
\boldsymbol{y} &= \boldsymbol{z}^{(\mathtt{O})} + \boldsymbol{v}.
\end{aligned}$$

where $\boldsymbol{w}, \boldsymbol{b}$ are parameters, $\mathtt{L}$ is the number of hidden layers, $\boldsymbol{v}$ are the observation noise, $\boldsymbol{z}$ are hidden units, $\boldsymbol{y}$ are outputs, and $\tilde{\sigma}(\cdot)$ is the linearized activation function.

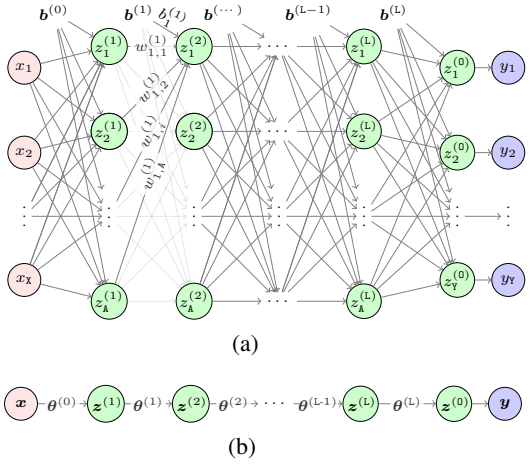

Figure 1. (a) A neural network representation. (b) A compacted NN representation, reproduced from (Goulet et al., 2021).

In Equation 3.1, the product of two Gaussian random variables is not Gaussian, despite of this, the authors proposed using the *Gaussian Multiplication Approximation* (GMA) to approximate the PDF of any product term $X_i X_j$ given by

$$\mathbb{E}\left[X_1 X_2\right] = \mu_1 \mu_2 + \text{cov}\left(X_1, X_2\right)$$

$$\text{cov}\left(X_3, X_1 X_2\right) = \text{cov}\left(X_1, X_3\right)\mu_2 + \text{cov}\left(X_2, X_3\right)\mu_1$$

$$
\begin{aligned}
\operatorname{cov}(X_1X_2, X_3X_4) = &\operatorname{cov}(X_1, X_3)\operatorname{cov}(X_2, X_4) \\
&+ \operatorname{cov}(X_1, X_4)\operatorname{cov}(X_2, X_3) \\
&+ \operatorname{cov}(X_1, X_3)\mu_2\mu_4 \\
&+ \operatorname{cov}(X_1, X_4)\mu_2\mu_3 \\
&+ \operatorname{cov}(X_2, X_3)\mu_1\mu_4 \\
&+ \operatorname{cov}(X_2, X_4)\mu_1\mu_3
\end{aligned}
$$

$$
\begin{aligned}
\operatorname{var}(X_1X_2) = &\sigma_1^2\sigma_2^2 + \operatorname{cov}(X_1, X_2)^2 \\
&+ 2\operatorname{cov}(X_1, X_2)\mu_1\mu_2 \\
&+ \sigma_1^2\mu_2^2 + \sigma_2^2\mu_1^2
\end{aligned}
$$

where μ, σ are respectively the mean and standard deviation of a random variable. Unlike a traditional NN where parameters are updated using backpropagation, model parameters in a TAGI-NN can be obtained analytically by a layer-wise scheme as depicted in Figure 2. First, given the observations

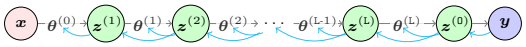

Figure 2. Recursive layer-wise inference, reproduced from (Goulet et al., 2021). Cyan arrows indicate inference directions.

$\boldsymbol{y}$, the output layer is updated as

$$
\begin{aligned}
f\left(\boldsymbol{z}^{(\mathtt{0})} \mid \boldsymbol{y}\right) &= \mathcal{N}\left(\boldsymbol{z}^{(\mathtt{0})}; \boldsymbol{\mu}_{\boldsymbol{Z}^{(\mathtt{0})}|\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{Z}^{(\mathtt{0})}|\boldsymbol{y}}\right) \\
\boldsymbol{\mu}_{\boldsymbol{Z}^{(\mathtt{0})}|\boldsymbol{y}} &= \boldsymbol{\mu}_{\boldsymbol{Z}^{(\mathtt{0})}} + \boldsymbol{\Sigma}_{\boldsymbol{Y}\boldsymbol{Z}^{(\mathtt{0})}}^{\top}\boldsymbol{\Sigma}_{\boldsymbol{Y}}^{-1}\left(\boldsymbol{y} - \boldsymbol{\mu}_{\boldsymbol{Y}}\right) \\
\boldsymbol{\Sigma}_{\boldsymbol{Z}^{(\mathtt{0})}|\boldsymbol{y}} &= \boldsymbol{\Sigma}_{\boldsymbol{Z}^{(\mathtt{0})}} - \boldsymbol{\Sigma}_{\boldsymbol{Y}\boldsymbol{Z}^{(\mathtt{0})}}^{\top}\boldsymbol{\Sigma}_{\boldsymbol{Y}}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{Y}\boldsymbol{Z}^{(\mathtt{0})}}.
\end{aligned}
$$

Next, the hidden units and parameters of the layer i^{th} are updated using a layer-wise procedure as

$$
\begin{aligned}
f(\boldsymbol{z} \mid \boldsymbol{y}) &= \mathcal{N}\left(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}}\right) \\
\boldsymbol{\mu}_{\boldsymbol{Z}|\boldsymbol{y}} &= \boldsymbol{\mu}_{\boldsymbol{Z}} + \mathbf{J}_{\boldsymbol{Z}}\left(\boldsymbol{\mu}_{\boldsymbol{Z}^{+}|\boldsymbol{y}} - \boldsymbol{\mu}_{\boldsymbol{Z}^{+}}\right) \\
\boldsymbol{\Sigma}_{\boldsymbol{Z}|\boldsymbol{y}} &= \boldsymbol{\Sigma}_{\boldsymbol{Z}} + \mathbf{J}_{\boldsymbol{Z}}\left(\boldsymbol{\Sigma}_{\boldsymbol{Z}^{+}|\boldsymbol{y}} - \boldsymbol{\Sigma}_{\boldsymbol{Z}^{+}}\right)\mathbf{J}_{\boldsymbol{Z}}^{\top} \\
\mathbf{J}_{\boldsymbol{Z}} &= \boldsymbol{\Sigma}_{\boldsymbol{Z}\boldsymbol{Z}^{+}}\boldsymbol{\Sigma}_{\boldsymbol{Z}^{+}}^{-1}, \\
f(\boldsymbol{\theta} \mid \boldsymbol{y}) &= \mathcal{N}\left(\boldsymbol{\theta}; \boldsymbol{\mu}_{\boldsymbol{\theta}|\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}|\boldsymbol{y}}\right) \\
\boldsymbol{\mu}_{\boldsymbol{\theta}|\boldsymbol{y}} &= \boldsymbol{\mu}_{\boldsymbol{\theta}} + \mathbf{J}_{\boldsymbol{\theta}}\left(\boldsymbol{\mu}_{\boldsymbol{Z}^{+}|\boldsymbol{y}} - \boldsymbol{\mu}_{\boldsymbol{Z}^{+}}\right) \\
\boldsymbol{\Sigma}_{\boldsymbol{\theta}|\boldsymbol{y}} &= \boldsymbol{\Sigma}_{\boldsymbol{\theta}} + \mathbf{J}_{\boldsymbol{\theta}}\left(\boldsymbol{\Sigma}_{\boldsymbol{Z}^{+}|\boldsymbol{y}} - \boldsymbol{\Sigma}_{\boldsymbol{Z}^{+}}\right)\mathbf{J}_{\boldsymbol{\theta}}^{\top} \\
\mathbf{J}_{\boldsymbol{\theta}} &= \boldsymbol{\Sigma}_{\boldsymbol{\theta}\boldsymbol{Z}^{+}}\boldsymbol{\Sigma}_{\boldsymbol{Z}^{+}}^{-1},
\end{aligned}
$$

where $\{\boldsymbol{\theta}^{+}, \boldsymbol{Z}^{+}\}$ is the short-hand notation of $\left\{\boldsymbol{\theta}^{(j+1)}, \boldsymbol{Z}^{(j+1)}\right\}$, and $\{\boldsymbol{\theta}, \boldsymbol{Z}\}$ is the short-hand notation of $\left\{\boldsymbol{\theta}^{(j)}, \boldsymbol{Z}^{(j)}\right\}$. While this section only introduces the TAGI method for a simple feedforward neural network, this method is applicable for any NN architectures that are already developed. This is because the difference between TAGI-NN and traditional NN models is the parameter update mechanism but the architecture is unchanged.

I NEED TO FURTHER ADD WHY THE ASSUMPTIONS HOLD.

4. Probabilistic Softmax

In this section we will present how the current classification in TAGI is performed, we will present a probabilistic softmax formulation and a new Remax activation.

4.1. Classification with TAGI

TAGI addresses classification tasks through a hierarchical binary tree structure, where each class is uniquely represented as a path in the tree. The tree's height is given by $\mathtt{H} = \lceil \log_2(\mathtt{K}) \rceil$, and the number of hidden states is $\mathtt{Y} = \mathtt{K} - 1$, provided $\log_2(\mathtt{K}) \in \mathbb{Z}^{+}$.

For instance, when $\mathtt{K} = 8$, the hierarchical decomposition results in $\mathtt{H} = 3$ layers, where each class $y_{\mathcal{C}}^{(\mathtt{C})}$ is uniquely identified by a binary sequence $\mathcal{C} = \{j, k, l\} \in \{0, 1\}^3$, corresponding to the indices of the path in the tree.

This approach reformulates the classification task as a regression problem, eliminating the need for a softmax activation function. However, the current implementation has limitations in capturing epistemic uncertainty effectively, as it enforces constraints on the probabilities of each hidden state to remain within the interval $[0, 1]$. Although a solution to this issue has been proposed by the authors, it has not yet been implemented.

Additionally, directly applying the softmax function is impractical in this context because leveraging uncertainty requires a probabilistic representation rather than a deterministic one. This work aims to address these challenges by implementing and evaluating the probabilistic softmax framework previously proposed by the authors, which is expected to overcome the limitations of the current implementation.

4.2. Closed-form softmax using a log-transformation

In neural networks, the softmax function is commonly defined as:

$$
\operatorname{softmax}(\boldsymbol{z}, i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}. \tag{1}
$$

While effective in standard implementations, this function poses challenges in the context of Tractable Approximate Gaussian Inference (TAGI) (Goulet et al., 2021). Specifically, performing this operation within TAGI requires dividing two Gaussian-distributed variables. However, the result of such a division does not follow a Gaussian distribution, which disrupts the probabilistic consistency of the framework.

To address this limitation, the authors of TAGI (Goulet et al., 2021) proposed a novel approach to adapt the softmax function for probabilistic contexts where z_i follows a Gaussian distribution. The method leverages a log-normal transformation to ensure mathematical tractability. By matching the moments of the neural network's output in the

log-normalized space, the division inherent to the softmax function can be reformulated as a subtraction between log-normal distributions. Subsequently, the moments of the resulting subtraction are remapped to a Gaussian distribution, preserving the probabilistic framework.This adaptation enables the application of softmax within TAGI while maintaining the Gaussian assumptions.

For hidden units $\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu_Z}, \boldsymbol{\sigma_Z^2} \cdot \mathbf{I})$ such that $\boldsymbol{z} \in \mathbb{R}^{\mathsf{z}}$, then taking the exponential function of z_i leads to a lognormal distribution

$$\begin{aligned} E = \exp(Z) &\sim \ln\mathcal{N}(e; \mu_Z, \sigma_Z) \\ \mu_E &= \exp\left(\mu_Z + \tfrac{1}{2}\sigma_Z^2\right) \\ \sigma_E^2 &= \exp\left(2\mu_Z + \sigma_Z^2\right) \cdot \left(\exp(\sigma_Z^2) - 1\right), \end{aligned} \tag{2}$$

where the covariance between Z_i and E_i is

$$\operatorname{cov}(Z, E) = \sigma_Z^2 \cdot \mu_E. \tag{3}$$

Given that $Z_i \perp\!\!\!\perp Z_j, \forall i \neq j$, the sum on the softmax's denominator $\tilde{E}$ is modelled by a Gaussian R.V.,

$$\begin{aligned} \textstyle\sum_j E_j = \tilde{E} &\sim \mathcal{N}(\tilde{e}; \mu_{\tilde{E}}, \sigma_{\tilde{E}}^2) \\ \mu_{\tilde{E}} &= \textstyle\sum_j \mu_{E,j} \\ \sigma_{\tilde{E}}^2 &= \textstyle\sum_j \sigma_{E,j}^2. \end{aligned} \tag{4}$$

Given the independence hypothesis between hidden units the covariance and coefficient of correlation between the softmax's numerator and denominator are only influenced by the presence of a same variable E_i in both terms so that,

$$\operatorname{cov}\big(E_i, \tilde{E}\big) = \sigma_{E_i}^2 \tag{5}$$

For a lognormal random variable X, the relations between the R.V.'s moments and the moment in the log-transformed space λ and ζ are

$$X \sim \ln\mathcal{N}(x; \lambda, \zeta) \begin{cases} \zeta &= \sqrt{\ln\left(1 + \frac{\sigma_X^2}{\mu_X^2}\right)} \\ \lambda &= \ln\mu_X - \frac{1}{2}\zeta^2. \end{cases} \tag{6}$$

Using these relations, we can obtain the softmax's denominator in the log-transformed space so that we can replace the division by a substraction,

$$\begin{aligned} \ln\tilde{E} &\sim \ln\mathcal{N}(\ln e; \mu_{\ln E}, \sigma_{\ln E}) \\ \sigma_{\ln E}^2 &= \ln\left(1 + \tfrac{\sigma_{\tilde{E}}^2}{\mu_{\tilde{E}}^2}\right) \\ \mu_{\ln E} &= \ln\mu_{\tilde{E}} - \tfrac{1}{2}\sigma_{\ln E}^2. \end{aligned} \tag{7}$$

The covariance between the softmax's numerator and denominator in the log-transformed space is

$$\operatorname{cov}\big(\ln E_i, \ln\tilde{E}\big) = \ln\left(1 + \operatorname{cov}\big(E_i, \tilde{E}\big) \cdot \tfrac{1}{\mu_{E,i}} \cdot \tfrac{1}{\mu_{\tilde{E}}}\right). \tag{8}$$

The i-th attention unit in the log-space $\check{A}_i = \ln E_i - \ln\tilde{E}$ defines a lognormal R.V. $A_i = \exp(\check{A}_i)$ so that

$$\begin{aligned} A_i &\sim \ln\mathcal{N}(\check{a}_i; \mu_{\check{A},i}, \sigma_{\check{A},i}) \\ \mu_{\check{A},i} &= \mu_{\ln E,i} - \mu_{\ln\tilde{E}} \\ \sigma_{\check{A},i}^2 &= \sigma_{\ln E,i}^2 + \sigma_{\ln\tilde{E}}^2 - 2\operatorname{cov}\big(\ln E_i, \ln\tilde{E}\big). \end{aligned} \tag{9}$$

The covariance between $\check{A}_i$ and $\ln E_i$ is given by

$$\operatorname{cov}(\check{A}_i, Z_i) = \sigma_{\ln E,i}^2 - \operatorname{cov}\big(\ln E_i, \ln\tilde{E}\big). \tag{10}$$

The moments of the lognormal variable A_i are

$$\begin{aligned} \mu_{A,i} &= \exp(\mu_{\check{A},i} + \tfrac{1}{2}\sigma_{\check{A},i}^2) \\ \sigma_{A,i}^2 &= \mu_{A,i}^2\big(\exp(\sigma_{\check{A},i}^2) - 1\big) \end{aligned} \tag{11}$$

In order to employ attention units with Gaussian inference, we use the approximation that A_i is Gaussian with

$$A_i \sim \mathcal{N}(a_i; \mu_{A,i}, \sigma_{A,i}^2),$$

with the assumption that $A_i \perp\!\!\!\perp A_j, \forall i \neq j$. The regression coefficient $\beta_{\check{A}|A}$ expressing the linear relationship between $\tilde{A}_i$ and A_i is given by

$$\beta_{A|\check{A}} = \frac{\operatorname{cov}(\check{A}_i, A_i)}{\sigma_{\check{A},i}^2}, \tag{12}$$

so that the covariance between A_i and Z_i is

$$\begin{aligned} \operatorname{cov}(A_i, Z_i) &= \tfrac{\operatorname{cov}(\check{A}_i, A_i)}{\sigma_{\check{A},i}^2} \cdot \operatorname{cov}(\check{A}_i, Z_i) \\ &= \exp(\mu_{\check{A},i} + \tfrac{1}{2}\sigma_{\check{A},i}^2) \cdot \operatorname{cov}(\check{A}_i, Z_i). \end{aligned} \tag{13}$$

4.3. Remax

The authors of the formulation Goulet and Nguyen observed through experimentation and comparison of the probabilistic softmax with results obtained via Monte Carlo sampling that high variance values ($\sigma^2 \approx 1$) can lead to exploding gradients. To mitigate this issue, they proposed replacing the exponential function with a novel activation function, termed the *Mixture Rectified Linear activation Unit* (Mixture ReLU).

The concept behind the *Mixture Rectified Linear activation Unit* is to propagate the hidden unit's uncertainty $Z \sim \mathcal{N}(\mu_Z, \sigma_Z^2)$ through a ReLU $\phi_{\mathrm{R}}(z) = \max(0, z)$ by using a Gaussian mixture between a truncated Gaussian and a zero-valued component. The expected value of the

Gaussian PDF of $\tilde{Z}$ truncated at $z \geq 0$ are

$$\begin{aligned} \phi_{\mathrm{R}}(z) &= \max(0, z) \\ \mu_A &= \mu_Z \cdot \Phi\left(\frac{\mu_Z}{\sigma_Z}\right) + \sigma_Z \cdot \varphi\left(\frac{\mu_Z}{\sigma_Z}\right) \\ \sigma_A^2 &= -\mu_A^2 + 2\mu_A\mu_Z - \mu_Z\sigma_Z\phi\left(\frac{\mu_Z}{\sigma_Z}\right) \\ &\quad + \left(\sigma_Z^2 - \mu_Z^2\right)\Phi\left(\frac{\mu_Z}{\sigma_Z}\right) \\ \mathrm{cov}(A, Z) &= \sigma_Z^2\Phi\left(\frac{\mu_Z}{\sigma_Z}\right) \end{aligned} \tag{14}$$

Then, the remax activation function is defined as

$$\mathrm{remax}(\boldsymbol{z}, i) = \frac{\mathrm{mrelu}(z_i)}{\sum_j \mathrm{mrelu}(z_j)}. \tag{15}$$

All the calculations remain the same as for the probabilistic softmax but instead of $E = \exp(Z)$, we will have

$$\boldsymbol{M} \sim \mathcal{N}(\boldsymbol{m}; \boldsymbol{\mu}_{\boldsymbol{M}}, \boldsymbol{\sigma}_{\boldsymbol{M}}^2 \cdot \mathbf{I}) = \mathrm{mrelu}(\boldsymbol{Z}).$$

Then, the convariance between A_i and Z_i is

$$\begin{aligned} \mathrm{cov}(A_i, Z_i) &= \frac{\mathrm{cov}(\check{A}_i, A_i)}{\sigma_{\check{A},i}^2} \cdot \frac{\mathrm{cov}(\check{A}_i, M_i)}{\sigma_{M,i}^2} \cdot \mathrm{cov}(M_i, Z_i) \\ &= \exp\left(\mu_{\check{A},i} + \frac{1}{2}\sigma_{\check{A},i}^2\right) \\ &\quad \cdot \frac{\mathrm{cov}(\check{A}_i, M_i) \cdot \mathrm{cov}(M_i, Z_i)}{\sigma_{M,i}^2}. \end{aligned} \tag{16}$$

5. Remax results

In this section, we evaluate the performance of the Remax activation function, not by directly assessing model accuracy, but through its application in a classification task on the MNIST and CIFAR-10 datasets. The evaluation is conducted using a simple convolutional neural network (CNN) architecture consisting of three convolutional layers with batch normalization.

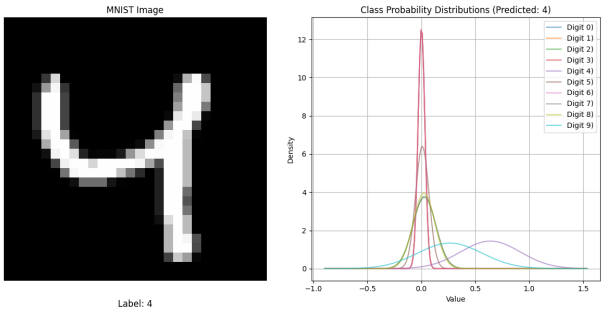

Figure 3. Ratio of predictions and epistemic uncertainty associated to each class on a MNIST *4* image.

Figures 3 and 4 illustrate the utility of the Remax activation function in classification tasks. In Figure 3, the class probabilities are represented by the means of Gaussian distributions, and the epistemic uncertainty—stemming from the model's lack of knowledge due to insufficient data—is captured by the standard deviations. In Figure 4, a misclassification example is shown where a frog is incorrectly predicted as a cat. However, the second most probable class is correctly identified as a frog, with both classes exhibiting considerable epistemic uncertainty. These examples highlight how Remax not only provides probabilistic predictions but also quantifies the model's uncertainty.

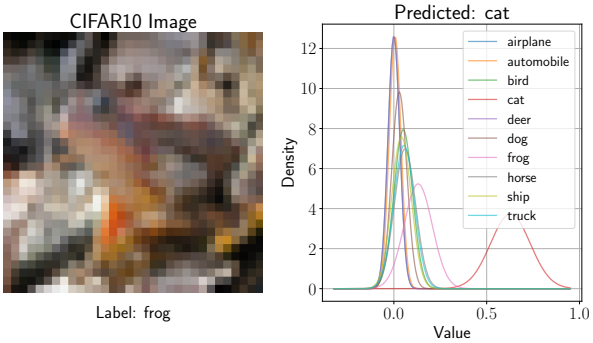

Figure 4. Ratio of predictions and epistemic uncertainty associated to each class on a CIFAR10 *frog* image.

6. Comparative Analysis: TAGI vs. Torch

In this section, we present a comprehensive comparison between the Tractable Approximate Gaussian Inference (TAGI) framework and a standard Torch-based backpropagation approach. Our analysis focuses on various aspects including overall performance, the impact of batch size, network depth, network size, the effect of the parameter σ_v on TAGI, and training time considerations. We highlight key findings using a combination of summary tables and selected figures. In addition, we incorporate results from our new TAGI-Remax implementation. While it often converged reliably and performed similarly on average to TAGI-HRC, it rarely achieved the absolute lowest error rates across experiments, suggesting that its stability does not always translate into peak performance.

6.1. Overall Performance

Across all experiments, TAGI consistently demonstrated better average performance in terms of test error compared to Torch. This advantage is most pronounced when increasing the number of layers in the network. While Torch-based models struggled to converge or exhibited deteriorated performance with deeper networks (particularly in convolutional neural networks without batch normalization), TAGI

maintained relatively stable error rates. This robustness suggests that TAGI may be less sensitive to architectural complexities and can effectively handle deeper networks without requiring additional normalization techniques.

The TAGI-Remax implementation followed a similar trend to TAGI-HRC in terms of average performance. As shown in Table 1, it also outperformed Torch in terms of test error. Although TAGI-Remax generally converged well, it did not frequently yield the absolute best results, indicating that while it is stable, it may not always produce the lowest achievable error.

Table 1. Overall Average Training and Test Error Rates (%) across Frameworks

Framework	Avg. Training Error	Avg. Test Error
Torch	10.05	10.69
TAGI-HRC	7.07	**7.73**
TAGI-Remax	7.14	**7.78**

The best overall performance was achieved by TAGI-HRC CNNs with three layers, 32 channels, and a sigma value of 0.5. Notably, the top experiments consistently featured TAGI-HRC CNNs rather than CNNs with batch normalization, highlighting TAGI's capability to deliver superior results without relying on normalization techniques. TAGI-Remax also benefited from similar settings, although it did not surpass the best TAGI-HRC configurations.

6.2. Effect of Batch Size

We observed a clear divergence in how each framework responds to varying batch sizes. Torch-based models tended to perform better with larger batch sizes, often relying on more stable gradient estimates to train effectively. Conversely, TAGI excelled with smaller batch sizes, where it could achieve remarkably low test errors even as network depth increased. This pattern is particularly striking in CNNs without batch normalization layers, where Torch's performance with smaller batch sizes was notably poor, while TAGI thrived under such conditions.

Interestingly, TAGI-Remax differed slightly from TAGI-HRC in terms of batch size sensitivity. While TAGI-HRC performed best with smaller batch sizes, TAGI-Remax appeared more flexible, showing stable performance across a range of batch sizes without a strong preference toward smaller batches.

6.3. Effect of Batch Normalization

We explore here the impact of batch normalization and batch size on test error rates for both frameworks. Torch-based models exhibited superior performance in CNNs with batch normalization, particularly with larger batch sizes, where they achieved the lowest test error rates. In contrast, TAGI demonstrated remarkable consistency and robustness with smaller batch sizes, regardless of whether batch normalization was applied. Interestingly, in TAGI experiments, CNNs without batch normalization performed slightly better than their batch-normalized counterparts, highlighting that lower batch sizes are a more significant factor for TAGI's performance than applying batch normalization.

Table 2. Test Error Rates (%) by Batch Size and Framework (FNN and CNN only)

	Small Batch (16)		Large Batch (512)	
Framework	FNN	CNN	FNN	CNN
Torch	9.31	31.20	**5.03**	16.22
TAGI-HRC	10.42	**1.74**	12.14	15.99
TAGI-Remax	9.14	7.30	7.02	5.43

For TAGI-Remax, a similar pattern emerged. Batch normalization did not provide a clear advantage, and good performance often came from configurations without it. This parallels the original TAGI-HRC results, suggesting that both TAGI-HRC and TAGI-Remax benefit more from careful selection of batch sizes than from normalization techniques.

Table 3. Test Error Rates (%) by Batch Size, Batch Normalization, and Framework (CNN only)

	Small Batch (16)		Large Batch (512)	
Framework	With BN	W.o BN	With BN	W.o BN
Torch	**1.35**	31.20	**1.06**	16.22
TAGI-HRC	1.97	**1.74**	4.11	15.99
TAGI-Remax	12.05	7.30	5.75	5.43

6.4. Influence of Network Depth

Depth-sensitive behavior was a crucial point of differentiation. As the number of layers increased, Torch-based CNNs (without batch normalization) struggled to converge, resulting in significantly higher test errors. In contrast, TAGI-HRC remained relatively stable, offering effective training and lower test errors even with deeper architectures. This is a key finding, suggesting TAGI-HRC's potential to alleviate the commonly observed training difficulties associated with very deep neural networks.

TAGI-Remax followed a similar trajectory to TAGI-HRC in this regard. While it did not consistently achieve the absolute lowest errors, it also did not succumb to the performance degradation seen with Torch models as the network depth increased. This indicates that the core resilience to depth-related training issues is preserved in the TAGI-Remax implementation.

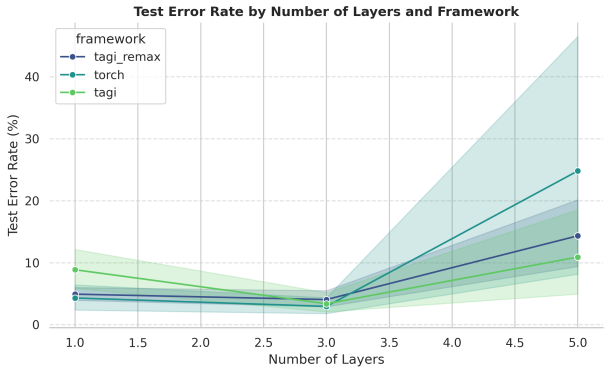

Figure 5. Test Error Rate by Number of Layers for TAGI-HRC vs. Torch. TAGI-HRC maintains low error rates as depth increases, while Torch performance degrades significantly with deeper networks (especially CNNs without batch normalization).

6.5. Effect of σ_v in TAGI

The parameter σ_v plays a pivotal role in TAGI's performance. Our experiments indicate that higher values of σ_v (e.g., $\sigma_v = 1.0$) yield better results, while a lower value (e.g., $\sigma_v = 0$) leads to poorer performance. This trend holds across different network depths, with deeper TAGI-HRC architectures benefiting even more from a larger σ_v. This finding underscores the importance of tuning σ_v to enhance TAGI-HRC's representational capability and stability.

TAGI-Remax experiments mirrored these findings. The same upward trend in performance with increasing σ_v values was observed, reinforcing the importance of this parameter for both implementations.

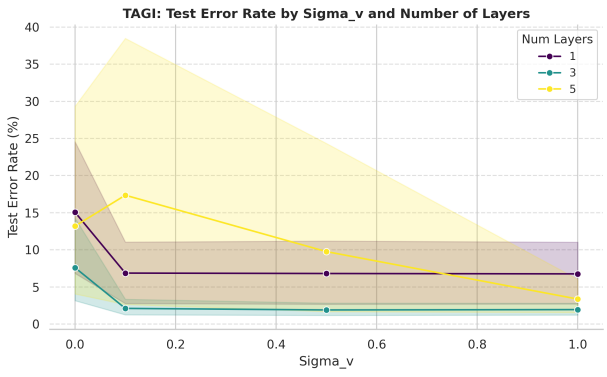

Figure 6. Impact of σ_v on TAGI Performance across Different Network Depths. Higher σ_v values consistently improve test error rates, especially in deeper networks.

6.6. Impact of Network Size (Neurons/Channels)

In fully-connected neural networks (FNNs), increasing the number of neurons per layer benefited TAGI-HRC more than Torch. TAGI-HRC seemed to leverage the additional network capacity effectively, reducing test error as network width grew. By contrast, Torch did not show significant improvement when neurons per layer increased, suggesting a possible limitation in how it explores the representational capacity of larger networks. Similar trends were noted with CNNs when increasing the number of channels, reinforcing the notion that TAGI-HRC may inherently avoid overfitting and achieve better generalization with larger models.

TAGI-Remax followed a similar pattern in FNNs: increasing the number of neurons per layer led to better performance. However, in CNNs, increasing the number of channels did not improve results for TAGI-Remax and even led to worse performance. This discrepancy highlights that while TAGI-Remax retains many of the advantageous characteristics of TAGI-HRC, certain architectural expansions may not universally improve its outcomes.

Table 4. FNN Test Error (%) by Neurons per Layer (Averaged across Layers)

Framework	Small (32) Neurons/layer	Large (512) Neurons/layer
Torch	7.03	7.32
TAGI-HRC	13.58	**8.98**
TAGI-Remax	9.25	**6.91**

Table 5. CNN Test Error (%) by Initial Channels per Layer (Averaged)

Framework	Small (8) Channels/layer	Large (32) Channels/layer
Torch	8.79	16.13
TAGI-HRC	6.29	**5.61**
TAGI-Remax	**5.46**	9.81

6.7. Training Time Considerations

While TAGI-HRC outperformed Torch in several qualitative aspects, it was generally more computationally expensive, especially as network depth increased. The training time for TAGI-HRC seemed to grow more rapidly than for Torch. In certain scenarios, Torch's faster (but sometimes less effective) training made it a more practical choice, especially when architectural complexity was moderate and batch normalization was used to stabilize training. Also, Torch experiments with CNNs without batch normalization were stopped early, explaining the longer training time for three layers compared to five.

TAGI-Remax followed a similar trend. Although it often

converged reliably, the computational overhead remained considerable. Its training time scaled in a manner similar to that of TAGI-HRC, suggesting that while stable and effective, both TAGI-HRC and TAGI-Remax incur higher computational costs than Torch as models grow in complexity.

Table 6. Training Time (minutes) vs. Test Error (%) by Number of Layers

Layers	Torch		TAGI-HRC		TAGI-Remax	
	Time	Err	Time	Err	Time	Err
1	4.5	4.3	5.7	8.9	5.6	4.9
3	5.6	3.0	6.6	3.4	6.5	4.1
5	5.2	24.8	**16.4**	10.9	**13.0**	14.3

6.8. Summary of Key Insights

In summary, the TAGI framework provides a more robust learning mechanism across various dimensions—batch sizes, network depths, batch normalization, and network widths—resulting in lower test errors compared to Torch. However, these improvements often come at a higher computational cost. Additionally, the choice of σ_v in TAGI can significantly influence performance, suggesting the importance of careful hyperparameter selection. Similarly, TAGI-Remax maintains these benefits and converges reliably, but does not consistently achieve the lowest possible error rates. It shares many of TAGI-HRC's characteristics, including sensitivity to σ_v and lower batch sizes, as well as improved performance with wider FNNs, but may not gain from wider CNNs. In contrast, Torch implementations benefit greatly from architectural and training heuristics like batch normalization and larger batch sizes, yet display notable limitations when confronted with deeper networks and smaller batch sizes.

Acknowledgements

Do not include acknowledgements in the initial version of the paper submitted for blind review.

If a paper is accepted, the final camera-ready version can (and probably should) include acknowledgements. In this case, please place such acknowledgements in an unnumbered section at the end of the paper. Typically, this will include thanks to reviewers who gave useful comments, to colleagues who contributed to the ideas, and to funding agencies and corporate sponsors that provided financial support.

References

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural network. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1613–1622, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/blundell15.html.

Duda, R. O., Hart, P. E., and Stork, D. G. *Pattern Classification*. John Wiley and Sons, 2nd edition, 2000.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/gal16.html.

Goulet, J.-A., Nguyen, L. H., and Amiri, S. Tractable approximate gaussian inference for bayesian neural networks. *Journal of Machine Learning Research*, 2021.

Graves, A. Practical variational inference for neural networks. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper_files/paper/2011/file/7eb3c8be3d411e8ebfab08eba5f49632-Paper.pdf.

Hobbhahn, M., Kristiadi, A., and Hennig, P. Fast predictive uncertainty for classification with bayesian deep networks. In Cussens, J. and Zhang, K. (eds.), *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pp. 822–832. PMLR, 01–05 Aug 2022. URL https://proceedings.mlr.press/v180/hobbhahn22a.html.

Kearns, M. J. *Computational Complexity of Machine Learning*. PhD thesis, Department of Computer Science, Harvard University, 1989.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. (eds.). *Machine Learning: An Artificial Intelligence Approach, Vol. I*. Tioga, Palo Alto, CA, 1983.

Mitchell, T. M. The need for biases in learning generalizations. Technical report, Computer Science Department, Rutgers University, New Brunswick, MA, 1980.

Neal, R. M. Mcmc using hamiltonian dynamics. *arXiv preprint arXiv:1206.1901*, 2012.

Newell, A. and Rosenbloom, P. S. Mechanisms of skill acquisition and the law of practice. In Anderson, J. R. (ed.), *Cognitive Skills and Their Acquisition*, chapter 1, pp. 1–51. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1981.

Samuel, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):211–229, 1959.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In Getoor, L. and Scheffer, T. (eds.), *ICML*, pp. 681–688. Omnipress, 2011. URL http://dblp.uni-trier.de/db/conf/icml/icml2011.html#WellingT11.

A. Do *not* have an appendix here

Do not put content after the references. Put anything that you might normally include after the references in a separate supplementary file.

We recommend that you build supplementary material in a separate document. If you must create one PDF and cut it up, please be careful to use a tool that doesn't alter the margins, and that doesn't aggressively rewrite the PDF file. pdftk usually works fine.

Please do not use Apple's preview to cut off supplementary material. In previous years it has altered margins, and created headaches at the camera-ready stage.