BAYESIAN TECHNIQUES FOR POST-HOC MODEL CALIBRATION

Jeegn Dani
Purdue University
jdani@purdue.edu
PUID: 0031432399

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated remarkable success across natural language processing tasks. However, standard training procedures fail to quantify uncertainty of, as predictions rely on a single optimized set of weights, W^* , obtained via maximum likelihood estimation (MLE) or the maximum a posteriori estimation (MAP). Bayesian inference addresses this limitation by treating the model parameters as random variables and modeling their posterior distribution given observed data. Unlike MLE, Bayesian methods provide a natural framework for uncertainty quantification, resulting in models that are more robust, calibrated, and interpretable.

Formally, the goal is to learn the conditional distribution $P(Y|X; D)$, where X represents input text sequences, Y represents target labels, and D is the training dataset. Instead of finding a point estimate W^*, Bayesian inference computes a posterior distribution over weights $P(W \mid D)$, enabling predictive uncertainty quantification. The predictive distribution under the Bayesian framework can be expressed as:

$$P(Y|X; D) = \int P(Y|X; W)P(W|D)dW \tag{1}$$

Direct computation of this integral is analytically intractable for deep neural networks due to their high-dimensional parameter spaces. As a result, we resort to approximation techniques that enable tractable estimation of the predictive distribution like:

$$P(Y|X; D) \approx \frac{1}{K} \sum_{k=1}^{K} P(Y|X; W^k), \quad W^k \sim P(W|D) \tag{2}$$

One of the primary goals of our project is to explore these Bayesian inference techniques in improving the calibration and uncertainty estimation of deep learning models.

Rather than building on a pretrained model, we chose to train GPT-2 Radford et al. (2019) from scratch in order to gain a deeper understanding of the transformer model architecture, training dynamics, and distributed-GPU learning. This end-to-end training approach allowed us to engage more directly with core deep learning concepts and infrastructure, beyond what is typically encountered in fine-tuning scenarios. After pretraining, we appended a classification head and fine-tuned the model on a multi-label emotion classification task using Reddit comments. To assess the model's generalization and uncertainty calibration under distribution shift, we evaluated this fine-tuned model on an OOD dataset which involves emotion classification on Tweets. This setup provided a meaningful downstream application for evaluating predictive confidence and robustness, as MLE models tend to be overconfident on OOD tasks.

Finally, we apply post-hoc Bayesian inference methods to the classification head. This enables us to approximate the posterior distribution over the output layer weights while keeping the transformer backbone fixed. This allows us to keep the computation feasible for Bayesian inference. In order to approximate $P(W|D)$, the primiary methods we investigated in this project were Laplace Approximation (LA) MacKay (1992), Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) Chen et al. (2014), Monte Carlo Dropout (MC Dropout) Gal & Ghahramani (2016) and Temperature scaling Guo et al. (2017). To evaluate the effectiveness of these methods, we use metrics like classification accuracy, Brier score, expected calibration error (ECE), and wall-clock runtime. This set of metrics allows us to assess both the reliability and computational efficiency of each technique. The code and experiments are publicly available at https://github.com/jeegn/gpt2-bayesian-inference.

2 DATASET

Pre-training: For pre-training the GPT-2 model from scratch, we sample approximately 5 billion tokens from the unlabelled FineWeb-Edu dataset Penedo et al. (2024), a large-scale curated web corpus focused on educational content. This subset enables the model to acquire broad linguistic and contextual knowledge.

Fine-tuning: We fine-tune the pre-trained model on the GoEmotions dataset Demszky et al. (2020), a multi-label emotion classification benchmark derived from Reddit comments. The labels contains 28 human-annotated emotion categories like gratitude, optimism, grief, excitement, and anger. It comprises of 58,000 Reddit comments, each labeled with one or more emotions. This dataset serves as our primary in-distribution benchmark.

Out-of-distribution evaluation: To measure generalization and model calibration under domain shift, we use the EmoInt dataset Duppada & Hiray (2017), which consists of emotion-labeled tweets across four categories: fear, joy, anger, and sadness. Since EmoInt uses a different label space than GoEmotions, we map the 28 GoEmotions emotions into these four categories based on the official GoEmotions-to-EmoInt mappings . This setup enables consistent evaluation across datasets and highlights the model's robustness when facing distributional shifts from Reddit to Twitter data.

3 PROPOSED APPROACH

3.1 TRAINING GPT2 FROM SCRATCH

To develop a deeper understanding of Transformer models and training dynamics, we implemented and trained a GPT-2 model from scratch. Our architecture follows the decoder-only Transformer design introduced by Vaswani et al. Vaswani et al. (2017), consisting of stacked masked self-attention blocks, feed-forward layers, and layer normalization. We used positional encodings to inject sequential information and trained the model with a causal language modeling (CLM) objective, predicting the next token given past context.

The model was trained the fineweb-edu dataset. Due to hardware constraints, training was limited to roughly 50% of the originally planned number of epochs. nevertheless, the model exhibited steadily decreasing loss during training. To validate the learning progress, we evaluated it on the HellaSwag benchmark and observed competitive results relative to GPT-2 models trained under similar conditions. This part was performed by other members in my group.

3.2 FINETUNING

After pre-training, we fine-tune our GPT-2 model for multi-label emotion classification on the GoEmotions dataset. To adapt the model, we append a lightweight classification head consisting of mean pooling, a linear projection with LeakyReLU activation, a dropout layer ($p = 0.2$), and a final Sigmoid-activated output layer predicting independent probabilities for each of the 28 emotion classes.

The model is trained with Binary Cross-Entropy (BCE) loss, which encourages probabilistic outputs suitable for later calibration. We use the AdamW optimizer Loshchilov & Hutter (2019) with a learning rate of 1×10^{-4}, weight decay of 0.01, batch size of 64, and train for 10 epochs. All parameters, including the GPT-2 backbone, are fine-tuned jointly.

During inference, we apply class-specific thresholds inversely proportional to label frequency, improving recall for rare emotions and mitigating majority-class bias. Model selection is based on validation accuracy and macro F1 score. This part was performed by other members in my group.

3.3 BAYESIAN INFERENCE

We use the following post-hoc Bayesian inference techniques to approximate the posterior distribution $P(W \mid D)$, where post-hoc refers to applying these methods after the model has already been trained using standard maximum likelihood estimation. This approach allows us to retrofit uncertainty estimation onto existing models without havin to change the orginal training pipeline.

3.3.1 LAPLACE APPROXIMATION (LA)

Laplace Approximation is a second-order method that approximates the posterior distribution $P(W \mid D)$ as a multivariate Gaussian centered at the maximum a posteriori (MAP) estimate W^*. Specifically, it assumes the posterior is approximately Gaussian in the neighborhood of W^*. In Deep Learning the model parameters $W \in \mathbb{R}^D$ are optimized as:

$$W^* = \arg\min_{W \in \mathbb{R}^D} \mathcal{L}(D; W) = \arg\min_{W \in \mathbb{R}^D} \left[r(W) + \sum_{n=1}^{N} \ell(x_n, y_n; W) \right], \tag{3}$$

where $\ell(x_n, y_n; W) = -\log P(y_n \mid f_W(x_n))$ is the negative log-likelihood, and $r(W) = -\log P(W)$ is a regularization term corresponding to the negative log-prior (e.g., weight decay $\frac{1}{2\gamma^2}\|W\|^2$). The posterior distribution is then given by Bayes' rule:

$$P(W \mid D) = \frac{1}{Z} P(D \mid W) P(W) = \frac{1}{Z} \exp(-\mathcal{L}(D; W)), \tag{4}$$

where $Z = \int \exp(-\mathcal{L}(D; W)) dW$ is the marginal likelihood or model evidence, however this is intractable. LA uses a second-order Taylor expansion around the MAP estimate W^*:

$$\mathcal{L}(D; W) \approx \mathcal{L}(D; W^*) + \frac{1}{2}(W - W^*)^\top H(W^*)(W - W^*), \tag{5}$$

where $H(W^*) = \nabla^2_W \mathcal{L}(D; W)\big|_{W^*}$ is the Hessian of the negative log-posterior evaluated at W^*. This leads to the Laplace posterior approximation:

$$P(W \mid D) \approx \mathcal{N}(W; W^*, \Sigma), \quad \text{where } \Sigma = H(W^*)^{-1}. \tag{6}$$

where H is the Hessian of the negative log-likelihood evaluated at W^*. In our case, we apply LA to only the classification head, where W corresponds to the final layer parameters, allowing us to reduce computational cost. We use the library laplace-redux Daxberger et al. (2022) to handle the computation and storage of the inverted Hessian Matrix and the approximation of the Predictive Distribution $P(Y|X; D)$. We get the predictive distribution under this posterior is by collect K samples from the posterior distribution and integrate over it like Eqn 2, where $W^k \sim \mathcal{N}(W; W^*, \Sigma)$.

3.3.2 MC DROPOUT

Dropout Srivastava et al. (2014) is a regularization technique introduced to prevent overfitting in neural networks by randomly setting a subset of activations to zero during training. During each training iteration, neurons are dropped independently with a fixed probability p, effectively sampling a different sub-network on every forward pass.

Monte Carlo Dropout (MC Dropout) builds on this by treating dropout as a form of approximate variational inference Gal & Ghahramani (2016). In this framework, the stochasticity induced by dropout is seen as implicitly sampling from a variational posterior distribution $q(W)$ over the model weights. At test time, dropout is kept active to simulate sampling from this posterior, enabling uncertainty estimation through repeated stochastic forward passes. So the predicitive distribution in this case will be given by Eqn 2, where $q(W)$ will be the approximate posterior distribution over weights induced by dropout.

3.3.3 STOCHASTIC GRADIENT HAMILTON MONTE CARLO (SGHMC)

We can use MCMC methods to get samples from the posterior distribution, and Hamiltonian Monte Carlo (HMC) Neal (2011) is one such technique that leverages Hamiltonian dynamics to efficiently sample from complex posterior distributions. HMC augments the model parameters $W \in \mathbb{R}^D$ with auxiliary momentum variables $V \in \mathbb{R}^D$ and simulates their joint evolution according to Hamiltonian dynamics. The total energy of the system is defined as:

$$\mathcal{H}(W, V) = U(W) + K(V), \tag{7}$$

where $U(W) = -\log P(W \mid D)$ is the potential energy (negative log-posterior) and $K(V) = \frac{1}{2} V^\top M^{-1} V$ is the kinetic energy, with M typically being the identity matrix. By simulating the Hamiltonian system using a leapfrog integrator, HMC proposes new states (W', V') that are accepted or rejected using a Metropolis-Hastings step. These new states preserve the total energy,

leading to efficient exploration with high acceptance rates. The SGHMC update rule using leapfrog integration for sampling is:

$$V_{t+\frac{1}{2}} = V_t - \frac{\eta}{2}\nabla U(W_t), \tag{8}$$

$$W_{t+1} = W_t + \eta V_{t+\frac{1}{2}}, \tag{9}$$

$$V_{t+1} = V_{t+\frac{1}{2}} - \frac{\eta}{2}\nabla U(W_{t+1}), \tag{10}$$

However, traditional HMC requires full-batch gradients and expensive numerical integration steps. To address this, Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) introduces two key modifications that make HMC scalable to modern deep networks:

- **Stochastic gradients:** Instead of computing gradients over the full dataset, SGHMC uses mini-batch stochastic estimates of the gradient, $\nabla\tilde{U}(W)$, where $\tilde{U}(W)$ is the stochastic approximation of the potential energy $-\log P(W \mid D)$.
- **Friction and noise:** To compensate for the noise introduced by stochastic gradients, SGHMC adds a friction term α and Gaussian noise ϵ_t, ensuring the correct stationary distribution is preserved.

The SGHMC update rules are given by:

$$V_{t+1} = (1-\alpha)V_t - \eta\nabla\tilde{U}(W_t) + \sqrt{2\eta\alpha}\,\epsilon_t, \tag{11}$$

$$W_{t+1} = W_t + V_{t+1}, \tag{12}$$

here V_t is the momentum variable, η is the learning rate, α is the friction coefficient, and $\epsilon_t \sim \mathcal{N}(0, I)$ is Gaussian noise. The term $\nabla\tilde{U}(W_t)$ represents the mini-batch gradient of the negative log-posterior.

In our implementation, we apply SGHMC only to the classification head of the fine-tuned GPT-2 model, keeping the backbone frozen. After burn-in of 50 steps, we collect about 50 samples of the classifier weights, enabling approximate posterior predictive estimation through Equation 2.

3.3.4 Temperature Scaling

Neural networks trained with MLE or MAP often produce poorly calibrated confidence estimates, typically being overconfident even when wrong Guo et al. (2017). Temperature scaling is a simple, non-Bayesian post-hoc method that addresses this issue by rescaling the model's logits at inference time without modifying the model's weights or introducing a posterior distribution.

Given uncalibrated logits z, temperature scaling introduces a learnable scalar $T > 0$ and modifies the predictive probabilities as:

$$\hat{p}_i = \sigma\left(\frac{z_i}{T}\right), \tag{13}$$

where $\sigma(\cdot)$ is the sigmoid function. Higher values of T lead to softer predictions, improving calibration without affecting classification accuracy.

To find the optimal temperature, we minimize the negative log-likelihood (NLL) on a held-out validation set:

$$T^* = \arg\min_{T>0} \frac{1}{N}\sum_{i=1}^{N} \ell\left(y_i, \sigma\left(\frac{z_i}{T}\right)\right), \tag{14}$$

Since only a single parameter is tuned, the optimization is lightweight and converges quickly. Temperature scaling serves as a strong, computationally cheap baseline for improving model calibration.

4 Related Works

Several alternative Bayesian inference methods for uncertainty estimation in deep learning have been proposed beyond those explored in this work.

Mean-field Variational Bayes (MFVI) Blundell et al. (2015) approximates the posterior by assuming a fully factorized Gaussian distribution over the model weights, leading to a tractable but potentially oversimplified uncertainty representation. Bayes by Backprop (BBB) Blundell et al. (2015) is a practical implementation of MFVI for neural networks, leveraging the reparameterization trick to optimize the evidence lower bound (ELBO) via stochastic gradient descent.

A counterpart to SGHMC is Stochastic Gradient Langevin Dynamics (SGLD) Welling & Teh (2011) injects carefully scaled Gaussian noise into the gradients during SGD updates, allowing the training trajectory to asymptotically sample from the true posterior distribution.

SWAG (Stochastic Weight Averaging Gaussian) Maddox et al. (2019) approximates the posterior by fitting a Gaussian distribution with a low-rank plus diagonal covariance structure to the trajectory of weights observed during SGD, offering a scalable and efficient posterior approximation without retraining.

Beyond explicitly Bayesian approaches, Deep Ensembles Lakshminarayanan et al. (2017) train multiple independently initialized models and combine their predictions to estimate epistemic uncertainty, often outperforming more principled Bayesian approximations in practice. Bootstrap Ensembles Osband et al. (2016) extend this idea by training models on different bootstrapped (resampled) versions of the dataset, encouraging diversity in learned representations and improving uncertainty quantification.

Finally, for post-hoc calibration, Dirichlet Calibration and Vector Scaling Kull et al. (2019) generalize Temperature Scaling by fitting more flexible transformations over the logit outputs, enabling improved calibration at the cost of slightly more complex post-processing.

5 RESULTS

In order to evaluate both the predictive performance, uncertainty calibration and scalability of our Bayesian inference methods, we employ the following metrics: accuracy, log loss, expected calibration error (ECE), Brier score and wall clock time.

Accuracy and log loss assess the model's correctness but they do not quantify how well predicted probabilities align with true outcomes. ECE measures this alignment, how calibrated a model is. To compute ECE, predictions are grouped into M bins based on their predicted confidence scores. For each bin B_m, we calculate:

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}(\hat{y}_i = y_i), \quad \text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i, \tag{15}$$

where $\hat{p}_i$ is the predicted probability (confidence) of the predicted class $\hat{y}_i$.

ECE is then computed as:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\text{acc}(B_m) - \text{conf}(B_m)\right|, \tag{16}$$

where n is the total number of predictions. A lower ECE indicates better calibration, meaning the predicted probabilities are more trustworthy. The Brier score provides a strictly proper scoring rule to evaluate the accuracy of probabilistic predictions. It is defined for binary and multi-label classification tasks as:

$$\text{Brier} = \frac{1}{nC} \sum_{i=1}^{n} \sum_{c=1}^{C} \left(\hat{p}_{i,c} - y_{i,c}\right)^2, \tag{17}$$

where n is the number of samples, C is the number of classes, $\hat{p}_{i,c}$ is the predicted probability for class c on instance i, and $y_{i,c} \in \{0, 1\}$ is the one-hot encoded ground truth label.

Unlike accuracy, which treats predictions as hard decisions, the Brier score penalizes both overconfidence and underconfidence, making it a valuable measure for calibrated probabilistic models. Lower scores indicate more accurate and better-calibrated predictions.

We evaluate each inference method under two settings: in-distribution (ID) performance on the GoEmotions test set, and out-of-distribution (OOD) generalization using the EmoInt dataset.

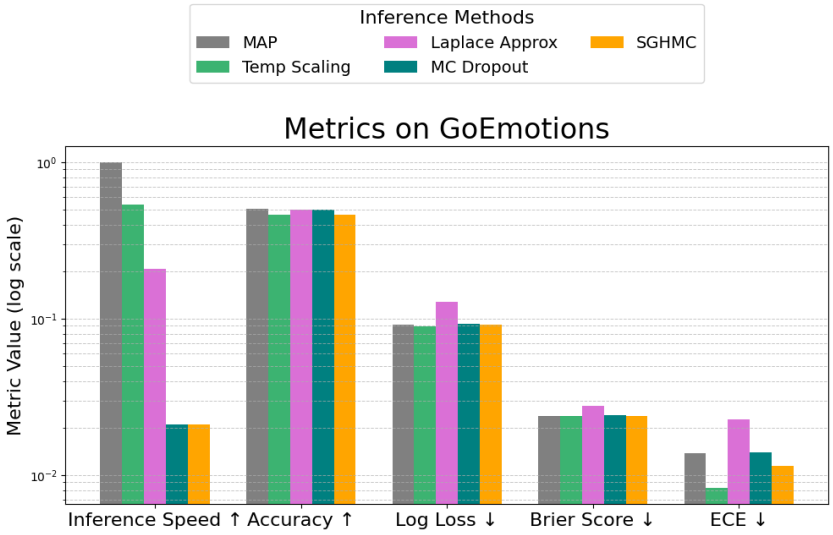

Figure 1: Comparison of inference methods across multiple metrics on the in-distribution GoEmotions test set. Inference speed is computed as the ratio of the fastest method's inference time to each method's inference time, where higher values indicate faster inference relative to the MAP baseline. The y-axis is shown on a logarithmic scale to highlight differences.

As shown in Figure 1, all methods achieve comparable accuracy and log loss on GoEmotions test set, with Bayesian inference techniques having minimal impact on predictive performance. In terms of calibration, Temperature Scaling performs the best, achieving the lowest ECE and Brier score, followed closely by SGHMC. MC Dropout does not offer significant improvements in either accuracy or calibration, suggesting that its variance injection is less effective in this setting. Surprisingly, Laplace Approximation performs the worst across all metrics, degrading both the log loss and calibration compared to the MAP baseline. Despite trying multiple configurations within the `laplace-redux` framework, Laplace Approximation consistently underperformed. A likely reason for this behavior is the use of Kronecker-factored approximations to the Hessian, as computing and storing the full inverted Hessian matrix was computationally infeasible given the 22k parameters in the fine-tuned classification head. Efficient inference is a critical consideration when selecting uncertainty quantification methods for real-world deployment. In high-throughput or latency-sensitive applications, the computational overhead introduced by Bayesian approximations must be weighed against their calibration benefits. For our experiments we chose these hyperparameters across inference methods which balance computational feasibility and posterior approximation quality:

- **Laplace Approximation (LA):** We use a Kronecker-factored Hessian approximation over the final classification head, drawing $n = 100$ Monte Carlo samples from the Laplace posterior during inference.
- **MC Dropout:** We perform 30 stochastic forward passes at test time, each with independently sampled dropout masks, and average the resulting logits.
- **SGHMC:** We collect 50 posterior samples of the classification head after a burn-in phase of 50 iterations, using a small learning rate (1×10^{-4}) and low injected noise (1×10^{-4}).
- **Temperature Scaling:** We tune a single scalar temperature parameter on the validation set using L-BFGS optimization, with a maximum of 50 iterations.

From Figure 1, we observe that methods like Laplace Approximation and Temperature Scaling introduce only a small inference overhead relative to the MAP baseline, where the overhead for Temp. Scaling is a fixed one of tuning the value of T on the validation set. In contrast, MC Dropout and SGHMC are significantly slower. This slowdown is expected: MC Dropout inference time grows linearly with the number of forward passes, and SGHMC prediction cost scales linearly with the

number of posterior samples collected. Thus, while these methods offer stronger Bayesian approximations, their computational burden must be carefully considered, particularly for deployment at scale.

Distribution shift presents a major challenge for predictive uncertainty: models trained on a source distribution often become miscalibrated and overconfident when evaluated on unseen, out-of-distribution (OOD) data. Accurate uncertainty estimation under such shifts is crucial for detecting prediction failures and enabling robust deployment.

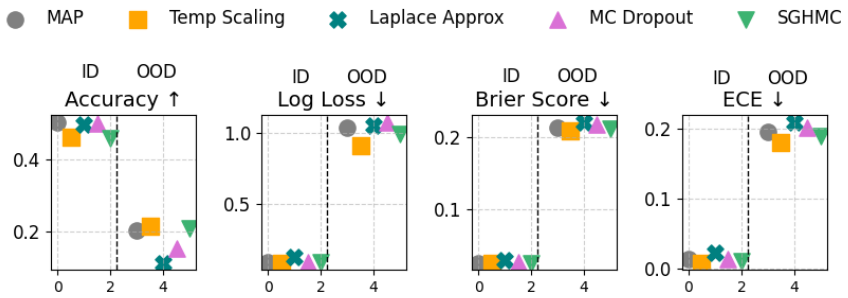

Figure 2: Comparison of inference methods on in-distribution (ID) and out-of-distribution (OOD) settings across key metrics. Each subplot shows performance on GoEmotions (ID) and EmoInt (OOD)

Figure 2 shows a comparison between in-distribution (ID) and OOD performance across key metrics. As expected, all methods experience a degradation in accuracy and an increase in log loss and calibration errors under OOD conditions. Notably, Temperature Scaling and SGHMC maintain superior calibration (lower ECE and Brier scores) even under distribution shift. MC Dropout, while modestly improving calibration compared to MAP, still exhibits notable overconfidence. Laplace Approximation continues to perform poorly relative to other methods, further increasing log loss and calibration error on the OOD benchmark. These results reinforce that lightweight techniques like Temperature Scaling can significantly improve model reliability under shift, while heavier Bayesian approximations such as SGHMC offer further gains at the cost of inference speed.

REFERENCES

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pp. 1613–1622. PMLR, 2015.

Tianqi Chen, Emily B. Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo, 2014. URL https://arxiv.org/abs/1402.4102.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux – effortless bayesian deep learning, 2022. URL https://arxiv.org/abs/2106.14806.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. Goemotions: A dataset of fine-grained emotions, 2020. URL https://arxiv.org/abs/2005.00547.

Venkatesh Duppada and Sushant Hiray. Seernet at emoint-2017: Tweet emotion intensity estimator, 2017. URL https://arxiv.org/abs/1708.06185.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 1050–1059, 2016. URL https://proceedings.mlr.press/v48/gal16.html.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Meelis Kull, Tiago Silva Filho, and Peter Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

David J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992. doi: 10.1162/neco.1992.4.3.415. URL https://doi.org/10.1162/neco.1992.4.3.415.

Wesley Maddox, Timur Garipov, Pavel Izmailov, Dmitry Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Radford M. Neal. MCMC using hamiltonian dynamics. In Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng (eds.), *Handbook of Markov Chain Monte Carlo*, pp. 113–162. Chapman and Hall/CRC, 2011. URL https://doi.org/10.1201/b10905-6.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, 2016.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale, 2024. URL https://arxiv.org/abs/2406.17557.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI*, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017. URL https://papers.nips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pp. 681–688. PMLR, 2011.