Fine-Tuning Language Models via Epistemic Neural Networks

Ian Osband 1  Mohammad Asghari 1  Benjamin Van Roy 1
Nat McAleese 1  John Aslanides 1  Geoffrey Irving 1

Abstract

Language models often pre-train on large unsupervised text corpora, then fine-tune on additional task-specific data. However, typical fine-tuning schemes do not prioritize the examples that they tune on. We show that, if you can prioritize informative training data, you can achieve better performance while using fewer labels. To do this we augment a language model with an epinet: a small additional network that helps to estimate model uncertainty and forms an epistemic neural network (ENN). ENNs are neural networks that can know what they don’t know. Using an epinet to prioritize uncertain data, we can fine-tune BERT on GLUE tasks to the same performance while using 2x less data than training without prioritization. We also investigate performance in synthetic neural network generative models designed to build understanding. In each setting, using an epinet outperforms heuristic active learning schemes.

1. Introduction

Large language models (LLMs) have emerged as one of the most exciting new developments in artificial intelligence research (Brown et al., 2020). Typically, large autoregressive transformers (Vaswani et al., 2017) are trained on large corpora of internet text data simply to predict the next token in an unsupervised manner. In spite of this simple training objective, these models have gone on to demonstrate impressive performance across a wide variety of tasks, and under different evaluation protocols (Rae et al., 2021). Importantly, these capabilities seem to improve with both increased computation and increased training data (Kaplan et al., 2020; Hoffmann et al., 2022).

LLMs are famously able to demonstrate impressive few-shot learning capabilities, induced by their ability to complete specially-designed text ‘prompts’ (Brown et al., 2020). However, these capabilities can typically be improved by fine-tuning the model on additional training data specialized to the particular application of interest (Ziegler et al., 2019; Cobbe et al., 2021). Beyond raw capability, this fine-tuning is also essential for language model alignment: to make sure the system accurately reflects the goals of the algorithm designer (Ouyang et al., 2022). This setting has even been proposed as an important case-study for the more general challenge of AI alignment and existential risk (Aspell et al., 2021).

One of the key challenges at the heart of the alignment problem comes from dealing with uncertainty. LLMs, as they are currently instantiated, are conventional neural networks that do not distinguish irreducible uncertainty over the next token (e.g. when the outcomes is produced by a rolling a die), from uncertainty that could be resolved with more training data (e.g. when the outcome is determined by recent history but the model has not been trained enough to know this). This distinction is sometimes referred to as the distinction between aleatoric (relating to chance) and epistemic (relating to knowledge) uncertainty (Kendall & Gal, 2017). Even though experimental results suggest LLMs can be trained to estimate uncertainty over the validity of their claims (Kadavath et al., 2022), this does not address this key distinction in the sources of uncertainty in ways that are necessary to accelerate learning. Previous attempts to augment LLMs with ensemble-based uncertainty have not been successful, and suggest that maybe a new training paradigm is necessary to incorporate this distinction (Gleave & Irving, 2022).

In this paper, we explore an alternative approach to supplement LLMs with an epinet: a small additional network trained to estimate model uncertainty (Osband et al., 2021). Taken together, the base LLM and epinet form an epistemic neural network (ENN). ENNs are neural networks that can know what they don’t know. Figure 1 offers a preview of the results of Section 4, where we examine a model based on BERT fine-tuned.
on GLUE datasets (Devlin et al., 2019; Wang et al., 2018). We consider a baseline that does not prioritize training data. We see that, using an epinet to prioritize training labels, we can match the performance of this baseline with 2x less data. We also see significant improvements in final accuracy from using an an epinet.

1.1. Key contributions

We show that LLMs can be augmented with an epinet to deliver effective results in a computationally scalable manner. We first present results with a synthetic neural-net generative model (Osband et al., 2022). This model has been used previously to demonstrate that the epinet produces useful estimates of epistemic uncertainty (Osband et al., 2021). We show that prioritizing data using an epinet-enhanced model reduces the number of training labels required to match baseline performance. We then demonstrate a similar outcome fine-tuning BERT, where we obtain baseline performance using 2x less data (Section 4).

We demonstrate improvement over prioritizing based on other heuristic approaches. Prior work has suggested that prioritizing by LLM class probabilities, which do not distinguish epistemic from aleatoric uncertainty, can be an effective heuristic for active learning. We reproduce these results, but find that our epinet-enhanced model leads to further improvement.

Finally, the epinet is at least as effective as existing approaches to Bayesian deep learning. Where prior work had concluded that ensemble approaches to uncertainty estimation were not effective for LLMs, we find that both ensemble-based approaches and dropout can also drive improved active learning (Gleave & Irving, 2022). However, even when compared to these well-tuned baselines, we find that using an epinet is able to match performance at a lower computational cost. Although this paper focuses on data efficiency, computational concerns may be an important consideration in other applications.

1.2. Related work

Our work is motivated by the emerging importance of large pre-trained models across language (Brown et al., 2020), images (Radford et al., 2021) and beyond (Bommasani et al., 2021). While these ‘foundation models’ are trained on huge amount of unsupervised data, their performance on downstream tasks can be typically boosted by fine-tuning on additional task-specific examples (Ziegler et al., 2019; Cobbe et al., 2021). Fine-tuning a ‘foundation model’ can be much more sample efficient than re-training from scratch, but may still require substantial amounts of data (Stiennon et al., 2020).

The field of active learning has studied the problem of optimizing which data to collect (Settles, 2009). Research in this area has unearthed a wide variety of heuristic approaches (Lewis, 1995; Seung et al., 1992), and which can be successful in particular benchmarks (Settles & Craven, 2008; Sadigh et al., 2017; Beluch et al., 2018). The problem has been most comprehensively studied through a Bayesian lens, through which the problem can naturally be framed as one to maximize the information gain through training data (Houlsby et al., 2011). While the Bayesian perspective offers a coherent statistical perspective, the computational costs of exact Bayesian inference can become intractable even for small systems (Welling & Teh, 2011).

Our paper seeks to find tractable computational solutions to active learning that are compatible with LLMs. Related work has studied similar approximation schemes applied to deep learning, but with a focus on image datasets (Gal et al., 2017; Christiano et al., 2017; Kirsch et al., 2019). However, other papers have shown that the proposed method of dropout for Bayesian inference can be of very poor quality (Osband, 2016; Hron et al., 2017; Osband et al., 2022). These results have been complemented by an empirical evaluation that concluded a fundamental rethink was necessary for effective uncertainty in LLMs (Gleave & Irving, 2022).

In this paper, we build on the recent development of epistemic neural networks (ENNs), which can offer a new and more effective approach for neural networks that know what they don’t know (Osband et al., 2021). Our work complements other recent investigations into the feasibility of active learning with LLMs (Dor et al., 2020; Margatina et al., 2022). Both of these papers show that heuristic methods for active learning can
improve data efficiency in fine-tuning. Our work reproduces this finding, and shows that prioritization based on more principled uncertainty estimates, as afforded by ENNs, can be even more effective.

2. Problem formulation

This section reviews the notation necessary to describe the agents we consider. We formalize the active learning problem, and review benchmark approaches to label prioritization in the literature. As part of our submission we also open-source our code in Appendix A.

2.1. Epistemic neural networks

We examine the performance of agents based on epistemic neural networks (ENN) (Osband et al., 2021). A conventional neural network is specified by a parameterized function class \( f \), which produces an output \( f_\theta(x) \) given parameters \( \theta \) and an input \( x \). An ENN is specified by a parameterized function class \( f \) and a reference distribution \( P_Z \). The output \( f_\theta(x, z) \) of an ENN depends additionally on an epistemic index \( z \), sampled from the reference distribution \( P_Z \). Variation of the network output with \( z \) indicates uncertainty that might be resolved by future data. All conventional neural networks can be written as ENNs, but this more general framing allows an ENN to represent the kinds of uncertainty necessary for useful joint predictions (Wen et al., 2022).

ENN share many similarities with Bayesian neural networks (BNNs), which maintain a posterior distribution over plausible neural nets. However, unlike BNNs, ENNs do not necessarily ascribe Bayesian semantics to the unknown parameters of interest, and do not generally update with Bayes rule. All BNNs can be expressed as ENNs; for example, an ensemble of \( K \) networks \( f_{\theta_1}, ..., f_{\theta_K} \) can be written as an ENN \( \bar{f} \) with index \( z \sim \text{Unif}(\{1, ..., K\}) \) and \( f_\theta(x, z) := f_{\theta_z}(x) \) (Osband & Van Roy, 2015; Lakshminarayanan et al., 2017). However, there are some ENNs that cannot be expressed naturally as BNNs.

One such example of novel ENNs is the epinet: a small additional network designed to estimate uncertainty. An epinet is added to a base network: a conventional NN with base parameters \( \zeta \) and input \( x \) with output \( \mu_\zeta(x) \). The epinet acts on a subset of features \( \phi_\zeta(x) \) derived from the base network, as well as an epistemic index \( z \) sampled from the standard normal in \( D_Z \) dimensions. For concreteness, you might think of \( \mu_\zeta(z) \) as an LLM, with features \( \phi_\zeta(x) \) as the final hidden layer in the model. For epinet parameters \( \eta \), this produces a combined output:

\[
\begin{align*}
\text{ENN} & \quad \mu_\zeta(x) + \sigma_\eta(\phi_\zeta(x), z).
\end{align*}
\]

The ENN parameters \( \theta = (\zeta, \eta) \) include those of the base network and epinet. The epinet \( \sigma_\eta \) is a simple MLP architecture, with an internal prior function designed to create an initial variation in index \( z \) (Osband et al., 2018). The “stop gradient” notation \( \text{sg}[\cdot] \) indicates the argument is treated as fixed when computing a gradient. For example, \( \nabla_\theta f_\theta(x, z) = \nabla_\zeta \mu_\zeta(x), \nabla_\eta \sigma_\eta(\phi_\zeta(x), z) \). This approach has been shown to outperform ensembles of hundreds of particles at a computational cost less than that of two particles. For more detail on the epinet we recommend reviewing (Osband et al., 2021).

2.2. Active learning

We consider the active learning setting, where a learning agent is able to prioritize training examples in order to improve performance on held out data. More formally, we consider a training dataset \( D = \{(x_i, y_i)\}_{i=1}^N \) for inputs \( x_i \) and class labels \( y_i \). Initially, at timestep \( t = 0 \), the agent can observe the training inputs \( D_X = \{(x_i, i)\}_{i=1}^N \) but none of the associated class labels. At each timestep \( t \), the agent selects \( a_t \in \{1, ..., N\} \) and reveals the corresponding class label \( y_{a_t} \). The agent can use the data \( D_{t-1} = D_X \cup \{y_{a_s}\}_{s=0}^{t-1} \) to update its beliefs, which we encode in the ENN parameters \( \theta_t \).

We assess the quality of active learning agents through their loss evaluated on a held-out dataset \( \mathcal{D} = \{(\tilde{x}_i, \tilde{y}_i, i)\}_{i=1}^N \). Examples of such evaluation might include classification error or log-loss evaluated on a test set. Agents that successfully prioritize their training labels to achieve lower held-out loss with fewer training labels can be said to perform better. Our next subsection provides some concrete approaches to prioritizing training labels.

2.3. Priority functions

We focus on active learning schemes that prioritize training labels based on a priority function \( g \). The priority function maps ENN parameters \( \theta \) and input \( x \) to a score \( g(\theta, x) \in \mathbb{R} \). We focus our attention on the problem of classification, where the ENN makes predictions over class labels \( c \in \{1, ..., C\} \). We introduce notation for the class probabilities as predicted by an ENN,

\[
\begin{align*}
\text{ENN} & \quad \mu_\zeta(x) + \sigma_\eta(\phi_\zeta(x)), \\
\text{epinet} & \quad \int_Z P_Z(dz)p(c|\theta, x, z).
\end{align*}
\]

The ENN parameters \( \theta = (\zeta, \eta) \) include those of the base network and epinet. The epinet \( \sigma_\eta \) is a simple MLP architecture, with an internal prior function designed to create an initial variation in index \( z \) (Osband et al., 2018). The “stop gradient” notation \( \text{sg}[\cdot] \) indicates the argument is treated as fixed when computing a gradient. For example, \( \nabla_\theta f_\theta(x, z) = [\nabla_\zeta \mu_\zeta(x), \nabla_\eta \sigma_\eta(\phi_\zeta(x), z)] \). This approach has been shown to outperform ensembles of hundreds of particles at a computational cost less than that of two particles. For more detail on the epinet we recommend reviewing (Osband et al., 2021).

2.2. Active learning

We consider the active learning setting, where a learning agent is able to prioritize training examples in order to improve performance on held out data. More formally, we consider a training dataset \( D = \{(x_i, y_i)\}_{i=1}^N \) for inputs \( x_i \) and class labels \( y_i \). Initially, at timestep \( t = 0 \), the agent can observe the training inputs \( D_X = \{(x_i, i)\}_{i=1}^N \) but none of the associated class labels. At each timestep \( t \), the agent selects \( a_t \in \{1, ..., N\} \) and reveals the corresponding class label \( y_{a_t} \). The agent can use the data \( D_{t-1} = D_X \cup \{y_{a_s}\}_{s=0}^{t-1} \) to update its beliefs, which we encode in the ENN parameters \( \theta_t \).

We assess the quality of active learning agents through their loss evaluated on a held-out dataset \( \mathcal{D} = \{(\tilde{x}_i, \tilde{y}_i, i)\}_{i=1}^N \). Examples of such evaluation might include classification error or log-loss evaluated on a test set. Agents that successfully prioritize their training labels to achieve lower held-out loss with fewer training labels can be said to perform better. Our next subsection provides some concrete approaches to prioritizing training labels.

2.3. Priority functions

We focus on active learning schemes that prioritize training labels based on a priority function \( g \). The priority function maps ENN parameters \( \theta \) and input \( x \) to a score \( g(\theta, x) \in \mathbb{R} \). We focus our attention on the problem of classification, where the ENN makes predictions over class labels \( c \in \{1, ..., C\} \). We introduce notation for the class probabilities as predicted by an ENN,

\[
\begin{align*}
\text{ENN} & \quad \mu_\zeta(x) + \sigma_\eta(\phi_\zeta(x)), \\
\text{epinet} & \quad \int_Z P_Z(dz)p(c|\theta, x, z).
\end{align*}
\]
For any probability distribution \( p \) over \( C \) classes we recall that the \emph{entropy} is defined \[ \mathbb{H}[p] := - \sum_c p(c) \log p(c) \] (Shannon, 2001). We can now describe the priority functions used in this paper. The key baseline for our experiment comes from the \textbf{uniform} prioritization, which does not distinguish between inputs, \( g^{\text{uniform}} = 0 \).

Our next two priorities require only conventional neural network predictions, and do not account for variability in the epistemic index \( z \). We call these approaches \textit{marginal} priority functions since they only depend on the marginal probability estimates for a single input \( x \). We define the \textbf{entropy} prioritization \[ g^{\text{entropy}}(\theta, x) = \mathbb{H}[p(\cdot|x, \theta)] \]

Similarly, \textbf{margin} (Roth & Small, 2006) prioritizes based on the smallest difference between the highest and second highest class probabilities, \[ g^{\text{margin}}(\theta, x) = p(c_2|\theta, x) - p(c_1|\theta, x), \]

where \( c_1, c_2 \in \arg\max_y p(c|\theta, x) \) and \( c_2 \neq c_1 \). Both of these priorities are maximized by the uniform distribution, but neither distinguish genuinely ambiguous examples, from those with insufficient data (Osband et al., 2021).

Our final approaches use the ENN to prioritize uncertain, as opposed to ambiguous, datapoints. We can call these priority functions \textit{epistemic} priority functions, since they depend on the joint distribution of predictions. We define \textbf{bald} (Houlsby et al., 2011) priority based on the mutual information gain: \( g^{\text{bald}}(\theta, x) = \mathbb{H}[p(\cdot|\theta, x)] - \int_z \mathbb{E}_z [p(\cdot|\theta, x, z)] \), Similarly, \textbf{variance} uses the variation in predicted probabilities, \[ g^{\text{variance}}(\theta, x) = \mathbb{E}_z \left[ \mathbb{E}_z [p(\cdot|\theta, x, z)] - p(c|\theta, x) \right]^2. \]

Both of these methods prefer to select training examples with high variability in prediction in the epistemic index \( z \).

### 2.4. Training algorithm

In our experiments, we consider a variant of stochastic gradient descent that proceeds in steps \( s = 0, 1, 2, \ldots \). At gradient step \( s \), a large candidate batch of size \( N_B \) is selected uniformly at random from the training dataset \( D \) without replacement. Then, the agent selects \( N_b \leq N_B \) elements of this candidate batch to perform SGD upon. Depending on the choice of data selected, this can require between 0 and \( N_b \) new training labels to be revealed. It is most typical in the active learning literature to consider the case \( N_B = N \) and \( N_b = 1 \). However, in large scale distributed frameworks it can be useful to consider other variants that don’t require passing through the entire dataset \( N_B < N \), and which may select more than one training datapoint at a time \( N_b > 1 \) (Kirsch et al., 2019).

Algorithm 1 formalizes the training procedure we use for the experiments in this paper. For all of the experiments in this paper we use a standard cross-entropy loss term with L2 regularization,

\[ \ell^{\text{XENT}}(\theta, x, y, i):= -\ln(\text{softmax}(f_\theta(x, i)))_{y_i} + \lambda \| \theta \|^2. \]

Here, \( \lambda \) is a hyperparameter that scales the regularization penalty. Algorithm 1 is not meant to be the best possible active learning scheme, but provides a lower bound on what is possible with active learning. We compare this approach varying the priority function and ENN like-for-like, as well as a non-prioritized baseline together with optimal early stopping. We see that, compared to both baselines, the epinet-enabled model is able to learn much faster.

### 3. Synthetic data

This section evaluates active learning schemes in a simple synthetic problem designed around a neural network generative model. We make minor modifications to the Neural Testbed (Osband et al., 2022) to create a new open-source active learning benchmark (Appendix A). We find that, using an epinet, we can greatly reduce the necessary amount of training labels.

#### 3.1. Environment

We consider a generative model based on data generated by a random MLP. The inputs \( x_i \in \mathbb{R}^D \) are sampled from a standard normal \( N(0, I) \). Class labels are then assigned,

\[ P(y_i = y|\theta^*) = \text{softmax}(h_{\theta^*}(x_i)/\rho)_y. \]

Here the function \( h_{\theta^*}(\cdot) \in \mathbb{R}^2 \) is a 2-layer MLP with ReLU activations and 50 hidden units at each layer. The network weights \( \theta^* \) are sampled according to Glorot initialization (Glorot & Bengio, 2010), and \( \rho > 0 \) is a temperature parameter.

In our experiments, we set input dimension \( D = 10 \), temperature \( \rho = 0.1 \), and generate \( N = 200 \) training examples. We evaluate performance through the average test log-likelihood over \( N = 1000 \) testing examples, and average our results over 10 random seeds.

This problem is clearly a small-scale and extremely simplified model of active learning for language models.

---

1By evaluating algorithms with a controlled generative model, we can avoid some of the pitfalls of evaluation on a fixed benchmark dataset. Benchmarks that rank performance on a fixed dataset are vulnerable to overfitting through iterative hill-climbing (Russo & Zou, 2015). As such, optimized solutions on fixed datasets often don’t generalize beyond the benchmark (Recht et al., 2018).
We consider baseline agents that learn based on ENN variants of a 2-layered MLP with 50 hidden units in each setting, we average the results over three random seeds to obtain standard error estimates for the quality of this baseline. We then pick the best hyperparameters per data fraction. As such, although we call this agent a baseline, this procedure is meant to provide an upper bound on the performance for any agent that does not prioritize its training data.

### 3.3. Results

Figure 2 compares learning with an epinet against our tuned supervised baseline. The agents epinet:bald and epinet:variance refer to epinet trained with bald and variance priorities. We can see that these two methods are essentially statistically indistinguishable in this setting. Our results clearly show that, by using an epinet, the learning agent can obtain the same test performance with significantly less data.

![Figure 2](image)

**Figure 2.** Learning with an epinet is able to match the baseline with fewer training labels.
uncertainty. By contrast, agents that prioritize by epistemic uncertainty (including bald and variance) are not drawn to these points.

The performance of epinet is also impressive when we compare against other approximate Bayesian approaches to uncertainty quantification. Figure 3(b) compares these agents varying only the choice of ENN architecture from epinet to dropout to ensemble. We see the epinet performs better than the competing approaches, and that the ensemble performs better than dropout in this setting. The computational cost of the epinet is much smaller than either an ensemble or dropout, since only the epinet, not the base network, needs to make a forward pass per index $z$. These results show that in this simple and sanitized setting, active learning with an epinet can be competitive with existing approaches to Bayesian deep learning.

4. **Language models**

In this section, we present the main empirical results of this paper. We show that the key insights afforded by the toy problem of Section 3 also carry over to fine-tuning language models. We review the problem formulation, and the experimental setup we use based on fine-tuning BERT (Devlin et al., 2019) in Jaxline (Babuschkin et al., 2020). Using an epinet, we can greatly reduce the necessary amount of training labels.

4.1. **Environment**

The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) is a collection of diverse natural language understanding tasks. These tasks are widely accepted as a benchmark for performance in the field, and improving performance on GLUE was one of the key markers that made BERT (Devlin et al., 2019) a seminal contribution to the field. For simplicity in a short paper, we only consider the classification tasks of GLUE, and so exclude 3 of the 11 benchmark tasks. We do not believe that including these tasks would pose any fundamental difficulty, but would require considering an alternative to the cross-entropy loss (4), such as mean-squared error.

BERT, with just over 100M parameters, is actually quite small compared to later LLMs that have scaled up to 1T+ (Chowdhery et al., 2022; Fedus et al., 2021). Larger models, trained with more data, have been shown to perform better on downstream tasks (Kaplan et al., 2020; Rae et al., 2021). These improvements have been so significant that there is a newer SuperGLUE benchmark (Wang et al., 2019), which replaces some of the easier tasks in GLUE (Wang et al., 2018). Nonetheless, we focus on BERT and GLUE because it is a clear benchmark transformer (Vaswani et al., 2017), with open-source implementations accessible to researchers without massive budgets for cloud compute. Our research provides a proof of concept that LLMs can benefit from the addition of epinet, which we expect to transfer to other related tasks. We’ve already shown in Figure 1 that epinet can lead to significant improvements on the MNLI dataset; in fact, these results extend more broadly to the full GLUE suite.

We consider a simple per task fine-tuning setting where the agents are trained on each GLUE task separately.
Our baseline agent follows the procedure from Section 3; it is trained by selecting a fixed and random subset of the data. We sweep over batch size $\in \{4, 16, 64\}$, learning rate $\in \{1e-6, 3e-6, 1e-5, 3e-5, 1e-4, 3e-4\}$. For each setting we perform 10 epochs of SGD training and select the best training step in hindsight. This baseline agent is not trained online, but is meant to provide an upper bound on how well any agent that does not prioritize its data can do.

4.2. Agents

We consider ENNs based around fine-tuning a pre-trained BERT architecture. At a high level, these ENNs work by branching from the final hidden layer in the BERT network. Table 2 outlines these architectures, together with the hyperparameters we tune. For all of our agents apart from epinet, we found no benefit to using an MLP head over a simple linear layer. This is perhaps unsurprising since BERT is already a very large model, so one additional hidden layer makes little difference. For the epinet, we use a network \[
\sigma_0(\tilde{x}, z) = (h_0(\text{concat}(\tilde{x}, z)) + \lambda h^P(\text{concat}(\tilde{x}, z)))^T z.
\]

Here $\tilde{x} := \text{sg}[\phi_\zeta(x)]$ is the last hidden feature for BERT, and $h$ is a 2-layer MLP with 50 hidden units. The MLP $h^P$ is initialized as a ‘prior network’ and has no trainable parameters (Osband et al., 2018), and $\lambda > 0$ is a scaling parameter. This part of the network serves to drive initial variance in index $z$, but that can be resolved with data. We push the details of the network implementation and open-source code to Appendix A.

We train all of our agents according to Algorithm 1 by examining $N_B = 40$ candidate inputs and then selecting $n_b = 4$ examples to train with $n_Z = 10$ index samples. In each case we tune the agent hyperparameters on the MNLI dataset, and then repeat the training and evaluation across the other GLUE tasks. For all agents, we found that learning rate $1e-5$ matched the best performance observed.

4.3. Results

Figure 1 shows that, using an epinet, we can match baseline performance on the MNLI dataset while using 2x fewer labels. Figure 4 shows that these results are impressive even when compared against other benchmark approaches to active learning. Figure 4(a) compares epinet prioritized by variance against other methods that do not make use of epistemic uncertainty. Their performance plateaus earlier as they prioritize potentially noisy examples, rather than informative training data. This mirrors our earlier result on the Neural Testbed (Figure 3(a)). Figure 4(b) compares agent performance when prioritizing by variance, changing the ENN architecture. We see that epinet performs better than the competing approaches, and the ensemble performs better than dropout in this setting. Once again, the qualitative results mirror the synthetic testbed (Figure 3(b)). Unlike the testbed, we see that using an epinet also improves the final performance of the model, not just the required number of training labels to reach that performance.

Figure 4. Active learning with an epinet is able to provide larger improvements with fewer labels than competing approaches for BERT on MNLI (Williams et al., 2017).

Figure 5 repeats the the analysis of Figure 1 and MNLI on all GLUE classification tasks. We include results for epinet prioritized by both bald and variance, as well as an epinet with uniform data prioritization. These bald and variance are essentially indistinguishable from each other in our experiments, and both significantly better than the baseline. The epinet with uniform data selection performs much worse, showing that the benefits from epinet do not come purely from the network architecture, but also the data selection scheme. In fact, on many tasks the uniform data selection does not match the baseline final performance. We provide more details on epinet performance under different priority schemes in Appendix C.
Fine-Tuning Language Models via Epistemic Neural Networks

Table 2. Summary of agent architectures, full details in Appendix C.

| agent   | description                                    | hyperparameters                        |
|---------|------------------------------------------------|-----------------------------------------|
| baseline| MLP classification layer (Devlin et al., 2019)  | learning rate, network                  |
| dropout | MLP dropout (Gal & Ghahramani, 2016)           | learning rate, network, dropout rate    |
| ensemble| MLP deep ensembles (Lakshminarayanan et al., 2017) | learning rate, network, ensemble size   |
| epinet  | MLP + MLP epinet (Osband et al., 2021)         | learning rate, network, prior, index dimension |

Figure 5. Fine-tuning BERT models across GLUE tasks. The baseline BERT is tuned over learning rate, batch size and SGD steps for each random subset of training data. Using an epinet with epistemic prioritization consistently learns with fewer labels and better final performance. Using an epinet with uniform data selection does not produce the same benefit. The choice of epistemic priority (bald vs variance) is relatively unimportant.

Figure 6. Learning with an epinet matches BERT performance in 2x less data. Using either dropout or ensemble as ENNs also performs well, but prioritizing by entropy/margin does not always reach baseline performance. The heuristic approaches of entropy and margin prioritization do not match the baseline performance on MNLI tasks, and so their ratio is taken to be infinite. Agents that use either dropout or ensemble ENNs were also able to see significant improvements in data efficiency. However, both of these network architectures have much higher computational costs than the epinet, and so are less promising for scaling to future LLM research at scale. We include further analysis and learning curves in Appendix C.

5. Conclusion

This paper looks at the problem of active learning in fine-tuning language models. We show that, using an epinet, we can augment LMs to output the kind of uncertainty estimates that are useful to prioritize label acquisition. On GLUE tasks, this allows us to train a BERT model using 2x fewer labels. Further, as we continue to train on the full datasets we actually reach better final performance. These improvements are significant, and at least match those attained by previous approaches to Bayesian deep learning. Importantly, the epinet architecture offers a way to scale this epistemic uncertainty to large pretrained language models with only model incremental computation.
Although these results are significant, we are motivated by the further potential of LLMs that know what they don’t know. In doing this, we can unlock new families of algorithms and approaches that can may lay the foundation for much larger improvements in data efficiency (Lu et al., 2021). We view this paper as an important ‘proof of concept’ that adding epistemic uncertainty estimates to language models is possible in a computationally tractable manner, and that even quite simplistic approaches that leverage this uncertainty can significantly improve data efficiency.

References

Askell, A., Bai, Y., Chen, A., Drain, D., Ganguli, D., Henighan, T., Jones, A., Joseph, N., Mann, B., DasSarma, N., et al. A general language assistant as a laboratory for alignment. arXiv preprint arXiv:2112.00861, 2021.

Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovy, P., Budden, D., Cai, T., Clark, A., Danilhka, I., Fantacci, C., Godwin, J., Jones, C., Hennigan, T., Hessel, M., Kapturowski, S., Keck, T., Kemaev, I., King, M., Martens, L., Mikulik, V., Norman, T., Quan, J., Papamakarios, G., Ring, R., Ruiz, F., Sanchez, A., Schneider, R., Sezener, E., Spencer, S., Srivivasan, S., Stokowiec, W., and Viola, F. The DeepMind JAX Ecosystem, 2020. URL http://github.com/deepmind.

Beluch, W. H., Genewein, T., Nürnberger, A., and Köhler, J. M. The power of ensembles for active learning in image classification. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 9368–9377, 2018.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258, 2021.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. arXiv preprint arXiv:2005.14165, 2020.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. arXiv preprint arXiv:2204.02311, 2022.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. Advances in neural information processing systems, 30, 2017.

Cobbe, K., Kosaraju, V., Bavarian, M., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Dor, L. E., Halfon, A., Gera, A., Shnarch, E., Dankin, L., Choshen, L., Danilevsky, M., Aharonov, R., Katz, Y., and Slonim, N. Active learning for bert: an empirical study. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 7949–7962, 2020.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2021.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In International Conference on Machine Learning, 2016.

Gal, Y., Islam, R., and Ghahramani, Z. Deep Bayesian active learning with image data. In International Conference on Machine Learning, pp. 1183–1192. PMLR, 2017.

Gleave, A. and Irving, G. Uncertainty estimation for language reward models. arXiv preprint arXiv:2203.07472, 2022.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the 13th international conference on artificial intelligence and statistics, pp. 249–256, 2010.
Fine-Tuning Language Models via Epistemic Neural Networks

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. arXiv preprint arXiv:2203.15556, 2022.

Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. Bayesian active learning for classification and preference learning. arXiv preprint arXiv:1112.5745, 2011.

Hron, J., Matthews, A. G. d. G., and Ghahramani, Z. Variational Gaussian dropout is not Bayesian. arXiv preprint arXiv:1711.02989, 2017.

Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., Schiefer, N., Dodds, Z. H., Das-Sarma, N., Tran-Johnson, E., et al. Language models (mostly) know what they know. arXiv preprint arXiv:2207.05221, 2022.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. arXiv preprint arXiv:2001.08361, 2020.

Kendall, A. and Gal, Y. What uncertainties do we need in Bayesian deep learning for computer vision? In Advances in Neural Information Processing Systems, volume 30, 2017.

Kingma, D. and Ba, J. Adam: A Method for Stochastic Optimization. Proceedings of the International Conference on Learning Representations, 2015.

Kirsch, A., Van Amersfoort, J., and Gal, Y. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. Advances in neural information processing systems, 32, 2019.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In Advances in Neural Information Processing Systems, pp. 6405–6416, 2017.

Lewis, D. D. A sequential algorithm for training text classifiers: Corrigendum and additional data. In Acm Sigir Forum, volume 29, pp. 13–19. ACM New York, NY, USA, 1995.

Lu, X., Van Roy, B., Dwaracherla, V., Ibrahimi, M., Osband, I., and Wen, Z. Reinforcement learning, bit by bit. arXiv preprint arXiv:2103.04047, 2021.

Margatina, K., Barrault, L., and Aletras, N. On the importance of effectively adapting pretrained language models for active learning. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 825–836, 2022.

Osband, I. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In NIPS Workshop on Bayesian Deep Learning, volume 192, 2016.

Osband, I. and Van Roy, B. Bootstrapped Thompson sampling and deep exploration. arXiv preprint arXiv:1507.00900, 2015.

Osband, I., Aslanides, J., and Cassirer, A. Randomized prior functions for deep reinforcement learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), Advances in Neural Information Processing Systems 31, pp. 8617–8629. Curran Associates, Inc., 2018. URL https://bit.ly/rpf_neurips.

Osband, I., Wen, Z., Asghari, M., Ibrahimi, M., Lu, X., and Van Roy, B. Epistemic neural networks. arXiv preprint arXiv:2107.08924, 2021.

Osband, I., Wen, Z., Asghari, S. M., Dwaracherla, V., Hao, B., Ibrahimi, M., Lawson, D., Lu, X., O’Donoghue, B., and Van Roy, B. The neural testbed: Evaluating joint predictions. In Advances in Neural Information Processing Systems, volume 35. Curran Associates, Inc., 2022.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. arXiv preprint arXiv:2203.02155, 2022.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In International Conference on Machine Learning, pp. 8748–8763. PMLR, 2021.

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., et al. Scaling language models: Methods, analysis & insights from training gopher. arXiv preprint arXiv:2112.11446, 2021.

Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. Do cifar-10 classifiers generalize to cifar-10? arXiv preprint arXiv:1806.00451, 2018.

Roth, D. and Small, K. Margin-based active learning for structured output spaces. In European Conference on Machine Learning, pp. 413–424. Springer, 2006.
Russo, D. and Zou, J. How much does your data exploration overfit? controlling bias via information usage. *arXiv preprint arXiv:1511.05219*, 2015.

Sadigh, D., Dragan, A. D., Sastry, S., and Seshia, S. A. *Active preference-based learning of reward functions*. 2017.

Settles, B. *Active learning literature survey*. 2009.

Settles, B. and Craven, M. An analysis of active learning strategies for sequence labeling tasks. In *proceedings of the 2008 conference on empirical methods in natural language processing*, pp. 1070–1079, 2008.

Seung, H. S., Opper, M., and Sompolinsky, H. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 287–294, 1992.

Shannon, C. E. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55, 2001.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D. M., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize from human feedback. *CoRR*, abs/2009.01325, 2020. URL https://arxiv.org/abs/2009.01325.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688. Citeseer, 2011.

Wen, Z., Osband, I., Qin, C., Lu, X., Ibrahimi, M., Dwaracherla, V., Asghari, M., and Van Roy, B. From predictions to decisions: The importance of joint predictive distributions, 2022.

Williams, A., Nangia, N., and Bowman, S. R. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.
A. Open source code

Two related github repositories complement this paper:

1. enn: https://github.com/deepmind/enn
2. neural_testbed: https://github.com/deepmind/neural_testbed

These libraries contain the code necessary to reproduce the key results in our paper, divided into repositories based on focus. Together with each repository, we include several ‘tutorial colabs’ – Jupyter notebooks that can be run in a browser without requiring any local installation. Each of these libraries is written in Python, and relies heavily on JAX for scientific computing (Bradbury et al., 2018). We view this open-source effort as a major contribution of our paper.

The first library, enn, was introduced as part of Osband et al. (2021). It focuses on the design of epistemic neural networks and their training. In this submission, we contribute additional code around the BERT network enn/networks/bert as well as the priority functions enn/active_learning.

The second library, neural_testbed, was introduced as part of The Neural Testbed (Osband et al., 2022). We add experiments around the active learning experiments considered in Section 3, which can be found in neural_testbed/active_learning.

B. Synthetic data

This section provides details about the Neural Testbed experiments in Section 3. We begin with a review of the neural testbed as a benchmark problem, and the associated generative models. We then give an overview of the baseline agents we compare against in our evaluation.

B.1. Environment

The Neural Testbed (Osband et al., 2022) is a collection of neural-network-based, synthetic classification problems that evaluate the quality of an agent’s predictive distributions. We make use of the open-source code at https://github.com/deepmind/neural_testbed. The Testbed problems use random 2-layer MLPs with width 50 to generate training and testing data. The specific version we test our agents on entails binary classification, input dimension $D = 10$, number of training samples $T = 200$, temperature $\rho = 0.1$ for controlling the signal-to-noise ratio, and 10 random seeds for generating different problems. The performance metrics are averaged across problems to give the final performance scores.

B.2. Agents

We follow Osband et al. (2022) and consider the benchmark agents as in Table 1, which includes the epinet from Osband et al. (2021). We use the open-source implementation and hyperparameter sweeps at /agents/factories. According to Osband et al. (2022), the benchmark agents are carefully tuned on the Testbed problems, so we do not further tune these agents.

C. Language models

This section provides details about the BERT fine-tuning experiments in Section 4. We begin with a review of the ‘baseline’ agent we use for evaluation in BERT. We then give an overview of hyperparameter tuning and sensitivities in the agents we investigate.

C.1. Baseline agent

A key component of our research was to set a strong baseline for BERT in this active learning setting. Since prior works have focused on a purely supervised learning setting, where the entire training set is made available at once, typical BERT implementations are not optimized for active learning. To sidestep these issues, we instead use a baseline that separately has the opportunity to retrain on different fractions of training data. For each
GLUE task, we sample a fraction $\psi$ of the data, and then optimize a supervised learning BERT baseline trained only on that fraction of the data. We then obtain the baseline learning curve shown in our figures by sweeping,

$$\psi \in \{0.01, 0.03, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}.$$ 

For each of these sub-datasets, we sweep over batch size $\in \{4, 16, 64\}$, learning rate $\in \{1e-6, 3e-6, 1e-5, 3e-5, 1e-4, 3e-4\}$, and select the best SGD step over 10 training epochs in hindsight. For each setting, we average the results over three random seeds to obtain standard error estimates for the quality of this baseline.

The performance of this ‘baseline’ agent is therefore much stronger than any naive application of BERT training to active learning. We allow the supervised learning baseline to optimize its parameters and hyperparameters purely for that problem setting (and data fraction) of interest. This means that the resultant baseline is really an upper bound on the performance that can reasonably be expected by any learning scheme that does not actively prioritize its training data.

C.2. Robustness and hyperparameters

This subsection outlines some of the key hyperparameter tunings and robustness analysis that we performed during our experiments. Due to computational costs, as well as concerns on overfitting, we tuned our agents on the MNLI matched dataset and then used these settings for the remainder of the GLUE tasks. We chose MNLI as it was the largest dataset in the GLUE task, and widely regarded as one of the most challenging tasks. We imagine that it is possible to further improve our results by tuning each approach per task, although this probably is not of much interest in terms of research contribution.

Except where otherwise stated, all of our agents use an ADAM optimizer with learning rate $1e^{-5}$, decay parameter $b_2 = 0.95$ and clipping global norm of the gradient to be at most 1, apart from that we use the default settings provided by optax (Kingma & Ba, 2015; Babuschkin et al., 2020). We train all of our agents according to Algorithm 1 by examining $N_B = 40$ candidate inputs and then selecting $n_b = 4$ examples to train on with $n_Z = 10$ index samples. This choice of $N_B = 40, N_b = 4$ is chosen for computational convenience as we distribute compute across several TPU cores, and running a fully-batched active learning scheme that evaluated much higher $N_B$ and trained on only $N_b = 1$ would have taken longer to run in wall clock time. For each batch we train with $n_Z = 10$ index samples.

C.2.1. Marginal methods

For our marginal prioritization methods we tuned the learning rate $\in \{3e-6, 1e-5, 3e-5\}$. We found that performance was quite sensitive to this parameter across both entropy and margin priority schemes. Figure 7 compares their performance on the MNLI matched dataset. We can see that none of the learning rates are able to match the performance of the baseline agent will full data.

![Figure 7](image-url)  

(a) Tuning learning rate for entropy.  
(b) Tuning learning rate for margin.

*Figure 7. Tuning learning rate on the MNLI matched dataset. Marginal prioritization methods can sometimes lead to faster learning, but have a tendency to plateau at higher test error. We found that margin and entropy prioritization consistently perform similarly.*
C.2.2. Dropout

For the dropout agent we tuned the dropout rate $p \in \{0, 0.05, 0.1, 0.2, 0.5\}$ over three seeds each on the MNLI matched dataset. We found that the dropout rate 0.1 performed best, and was the only one that matched our baseline performance over all three seeds.

![Figure 8. Varying dropout rate on the MNLI matched dataset. Only dropout rate 0.1 is able to match the baseline performance in our experiment.](image)

C.2.3. Ensemble

For the ensemble agent we vary the ensemble size $K \in \{1, 3, 10, 30\}$ over three seeds each on the MNLI matched dataset. We found that the ensemble size 10 performed best, in terms of the average number of labels required to match the baseline performance. In general, we would expect larger ensembles to perform better, but we did run with $n_Z = 10$ which may have impacted the results. We opted for size 10 since we reported our main headline numbers in terms of number of labels required to match the baseline, and wanted to run our experiments quickly in terms of wall clock time.

![Figure 9. Varying ensemble size on the MNLI matched dataset.](image)

C.2.4. Epinet

For the epinet agent we took the ‘off the shelf’ solution provided by the ENN library for ImageNet (Osband et al., 2021). This consists of a base MLP with 2-layers of 50 hidden units and ReLU activations, and an epinet with the same form and a matched prior function. Following that work, we used an index dimension of 10 and a prior scale of 1. Similar to past work, we found that the results were not very sensitive to the choice of these parameters.

To investigate the sensitivity of the epinet to the prioritization scheme, we ran the same epinet architecture with each of the prioritization schemes for three seeds on the MNLI matched dataset. We present these results in Figure 10. We can see that the epinet performs better when prioritizing by epistemic uncertainty (variance),
although the marginal prioritization methods (margin, entropy) also lead to significant improvements over the baseline. However, learning with uniform prioritization does not match baseline performance after one epoch.

![Figure 10](image)

*Figure 10.* Varying the prioritization scheme on the MNLI matched dataset. Using an epinet, we are able to beat baseline performance using marginal priority methods. However, we can obtain even better performance by focusing on datapoints with high epistemic uncertainty.

The results of Figure 10 are interesting in that they show that the epinet with marginal prioritization performs better than the baseline BERT model with the same priority scheme (Figure 7). This shows that some of the benefits of epinet:variance come from the epinet architecture without prioritizing on epistemic uncertainty. To get an idea of the benefits purely from the epinet architecture, we re-run the baseline agent of Appendix C.1 but using the epinet architecture. We call the resultant supervised learning agent the baseline-epinet and compare its performance against the baseline and epinet:variance in Figure 11. Here we can see that the baseline-epinet is essentially indistinguishable from the baseline agent. This shows that the principle benefits of learning with epinet come from the ability to prioritize data more effectively, rather than purely from improvements in network architecture in supervised learning.

![Figure 11](image)

*Figure 11.* Comparing epinet performance against supervised baselines on the MNLI matched dataset. When trained using supervised learning alone, the baseline-epinet offers no significant benefit in test accuracy over baseline.

At first glance, these results might appear confusing: how can the epinet lead to better performance with marginal prioritization methods (Figure 10) when the purely-supervised learning schemes lead to no improvements (Figure 11)? Actually, since the MNLI task is a fixed dataset without noisy labels, there is in practice a high correlation between marginal priority and epistemic uncertainty. As such, even the marginal priority functions are able to benefit from the uncertainty estimates provided by the epinet to drive faster learning in this setting.
C.3. Results by task

Figure 6 presents a summary of agent performance across all GLUE tasks. Figure 12 breaks down the performance for each agent across each task. We see that, although there is some variability across tasks, the general pattern of the results is somewhat consistent. Importantly, marginal priority methods are unable to match the baseline performance even after tuning on MNLI matched or mismatched datasets. However, on other datasets, they can be tuned to provide reasonable improvements in data-efficiency over the baseline.

Figure 12. Comparison of amount of labels required to match baseline performance across GLUE tasks. The geometric mean of these results is presented in Figure 6.