Efficient Uncertainty Quantification for Multilabel Text Classification

Jialin Yu*, Alexandra I. Cristea†, Anoushka Harit‡ Zhongtian Sun§
Olanrewaju Tahir Aduragba¶, Lei Shi‖ and Noura Al Moubayed**
Department of Computer Science, Durham University
Durham, UK
Email: {*jialin.yu, †alexandra.i.cristea, ‡anoushka.harit, § zhongtian.sun}@durham.ac.uk
{¶olanrewaju.m.aduragba, ‖lei.shi, **noura.al-moubayed}@durham.ac.uk

***Abstract*—Despite rapid advances of modern artificial intelligence (AI), there is a growing concern regarding its capacity to be *explainable, transparent, and accountable*. One crucial step towards such AI systems involves *reliable and efficient uncertainty quantification methods*. Existing approaches to uncertainty quantification in natural language processing (NLP) take a Bayesian Deep Learning approach. However, the latter is known to not be computationally efficient in testing time, thus hindering its applicability in real-life scenarios. This paper proposes a new focus on the *efficiency of uncertainty quantification methods*, evaluating them on four multi-label text classification tasks. Our novel methods of representing epistemic and aleatoric uncertainties enable efficient uncertainty quantification (around 13 to 45 times faster than existing approaches, depending on architecture) with posterior analysis in the (approximated) latent- and data space. We conduct extensive experiments and studies on diverse neural network architectures (LSTM, CNN and Transformer) to analyse their power. Our results prove the benefits of explicitly modelling uncertainty in neural networks.**

***Index Terms*—Bayesian method, neural network, deep learning, natural language processing, uncertainty quantification**

I. Introduction

Deep neural networks have been successfully applied in a wide range of natural language processing (NLP) tasks, such as text classification, question answering, and natural language inference [1]. However, modern deep neural networks are (mainly) discriminative models, with only point estimation. They can make predictions *'blindly'* [2], raising, in practice, concerns over AI safety and social bias [3]. One natural solution towards such trustworthy and robust AI systems is to combine the predictive power of Deep Learning with the statistical robustness of Bayesian Learning [4].

Combining these two powerful tools inspires two different directions, as shown in Figure 1: *Bayesian Deep Learning (BDL)* and *Deep Bayesian Learning (DBL)*. In the deep learning approach, each neuron learns a fixed value representing its parameters; in contrast, the BDL approach allows each neuron to learn a distribution of its parameters; and the DBL approach infers a latent variable and learns its distribution instead. A widely adopted BDL approach is the Bayesian Neural Network (BNN) [5]–[7]; while a known DBL approach is the Deep Generative Model (DGM) [8]–[11].

From a Bayesian modelling perspective, there exist two main types of uncertainty inside a neural network [12], namely

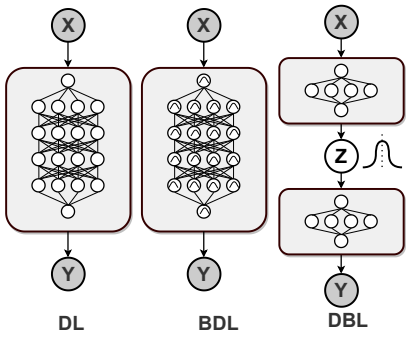

Fig. 1. Graphical illustrations of the differences between 'typical' Deep Learning (DL), Bayesian Deep Learning (BDL) and Deep Bayesian Learning (DBL).

epistemic uncertainty and *aleatoric* uncertainty. Epistemic uncertainty represents model uncertainty resulting from ignorance about model assumptions. It can be reduced when more data are observed, as more information leads to better model assumptions. Aleatoric uncertainty is known as data uncertainty and reflects noise inherent in the data, i.e., the deviation between ground truth and observed values. It cannot be reduced, even if more data are observed, as the deviation comes from the data itself. Aleatoric uncertainty can be further categorised as: *homoscedastic*, which captures the data-invariant noise across the whole dataset; and *heteroscedastic*, which captures the data-dependent noise over each data instance [2].

BNN is widely used for quantifying epistemic uncertainty on the strength of its robustness to the distribution shift of data [13]. However, for modern deep neural networks, it is computationally expensive to build and train them as BNNs. Alternatively, several works [2], [14] model epistemic uncertainty in deep learning with BNN, via an approximation technique named Monte Carlo Dropout (MCD) [15]. The MCD only requires performing a Dropout operation [16] before every weight matrix in a standard neural network, hence eliminating the extra parameters cost for BNN. However, it is still computationally expensive for real-time rendering, due to sampling through a deep neural network many times at each layer, which is regarded as a critical challenge [2].

Recently, amortised variational inference-based [8] DGMs are proposed as an alternative approach to uncertainty quantification [17], [18]. Compared to BNN, a DGM enables computationally tractable uncertainty quantification in the form of posterior analysis in the approximated latent space. Hence it avoids expensive sampling as in MCD. Therefore, in this paper, we explore using DGM for quantifying epistemic uncertainty as an alternative replacement for MCD in four multi-label text classification tasks. For quantifying aleatoric uncertainty, we place a distribution on prediction outputs as in [2], [14]. Our contributions thus include the following:

1) we present novel methods of representing epistemic and aleatoric uncertainties conditioned on text;
2) we compare the effectiveness of modelling epistemic uncertainty between DGM and the widely adopted technique, MCD [2]; and show that the DGM-based method achieves competitive performance, while the DGM method is around 13 to 45 times faster than the MCD method, depending on architecture;
3) we discuss various strategies for training a DGM for epistemic uncertainty modelling in multi-label text classification tasks;
4) we show the benefits of modelling epistemic and aleatoric uncertainties in four text classification tasks with diverse neural network architectures.

II. Related Work

Research on practical methods of quantifying uncertainty in deep learning from a Bayesian perspective has only recently been endeavoured [19]. Methods were initially proposed to capture either epistemic uncertainty or aleatoric uncertainty, alone. For epistemic uncertainty, the approach involved a Bayesian approximation technique named Monte Carlo Dropout (MCD) [15], based itself on a widely adopted regularisation technique called Dropout [16]. MCD allows Dropout to be considered equivalent to applying Variational Inference (VI) [20]–[22] over the full parameter set in a deep neural network. The posterior distribution can then be approximated via multiple runs of the same model with Dropout applied, using the same input data. This practical tool for epistemic uncertainty estimation has been successfully used on a wide range of applications, such as semantic segmentation [23], language modelling [24], diabetic retinopathy [25], transport data analysis [26], magnetic resonance imaging (MRI) segmentation [27], text classification [28] and learning analytics [29].

Most of the works mentioned above focused on modelling epistemic uncertainty alone; and they overlooked the existence of the aleatoric uncertainty, which is equally essential for real-life applications [12]. To bridge this research gap, an uncertainty quantification framework, jointly modelling these two types of uncertainties, was proposed and applied first to computer vision [2] and later extended to natural language processing [14]. It modelled the heteroscedastic aleatoric uncertainty, by placing a distribution on prediction outputs. This distribution is jointly learnt with an additional neural network, during the training process with the original network. The epistemic uncertainty is modelled by creating a BNN using MCD; however, when applying MCD on modern neural network models, the computational cost in testing time induced by sampling is a critical challenge [2].

Recently, the deep generative model (DGM) has been proposed, as an alternative approach to uncertainty quantification. It has been successfully applied to the inverse problem [17] and to image denoising [18]. Compared with BNN, DGM enables computationally tractable uncertainty quantification in the form of posterior analysis in approximated latent space. Hence, it avoids expensive sampling, as required by MCD during testing time. DGM for uncertainty quantification purposes has not been well studied in the NLP domain. To address this research gap, we thus propose novel methods of quantifying uncertainties conditioned on text. We compare the performance of DGM epistemic modelling on text with the widely adopted method, MCD. We demonstrate the benefits of modelling uncertainties with our novel methods on four multi-label text classification problems. To the best of our knowledge, this is the first time *uncertainty quantification methods conditioned on text* are proposed, allowing for *efficient posterior analysis*. We apply them to diverse neural architectures on text classification tasks with empirical experiments.

III. Methodology

This section presents a detailed explanation of our novel methods of modelling uncertainties in deep neural networks. We start with defining the problem and introducing our baseline deep learning approach, and then we articulate how epistemic uncertainty and aleatoric uncertainty can be modelled.

We use three diverse neural network architectures (LSTM, CNN, and Transformer) throughout our experiments. Since our methods are invariant to specific architectures, we use an encoder network, '$\boldsymbol{Encoder}()$', to represent a generic architecture choice. In the context of this study, it can refer to one of these three network architectures; however, it is not limited to the architectures mentioned above. Other neural networks that can produce a fixed dimensional vector representation of a given text could be interchanged as the '$\boldsymbol{Encoder}()$'.

A. Problem Definition

A multi-label text classification task can be defined as: given training data in the form of N data pairs $\{(x_n, y_n)\}_{n=1}^{N}$, with each pair consisting of the text (denoted by x_n) and their associated label (denoted by y_n). For the n^{th} pair, $x_n = \{w_1, ..., w_L\}$ denotes the set of L words from the input, where each word $w_l \in \mathcal{V}_x$ is an instance of a discrete random variable from the dictionary $\mathcal{V}_x$; $y_n \in \mathcal{V}_y$, an integer, is an instance of a discrete random variable from the set $\mathcal{V}_y$. The purpose is to find the right prediction $\hat{y}^*$, given new data x^*.

In the following descriptions, we omit the data pair index n and use bold characters to represent vector form representations, i.e., $\boldsymbol{x}$ and $\boldsymbol{y}$. These representations will be learnt in an end-to-end fashion.

B. Deep Learning Approach (Baseline)

In a traditional deep learning (DL) approach, used here as a baseline, we build a deep neural network to learn a deterministic function as the approximation for the probability $p(\hat{y}|x)$ of the prediction $\hat{y}$, given input x. In our experiments, we use a standard architecture setup for text classification. Our architecture consists of an encoder network, as explained above, followed by an affine transformation with an output, where the dimension is equal to the associated classes.

Given an input sequence of words $x = \{w_1, ..., w_L\}$, the encoder network outputs a representation:

$$x_{rep} = Encoder(E_{w^l}(x)) \tag{1}$$

where E_{w^l} is the learnt word embedding for the l^{th} word $w^l \in x$; x_{rep} is then fed through the affine transformation for the prediction $\hat{y}$. With negative cross-entropy loss, for a mutually exclusive K-class multinomial classification, we have the following cost function:

$$\theta^* = \arg\min_\theta \sum_{k=1}^{K} (-y^{(k)}) \log \hat{y}^{(k)} \tag{2}$$

As suggested in [2], entropy (denoted as H) is a measurement of prediction uncertainty[1], and a low entropy indicates the neural network is confident when making predictions. Entropy can be calculated via the logits layer values:

$$H(p(\hat{y})) = -\sum_{k=1}^{K} p(\hat{y}^{(k)}) \log p(\hat{y}^{(k)}) \tag{3}$$

Note that in a standard deep learning approach, however, neither epistemic nor aleatoric uncertainty is explicitly modelled. This entropy value is an estimate of 'uncertainty' for 'in-domain' data. A plethora of research has demonstrated that it is easy to find or synthesise inputs for which a standard neural network is highly confident, yet wrong [30].

C. Modelling Epistemic Uncertainty

To model epistemic uncertainty in a standard neural network model, we introduce an additional unobserved random variable z and place a distribution on it. This essentially turns our model into a conditional variational auto-encoder (CVAE) [10], [11]. CVAE has been explored as a supervised generative model for text classification [31], [32]. In this paper, we follow [17], [18], and study the effectiveness of CVAE as a tool to model the epistemic uncertainty in a deep neural network. For each observed data pair $\{x, y\}$, the joint distribution can be factorised as follows:

$$p_\theta(y, z|x_{rep}) = p_\theta(y|z, x_{rep}) p_\theta(z|x_{rep}) \tag{4}$$

where θ is the set of neural network parameters and x_{rep} comes from the same encoder architecture as for the standard deep learning approach. We use amortised variational inference (amortised VI) [8] and adopt the same assumption for continuous multivariate Gaussian with diagonalised covariance matrix as in [31], [32]. The evidence lower bound ($\mathcal{L}(ELBO)$) for the marginal likelihood is:

$$\begin{aligned} \log p_\theta(y|x) \geq \mathcal{L}(ELBO) = E_{q_\phi(z)}[\log p_\theta(y|z, x_{rep})] \\ - D_{KL}[q_\phi(z|rep, y)||p_\theta(z|x_{rep})] \end{aligned} \tag{5}$$

The first term of $\mathcal{L}(ELBO)$ is the reconstruction loss and is measured via a multi-class cross-entropy loss. The second term is the Kullback–Leibler (KL) divergence between $p_\theta(z|x_{rep})$ and $q_\phi(z)$. The variational family $q_\phi(z)$ here approximates the posterior distribution as in Figure 2:

$$q_\phi(z|x_{rep}, y) = \mathcal{N}(z|\mu_\phi(x_{rep}, y), \text{diag}(\sigma^2_\phi(x_{rep}, y))) \tag{6}$$

where we have:

$$\begin{aligned} \mu_\phi(x_{rep}, y) &= l_1(\pi_\phi) \\ \log \sigma_\phi(x_{rep}, y) &= l_2(\pi_\phi) \\ \pi_\phi &= g_\phi(x_{rep}, y) \end{aligned} \tag{7}$$

where l_1 and l_2 are two separate affine transformation functions from π_ϕ, g_ϕ is an MLP unit, and y is the one-hot encoding form of the label. The latent variable z can be reparameterised as $z = \mu + \sigma \cdot \epsilon$, known as the "reparameterisation trick" [33], with sample $\epsilon \sim \mathcal{N}(0, I)$. We adopt the inference and generation network for both p and q distributions, similar to [31], [32], shown in Figure 2. For the conditional distribution $p_\theta(z|x_{rep})$, we model it as:

$$p_\theta(z|x_{rep}) = \mathcal{N}(z|\mu_\theta(x_{rep}), \text{diag}(\sigma^2_\theta(x_{rep}))) \tag{8}$$

where we have:

$$\begin{aligned} \mu_\theta(x_{rep}) &= l_3(\pi_\theta) \\ \log \sigma_\theta(x_{rep}) &= l_4(\pi_\theta) \\ \pi_\theta &= g_\theta(x_{rep}) \end{aligned} \tag{9}$$

Similarly, l_3 and l_4 are two separate affine transformation functions from π_θ, and g_θ is an MLP unit. The KL term in ELBO has a closed-form solution [8], and the first term of ELBO can be calculated via a Monte Carlo approximation, as:

$$E_{q_\phi(z)}[\log p_\theta(y|z, x_{rep})] \approx \frac{1}{M} \sum_{m=1}^{M} \log p_\theta(y|z^{(m)}, x_{rep}) \tag{10}$$

The Monte Carlo approximation term here is an unbiased estimator. $z^{(m)}$ is the m^{th} sample from the probability distribution $p(z)$, and M is the total number of samples. We use $M = 1$ during training as per standard practice for

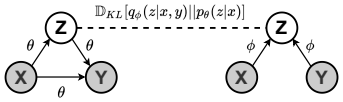

Fig. 2. Graphical model for generation network (left) and inference network (right) for CVAE, using amortised VI [8].

[1]We discuss applicability of entropy as uncertainty estimation in section V.

amortised VI [8] and adopt maximum a-posteriori (MAP) as estimation during testing. During training, $\boldsymbol{z}$ is sampled from $\boldsymbol{q_\phi(z|x_{rep}, y)}$ and during testing, MAP is calculated based on $\boldsymbol{p_\theta(z|x_{rep})}$. Given our prediction $\hat{\boldsymbol{y}}$, the cost function is:

$$\begin{aligned}\boldsymbol{\theta^* = \arg\min_{\theta,\phi}} \sum_{k=1}^{K} (-\boldsymbol{y^{(k)}}) \log \hat{\boldsymbol{y}}^{(k)} \\ +\boldsymbol{\beta D_{KL}[q_\phi(z)||p_\theta(z|x_{rep})]}\end{aligned} \tag{11}$$

We obtain the real $\boldsymbol{\mathcal{L}(ELBO)}$ when $\boldsymbol{\beta = 1}$. CVAE models suffer from posterior collapse as discussed in [34]. Here, 'posterior collapse' refers to the issue where the latent variable z does not contribute much to the model output. We explore three different methods to address this issue during our training in the later section (Experiment 1).

Compared with the Monte Carlo Dropout (MCD), CVAE enables computationally tractable uncertainty quantification in the form of posterior analysis in (approximated) latent space. To calculate the epistemic entropy, we use the same formula as in equation 3. Due to the isotropic Gaussian assumption for the latent space, the prediction entropy for a single data point can be calculated via posterior analysis in the latent space. As opposed to MCD, CVAE allows more efficient estimates, without computationally expensive sampling [2].

D. Modelling Heteroscedastic Aleatoric Uncertainty

We follow [2], [14] and focus only on *heteroscedastic* aleatoric uncertainty. To model the heteroscedastic aleatoric uncertainty in a standard neural network model, we place a distribution on the network output and define the following generative process:

$$\begin{aligned}\boldsymbol{\mu_{y_a}} &= \boldsymbol{l_5(\hat{y})} \\ \boldsymbol{\sigma_{y_a}} &= \boldsymbol{l_6(\hat{y})} \\ \boldsymbol{y_a} &\sim \boldsymbol{\mathcal{N}(\mu_{y_a}, \sigma_{y_a})} \\ \boldsymbol{y} &\sim \boldsymbol{Categorical(y_a)}\end{aligned} \tag{12}$$

where $\boldsymbol{l_5}$ and $\boldsymbol{l_6}$ are two separate affine transformation functions and, during training, the empirical mean ($\boldsymbol{\bar{y_a}}$) can be calculated based on sampling. For the aleatoric uncertainty, we have the following loss function, where $\boldsymbol{K}$ is the number of class labels:

$$\boldsymbol{\theta^* = \arg\min_{\theta}} \sum_{k=1}^{K} (-\boldsymbol{y^{(k)}}) \log \boldsymbol{\bar{y_a}}^{(k)} \tag{13}$$

To quantify the aleatoric uncertainty, we can also calculate the entropy value based on the layer value of $\boldsymbol{\bar{y_a}}$. Note that here $\boldsymbol{y_a}$ is not learnt through variational inference, hence does not measure the epistemic uncertainty of deep learning. Thus, the entropy here only represents the uncertainty associated with the noise in the data. To calculate the aleatoric entropy, we use the same formula as in equation 3, but replace $\hat{\boldsymbol{y}}$ with $\boldsymbol{y_a}$. Emulating the epistemic modelling case, we use the analytical solution to calculate individual aleatoric uncertainty.

IV. Experiments

A. Data

We conduct experiments on three public text classification benchmark datasets, presented in [35]. These datasets can be used in two different types of tasks: (1) *topic classification* (using AG's News and DBPedia) and (2) *sentiment analysis* (using Yelp-P). We denote these datasets as 'AG', 'DB', and 'Y-P', respectively, shown in Table I, II, III, IV, and V. A summary of the datasets is provided in Table I. In our experiments, we use the full testing data set (hence the difficulty of the task is the same) and use only partially the original training data, split as our training set, which we will explain in the following section. We set the maximum token length as 110 (by removing the tokens beyond the first 110) for the 'AG' and 'DB' datasets; and 450 (by removing the tokens beyond the first 450) for the 'Y-P' dataset. The maximum token length is selected based on Table VIII.

B. Vocabulary and Sampling

Before performing sampling, we first create the vocabulary for each dataset, which is shared across all the models. We create each vocabulary based on a minimum frequency of 5 and a maximum size of top $20K$ tokens from the complete training data points, with four additional special tokens: $< pad >$, $< unk >$, $< bos >$, and $< eos >$. These tokens are used to denote batched computation padding, out-of-vocabulary words, the beginning and the end of the sequence, respectively.

We first perform stratified sampling on the original data points to retrieve our class-balanced data points (we sample $20K$ points per class in each task), and then we apply again stratified sampling, to further split these data points into the class-balanced training set and validation set, with a percentage of 80% and 20%, respectively.

TABLE I
DATASET SUMMARY

Dataset	Training	Validation	Testing	Number of Classes
AG	64,000	16,000	7,600	4
DBP	224,000	56,000	70,000	14
Y-P	32,000	8,000	38,000	2

C. Experimental Setup

We conduct experiments over three diverse neural network architectures as the encoder network: LSTM [36], CNN [37], and Transformer [38]. For epistemic uncertainty modelling and aleatoric uncertainty modelling, we add '+EP' and '+AL' to the model name, respectively, in the tables below (Tables III, IV and V).

To compare with the widely adopted epistemic uncertainty modelling technique [15], we implement the Monte Carlo Dropout technique for each network, denoted by the addition of '+MC', following the implementation guideline in [14]. We fix the dropout rate as 0.5 based on our empirical experiments. We use Adam [39] as our optimiser in all experiments. The batch size is set to 32 and all training runs for a maximum of

10 epochs. All experiments are conducted on a computer with an Ubuntu operating system and a single RTX 2080 Ti GPU.

Next, we introduce the setup of the three model architectures (LSTM, CNN, and Transformer) below. Each architecture is based on a widely adopted model for text classification tasks in the NLP domain.

1) LSTM: For the LSTM experiment, we use an embedding size of 56 for each unique word and build a single layer bi-directional LSTM (Bi-LSTM) network architecture with a hidden dimension of 56 (resulting in 112 latent dimensions). We use a size of 56 for the CVAE latent variable $\boldsymbol{z}$. We use the last hidden state of the Bi-LSTM network as the $\boldsymbol{x_{rep}}$.

2) CNN: For the CNN experiment, we apply the $2d$-convolution operation (over sequence length and embedding dimension) as in [40] on our text input and use learnable filter sizes of 1, 2 and 3 to represent 'unigram', 'bigram', and 'trigram' information from the text sequence. We use a max pooling operation over each filter output to alleviate various sequence length issue and concatenate them as $\boldsymbol{x_{rep}}$. For each $2d$-convolution operation, we use an input channel size of 1 and output channel size of 56. We use an embedding size of 56 for each unique word and a dimension of 56 for the CVAE latent variable $\boldsymbol{z}$ for our epistemic modelling experiments.

3) Transformer: For the Transformer experiment, we use the same architecture setup as in [38], but with a stack of $N = 1$ encoder layer and with no decoder (as it is not required for text classification). For a detailed description of the transformer network, please refer to [38]. We use the same positional encoding methods (sine and cosine) as in [38] and use a hidden dimension of 56, feedforward neural network intermediate layer dimension of 256, and 8 attention heads. This grants us an output dimension of 56, and we use the same size (56) for the CVAE latent variable $\boldsymbol{z}$. As the transformer encoder creates a sequence of contextual representations for a given input sequence, we pre-process text by adding additional special tokens $< bos >$ and $< eos >$, before and after the sequence input. We use the output at the $< bos >$ position as $\boldsymbol{x_{rep}}$, which is similar to how a BERT model [1] is used for text classification.

4) Model Architecture Complexity: For a fair comparison, we set each model architecture with a similar complexity in terms of the number of trainable parameters. For the LSTM, CNN, and Transformer models defined in the experiments, there are around $1.4M$, $1.5M$, and $1.4M$ trainable parameters, respectively (calculated based on the epistemic uncertainty experiment). There are slight variations in the number of parameters in other experiments (Table III, IV and V, results without the '+EP' flag), but the model complexities always remain on the same scale.

D. Experimental Run

For reporting results, we first select the best *learning rate* from among $1e{-}2$, $1e{-}3$, $1e{-}4$ and $1e{-}5$, based on the mean average run from three random seeds: 1000, 2000, and 3000. Then, we run each model 3 times with the best *learning rate* and use a seed from 1111, 2222, and 3333 to initialise model parameters. For each run of the model, we use the training, validation, and testing sets as shown in Table I. We evaluate the model performance on the validation set after each epoch and save the best model based on the $F1$ score, calculated over the validation set. For testing, we apply the best performance model on the test set and report the evaluation metric.

Based on our experiments, the optimal *learning rates* for our models are $1e-3$ and $1e-4$ in general, while most of the time $1e-3$ performs the best. This is thus the *learning rate* we use when reporting the results, unless specified otherwise. During training, we use a gradient clip of 1, to avoid gradient overflow.

E. Evaluation Metric

We report the *mean* and the *variance* (in brackets) amongst three runs on the standard evaluation methods for the text classification task (macro-averaged $F1$) in *Experiment* 1 and *Experiment* 2. For the $F1$ scores, all values are reported in percentage. We report additional entropy measurement [2] calculated with Equation 3 over the test set in *Experiment* 2. For the entropy measurement, we report the mean average over the three runs.

V. Results and Discussion

A. Experiment 1: Posterior Collapse

In the first experiment, we address the commonly encountered problem in CVAE training, i.e., the posterior collapse [34], [41], [42], in the context of text classification. We compare three different methods for training a CVAE, including using standard KL, KL Annealing, and KL Coefficient. We apply each of the three methods on the epistemic uncertainty modelling task over three diverse architectures. Here, we briefly describe these three methods:

1) **Standard KL**
   During training, we use the standard CVAE formula (as in Equation 11) with $\beta = 1$ as the KL term.
2) **KL Annealing**
   During training, we apply KL annealing on β, as in [34], [41], to induce the cost function and thus maintaining a substantial KL value. In all of our experiments, we run a total of 10 epochs. Specifically, we linearly anneal β from 0 to 1 over the first 5 epochs and then use value of 1 for the remain 5 epochs. The KL annealing essentially forces the model to explore and utilise the information from the latent variable $\boldsymbol{z}$ [42].
3) **KL Coefficient**
   During training, we use a reduced $\beta = 0.2$ for the KL term as in [34], [43]. This allows us to adjust the weight of the KL penalty related to the reconstruction loss.

Experimental results on macro-average $F1$ for the LSTM, CNN, and Transformer architectures, respectively, are presented in Table II. Training with standard KL, KL Annealing, and KL Coefficient are denoted as 'EP ($\beta = 1$)', 'EP (Ann.)', and 'EP (Coe.)', respectively. Throughout the experiments, we have discovered that training with the KL annealing method provides us with the best $F1$ score in general on both *mean*

TABLE II
POSTERIOR COLLAPSE EXPERIMENT ON LSTM, CNN AND TRANSFORMER ARCHITECTURES. RESULTS PRESENTED IN MACRO F1-SCORE BASED ON THE TEST DATASET.

Model	AG	DBP	Y-P
LSTM (%)			
EP (Ann.)	**91.13 (0.33)**	**98.36 (0.08)**	**91.70 (0.23)**
EP (Coe.)	90.94 (0.44)	98.33 (0.04)	90.69 (0.12)
EP ($\beta = 1.$)	91.12 (0.39)	98.33 (0.02)	91.52 (0.15)
CNN (%)			
EP (Ann.)	91.38 (0.30)	**98.22 (0.04)**	**92.69 (0.01)**
EP (Coe.)	91.42 (0.52)	98.15 (0.08)	92.21 (0.16)
EP ($\beta = 1.$)	**91.47 (0.28)**	98.15 (0.02)	92.33 (0.21)
Transformer (%)			
EP (Ann.)	91.29 (0.15)	97.52 (0.14)	**92.16 (0.10)**
EP (Coe.)	91.03 (0.19)	**97.68 (0.01)**	91.88 (0.11)
EP ($\beta = 1.$)	**91.50 (0.29)**	97.63 (0.04)	91.99 (0.26)

and variance across three random runs with different seeds (see Table II) when the MAP decoding method is adopted, especially for the LSTM architecture, which is reported similarly in literature [34], [41], [42]. Based on results shown in Table II, we adopt the KL Annealing training technique in the follow-up experiments, although it requires an additional hyperparameter tuning (rate for linear annealing reduction).

To demonstrate the efficacy of the results, we apply the Wilcoxon signed-rank test on the results (KL Annealing against the other two in Table II) from multiple 3 runs of our models. In terms of the AG dataset, the $F1$ score is better, although not statistically significant ($p > .05$) for Transformer; and statistically significantly better ($p < .05$) for LSTM and CNN. For the DB dataset, $F1$ is better, but not statistically significant ($p > .05$) for LSTM and CNN; and not better for Transformer. For the YP dataset, $F1$ is statistically significantly better ($p < .05$) for LSTM and Transformer; and better, but not statistically significant ($p > .05$), for CNN.

B. Experiment 2: Uncertainty Modelling

In the second experiment, we apply uncertainty modelling over the four datasets with three diverse neural network architectures. For epistemic uncertainty modelling, we use the KL annealing technique during training, followed by the results in experiment 1 (Table II). The experimental results on macro-average $F1$ and entropy for LSTM, CNN, and Transformer architectures are presented in Tables III, IV, and V, respectively. The results suggest that uncertainty plays an important role in text classification tasks. In general, modelling uncertainty, as we observe, mostly comes with a benefit. We use bold font to denote when results are better than the deep learning baseline models, and use underline to denote the best performance methods in Tables III, IV, and V.

1) Epistemic Uncertainty: Compared with the '+MC' method using Monte Carlo dropout, we observe that our epistemic uncertainty modelling method '+EP' achieves competitive or better performance across most experiments (based on mean and variance) for the $F1$ score. However, the '+EP' model grants much lower entropy in general, compared to the MC dropout method, as shown in Tables III, IV, and V.

TABLE III
UNCERTAINTY MODELLING RESULTS FOR LSTM MODEL, RESULTS PRESENTED IN MACRO F1-SCORE AND ENTROPY BASED ON THE TEST DATASET.

Model	AG	DBP	Y-P
Macro-F1%			
LSTM	91.17 (0.28)	98.39 (0.03)	91.20 (0.17)
LSTM + EP	91.13 (0.33)	98.36 (0.08)	**91.70 (0.23)**
LSTM + MC	**91.49 (0.21)**	**98.58 (0.02)**	**91.74 (0.35)**
LSTM + AL	91.16 (0.20)	**98.40 (0.03)**	91.20 (0.29)
Entropy (Ave.)			
LSTM	0.1866	0.0360	0.2166
LSTM + EP	**0.0434**	**0.0262**	**0.0338**
LSTM + MC	**0.1766**	**0.0193**	**0.1816**
LSTM + AL	0.2359	**0.0315**	**0.1589**

TABLE IV
UNCERTAINTY MODELLING RESULTS FOR CNN MODEL, RESULTS PRESENTED IN MACRO F1-SCORE AND ENTROPY BASED ON THE TEST DATASET.

Model	AG	DBP	Y-P
Macro-F1%			
CNN	91.49 (0.27)	98.32 (0.06)	92.49 (0.12)
CNN + EP	91.38 (0.30)	98.22 (0.04)	**92.69 (0.01)**
CNN + MC	**91.77 (0.18)**	**98.39 (0.01)**	92.20 (0.23)
CNN + AL	**91.70 (0.37)**	**98.35 (0.06)**	92.40 (0.10)
Entropy (Ave.)			
CNN	0.1960	0.0327	0.1619
CNN + EP	**0.0651**	**0.0203**	**0.0240**
CNN + MC	0.2014	0.0379	**0.1607**
CNN + AL	**0.1762**	0.0357	**0.1261**

The entropy value is regarded as a critical measurement of information uncertainty for the predictive distribution [2]. The lower the entropy, the more confident the classifier is in its decision. '+EP' results in a lower entropy than the '+MC' method in general, demonstrating its effectiveness in reducing epistemic uncertainty.

TABLE V
UNCERTAINTY MODELLING RESULTS FOR TRANSFORMER MODEL, RESULTS PRESENTED IN MACRO F1-SCORE AND ENTROPY BASED ON THE TEST DATASET.

Model	AG	DBP	Y-P
Macro-F1%			
Transformer	91.32 (0.16)	97.88 (0.13)	92.10 (0.03)
Transformer + EP	91.29 (0.15)	97.52 (0.14)	**92.16 (0.10)**
Transformer + MC	**91.43 (0.18)**	97.84 (0.14)	91.87 (0.32)
Transformer + AL	**91.41 (0.22)**	**97.89 (0.01)**	92.07 (0.06)
Entropy (Ave.)			
Transformer	0.2274	0.0543	0.2548
Transformer + EP	**0.0592**	**0.0341**	**0.0319**
Transformer + MC	**0.2106**	**0.0339**	**0.2164**
Transformer + AL	**0.1969**	**0.0444**	**0.1559**

To demonstrate the efficacy of the results, we apply the Wilcoxon signed-rank test on all the results (compare baseline model, i.e. LSTM, CNN, and Transformer; and with uncertainty modelling, i.e.'+EP') from the multiple runs of our models. The $F1$ score is generally competitive, or better than the baseline model. The entropy value is mostly statistically significantly better ($p < .05$), in comparison with the baseline

TABLE VI
HIGH VERSUS LOW ALEATORIC UNCERTAINTY EXAMPLES FOR YELP-P DATASET.

High Aleatoric Uncertainty
The price is a **little high** for me. They don't give enough bean sprouts without asking for more. The pho broth is a **little too sweet** for me also.
I like their beer ... seriously.
Was a **great** place to eat, now food is **greasy**, and it doesn't taste like it used to. Used to have a more of a homemade taste, now tastes like Costco business center. **Sad**. I **loved** going here .
Low Aleatoric Uncertainty
I went there last night and I ordered the calamari. It had **no taste** and was **very expensive**. And the service was **not very good**. I **will not be back**
Oh my... **dont like** the food here. Tried the Pad thai and the chicken fried rice ... the pad thai was **disgusting** and the fried rice was not $< unk >$ either ... **wont be** going back ... ugh
Wow this place has gone down hill. **Old** smelly rooms. Service is **horrible**

model. When combining results from both the $F1$ score and entropy value, we can claim that it is beneficial to model epistemic uncertainty in deep neural networks.

2) Aleatoric Uncertainty: Compared with the baseline model, our aleatoric uncertainty modelling method '+AL' achieves competitive or better performance across most experiments (based on *mean* and *variance*) for the $F1$ score. Again, for the Wilcoxon signed-rank test on all results (comparing baseline model, i.e. LSTM, CNN, and Transformer; with uncertainty modelling, i.e.'+AL') from the multiple runs of our model, $F1$ is better, although not statistically significantly so. The entropy value is mostly statistically significantly better ($p < .05$), in comparison to the baseline model. When combining results from both the $F1$ score and entropy value, we can claim that it is beneficial to model aleatoric uncertainty in deep neural networks. Additionally, since modelling aleatoric uncertainty only requires tiny changes on the cost function during training, it is inexpensive and hence is recommended for text classification tasks whenever possible.

To illustrate what we modelled for aleatoric uncertainty, we provide examples of high and low uncertainty of data in the Yelp-P dataset, presented in Table VI (based on the LSTM model). The Yelp-P data is used for the polarised sentiment analysis task (negative or positive). We can observe that the low aleatoric uncertainty examples contain clear sentiment words, such as *'not very good'*, *'dont like'*, and *'horrible'*. On the contrary, the high aleatoric uncertainty examples contain vague sentiment words, such as *'little high'* and *'little too sweet'*; or word with vague meaning, such as *'seriously.'*; or even sentiment words with contradictory meanings, such as *'great'*, *'greasy'* and *'sad'* concomitantly. These observations align with the results presented in [14].

C. Discussion on Entropy as Uncertainty Estimation

In this paper, entropy is used as an uncertainty estimation, as in [2], [14]. However, the reason for choosing entropy as the measurement is not clear in the literature, so here we discuss briefly the reason and whether it is applicable. From a Bayesian modelling perspective, uncertainty is best presented as the variance of the density function for the posterior distribution in regression tasks. In this paper, we instead explore classification tasks, and thus the variance is best represented as the entropy measurement over the posterior distribution, as in equation 3. In this case, a lower entropy value indicates a decrease in classification variance for a predictor, which represents a decrease in uncertainty. However, a lower entropy value only can not denote an improvement in model performance, so we additionally adopt the $F1$ score as another metric measurement.

TABLE VII
RUN TIME RESULTS FOR UNCERTAINTY MODELLING FOR LSTM, CNN AND TRANSFORMER MODELS, RESULTS PRESENTED IN SECONDS BASED ON THE TEST DATASET (IN BRACKETS THE NUMBER OF TIMES IT IS FASTER, COMPARED TO ITS RESPECTIVE BASELINE MODEL).

Model	**AG**	**DBP**	**Y-P**
	LSTM (s)		
LSTM	0.700	7.746	12.607
LSTM + EP	**0.834 (1.08)**	**8.132 (1.04)**	**14.51 (1.15)**
LSTM + MC	26.990 (34.91)	299.867 (38.70)	582.844 (46.20)
LSTM + AL	0.783 (1.01)	8.034 (1.03)	12.775 (1.01)
	CNN (s)		
CNN	0.278	2.426	1.877
CNN + EP	**0.358 (1.28)**	**3.077 (1.26)**	**2.100 (1.11)**
CNN + MC	3.597 (12.90)	33.350 (13.7)	46.011 (24.50)
CNN + AL	0.282 (1.01)	2.550 (1.05)	1.887 (1.01)
	Transformer (s)		
Transformer	0.412	4.186	19.689
Transformer + EP	**0.439 (1.06)**	**4.421 (1.05)**	**20.185 (1.02)**
Transformer + MC	11.185 (27.10)	129.108 (30.80)	848.486 (43.10)
Transformer + AL	0.415 (1.01)	4.604 (1.09)	20.878 (1.06)

D. Analysis on Run Time Efficiency

Based on information provided for the experimental setup in sections IV-C and IV-C4, we present an analysis on the run time efficiency in Table VII. In particular, we present the run time for each testing set and the number of times it is faster than its base model (i.e. LSTM, CNN, and Transformer). We also underline the multipliers for '+MC' and '+EP'. We observe that, in general, modelling aleatoric uncertainty ('+AL') does not increase run time during testing. However, with epistemic uncertainty ('+EP' and '+MC'), the current widely adopted approach ('+MC') requires significantly more time, as noted in [2]. While our approach barely changes the run time and is a lot faster (between 13 to 45 times faster).

VI. CONCLUSION

This paper presents novel uncertainty quantification methods that allow efficient analysis of posterior inference. We demonstrate their effectiveness on four multi-label text classification tasks. Our framework allows efficient posterior analysis and our experiments affirm the benefits of modelling uncertainty. We show the consistency of our results over multiple experiments with diverse neural network architectures. Our work can serve as a baseline for applying uncertainty quantification to text classification tasks and contributes to further research in this domain.

REFERENCES

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Advances in neural information processing systems*, 2017, pp. 5574–5584.

[3] R. McAllister, Y. Gal, A. Kendall, M. Van Der Wilk, A. Shah, R. Cipolla, and A. Weller, "Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning." International Joint Conferences on Artificial Intelligence, Inc., 2017.

[4] C. Li, "Towards better representations with deep/bayesian learning," Ph.D. dissertation, Duke University, 2018.

[5] D. J. MacKay, "A practical bayesian framework for backpropagation networks," *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.

[6] ——, "Probable networks and plausible predictions-a review of practical bayesian methods for supervised neural networks," *Network: computation in neural systems*, vol. 6, no. 3, p. 469, 1995.

[7] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

[8] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[9] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-supervised learning with deep generative models," *arXiv preprint arXiv:1406.5298*, 2014.

[10] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *Advances in neural information processing systems*, vol. 28, pp. 3483–3491, 2015.

[11] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *International conference on machine learning*. PMLR, 2016, pp. 1558–1566.

[12] A. Der Kiureghian and O. Ditlevsen, "Aleatory or epistemic? does it matter?" *Structural safety*, vol. 31, no. 2, pp. 105–112, 2009.

[13] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information Fusion*, 2021.

[14] Y. Xiao and W. Y. Wang, "Quantifying uncertainties in natural language processing tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 7322–7329.

[15] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[17] V. Böhm, F. Lanusse, and U. Seljak, "Uncertainty quantification with generative models," *arXiv preprint arXiv:1910.10046*, 2019.

[18] M. Prakash, A. Krull, and F. Jug, "Fully unsupervised diversity denoising with convolutional variational autoencoders," *arXiv preprint arXiv:2006.06072*, 2020.

[19] Y. Gal, "Uncertainty in deep learning," *University of Cambridge*, vol. 1, no. 3, p. 4, 2016.

[20] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[21] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine learning*, vol. 37, no. 2, pp. 183–233, 1999.

[22] M. J. Wainwright and M. I. Jordan, *Graphical models, exponential families, and variational inference*. Now Publishers Inc, 2008.

[23] A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," *arXiv preprint arXiv:1511.02680*, 2015.

[24] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in neural information processing systems*, 2016, pp. 1019–1027.

[25] C. Leibig, V. Allken, M. S. Ayhan, P. Berens, and S. Wahl, "Leveraging uncertainty information from deep neural networks for disease detection," *Scientific reports*, vol. 7, no. 1, pp. 1–14, 2017.

[26] L. Zhu and N. Laptev, "Deep and confident prediction for time series at uber," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2017, pp. 103–110.

[27] A. G. Roy, S. Conjeti, N. Navab, C. Wachinger, A. D. N. Initiative *et al.*, "Bayesian quicknat: model uncertainty in deep whole-brain segmentation for structure-wise quality control," *NeuroImage*, vol. 195, pp. 11–22, 2019.

[28] W. Chen, B. Zhang, and M. Lu, "Uncertainty quantification for multi-label text classification," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 10, no. 6, p. e1384, 2020.

[29] J. Yu, L. Alrajhi, A. Harit, Z. Sun, A. I. Cristea, and L. Shi, "Exploring bayesian deep learning for urgent instructor intervention need in mooc forums," in *International Conference on Intelligent Tutoring Systems*. Springer, 2021, pp. 78–90.

[30] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan, "Do deep generative models know what they don't know?" *arXiv preprint arXiv:1810.09136*, 2018.

[31] Y. Miao, L. Yu, and P. Blunsom, "Neural variational inference for text processing," in *International conference on machine learning*, 2016, pp. 1727–1736.

[32] J. Qian, M. ElSherief, E. Belding, and W. Y. Wang, "Hierarchical cvae for fine-grained hate speech classification," *arXiv preprint arXiv:1809.00088*, 2018.

[33] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Advances in neural information processing systems*, 2015, pp. 2575–2583.

[34] A. Pagnoni, K. Liu, and S. Li, "Conditional variational autoencoder for neural machine translation," *arXiv preprint arXiv:1812.04405*, 2018.

[35] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[40] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. [Online]. Available: https://aclanthology.org/D14-1181

[41] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *arXiv preprint arXiv:1511.06349*, 2015.

[42] J. He, D. Spokoyny, G. Neubig, and T. Berg-Kirkpatrick, "Lagging inference networks and posterior collapse in variational autoencoders," *arXiv preprint arXiv:1901.05534*, 2019.

[43] Y. Miao and P. Blunsom, "Language as a latent variable: Discrete generative models for sentence compression," *arXiv preprint arXiv:1609.07317*, 2016.

APPENDIX A
DATASET STATISTICS

TABLE VIII
TOKEN LENGTH STATISTICS, ALL NUMBERS ROUND TO INTEGER.

Model	Mean	Standard Deviation	Min	Max
AG	45	13	12	214
DBP	57	26	3	1500
Y-P	156	143	1	1202