Variational Auto-Encoder for Latent Uncertainty Encoding in Large Language Models

Stefan Paun

21st January, 2025

University of Twente

Variational Auto-Encoder for Latent Uncertainty Encoding in Large Language Models

MSc Thesis

Under the supervision of
dr. Alexandros G. Stergiou (DMB, University of Twente)
dr. Arjan J. van Hessen (HMI, University of Twente)

Stefan Paun

Contents

1 Introduction

Uncertainty is both a phenomenon that is an integral part of the human experience and a fundamental concept that spans a multitude of disciplines, including psychology, cognitive sciences, mathematics, physics, engineering and machine learning. In the field of cognitive science, uncertainty is often viewed from the perspective of decision making, where future outcomes are difficult to predict due to unreliable or incomplete information, or due to unexpected changes in the environment [1]. In the field of physics, uncertainty manifests itself at the most fundamental level, through Heisenberg's Uncertainty Principle, which imposes a limit to the level of precision with which the position and momentum of sub-atomic particles can be simultaneously measured [2]. In the engineering, uncertainty plays a role in modeling real-world systems, where measurements that are noisy and have a higher degree of uncertainty, lead to less accurate models.

From an information theory and a computational perspective, uncertainty is characterized by probability distributions which quantify the likelihood of various outcomes, given incomplete or noisy information [3]. In machine learning, however, models often rely on fixed representations, called embeddings, to represent information. While these fixed-point embeddings are useful for many tasks, they fall short in capturing the inherent uncertainty about the data these models encode. Representing these embeddings as a probability distribution, rather than a fixed point, in an n-dimensional space, could be a way to mitigate this shortfall and enable machine learning models to encode uncertainty.

This thesis aims to explore methods for encoding uncertainty in machine learning, focusing on Large Language Models (LLMs) in particular. To achieve this, we make use of a Variational Auto-Encoder (VAE) architecture to learn a probabilistic latent representation of the internal embeddings of LLMs. Our findings show that this architecture, in combination with a Kullback–Leibler (KL) training objective designed to capture the inherent uncertainty, is a promising method for encoding and estimating uncertainty for LLMs.

The rest of the chapter is organized as follows. In section 1.1 we introduce LLMs and provide a brief overview of language-related tasks. We then discuss how uncertainty can be quantified in section 1.2. Our contributions are summarized in section 1.3 and the thesis structure is presented in section 1.4.

1.1 Natural Language Processing

Natural Language Processing (NLP) is a sub-field of artificial intelligence which focuses on enabling computers to comprehend, manipulate and generate human language. In recent years, LLMs, also referred to as foundational models, have revolutionized the field of NLP and have gained widespread adoption due to their impressive performance across various language-related tasks, ranging from comprehension to generation. These models have become an integral com-

ponent in numerous real-world applications. This adoption was driven in particular by the development of ChatGPT and other open-source models such as Llama 2 [4], which have shown impressive capabilities to generate human-like text and solve complex tasks such as coding or reasoning.

However, the widespread adoption of LLMs has also highlighted significant risks. These risks include issues such as bias [5], safety [6] and hallucinations [7]. These are especially important in instances where LLMs are deployed in applications, such as healthcare, legal advice or automated decision-making systems, where incorrect or misleading information can lead to serious consequences [8].

1.2 Quantifying Uncertainty

Uncertainty quantification for deep learning neural networks, used in tasks such as classification or image segmentation, is a relatively well established area of research [9]. Common uncertainty estimation approaches can be broadly categorized in Bayesian methods or ensemble methods. In contrast, uncertainty quantification in the context of LLMs is a relatively novel area of research that is still under active development.

One approach to addressing some of the risks posed by LLMs, as suggested by Liu *et al.* [10], is to enhance foundational models with uncertainty quantification capabilities. By providing accurate estimates of an LLM's confidence in the reliability of its generated response, users can be empowered to make informed decisions about whether to trust the model's output, thereby mitigating issues such as the spread of misinformation or biases. Another potential advantage of uncertainty estimation is the handling of out-of-distribution data. Such responses can be flagged, based on the low model confidence, prompting for human intervention in automated systems, or driving improvements in model training and leading to more robust LLMs.

Most common approaches for uncertainty estimation in LLMs rely on Bayesian methods or make use of sampling [11]. Despite being effective, these methods are often computationally expensive as they require multiple inferences for a given input, hindering scalability. Single-inference methods, relying on output logits of an LLM, by means of using token probabilities or entropy measures, are computationally efficient but fail to to fully capture the uncertainty of LLM outputs [11]. Another approach involves prompting LLMs to self-verbalize their confidence [12, 13]. While this approach has the advantage of working for black-box models, whose internal model states are not accessible, it has been found that this method often performs poorly, as LLMs tend to be significantly over-confident. More recently, approaches using the intermediate hidden states of LLMs to infer uncertainty have shown promise, offering a good trade-off between efficiency and performance [14, 15]. The major limitation is that these methods typically require access to internal model states, thus only working with white-box models. However, research suggests that this type of methods can be adapted for use with black-box models with limited loss in performance [10].

1.3 Contributions

In this work, we propose Variational Auto-encoder for Latent Uncertainty Encoding (VALUE), for encoding the uncertainty inherent to LLMs. We build upon basic notions of VAEs to learn uncertainty-aware latent probabilistic representation of the internal states of an LLM, thus modeling the uncertainty of the LLM's responses within the latent space. The underlying intuition is that a larger variance in the latent space for a generated sequence will correspond to a lower confidence in the model's response. The method uses a KL divergence-based uncertainty objective which allows the VALUE model to encode the uncertainty inherent to an LLM's internal states. We demonstrate that our proposed method is LLM-agnostic by evaluating it using both an Open Llama 2 3B model and Meta's Llama 2 7B model. Both the quantitative results and the qualitative analysis showcases the VALUE model's ability to represent uncertainty in the learned latent probabilistic distribution, while maintaining the coherence and the quality of the text generated by the LLM using the VALUE-processed hidden states.

1.4 Thesis Structure

An overview of the thesis is provided below, briefly mentioning each chapter's objective.

Chapter 1 introduces the background, the motivation and the contributions of this thesis.

Chapter 2 presents a literature review covering LLMs, uncertainty and methods of estimating uncertainty in LLMs. It also briefly introduces VAEs and covers related works that employ this architecture for uncertainty estimation.

Chapter 3 provides the theoretical foundation for this thesis by covering the language modeling task and the Transformer architecture specific to LLMs. In addition, it also covers the underlying principles of VAEs.

Chapter 4 formalizes the VALUE approach and details the training objectives used.

Chapter 5 starts by presenting the dataset used, the process through which the VALUE model has been trained and the metrics used to evaluate it. Next, it presents the main quantitative results of the evaluation, followed by a brief qualitative analysis.

Chapter 6 presents some of the challenges encountered in the development of our model and describes how these issues were mitigated.

Chapter 7 concludes the thesis and summarizes the contribution and the findings, while also highlighting directions for future work.

2 Literature review

This chapter provides an overview of LLMs and the current state of research in the field of uncertainty quantification. Section 2.1 focuses on the evolution of LLMs, highlighting some of the most important models and the innovations they have brought to the field, while also briefly touching on evaluation and risks of LLMs. Section 2.2 addresses the topic of uncertainty estimation in neural networks, with a specific focus on uncertainty estimation for LLMs, and explores the use of VAEs for uncertainty quantification.

2.1 Large Language Models (LLMs)

This section outlines the evolution of LLMs, from the introduction of the foundational transformer architecture to the latest developments of models such as GPT and Llama, briefly mentioning some of the key innovations. The evaluation of LLMs is also covered in this section, with commonly used evaluation metrics and benchmarks. Lastly, some challenges such as bias and hallucinations are mentioned, as well as how uncertainty estimation could be used to mitigate these risks.

2.1.1 Evolution of LLMs

One of the most important developments of the last years, in the field of NLP, has been the introduction of the transformers. The encoder-decoder architecture proposed by Vaswani *et al.* [16] introduced several key innovations such as scaled dot-product attention, multi-head attention and positional encoding. With this architecture, a text sequence is split into small units, called tokens, which are processed in parallel. The parallel processing requires positional information of the tokens inside the sequence, achieved by means of positional encodings. The attention mechanism, which is essentially a matrix of token correspondence, allows transformers to capture contextual information. These developments allowed transformers to handle long-range dependencies and efficiently process data in parallel, proving a significant advantage over recurrent neural networks (RNNs) which were widely adopted in NLP. The scalability and efficiency of the transformer architecture enabled the development of LLMs.

LLMs are a class of artificial intelligence systems which are designed to process, understand and generate human language. Such models are trained on large amounts of textual data, often in an unsupervised manner, enabling them to learn linguistic patterns. The most important aspect of LLMs is the concept of language modeling, which involves learning to predict a probability distribution over a vocabulary. LLMs are able to perform a wide range of NLP tasks such as text classification, summarization or translation. The scale of the models enabled LLMs to exhibit emergent capacities, such as in-context learning or chain-of-thought reasoning, properties that are not present in smaller NLP models [17].

Generative Pre-trained Transformer (GPT) - Open AI. Radford *et al.* [18] introduced GPT-1, the first model in the GPT family of models. The model is autoregressive and is using a

left-to-right decoder-only architecture. An autoregressive model generates text one token at a time, based on all previous tokens. It is trained by means of a two-stage training process; the first stage is an unsupervised pre-training with next token prediction as a task; the second stage involves supervised, task-specific fine-tuning. Due to the model being trained on long text sequences, this enabled it to learn long-range dependencies. GPT-1 achieved state-of-the-art on a number of various NLP tasks and demonstrated zero-shot capabilities. Radford *et al.* [19] continued the development of GPT models, by introducing GPT-2. Their work highlighted the positive effect of large and diverse datasets in improving model performance, alongside with the importance of a larger number of parameters for the model. GPT-2 showed good zero-shot capabilities and eliminated the need for downstream task-specific fine-tuning. Brown *et al.* [20] made an important leap with the introduction of GPT-3. It keeps the same autoregressive architecture of GPT-2 with minor changes, but significantly scales up the number of model parameters. One of the key contributions was the introduction of prompts as a way of eliciting model-stored knowledge, enabling meta-learning and in-context learning. The model displayed improved performance in zero-, one, and few-shot learning scenarios. GPT-3 also sparked important discussions about the bias, fairness, safety and misuse of LLMs [20].

Large Language Model (Llama) - Meta AI. Touvron *et al.* [21] introduced the first iteration in the Llama family of models. Llama 1 relied on the original autoregressive transformer architecture, similar to the GPT family of models, however shifted focus on inference efficiency and made use of publicly available data for training. Touvron *et al.* [4] later introduced Llama 2, along with Llama 2-Chat fine-tuned for conversational use cases. The model closely resembled the previous iteration but made use of an increased context window length. The key improvement over the previous iteration stemmed from the training data, with a 40% increase in dataset size and an improved data cleaning process. The chat-focused model was developed using Supervised Fine-Tuning (SFT) and Reinforcement Learning with Human Feedback (RLHF), optimizing the model for both helpfulness and safety of its answers. SFT involves training the model in a supervised manner using labeled data to improve performance on certain tasks, while RLHF is a reinforcement learning paradigm which enhances the model by making use of human feedback to further refine and align the model with human expectations. More recently, Llama 3 [22] was released with an increased vocabulary size and context window length. The training dataset was significantly expanded compared to the previous iteration, amounting to 15 trillion tokens.

2.1.2 Metrics, benchmarks and challenges

Metrics. Surveys by Minaee *et al.* [17] and Chang *et al.* [23] describe how the evaluation process of LLMs has also evolved over time. Initially, the focus has been on traditional NLP tasks such as sentiment analysis, text classification, natural language inference (NLI), while later the focus has shifted towards natural language generation (NLG) tasks such as summarization, translation or question answering. More recently, driven in particular by the advancement of dialogue-focused LLMs, there was a shift in evaluation towards reasoning, factuality, robust-

ness, ethics and bias. The metrics used for evaluation are usually task specific. For instance, ROUGE and BLUE scores are commonly used to measure the accuracy of text generation by comparing the overlap between generated and expected texts. F1 score is the harmonic mean of precision and recall and is often used in classification tasks. Area under the curve (AUC) is also commonly used for classification tasks and is used to determine a model's capability of distinguishing between classes across different threshold values. Expected calibration error measures how well the predicted probabilities of a model align with the ground-truth probabilities, hence measuring how well a model is calibrated.

Benchmarks. Common benchmarks for LLMs include: *MMLU* [24], general knowledge and problem solving task covering 57 subject fields; *AGIEvalEnglish* [25], human-centric standardized exams task, covering exams such as college entrance exams, law school admission or math competitions; *CommonSenseQA* [26], question answering task with a focus on using background knowledge to answer common sense questions; *Winogrande* [27], common sense reasoning task, formulated as filling in the blank by choosing from 2 possible answers; *Big-Bench* [28], benchmark consisting of more than 200 tasks spread across diverse topics, designed to be beyond current LLMs capabilities; *ARC-Challenge* [29], complex reasoning task with a focus on abstraction and skill-acquisition; *TriviaQA* [30], trivia-style question answering task; *QuAC* [31], conversational-style question answering task with a focus of understanding and using the context; *BoolQ* [32], yes/no question answering task with a focus on reading comprehension; *DROP* [33], reading comprehension benchmark with tasks such as addition, counting and sorting across text paragraphs.

Challenges. LLMs often encode bias present in the training data and can subsequently reproduce or even amplify that bias when deployed in practical applications. The combination of encoded biases and the mirage of coherent generated text can lead to the perpetuation and reinforcement of stereotypes, or, when used in automated system, can lead to unfair and discriminatory outcomes [5]. The potential of malicious uses is also of great concern in LLM development. As discussed by Weidinger *et al.* [6], models can assign high probabilities to factually incorrect sequences of tokens, and because users may develop confidence in the outputs of an LLM, there is a risk of models spreading misinformation. A different study [7] notes that hallucinations of LLMs, can undermine the reliability and trustworthiness of language models. Liu *et al.* [10] mentions that a potential approach to addressing this issue could be to incorporate uncertainty estimation with regards to the model's outputs. These estimates can be used to detect when the hallucination phenomenon occurs and prevent the model from outputting misleading or fabricated information.

2.2 Uncertainty

This section addresses the topic of uncertainty estimation. It starts with an outline of the different types and sources of uncertainty, followed by an overview of general methods for quantifying uncertainty and their categorization. The focus then shifts to developments in uncertainty

estimation, specifically for LLMs, covering heuristic approaches, such as the use of likelihoods and entropy, sampling and perturbation methods, and verbalized confidence elicitation, as well as approaches which leverage the hidden states of LLMs to train modules capable of quantifying uncertainty. Lastly, this section covers the use of VAE networks for estimating uncertainty mentioning approaches based on heuristics, sampling and the variation encoded in the latent distributional representations of VAEs.

Uncertainty types. Due to neural networks becoming ubiquitous for various applications in a real-world setting, building confidence in their predictions has gained a lot of attention from researchers. Gawlikowski *et al.* [34] provide a survey article highlighting types, sources and quantification methods for uncertainty in deep neural network, as well as the importance of calibration to ensure the applicability of uncertainty estimation methods. Hüllermeier & Waegeman [35] categorized two types of uncertainty. *Aleatoric* uncertainty is intrinsic to the data and is of stochastic nature. *Epistemic* uncertainty arises from the model and can be reduced or eliminated by additional data. According to Gawlikowski *et al.* [34], sources of uncertainty include variability in data, issues with data acquisition and measurement, flaws in model design and training procedures, and the use of out-of-distribution data during inference.

Uncertainty estimation methods can be broadly categorized into four types. *Single network methods* involve either training a neural network to estimate uncertainty, or including additional components into a pre-trained network with the specific purpose of modeling uncertainty. For example, Kirchhof *et al.* [36] introduced a pre-trained uncertainty estimation module for neural networks in the space of computer vision. The module is trained as a separate head on top of a backbone vision model and uses the outputs of the backbone model together with the loss values from a classification head to model aleatoric uncertainty. They use a ranking loss function to train the uncertainty head, such that an input with a higher classification loss value is associated with a higher uncertainty. *Bayesian methods* focus on learning a distribution over model parameters and can be further sub-categorized into variational inference, sampling and Laplace approximations approaches. *Ensemble methods* estimate uncertainty based on the variance of predictions from multiple models trained independently. *Test-time augmentation methods* involve the augmentation of data during inference and analyzing the resulting distribution over the predictions in order to quantify uncertainty [34].

2.2.1 Uncertainty estimation in LLMs

Heuristic approaches. Huang *et al.* [11] explore several approaches of estimating uncertainty in LLMs. The authors highlight challenges inherent to LLMs that prevent, to a certain extent, the adoption of uncertainty estimation techniques used for more general machine learning. These challenges include the complexity of LLMs, making the use of neuron activations probing methods prohibitively expensive; the black-box nature of many models, particularly concerning training data, which prevents the use of out-of-distribution techniques; and the wide range of tasks that LLMs are used for, requiring the development of more specialized methods. Fur-

thermore, they select and evaluate a number of uncertainty quantification methods, focusing on single-inference methods, based on likelihood and entropy, and multi-inference methods. The multi-inference methods can be further sub-divided into sample-based methods, which introduce stochasticity by means of higher temperature model values, and perturbation-based methods, which generate different outputs by selecting different variations of high-entropy tokens. Their findings show that sample-based methods significantly outperform single-inference methods in terms of uncertainty estimation, while perturbation-based methods show moderate effectiveness but suffer high model-specificity. Xiong *et al.* [12] empirically evaluate the ability of LLMs to self-express uncertainty in the generated answers and showed that LLMs tend to overestimate the confidence in their answers, similarly to Groot & Valdenegro-Toro [13]. Although certain strategies improve the uncertainty estimation, the self-verbalized uncertainty elicitation approach performs worse than more traditional, white-box-style uncertainty estimation approaches. Chen *et al.* [15] introduce a new approach to uncertainty estimation which also uses hidden states of LLMs. They propose a new metric, 'EigenScore', which relies on the self-consistency of LLM generations. The score is computed as the logarithm determinant of the covariance matrix of the last-token embedding from the middle layer of the LLM across several generated sentences. Their findings show that the middle layers outperform early or final layers, and that using the hidden state of the last token is more effective than averaging the hidden states of all generated tokens.

Trained modules. Azaria & Mitchell [14] leverage the hidden activations of an LLM to train a classifier capable of detecting false statements, either provided as input to the LLM or generated by the LLM. The authors also investigate which layers of the LLM are more effective to use for assessing the truthfulness of statements and find that middle layers tend to outperform the final layers. Their work highlights the the potential of using hidden states of LLMs for uncertainty estimation. Liu *et al.* [10] present a supervised approach to uncertainty estimation for LLMs by leveraging labeled datasets. They highlight the difference between uncertainty estimation in traditional machine learning and LLMs, and argue that, due to the choice of loss function for LLMs, the hidden activations corresponding to the prompt-response pairs should encode information about the uncertainty of the generated response, information that would usually be encoded in the output logits of a neural network. They perform a feature selection process to significantly reduce the dimensionality of the hidden activations, by means of Lasso regression, mutual information and Pearson correlation. The selected features are combined with a number of logit-based features such as entropy and likelihood and used to train a random forest model which predicts the level of uncertainty of the generated text sequence. Although the presented supervised approach outperforms other unsupervised methods, it has the limitation of relying on the availability of task-specific datasets.

2.2.2 Variational Auto-Encoders (VAEs)

The VAE was introduced by Kingma & Welling [37], and it is a neural network architecture that comprises of an encoder and a decoder module. The encoder learns a probabilistic latent rep-

resentation of the input data, while the decoder learns to reconstruct the data by sampling from the latent distribution. The VAE is typically used as a generative model, in particular in the field of computer vision, however there have been a number of works which employ the VAE for the purpose of uncertainty estimation based on the intuition that the probabilistic representation can also encode information about data uncertainty.

Heuristic approaches. Böhm *et al.* [38] adopt a generative approach to address the Bayesian inverse problem of image reconstruction from corrupted data, by means of training a VAE on uncorrupted data. The use of a VAE enables estimating the uncertainty by inferring the latent distribution of the corrupted data, sampling from this distribution and using the decoder to obtain a range of possible reconstructions. The uncertainty of the corrupted data correlates with the variability of the reconstructed images.

Sampling. Belen *et al.* [39] train a network to classify ECG signals for the scope of detecting Atrial Fibrillation. The authors use a VAE network to process the input data and learn a distribution over latent variables. Samples from the latent representation are then passed through a multi-layer perceptron (MLP) to obtain a prediction. The uncertainty estimate is obtained by computing the standard deviation of the classifier's probability array after multiple passes through the network. Lin *et al.* [40] adopt a similar approach for a computer vision classification task. A VAE network is used to learn probabilistic latent representations from which samples are taken and fed through an MLP classifier. A probability distribution for the output classes, averaged over the samples, is then obtained. The final prediction is determined by selecting the class with the highest average probability across the samples, while the uncertainty estimate is given by the probability value for the predicted class.

Latent distribution variance. Catoni *et al.* [41] found that, in the context of image classification, there is mis-calibration between uncertainty estimates encoded in the latent probabilistic representations of VAEs and the informativeness of the classified images. To address this shortcoming, the authors propose an improved VAE architecture which introduces an additional latent variable acting as a scaling factor on the probabilistic latent representations. The uncertainty estimate is obtained by averaging the standard deviations of all latent variables in the network. One of the important findings is that this new architecture can correctly assign high uncertainty estimates to out-of-distribution data samples, unlike the standard VAE architecture.

3 Preliminaries

This chapter covers the preliminaries necessary to understand our proposed approach. It starts with an overview of LLMs in section 3.1, covering the language modeling task and the transformer architecture. Then, section 3.2 introduces the VAE model and covers the main theoretical concepts behind this architecture.

3.1 LLMs

This section aims to provide an understanding of the key concepts about language modeling, as well as a brief description of the key components underlying an LLM. It begins by discussing the task of language modeling, including the core concepts of autoregressive language modeling, the training objective, the loss function and perplexity as an evaluation metric. Following that, the transformer architecture, with the important mechanisms such as positional encoding, attention and feed-forward layers is covered. An important thing to note is that, throughout this work, we specifically make use the Llama 2 model architecture.

3.1.1 Language modeling

Language modeling (LM) is one of the core applications of NLP and a task performed by LLMs. The main objective of LM is to predict the probability of word sequences. There are two main approaches to the task of language modeling, namely masked language modeling and autoregressive language modeling.

Masked language modeling (MLM) is used with models such as BERT [42]. During the training, tokens in the input sequence are replaced by a special "mask" token, and the model's task is to predict the masked token given the surrounding context. One particularity of this approach is that the model has access to the context left of the masked token (all words preceding it) as well as the context right of the masked token (all words succeeding it). As such, the model learns a bidirectional representation, more specifically, a probability distribution over the vocabulary for the missing token that is conditioned on both the left and right contexts.

Autoregressive language modeling is used with generative models such as GPT [18–20] and Llama [4, 21, 22]. With this approach, the model predicts the most likely next token given a sequence of tokens. Unlike MLM, the model learns to predict a probability distribution over the vocabulary for the next token, in a sequence conditioned only on the left-hand context. Since the model depends only on the previous tokens, the model learns to generate coherent text. The generation is done in a sequential manner where, given an input sequence, the model predicts the next likely token. The predicted token is appended to the original sequence, and this updated sequence is used as input to predict the following token, this processes being repeated iteratively until a special token that represents the end-of-sequence is generated, or if other conditions are met, such as reaching the maximum specified number of tokens to be generated. Different strategies can be applied during the generation process which will determine which

token from the vocabulary will be selected. These are used as a mechanism to balance quality, reproducibility or diversity of the generated text. Such strategies include "greedy decoding" where the model always selects the highest probability token at each generation step, or "beam search" where the model explores multiple candidate generated sequences and selects the one with the highest overall probability, but can also include more complex decoding strategies. For the reminder of the current work we will focus exclusively on autoregressive language models.

More formally, language modeling can formulated as computing the joint probability for a given sequence of tokens $\boldsymbol{x} = (x_1, x_2, \ldots x_T)$, which using the chain rule, can be formulated as the product of conditional probabilities for each token in the sequence:

$$P(\boldsymbol{x}) = P(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} p(x_t \mid x_1, x_2, \ldots, x_{t-1}) \tag{1}$$

The training objective for an autoregressive language modeling task is to maximize the likelihood of the given training data under the model parameters Θ. The likelihood is formulated as the sum of log probabilities. Logs ensure numerical stability and avoid computational issues such as underflow which arise due to working with very small probability values. The training objective can be formulated as follows:

$$\mathcal{L} = \sum_{t=1}^{T} \log p(x_t \mid x_1, x_2, \ldots, x_{t-1}; \Theta) \tag{2}$$

The loss objective used in this context is cross-entropy loss, which is a measure of the distance between the predicted probability distribution $\hat{y}$ and the true probability distribution y, over the vocabulary V, for each predicted next token:

$$\mathcal{L}_{\text{cross-entropy}} = -\sum_{t \in V} \boldsymbol{y}_t \log \hat{\boldsymbol{y}}_t \tag{3}$$

Metrics. Perplexity (PPL) is a metric used to evaluate the performance of a language model on an unseen test dataset. It is defined as the exponentiated average negative log-likelihood of a sequence $\boldsymbol{x} = (x_1, x_2, \ldots x_T)$. According to Jurafsky & Martin [43], perplexity can be considered, more intuitively, as a "weighted branching factor" of a language model, which represents the average number of words the model can select from at each step. A lower perplexity score correlates with a model that is better at modeling language. However, one limitation of this metric is its dependency on the model's vocabulary, and test dataset used for evaluation it. Due to this limitation, the metric can mainly be used to compare models which have the same vocabulary, using the same test dataset.

$$\text{PPL}(\boldsymbol{x}) = \exp\left(-\frac{1}{T}\sum_{t=1}^{T} \log p(x_t \mid x_1, x_2, \ldots, x_{t-1}; \Theta)\right) \tag{4}$$

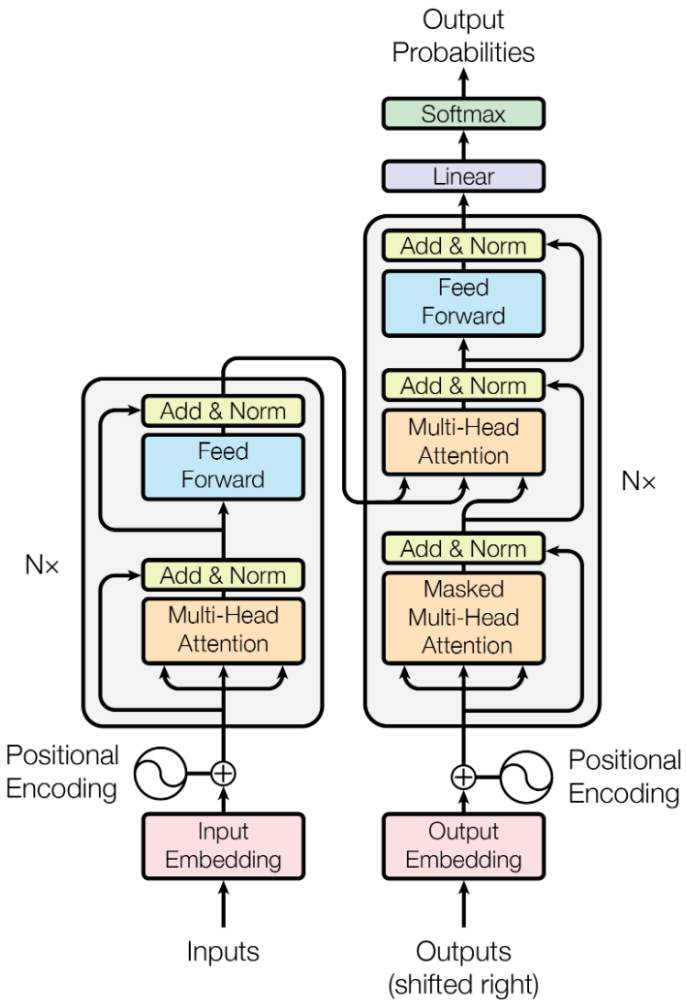

Figure 1: Transformer model architecture, from Vaswani *et al.* [16]. The architecture consists of an encoder block, on the left, and a decoder block, on the right. The decoder block, used in LLMs such as Llama 2, consists of an embedding layer, a positional encoding module, multiple stacked decoder layers and a combination of linear layer with a softmax activation. Each decoder layer consists of an attention mechanism followed by a feed-forward network.

3.1.2 Transformers

The transformer model introduced by Vaswani *et al.* [16] has become the building block of current NLP models, as well as more general machine learning models from areas such as computer vision or time-series analysis. In particular, LLMs such as GPT [18–20] and Llama [4, 21, 22] make use of this architecture. An overview of the transformer architecture can be seen in Figure 1.

While the original architecture is composed of both an encoder and a decoder block, we exclusively focus on the decoder block, as the Llama model used in this work employs a decoder-only architecture. A typical decoder block consists of an embedding layer, a positional encoding module, multiple stacked decoder layers, and a final language modeling head. Furthermore, it is worth noting that while the original transformer architecture includes an additional cross-attention module in each decoder layer which incorporates information from the encoder block,

the decoder-only model eliminates this cross-attention module. In the following paragraphs, we briefly cover some of the key components of the decoder block: the positional encoding, the attention mechanism and the feed-forward module.

Positional Encodings. The order of tokens in a sequence is very important for the task of language modeling. To capture this positional information, the transformer architecture makes use of "positional encodings" which are added to the input token embeddings to differentiate between positions within a sequence. Llama 2 deviates from the original transformer architecture by employing relative rotary positional embeddings (RoPE) instead of fixed positional embeddings. Additionally, Llama 2 applies the positional encoding at each decoder layer, as opposed to the original architecture where the positional encoding is applied only to the input embeddings.

Attention mechanism. One of the key innovations introduced by Vaswani *et al.* [16] is the scaled dot-product self-attention mechanism. For a given token, the self-attention mechanism computes a representation which is a weighted combination of all tokens in the sequence. For decoder-only transformers, an additional mask is used, to ensure that each token can only attend to earlier tokens in the sequence. The input token representation is projected into three matrices that are learned: the query (Q), key (K) and value (V). The queries and keys are used to compute attention scores which determine the weight of each token's importance. The value matrix is then used to generate a representation for the input token, by taking the weighted combination of the all tokens' representations in the sequence. The attention mechanism is formalized in Equation (5) and a schematic representation can be seen in Figure 2. This approach has a significant advantage over previous mechanisms used to capture long-range dependencies, such as RNNs, because the attention mechanism can be parallelized, enabling faster computations, while effectively capturing both short- and long-range dependencies. Multi-head attention extends this mechanism by using multiple self-attention heads. This allows the model to project the input into multiple vector spaces, thus capturing different syntactic and semantic meaningful information. These distinct representations are then concatenated together and processed further in the decoder layer. Llama 2 replaces the multi-head attention mechanism with a grouped-query attention mechanism introduced by Ainslie *et al.* [44], which is more efficient in terms of computation during inference.

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5}$$

Feed-forward network. After the attention mechanism in each decoder layer, the output passes through a feed-forward network (FFN), which consists of two linear layers with a non-linearity in-between.

Decoder layers also incorporate residual (skip) connections and layer normalization. Residual connections allow the input to one sub-module, such as attention, to bypass that sub-module and be added to its output. Layer normalization is applied to the output of a sub-module to rescale

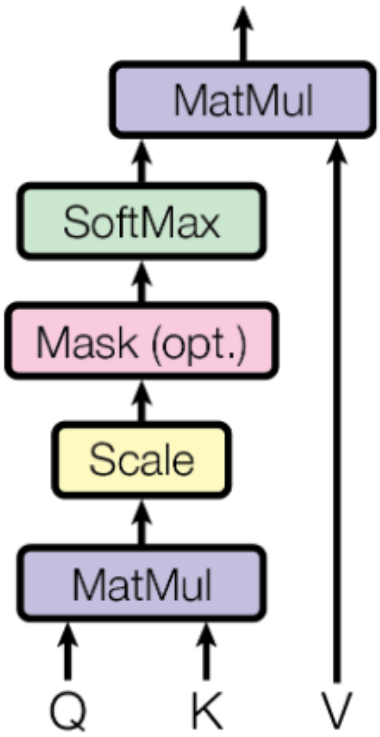

Figure 2: Schematic representation of dot-product attention, from Vaswani *et al.* [16], illustrating how the query (Q), key (K) and value (V) matrices are used to compute attention scores.

it such that the representations maintain a zero mean and unit variance. One change that Llama 2 incorporates is that the normalization step is applied to the input of a sub-module, rather than the output. Lastly, the output of the final decoder layer is passed through the language modeling head, which consists of a simple linear transformation, followed by a softmax layer. This produces a probability distribution over the model's vocabulary, enabling the model to predict the most likely token that would follow the given sequence of tokens.

Transformers have gained a lot of traction in the field of NLP, becoming the default architecture for a wide range of applications. However, the mechanisms through which these models store information and encode factual associations remain sparsely explored. In this work, we propose an approach that encodes the uncertainty level of these learned factual associations, such that we can gain a better understanding of these processes, as well as improve the reliability and interpretablity of transformers.

3.2 VAEs

This section provides a brief overview of the core concepts underlying the VAE. It begins by briefly covering what an auto-encoder is, then continues by going more in-depth about the key components of a VAE, the training objective and the loss function.

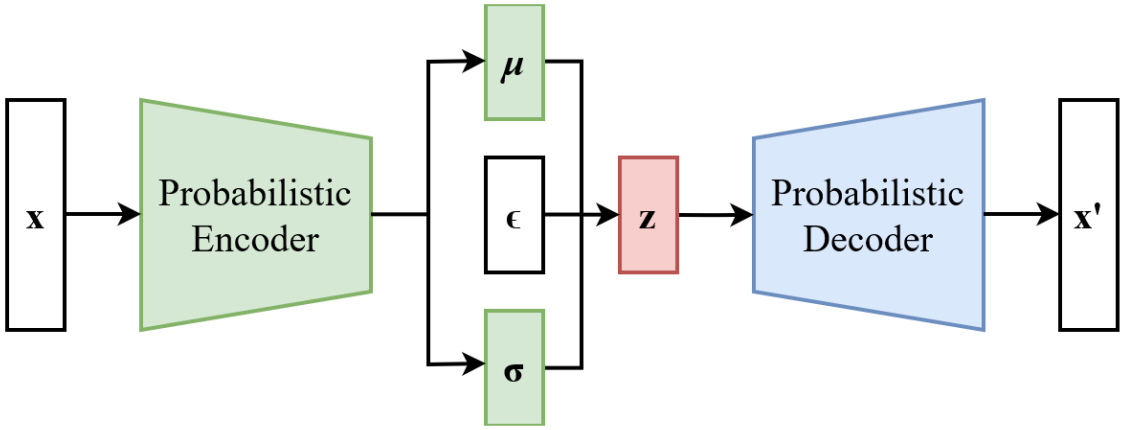

Figure 3: Variational Auto-Encoder model architecture, from [46]. The architecture consists of an encoder block that learns a latent probabilistic representation of the input data, and a decoder block that learns to reconstruct the input data by sampling from the latent representation.

Auto-encoders, first introduced by Rumelhart *et al.* [45], are a type of neural network that is trained in an unsupervised manner to reconstruct its input, in the process learning a meaningful latent (hidden) representation of it. An auto-encoder consists of two blocks, an encoder and a decoder, both typically fully connected feed-forward neural networks. The encoder maps the input data to a latent, usually lower dimensional space, while the decoder reverses the mapping and reconstructs the original data from this latent representation.

Variational Auto-Encoders, introduced by Kingma & Welling [37], is an extension of the original auto-encoder architecture, with a key improvement. Instead of learning a single latent representation for a given input, the VAE learns a distributional latent representation. More specifically, for an input $\boldsymbol{x}$, the encoder block of a VAE learns a mapping of the input to a latent probability distribution, characterized by a mean $\boldsymbol{\mu}$ and a standard deviation $\boldsymbol{\sigma}$. The input for the decoder block is $\boldsymbol{z}$, which is sampled from this learned latent probability distribution using the variable $\boldsymbol{\varepsilon}$ as a parameter. The decoder then reconstructs the original input as $\hat{\boldsymbol{x}}$. A schematic overview of this architecture can be seen in Figure 3.

More formally, a VAE models inputs $\boldsymbol{x}_i \in \boldsymbol{X}$, and aims to reconstruct each $\boldsymbol{x}_i$ through a process conditioned on a latent variable $\boldsymbol{z}$. The latent variable $\boldsymbol{z}$ is sampled from a prior distribution $p_\theta(\boldsymbol{z})$ and each input $\boldsymbol{x}_i$ is generated from a conditional distribution $p_\theta(\boldsymbol{x}_i \mid \boldsymbol{z})$. This conditional distribution corresponds to the probabilistic decoder block in Figure 3. As such, the goal of a VAE is to maximize the likelihood of the given data, as defined by Equation (6).

$$p_\theta(\boldsymbol{x}_i) = \int p_\theta(\boldsymbol{z}) p_\theta(\boldsymbol{x}_i \mid \boldsymbol{z})\, dz \tag{6}$$

However, this likelihood is intractable and cannot be evaluated since it involves integrating over all possible values of this latent variable $\boldsymbol{z}$. To overcome this, a VAE estimates how $\boldsymbol{z}$ is distributed given the observed data points $\boldsymbol{X}$ using the true posterior density:

$$p_\theta(\boldsymbol{z} \mid \boldsymbol{x}_i) = \frac{p_\theta(\boldsymbol{x}_i \mid \boldsymbol{z}) p_\theta(\boldsymbol{z})}{p_\theta(\boldsymbol{x}_i)} \tag{7}$$

Similarly to the likelihood, the true posterior is also intractable. In order to solve this issue, the VAE will approximate the posterior distribution $q_\phi(\boldsymbol{z} \mid \boldsymbol{x}_i)$, which corresponds to the probabilistic encoder block in Figure 3. The parameters θ and ϕ of the encoder and, respectively, the decoder block need to be learned from the training data. This is done by maximizing the evidence lower bound (ELBO), which is defined by Equation (8) [47].

$$\mathcal{L}_{ELBO}(\theta, \phi, \boldsymbol{x}_i) = -D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}_i) \parallel p_\theta(\boldsymbol{z})) + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x}_i)}[\log p_\theta(\boldsymbol{x}_i|\boldsymbol{z})] \tag{8}$$

The ELBO is balancing two complementary objectives, corresponding to the two terms of the equation. The first term in Equation (8) is based on the Kullback-Leibler (KL) divergence and quantifies how close the posterior probability distribution $q_\phi(\boldsymbol{z} \mid \boldsymbol{x}_i)$ is to the prior $p_\theta(\boldsymbol{z})$. The second term in Equation (8) ensures that the input data can be accurately reconstructed, and it is often referred to as the reconstruction loss. Typically, a loss function such as Mean-Squared Error (MSE) is used for this objective.

In order to train a VAE model with common optimization methods such as Stochastic Gradient Descent (SGD), by means of backpropagation, the model must be differentiable with respect to the model's parameters. Since the encoder and decoder blocks are typically feed-forward neural networks which are differentiable, these pose no issues for backpropagation. However, the sampling of the latent variable z is a stochastic process which is not inherently differentiable. To overcome this issue, the VAE uses a change of variables called the "reparametrization trick". An additional random variable $\varepsilon \sim \mathcal{N}(0, 1)$, which is treated as an input, is introduced and used to simulate the sampling of $\boldsymbol{z}$ from the distribution defined by mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ by computing $\boldsymbol{z} = \boldsymbol{\mu} + \varepsilon\boldsymbol{\sigma}$.

4 Approach

This chapter starts with an overview of our approach, VALUE, in section 4.1 and is followed by a brief explanation of the reconstruction and latent distribution objectives typical of a VAE in section 4.2. Next, the proposed uncertainty objectives are presented in section 4.3 and, lastly, the aggregated loss for uncertainty estimation used by the VALUE model is described in section 4.4.

4.1 Variational Auto-encoder for Latent Uncertainty Encoding (VALUE)

In this section, we will outline the approach we propose for encoding uncertainty for an LLM. A schematic overview of the architecture of our proposed approach can be seen in Figure 4.

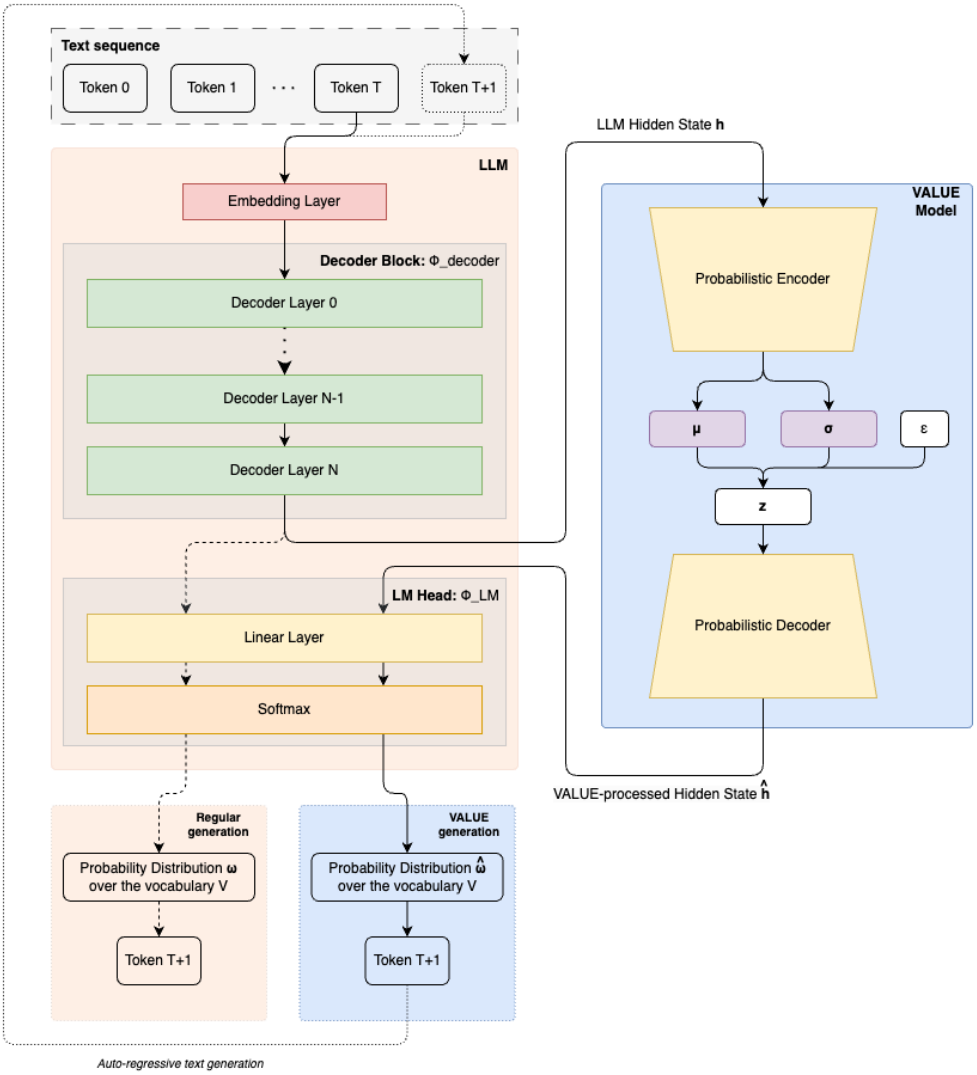

Figure 4: Diagram of VALUE framework. The diagram shows the architecture of the VALUE model, as well as the VALUE-enhanced text generation process for an LLM. Given a text sequence $\boldsymbol{x}$ of length T, we pass the last token of the input (x_T) to the LLM. We capture the hidden state $\boldsymbol{h}$ from the last decoder layer of the LLM and process it with the VALUE model to obtain $\hat{\boldsymbol{h}}$. We feed the hidden state $\hat{\boldsymbol{h}}$ back to the LLM, pass it to the language modeling head (LM) and generate a new token (x_{T+1}) on the basis of newly computed probability distribution $\hat{\boldsymbol{\omega}}$ over the LLM's vocabulary V. This new token is then fed back into the LLM as a new input and the text generation process continues autoregressively in the manner described above. The regular generation process of an LLM is shown for illustrative purpose and is only used for the harvesting of hidden states used to train the VALUE model.

The VALUE framework integrates a VAE model between the final decoder layer of an LLM and the language modeling (LM) head. Throughout the reminder of this work, we use the terms VAE model and VALUE model interchangeably. The VAE consists of an encoder and a decoder block, each block being made up of a two layer MLP with ReLU activation, layer normalization and dropout applied after each layer. The loss function of the VAE is updated, with respect to a standard VAE model, with an additional uncertainty-focused objective in order to allow the model to encode uncertainty in its latent representation. The uncertainty-focused loss term is described in more detail in section 4.3.

The VAE model receives as input the hidden states $\boldsymbol{h}_i$ from the LLM's last decoder layer. After being processed through the VAE, the output hidden states $\hat{\boldsymbol{h}}_i$ are injected back into the LLM, at the same position from which they were harvested. These hidden states thus become the input for the LM head which will select the next token to be generated, from a probability distribution $\hat{\boldsymbol{\omega}}$ over the vocabulary V of the LLM. An important thing to note is that, during the training of the VAE model, both the input prompt tokens and the newly generated tokens are processed by the VAE model. However, during inference, only the generated tokens are processed through VAE, while the input prompt tokens are processed by the LLM in an unaltered manner.

The main processes behind the VALUE framework, namely the harvesting of hidden states, the training of the VALUE model and the inference step of the VALUE framework, are detailed as follows:

Hidden states harvesting. Given a token x_i of a prompt input sequence $\boldsymbol{x} = (x_1, x_2, \ldots x_T)$ of length T and an autoregressive LLM Φ, we obtain the corresponding hidden state $\boldsymbol{h}_i$ from the output of the last decoder layer of the LLM, by processing each token x_i through the decoder layers of Φ: $\boldsymbol{h}_i = \Phi_{decoder}(x_i \mid x_1, x_2, \ldots, x_{i-1})$. Once the input prompt has been processed, the LLM will generate new tokens in an autoregressive manner by sampling the next token x_{T+1} from the probability distribution $\boldsymbol{\omega}_T = \Phi_{LM}(\boldsymbol{h}_T)$, obtained by processing $\boldsymbol{h}_T$ through the LM head of the LLM, until the end-of-sequence token is generated, or until the generation token limit has been reached. All hidden states $\boldsymbol{h}_i$, both from the input prompt tokens and the generated tokens, together with the corresponding next token x_{i+1} are stored as a dataset that will be used to train and evaluate the VALUE framework. This setup will allow the VAE model part of the VALUE framework to efficiently learn to map the hidden states of an LLM to a probabilistic latent space from which we can sample and generate new uncertainty-enhanced embeddings in the original vector space of the LLM.

VALUE Training. At each autoregressive step, the LLM's harvested hidden states $\boldsymbol{h}_i$, mentioned in the previous paragraph, are passed through the VAE. The encoder block of the VAE will map these hidden states to a latent probabilistic representation, characterized by a mean vector $\boldsymbol{\mu}_i$ and a log variance vector $\boldsymbol{\sigma}_i$. The VAE uses the "reparametrization trick", where ε is a scalar value sampled randomly from a normal distribution, in order to sample $\boldsymbol{z}_i$ from the latent probabilistic representation. This latent representation $\boldsymbol{z}_i$ is then processed through the

VAE decoder and mapped back to the embedding space of the LLM as $\hat{\boldsymbol{h}}_i$. The training loss, detailed in section 4.4 is computed, together with the gradients needed for the backpropagation step. Lastly, the backpropagation step updates the VAE's parameters, and the process continues with a new training batch of harvested hidden states. We apply an early stopping mechanism for the training process, based on the perplexity value of the LLM computed on the held-out test subset. It is important to note that the parameters of the LLM are frozen, and thus not affected by the backpropagation process.

VALUE Inference Once the VAE model is trained, it can be used together with the LLM for inference. During the inference step, the VALUE framework generates text starting from an input prompt. The prompt is processed by the LLM unaffected by any VALUE processing. Once the prompt has been processed and new tokens start being generated, the VALUE model modifies the hidden states of the LLM and consequently alters the generated text. The hidden state $\boldsymbol{h}_i$ from the LLM's last decoder layer, corresponding to the last token x_i of the input sequence is processed through the VALUE model and mapped to a latent probabilistic representation. We sample $\boldsymbol{z}_i$ from this probabilistic representation using ε as an input parameter, and map it back to the original embedding space of the LLM, as $\hat{\boldsymbol{h}}_i$. This VALUE-processed hidden state is then passed through the LM head to obtain a probability distribution $\hat{\boldsymbol{\omega}}_{i+1}$ over the LLM's vocabulary V. The next generated token x_{i+1} is sampled from $\hat{\boldsymbol{\omega}}_{i+1}$. The process is then repeated in an autoregressive manner, until the end-of-sequence token is generated, or the new token generation limit has been reached. In this formulation, $\boldsymbol{\sigma}$ defines a 'region' of uncertainty with regards to the latent representation of the token to be generated and ε is used as an input parameter to control the level of uncertainty in the generation process. More specifically, ε directly influences $\hat{\boldsymbol{\omega}}$, with larger values producing more uniform, and uncertain, distributions over V.

The VALUE framework aims to leverage the properties of VAEs, particularly the ability of the model to map the hidden states of the LLM to a probabilistic latent space, and in the processes, encoding the inherent uncertainty of LLM-generated text. In order to achieve that, the model needs to be trained using a well designed loss function, which encourages the VAE architecture to encode uncertainty in its latent representations. As such, the loss function is made up of the loss objective standard to a VAE, briefly covered in section 4.2, and an additional loss objective which aims to encode uncertainty in the latent representations, covered in section 4.1. The following sections provide a more in-depth explanation of the different objectives which contribute to the loss objective central to the VALUE model.

4.2 Reconstruction and latent distribution objective

The standard objective for a VAE, as outlined in section 3.2, is ELBO. This objective comprises two components: the reconstruction loss and the KL divergence loss. Minimizing both loss terms is important to obtain an effective VAE model which can learn meaningful probabilistic latent representations.

4.2.1 Reconstruction loss

The reconstruction objective encourages a VAE model to reconstruct the original LLM hidden activations with a high degree of accuracy. Therefore, we compute the MSE loss between the input hidden states $\boldsymbol{h}_i \in \mathbb{R}^D$ and the reconstructed output $\hat{\boldsymbol{h}}_i \in \mathbb{R}^D$. The formula for MSE loss, as used in our approach, is given by Equation (9), where D is the hidden dimension size. In addition, an attention mask is applied during the computation of the loss such that only valid tokens are considered for the calculation, while special tokens added by the tokenizer such as beginning-of-sequence (BOS), padding (PAD) and end-of-sequence (EOS) tokens are ignored. Minimizing this loss is vital for maintaining the coherence of the generation process in the LLM, as reconstruction errors can be amplified further along in the sequence due to the autoregressive nature of LLMs.

$$\mathcal{L}_{MSE}(\boldsymbol{h}_i, \hat{\boldsymbol{h}}_i) = \frac{1}{D} \sum_{j=1}^{D} \|\boldsymbol{h}_{i,j} - \hat{\boldsymbol{h}}_{i,j}\|_2^2 \tag{9}$$

4.2.2 KL divergence loss

The KL divergence objective is a regularization mechanism used to ensure a VAE maps its input to a continuous latent space, that follows a predefined prior distribution. This term measures the difference between the shape of the learned latent distribution and the shape of the prior distribution. In our approach, we select a Gaussian distribution with $\mu = 0$ and $\sigma = 1$ as the prior distribution. The formula for the KL divergence loss term, as used in our approach, is given by Equation (10), where D is the hidden dimension size. Similarly to the MSE loss, we also apply an attention mask to exclude from the calculation the BOS, PAD and EOS tokens. Minimizing this loss is important for maintaining a smooth latent space and a meaningful sampling process.

$$\mathcal{L}_{KL}(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i) = \frac{1}{D} \sum_{j=1}^{D} \left(-\frac{1}{2} \left(1 + \boldsymbol{\sigma}_{i,j} - \boldsymbol{\mu}_{i,j}^2 - e^{\boldsymbol{\sigma}_{i,j}}\right) \right) \tag{10}$$

4.3 Uncertainty objective

As mentioned in the beginning of this chapter, we adopt a VAE architecture with a modified loss function which includes an uncertainty objective. The goal of this uncertainty objective is to encourage the VAE model to encode uncertainty in its learned latent representations. Since the VALUE framework uses the VAE model to process hidden states that are fed back into an LLM - which then generates new tokens by selecting the most likely candidate based on the probability distribution ω over the LLM's vocabulary V - we also compute the altered probability distribution $\hat{\omega}$ over V using the VAE-processed hidden states $\hat{\boldsymbol{h}}$ during the training step of the VALUE model. This modified probability distribution is used in the loss computation for the uncertainty objective, the intuition behind this being that the uncertainty we aim to encode is reflected in the probability distribution over the LLM's vocabulary.

The uncertainty objective must be influenced by the sampling process used to obtain z. Intuitively, when sampling from the latent space close to the mean of the distribution, we expect to encode a low level of uncertainty. Conversely, sampling further away from the mean, we expect to encode a higher degree of uncertainty. Thus, a "distance" metric is needed to indicate how far away from the mean we sample. In a VAE model, the sampling process is determined by "reparametrization trick", which is controlled by the ε variable.

However, ε cannot directly be used as a distance metric as it is unbounded and can take both positive and negative values. In our approach we transform ε to create a useful, bounded, distance metric, as defined by Equation (11). By applying the hyperbolic tangent function tanh we scale ε in the range of $[-1,1]$. Furthermore, by taking the absolute value, we focus exclusively on the distance of the deviation from the mean, rather than also taking into account the direction, effectively bounding the value in the $[0,1]$ range. As such, ε_{norm} can be effectively used as a distance metric or, more precisely, as a scaling factor, which determines the degree of uncertainty that should be encoded by the VAE.

$$\varepsilon_{norm} = |\tanh(\varepsilon)| \tag{11}$$

In this section we present two candidate uncertainty-focused objectives, an entropy-based loss and an uncertainty KL divergence loss. For clarification, this uncertainty-focused KL divergence loss is different than the standard KL divergence loss used by a VAE model to regularize the latent space, and is used in conjunction with this standard KL divergence loss. These two approaches are evaluated and the final VALUE framework will adopt the best performing approach as the uncertainty objective used.

4.3.1 Entropy loss

This uncertainty objectives, defined by Equation (12), acts as a regularization term and is based on the absolute difference between the current entropy $H(\hat{\omega}_i)$, calculated based on the LLM's vocabulary distribution $\hat{\omega}_i$ using the VAE-processed hidden states $\hat{h}_i$, and a target entropy value $H_{target}(\omega_i)$, calculated on the basis of the original distribution ω_i.

$$\mathcal{L}_{Uncertainty}(\omega_i, \hat{\omega}_i) = \left| H_{target}(\omega_i) - H(\hat{\omega}_i) \right| \tag{12}$$

The entropy H of a probability distribution is a measure of the average level of uncertainty encoded by that distribution. The formula for entropy is given by Equation (13), where $P_i(x_j)$ is the probability of outcome x_j - in the context of the this uncertainty objective, $P_i(x_j)$ represents the probability of token x_j of the LLM's vocabulary V being generated next, given the current sequence of tokens $x = (x_1, x_2, \ldots x_i)$.

$$H(\omega_i) = -\sum_{j \in V} P_i(x_j) \log P_i(x_j) \tag{13}$$

The target entropy $H_{target}(\omega_i)$, defined in Equation (14), is an interpolation between the original vocabulary entropy of the unaltered LLM $H(\omega_i)$ and the maximum entropy value of a uniform categorical distribution of the same size as the LLM's vocabulary $H_{uniform}$. The interpolation is done using the ε_{norm} parameter, described in the previous section. As such, when ε_{norm} is close to zero, meaning the latent value z_i is close to the latent distribution's mean, we want the target entropy to be very similar to the original entropy. In contrast, when z_i is sampled from the tail ends of the latent distribution, we set the target entropy to be high, close to that of a uniform distribution.

$$H_{target}(\omega_i) = H(\omega_i) + \varepsilon_{norm} \cdot (H_{uniform} - H(\omega_i)) \tag{14}$$

4.3.2 Uncertainty KL divergence loss

This uncertainty objective employs an interpolation between two KL divergence terms, as defined in Equation (15). The ε_{norm} parameter is used as an scaling factor for this interpolation.

$$\mathcal{L}_{Uncertainty}(\omega_i, \hat{\omega}_i) = (1 - \varepsilon_{norm}) \cdot KL_{upper}(\omega_i, \hat{\omega}_i) + \varepsilon_{norm} \cdot KL_{lower}(\omega_i) \tag{15}$$

The first term, $KL_{upper}(\omega_i, \hat{\omega}_i)$ defined in Equation (16), measures the KL divergence between the current probability distribution $\hat{\omega}_i$ over the LLM's vocabulary V and the unaltered probability distribution ω_i. The second term, $KL_{lower}(\omega_i)$ defined in Equation (17), computes the KL divergence between the current probability distribution ω_i and a uniform distribution $\mathcal{U}$ of the same size as the LLM's vocabulary V.

$$KL_{upper}(\omega_i, \hat{\omega}_i) = D_{KL}(\hat{\omega}_i \parallel \omega_i) \tag{16}$$

$$KL_{lower}(\omega_i) = D_{KL}(\omega_i \parallel \mathcal{U}) \tag{17}$$

The intuition behind this uncertainty objective is similar to the one used for the entropy-based objective, however, using the KL divergence rather than the entropy should provide a stronger signal to the model, with the potential downside of being a harder constraint for the optimization. The objective with this approach is not only to control the "quality" of the distribution - that it should be more or less uncertain - but to also shape the distribution, such that it better aligns with the desired target.

4.4 Aggregated loss for uncertainty estimation

The aggregated loss for the VALUE framework, as defined by Equation (18), integrates the reconstruction loss and the KL divergence loss of a typical VAE model, together with an additional uncertainty loss. This aggregated objective guides the VALUE model to encode meaningful latent representations of the LLM's hidden states, which can then be accurately projected back into the LLM's embedding space, and which capture the inherent uncertainty of the LLM's generation process.

$$\mathcal{L}_{VALUE} = (1 - \varepsilon_{norm}) \cdot \lambda_1 \cdot \mathcal{L}_{MSE}(\boldsymbol{h}_i, \hat{\boldsymbol{h}}_i) + \lambda_2 \cdot \mathcal{L}_{KL}(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i) + \mathcal{L}_{Uncertainty}(\boldsymbol{\omega}_i, \hat{\boldsymbol{\omega}}_i) \tag{18}$$

The MSE loss term $\mathcal{L}_{MSE}$ is scaled by $(1 - \varepsilon_{norm})$, which reduces the influence of the reconstruction objective when samples are taken from the tail end of the latent distribution, allowing the model to prioritize uncertainty. At the same time, this scaling ensures an accurate reconstructions when samples are closer to the mean of the distribution. Furthermore, an additional scaling term λ_1, a hyper-parameter specific to the LLM used, is used to ensure that the MSE term remains balanced in magnitude with the other loss components and, as such, does not dominate the loss landscape.

The KL divergence loss term $\mathcal{L}_{KL}$ is scaled by λ_2, which is an annealing term. Following the work of Kingma & Welling [48], this annealing strategy prevents the model from being stuck in a local minima early in the training process. By gradually increasing the influence of the KL term, the model is encouraged to optimize the reconstruction loss first, ensuring a more effective training process. We investigate the effect of this annealing term in section 6.2.

The uncertainty loss term $\mathcal{L}_{Uncertainty}$ is defined as either the entropy-based term detailed in section 4.3.1, or the KL divergence-based term described in section 4.3.2. This term, influenced by ε_{norm}, adjusts accordingly to encode different levels of uncertainty based on the samples' distance to the mean of the latent distribution, and thus capturing the inherent uncertainty of an LLM's generation capabilities.

5 Results

This chapter covers the details relevant to the training process, and the evaluation of our uncertainty encoding approach for LLMs. The chapter starts with a brief description of the dataset used in our work in section 5.1. Next, section 5.2 goes into detail about the training process and choice of hyperparameters for the VALUE framework. The metrics used for evaluating the VALUE model are described in section 5.3. The main quantitative results of our work are presented in section 5.4. Lastly, a brief qualitative analysis is presented in section 5.5.

5.1 Dataset

We use `CounterFact`, introduced by Meng *et al.* [49], as the main benchmark. The dataset contains factual knowledge in free-form partial sentence pairs. Originally designed to evaluate methods for altering the factual associations learned by LLMs, this dataset is also suitable for assessing an LLM's ability to generate factually accurate text, and in turn, evaluate our model's ability to encode the uncertainty of the generated text.

The dataset contains 21919 text pairs, each representing a factual association between a subject and a target, across 34 different types of relationships. For our research, we make use of the *prompts*, *subjects*, and *true targets* fields from the original dataset. Each prompt describes a type of relationship and contains a placeholder (a pair of curly brackets) for the subject, which is replaced with the appropriate entity during the pre-processing of the dataset, making the statement factually accurate.

Table 1 showcases examples of prompts, subjects and expected answers. The first column contains different categories of relationships, while the second and third column contain a list of subjects and, respectively, a list of the corresponding targets. The processed prompts are used as inputs to the LLM for generating text. The true targets are used to evaluate the generated text in order to asses the factuality and the uncertainty of the LLM.

Prompts	**Subjects**	**True Targets**
{} is located in	[Vierlingsbeek, Helsinki, . . .]	[Netherlands, Europe, . . .]
The domain of activity of {} is	[Alan Turing, Edwin Hubble, . . .]	[logic, astronomy, . . .]
{} was developed by	[Internet Explorer 5, iPod, . . .]	[Microsoft, Apple, . . .]
{} professionally plays the sport	[Ryan Smyth, Otto Graham, . . .]	[hockey, basketball, . . .]

Table 1: Examples of Prompts, Subjects, and True Targets from the `CounterFact` dataset. The curly brackets in the prompts indicate placeholders where the subject must be inserted.

5.2 Training

The training of the VALUE model is an important aspect of the current research. This section covers the training process, as well as the configurations and hyperparameters for both the VALUE model and the LLMs used. In our work, we train two versions of the VALUE model, each using a different LLM. Both LLMs used are based on the Llama 2 architecture [4]: Open Llama 2 3B, with 3 billion parameters, and Meta's Llama 2 7B, with 7 billion parameters.

The first step of the training process is to pass `CounterFact` *subject* prompts to the corresponding LLM, to generate the unaltered predicted targets and capture the hidden states used to train the VALUE model. The settings used for Llama 2 3/7B to generate text can be seen in Table 2. As the prompts contained in the `CounterFact` dataset are relatively short, a sequence limit of 30 tokens long was used for the text generation. The parameters were chosen in such a manner that the generation process is deterministic, therefore no sampling is done and the temperature of the LLMs is set to 0. Furthermore, we use *Decoding By Contrasting Layers* (DoLa) [50] as a text generation strategy, to improve the factuality of the LLM. The parameters were chosen to reduce repetitions and hallucinations for the short-answer task. We terminate the generation process if a newline character or a period character is autoregressively generated.

Parameter	**Value**
max_length	30
do_sample	False
num_return_sequences	1
top_p	None
temperature	0
use_cache	True
dola_layers	"high"
no_repeat_ngram_size	2
early_stopping	True
repetition_penalty	1.2
stop_strings	["\n", "."]

Table 2: LLM Generation Configuration Parameters

The prompts are first tokenized using the tokenizer corresponding to the selected LLM, then batched and padded accordingly. For the training of the VALUE model, we generate text in batches, based on the given prompts. We obtain the hidden states of the last decoder layer of the LLM, together with the model's logits, which we use to compute the entropy over the LLM's vocabulary. Within our training set-up, we capture and use additional metadata such as the prompt length, the size of the padding, and the generated token ids. We cache the LLM's activations to the given prompts, which speeds up the training process of the VALUE model by eliminating the need to recompute the hidden activations for each epoch of the training process.

During the training loop, we process each batch of prompts, generated text sequences and their corresponding hidden states through our VALUE model. For the uncertainty-focused training objectives, the hidden states processed through the uncertainty-VAE are fed back into the LLM as input for the language modeling head. New text sequences are then generated from these variance-encoded hidden states. The new logits are used to compute a new probability distribution over the vocabulary of the LLM. We use the new probability distribution to compute the loss objective of the model. During training, we log the different loss values, and keep track of the perplexity scores over the evaluation set. We employ early stopping when the average perplexity on the evaluation subset begins to increase. We use the AdamW optimizer, together with a cosine annealing scheduler. The relevant hyperparameters are shown in Table 3.

Hyperparameter	**Open Llama 2 - 3B**	**Llama 2 - 7B**
Batch Size	128	128
Learning Rate (lr)	1e-4	1e-4
Optimizer Weight Decay	1e-4	1e-4
Scheduler T_0	800	800
Scheduler eta_min	5e-7	5e-7
KL Warmup (Epochs)	100	100
Latent Size	1500	3000
Reconstruction Weight	0.1	0.1

Table 3: Training Hyperparameters for Llama 2 (3B) and Llama 2 (7B).

5.3 Metrics

This section details the metrics used to evaluate the performance of the VALUE model and quantify its ability to encode LLM uncertainty. These metrics aim to quantify the quality of the generated text, the similarity between the original generated text and the VALUE-processed generated text, as well as the alignment of the VALUE-processed generated text with the expected answers. Namely, we use the semantic similarity and BLEU score between the original generated text and the VALUE-enhanced generated text, as well as the perplexity score, and answer coverage.

SemScore Aynetdinov & Akbik [51] evaluates how closely the text generated by the VALUE-processed hidden states matches, in semantic meaning, the original generated text. *SemScore* uses an additional verifier small language model, fine-tuned specifically for measuring sentence similarity, to encode the generated pieces of text and compute the cosine similarity between the corresponding embeddings.

BLEU is a metric commonly used in machine translation and text generation tasks, providing a measure of n-gram overlap between the VALUE-generated text and the original generated text.

This metric is complementary to *SemScore*, as it measures the similarity between the two pieces of text. However, *BLEU* does not take into account the semantics of the two texts but rather the overlap between generated tokens.

Perplexity (PPL) is a metric commonly used to evaluate the performance LLMs. We discuss it briefly in section 3.1.1. Furthermore, perplexity also correlates with the coherence of the generated text; i.e. a version of an LLM achieving a significantly higher perplexity than a different version of the same LLM is likely to generate noisier text that is less coherent. In our evaluation, we look at the difference between the perplexity of the original LLM on the evaluation dataset and the perplexity of the VALUE-modified LLM. Ideally, we expect the difference in perplexity to be as small as possible when we sample from the latent space of the VAE close to the mean of the latent distribution, and we expect this difference to grow as we start sampling further away from the mean.

Answer Coverage (AnsCov) is a simple metric that we employ to assess the factuality of the generated text across the different ε values, which serves as an "uncertainty slider", and is used to sample from the latent distribution learned by the VALUE model. A generated sample obtains an answer coverage score of 1 if the *true target* from the `CounterFact` dataset associated with the prompt is present in the generated text and 0 otherwise. We average this score over the prompts present in the evaluation set used for our VALUE framework.

5.4 Main results

This section covers the quantitative evaluation of the VALUE framework. In particular, we compare the two uncertainty objectives described in section 4.3.1 and section 4.3.2 for the VALUE model, and we evaluate them against a baseline consisting of a standard VAE architecture. Furthermore, we evaluate our proposed approach using both the Open Llama 2 3B model, in section 5.4.1, and with Meta's Llama 2 7B model in section 5.4.2, in order to determine whether the VALUE framework can be generalized to different LLMs. The main goal of this evaluation is to quantitatively assess the effect of the uncertainty objective and, in turn, VALUE's ability to encode uncertainty. We are interested in investigating whether these capabilities do indeed arise from the uncertainty objective, by comparing it to the baseline VAE architecture, and assessing the text generation ability of VALUE-enhanced LLMs.

5.4.1 Llama 2 3B

In this section, we report results on three VALUE settings: **BASE**, **VALUE-E** and **VALUE-KL**.

BASE. Table 4 shows the evaluation metrics for the VALUE framework with Open Llama 2 3B, using a standard VAE reconstruction objective, without our introduced uncertainty terms. Across metrics, we do not observe a significant variation as a function of ε. The generated text remains consistent in terms of both semantic similarity to the unaltered LLM-generated text

Metric	**Baseline**	$\varepsilon = -3$	$\varepsilon = -2$	$\varepsilon = -1.5$	$\varepsilon = -1$	$\varepsilon = -0.5$	$\varepsilon = -0.25$	$\varepsilon = 0$	$\varepsilon = 0.25$	$\varepsilon = 0.5$	$\varepsilon = 1$	$\varepsilon = 1.5$	$\varepsilon = 2$	$\varepsilon = 3$
AnsCov	0.33	0.302	0.301	0.300	0.298	0.300	0.301	0.300	0.300	0.300	0.299	0.299	0.299	0.299
BLEU	-	0.532	0.532	0.531	0.532	0.532	0.532	0.532	0.532	0.531	0.531	0.532	0.532	0.532
SemScore	-	0.613	0.614	0.615	0.613	0.614	0.613	0.613	0.614	0.614	0.614	0.614	0.613	0.613
PPL	15.897	31.154	30.775	30.616	30.468	30.374	30.335	30.306	30.268	30.250	30.232	30.258	30.284	30.378

Table 4: VALUE framework evaluation results for Open Llama 2 3B LLM with baseline VAE architecture (**BASE**). Metrics are reported across different values of ε. As baseline, the unaltered LLM achieves a *PPL* score of 15.897 and a *AnsCov* score of 0.33.

Metric	**Baseline**	$\varepsilon = -3$	$\varepsilon = -2$	$\varepsilon = -1.5$	$\varepsilon = -1$	$\varepsilon = -0.5$	$\varepsilon = -0.25$	$\varepsilon = 0$	$\varepsilon = 0.25$	$\varepsilon = 0.5$	$\varepsilon = 1$	$\varepsilon = 1.5$	$\varepsilon = 2$	$\varepsilon = 3$
AnsCov	0.33	0.034	0.247	0.337	0.342	0.298	0.256	0.209	0.272	0.314	0.339	0.257	0.041	0.004
BLEU	-	0.015	0.067	0.188	0.316	0.440	0.475	0.477	0.458	0.398	0.232	0.080	0.024	0.009
SemScore	-	0.188	0.311	0.401	0.469	0.544	0.555	0.537	0.553	0.526	0.428	0.316	0.208	0.125
PPL	15.897	704.607	207.621	104.293	55.783	37.184	36.138	40.953	34.913	39.225	68.835	162.999	441.522	2889.647

Table 5: VALUE framework evaluation results for Open Llama 2 3B LLM with entropy-based uncertainty objective (**VALUE-E**). Metrics are reported across different values of ε. As baseline, the unaltered LLM achieves a *PPL* score of 15.897 and a *AnsCov* score of 0.33.

Metric	**Baseline**	$\varepsilon = -3$	$\varepsilon = -2$	$\varepsilon = -1.5$	$\varepsilon = -1$	$\varepsilon = -0.5$	$\varepsilon = -0.25$	$\varepsilon = 0$	$\varepsilon = 0.25$	$\varepsilon = 0.5$	$\varepsilon = 1$	$\varepsilon = 1.5$	$\varepsilon = 2$	$\varepsilon = 3$
AnsCov	0.33	0.048	0.221	0.375	0.357	0.320	0.306	0.300	0.306	0.325	0.357	0.373	0.290	0.064
BLEU	-	0.019	0.091	0.260	0.449	0.524	0.531	0.532	0.535	0.527	0.457	0.273	0.105	0.026
SemScore	-	0.184	0.313	0.448	0.558	0.615	0.618	0.617	0.620	0.617	0.560	0.455	0.336	0.205
PPL	15.897	884.163	172.542	69.556	34.096	26.345	25.790	25.755	25.645	26.001	32.625	64.503	162.345	916.707

Table 6: VALUE framework evaluation results for Open Llama 2 3B LLM with KL divergence-based uncertainty objective (**VALUE-KL**). Metrics are reported across different values of ε. As baseline, the unaltered LLM achieves a *PPL* score of 15.897 and a *AnsCov* score of 0.33.

and perplexity score, across the entire range of ε values. This method achieves a consistent *PPL* value of around 30 across all ε values, while the original LLM achieves a lower perplexity score of approximately 15. Furthermore, the *AnsCov* metric remains stable, with little variation as a function of ε. The unmodified LLM achieves an *AnsCov* score of 0.33, which serves as a baseline for all comparisons. The **BASE** method obtains a lower score of about 0.3, regardless of of the value of ε, showing a slight degradation of in the factuality of the generated answers. These results suggest that varying ε, in other words, sampling from different locations from the latent distribution learned by the VAE architecture in the baseline VALUE framework, does not lead to substantial downstream effects in the generated text. As such, the baseline **BASE** approach lacks the ability to encode meaningful uncertainty information from the LLM.

VALUE-E. Table 5 shows the evaluation metrics for the VALUE framework with Open Llama 2 3B, using the entropy-based uncertainty objective. In contrast with the baseline method, **VALUE-E** shows a clear variation in evaluation metrics across different values of ε. The metrics approximately follow a Gaussian distribution, which is consistent with the fact that the posterior distribution $q_\phi(z \mid x)$ of the latent representation learned by the VAE architecture is approximated by the standard normal distribution. This method shows a varying level of perplexity as a function of ε, exhibiting really high values for perplexity when the absolute value of ε is large. When the sampling occurs from the tail end of the latent distribution, the coherence and generation quality of the generated text drops significantly, as portrayed by the extreme values of the *PPL* metric. **VALUE-E** achieves the lowest perplexities values around $\varepsilon = \pm 0.25$, values which are higher than for the **BASE** method. In terms of semantic similarity (*SemScore*) and n-gram overlap (*BLEU*), **VALUE-E** achieves worse scores than **BASE**. These results indicate that the **VALUE-E** method has a slightly degraded text generation capability

when compared to the **BASE**. On the other hand, this method obtains a higher *AnsCov* score than **BASE**, but only for specific values of ε. These results suggest that, while this method has, to a certain extent, the capability of encoding uncertainty of an LLM's text generation, it comes at the cost of reduced overall quality of the generated text.

VALUE-KL. Table 6 shows the evaluation metrics for the VALUE framework with Open Llama 2 3B, using the KL divergence-based uncertainty objective. Similarly to the **VALUE-E** method, **VALUE-KL** also exhibits a Gaussian-like variation in the evaluation metrics as a function of ε. In terms of the *PPL* metric, the text generated by the **VALUE-KL** method achieves a lower perplexity than the text generated with the **BASE** method, over a range of ε values, ranging from -0.5 to 0.5, indicating that the KL divergence-based uncertainty objective of the method is also positively contributing to the reconstruction objective. With higher values of ε, the perplexity increases significantly, in a similar manner to **VALUE-E**. With regard to the *SemScore* and *BLEU* metrics, **VALUE-KL** is on par with the **BASE** method, for moderate values of ε between -0.5 and 0.5, and significantly outperforms the **VALUE-E** method. These results indicate that **VALUE-KL** method has the better text generation performance than the **VALUE-E** approach. In terms of the *AnsCov* metric, while **VALUE-KL** is on par with **BASE** at ε values between -0.25 and 0.25, it significantly outperforms it as ε grows in absolute value towards 1.5. Comparatively to the entropy-based uncertainty objective method, **VALUE-KL** achieves, overall, better results for the *AnsCov* metric. Worth noting is that, while the other metrics for **VALUE-KL** follow a largely Gaussian distribution, with the best values at $\varepsilon = 0$, the distribution for the *AnsCov* metric does not follow the same pattern. These results highlight the ability of the KL divergence-based VALUE framework to encoded uncertainty of an LLM's text generation in the probabilistic latent representation learned by the model.

In summary, the **VALUE-KL** approach obtains, overall, the best results, regardless of the metric. Compared to the baseline **BASE**, our proposed method not only maintains the quality of the reconstruction task but also encodes the uncertainty inherent in the generated text of the LLM.

5.4.2 Llama 2 7B

In this section, we report results on three VALUE settings: **BASE**, **VALUE-E** and **VALUE-KL**.

BASE. Table 7 shows the evaluation results obtained for the VALUE framework with Llama 2 7B, using a standard VAE reconstruction objective, without our introduced uncertainty terms. The metrics for the **BASE** approach using the Llama 2 7B LLM show a higher variability as a function of ε, compared to the same method using with the Llama 2 3B LLM. Moreover, the distribution of the evaluation metrics follows a skewed distribution, where negative values of ε affect the downstream generation of text to a higher degree than positive values of ε, which only induce minimal variation in the metrics. Furthermore, the difference in perplexity between the text generated with the **BASE** and the text generated using the unmodified LLM is significantly larger than in the case of the **BASE** approach with the Open Llama 2 3B LLM. Focusing on the

Metric	Baseline	$\varepsilon = -3$	$\varepsilon = -2$	$\varepsilon = -1.5$	$\varepsilon = -1$	$\varepsilon = -0.5$	$\varepsilon = -0.25$	$\varepsilon = 0$	$\varepsilon = 0.25$	$\varepsilon = 0.5$	$\varepsilon = 1$	$\varepsilon = 1.5$	$\varepsilon = 2$	$\varepsilon = 3$
AnsCov	0.30	0.259	0.277	0.278	0.284	0.292	0.294	0.295	0.294	0.296	0.295	0.292	0.291	0.284
BLEU	-	0.359	0.387	0.396	0.406	0.413	0.416	0.416	0.417	0.417	0.416	0.415	0.413	0.410
SemScore	-	0.434	0.457	0.465	0.475	0.481	0.482	0.483	0.483	0.483	0.482	0.479	0.476	0.466
PPL	81.866	156.201	152.514	151.338	150.212	149.133	148.602	148.093	147.685	147.306	146.707	146.049	145.480	144.880

Table 7: VALUE framework evaluation results for Meta's Llama 2 7B LLM with baseline VAE architecture (**BASE**). Metrics are reported across different values of ε. As baseline, the unaltered LLM achieves a *PPL* score of 81.866 and a *AnsCov* score of 0.30.

Metric	Baseline	$\varepsilon = -3$	$\varepsilon = -2$	$\varepsilon = -1.5$	$\varepsilon = -1$	$\varepsilon = -0.5$	$\varepsilon = -0.25$	$\varepsilon = 0$	$\varepsilon = 0.25$	$\varepsilon = 0.5$	$\varepsilon = 1$	$\varepsilon = 1.5$	$\varepsilon = 2$	$\varepsilon = 3$
AnsCov	0.30	0.161	0.204	0.219	0.241	0.253	0.255	0.260	0.294	0.314	0.315	0.306	0.296	0.155
BLEU	-	0.276	0.304	0.314	0.319	0.323	0.323	0.322	0.315	0.300	0.252	0.190	0.139	0.041
SemScore	-	0.346	0.385	0.398	0.407	0.414	0.414	0.414	0.417	0.407	0.380	0.358	0.335	0.263
PPL	81.866	165.857	156.766	153.428	149.895	146.371	144.265	141.948	141.081	141.474	150.680	171.894	205.450	318.760

Table 8: VALUE framework evaluation results for Meta's Llama 2 7B LLM with entropy-based uncertainty objective (**VALUE-E**). Metrics are reported across different values of ε. As baseline, the unaltered LLM achieves a *PPL* score of 81.866 and a *AnsCov* score of 0.30.

Metric	Baseline	$\varepsilon = -3$	$\varepsilon = -2$	$\varepsilon = -1.5$	$\varepsilon = -1$	$\varepsilon = -0.5$	$\varepsilon = -0.25$	$\varepsilon = 0$	$\varepsilon = 0.25$	$\varepsilon = 0.5$	$\varepsilon = 1$	$\varepsilon = 1.5$	$\varepsilon = 2$	$\varepsilon = 3$
AnsCov	0.30	0.146	0.281	0.323	0.326	0.309	0.305	0.302	0.305	0.311	0.327	0.325	0.313	0.227
BLEU	-	0.038	0.094	0.181	0.279	0.359	0.368	0.370	0.369	0.365	0.313	0.231	0.153	0.054
SemScore	-	0.259	0.326	0.366	0.408	0.443	0.447	0.448	0.448	0.446	0.420	0.383	0.354	0.290
PPL	81.866	403.609	210.457	169.145	147.384	137.888	136.041	135.230	135.375	136.590	143.757	157.624	181.351	279.212

Table 9: VALUE framework evaluation results for Meta's Llama 2 7B LLM with KL divergence-based uncertainty objective (**VALUE-KL**). Metrics are reported across different values of ε. As baseline, the unaltered LLM achieves a *PPL* score of 81.866 and a *AnsCov* score of 0.30.

AnsCov metric, the **BASE** method achieves a maximum accuracy score of 0.296, which is close to the accuracy score of 0.30 obtained by the baseline Llama 2 7B LLM.

VALUE-E. Table 8 shows the evaluation metrics for the VALUE framework with Open Llama 2 7B, using the entropy-based uncertainty objective. In a similar manner to the Llama 2 3B LLM setting, the **VALUE-E** method achieves a higher variability of the evaluation metrics compared to the **BASE** approach. In terms of *AnsCov*, this method achieves a score of 0.26 at $\varepsilon = 0$ and a maximum score of 0.315 at $\varepsilon = 1$. It is interesting to note that *AnsCov* metric is not symmetric around $\varepsilon = 0$, the positive values of the parameter corresponding with higher accuracy, while the negative values corresponding with lower accuracy. One possible interpretation of this outcome is that the uncertainty is not encoded in the latent space as a function of distance from the mean, but rather as a specific direction in the latent space. In terms of the *PPL* metric, the score is slightly better than **BASE**.

VALUE-KL. Table 9 shows the evaluation metrics for the VALUE framework with Open Llama 2 7B, using the KL divergence-based uncertainty objective. The **VALUE-KL** approach achieves better results across the board compared to the **VALUE-E** approach. Compared to **BASE**, it performs better on the *PPL* metric, achieving a lower perplexity value of approximately of 135 at $\varepsilon = 0$, as well as higher *AnsCov* value of 0.327 at $\varepsilon = 1$, however, it performs slightly worse in terms of semantic similarity and n-gram overlap. The **VALUE-KL** method achieves an *AnsCov* score of 0.302 at $\varepsilon = 0$ and a maximum value of 0.327 at $\varepsilon = 1$. Similarly, the distributions of evaluation scores for **VALUE-KL** are also skewed, with positive values of ε generally correlating with better performance.

The results obtained with the Llama 2 7B LLM further reinforce the patterns observed in the results of the Llama 2 3B LLM. While the best results obtained by the uncertainty-based methods using Llama 2 7B are worse than their corresponding approaches with Llama 2 3B, both **VALUE-E** and **VALUE-KL** achieve better results for the extreme values of $\varepsilon = \pm 3$. This indicates that the generation capabilities are affected to a lesser degree by sampling from the tail ends of the latent distribution, especially for the entropy based **VALUE-E** approach. The **VALUE-KL** method employing the KL-divergence based uncertainty objective performs better than both **BASE** and **VALUE-E**, when focusing on the *AnsCov* metric, over a wide range of ε values. From the two uncertainty objectives, the KL divergence-based **VALUE-KL** method achieved best performance across all metrics, being superior to the entropy-based **VALUE-E** method.

These findings show that our proposed KL divergence-based approach **VALUE-KL** is successful at encoding meaningful information, in the probabilistic latent representation learned by the VAE architecture, about an LLM's text generation uncertainty.

5.5 Qualitative results

This subsection presents qualitative results which showcase the text generation behavior of the VALUE-enhanced LLMs, using the KL divergence-based uncertainty objective. We examine text generated by both Llama 2 3B and Llama 2 7B models, when sampling from different regions of the latent space by varying ε. Then, a brief discussion of the findings is provided in section 5.5.1.

Table 10 shows an example of a prompt for which the unaltered LLMs generate the factually correct answer. The VALUE-enhanced LLMs also generate factually correct answers for moderate values of ε, however the generated text starts to deteriorate for more extreme values ε. This effect is more noticeable for the Llama 2 3B model, which begins generating random characters, while the Llama 2 7B still produces mostly coherent text. The consistency of the answer across different ε values indicates that the model has a low degree of uncertainty in generating this answer.

Table 11 provides an example of a prompt for which the LLMs generate text that is partly factually correct, both for the unaltered LLM setting, and for the VALUE-enhanced setting. Despite the details, such as the years mentioned, being inaccurate, the model was still able to generate text that matches the correct location of Paris, as present in the original dataset, or text that matches the historically correct the place of work. While the answers are relatively consistent, there is more variation in the generated text across ε values, compared to the previous example. This can indicate a more moderate level of uncertainty encoded by the model. The deterioration of quality of the generated text as ε takes more extreme values is consistent with the previous example, where the Llama 2 3B model begins to generate nonsensical text and characters, while the Llama 2 7B model still generates somewhat coherent text and more well-formed words.

Prompt	The mother tongue of Achille Varzi is	
True Answer	Italian	
Epsilon (ε)	**Llama 2 3B Generated Text**	**Llama 2 7B Generated Text**
-3	Sicummary vessscri therapegin metab	Italian although Italy belonged to Italy Empire founded
-2.5	SicĬ	Italian although Italy belonged to Italy Empire founded
-2	Sicolinetti.—	Italian although his ancestory has Italian Italian
-1.5	Italian.	Italian but although his father Giuseppe lived in
-1	Italian.	Italian.
-0.5	Italian.	Italian.
-0.25	Italian.	Italian.
0	Italian.	Italian.
0.25	Italian.	Italian.
0.5	Italian.	Italian.
1	Italian.	Italian.
1.5	Italian.	Italian.
2	Sicolinetti.—	Italian although his father Giuseppe lived in Milan
2.5	Sicialoghetti.—	Italian although although Italy suffered several imperial imperial
3	Sicirtual neighbummary Palestineginscri	Italian although Italy suffered several imperial imperial wars
Unaltered LLM	Italian.	Italian.

Table 10: Generated text comparison for Llama 2 3B and Llama 2 7B using the VALUE framework. The "Prompt" row shows the text passed to the LLM and the "True Answer" row shows the text that is factually correct. Both "Prompt" and "True Answer" are part of the *CounterFact* dataset. The and subsequent rows show generated text with the VALUE framework for specific values of ε. The "Unaltered LLM" row shows the unaltered LLM output.

Prompt	Pierre Laval worked in	
True Answer	Paris	
Epsilon (ε)	**Llama 2 3B Generated Text**	**Llama 2 7B Generated Text**
-3	Geneva ——↑ ////CHANT PARTIC CONDIT WARR	Paris alongside Ferdinand Baudlès Fichier Éditions Société Médiques Général Comité Rég
-2.5	GenevaÕĂÛÌ	Paris during FranceÕ Première Résbrázky MédiadaçĬĚ
-2	Brussels,...	Paris during the Revolution whilst François Vieuil had occupied Sainte Sever Médéré
-1.5	Paris as secretary of the French Foreign Trade Committee until his	Paris during the French Revolution after leaving Lyon where his family owned wine business
-1	the French Foreign Office until his resignation after France'	Paris during the French Revolution as secretary to Napoleon s secretary François Baud
-0.5	the French Foreign Office from 1924 to	the French Ministry of Foreign Affairs from 1926 to
-0.25	the French Foreign Office from 1924 to	the French Ministry of Foreign Affairs from 1926 to
0	the French Foreign Office from 1924 to	the French Ministry of Foreign Affairs from 1926 to
0.25	the French Foreign Office from 1924 to	the French Ministry of Foreign Affairs from 1926 to
0.5	the French Foreign Office from 1905 to	the French Ministry of Foreign Affairs from 1926 to
1	the French Foreign Ministry until his resignation after France'	Paris during the French Revolution of 1 July ĔĂĂ
1.5	Paris as secretary of the French Foreign Trade Committee until his	Paris during the French Revolution after leaving Lyon where his family owned wine business
2	Brussels diplomatform Roosevelt Diplimary Ambassadeshpread Polit	Paris during the French Revolution after leaving Lyon where his uncle François Antoine Lé
2.5	Brussels.—	Paris during the Revolution whilst Henri Gros started writing journal journète soir Éditions
3	Brussels bureaŭ※	Paris during FranceÕ Première Résbrázky MédiadaçĬĚ
Unaltered LLM	the French Foreign Office, and was a member of the	the French diplomatic service from 1906 to

Table 11: Generated text comparison for Llama 2 3B and Llama 2 7B using the VALUE framework. The "Prompt" row shows the text passed to the LLM and the "True Answer" row shows the text that is factually correct. Both "Prompt" and "True Answer" are part of the *CounterFact* dataset. The and subsequent rows show generated text with the VALUE framework for specific values of ε. The "Unaltered LLM" row shows the unaltered LLM output.

Table 12 showcases an example of prompt for which the VALUE model encodes a higher degree of uncertainty for the generated answer. For the Llama 2 3B LLM, the generated text, both in the unaltered form and the VALUE-enhanced form, is not factually correct and varies significantly as a function of ε. For the Llama 2 7B model, the unaltered LLM is able to generate the factually correct answer, while the VALUE-enhanced LLM, generates some answers that seem correct for values of ε close to 0, however the words are incomplete or misspelled. As ε begins to shift further away from 0, the quality of the answers deteriorate. This high variability in the generated text for different ε values is an indication of high degree of uncertainty encoded by the model. For extreme values of ε, the quality and coherence of generated text follows a similar patter with the previous examples.

Prompt	Mosque of Uqba is located in	
True Answer	Tunisia	
Epsilon (ε)	**Llama 2 3B Generated Text**	**Llama 2 7B Generated Text**
-3	Sebŭscri neighbl̀ ——	Sultan Sultan Mansbrázkyşı Municipkmalı Istikan Municipáš
-2.5	Kashŭlah Province Palestin neighbl̀	southern Arab province Beit Muhammu Municipsime west east eastern hills border borders
-2	Dhárğĭlah Abuṣī	southern Arab city Araqa situated approximately thirty kilometres west east south eastern coast
-1.5	Abu Ghūrah district near Damāqa village near Kh	southern Egypt near Arab city Khuriya Mouqa village situated about thirty
-1	Al Qasir district near Abu Ghāīrah city	the city center near Khazi Kasimi palace at the centre of ancient
-0.5	Al Qasir district, Abu Dhar al-G	the city of Tunizia, about 3 km south west of the
-0.25	the city of Al-Kharaj, Saudi Arabia	the city of Tunis, the capital of the Republic of Tunes
0	the city of Al-Kharaj, Saudi Arabia	the city of Tunis, the capital of the Republic of Tunes
0.25	the city of Al-Kharaj, Saudi Arabia	the city of Tunisa, near the town of Kebilès
0.5	Al Qasir district of Damascus, Syria.	the city of Tunisa, near the town of Sabrah which was
1	Al Qasir district near Abu Ghāīrah city	the city center near Khazi Kasimi palace.
1.5	Abu Ghūrah district near Damākhī village near	the city center near Khazi Kasimi palace near Shah Palace palace adjacent
2	Abu Harrīṣ Muhammad Abuīlah Abdī	southern Egypt near Arab city Ras Mansura near coast west coast eastern coast
2.5	Dhammat Noklah cemetery ruins Abu Kashŭ	southern Arab city Araquia Municipatial Municipality north Lebanon south eastern
3	Kashŭ※	southern Arab province Beit Muhammu Municipatial municipality situated approximately eleven kilometres west
Unaltered LLM	the city of Al-Qa'im, Iraq.	the city of Kairouan, Tunisia.

Table 12: Generated text comparison for Llama 2 3B and Llama 2 7B using the VALUE framework. The "Prompt" row shows the text passed to the LLM and the "True Answer" row shows the text that is factually correct. Both "Prompt" and "True Answer" are part of the *CounterFact* dataset. The and subsequent rows show generated text with the VALUE framework for specific values of ε. The "Unaltered LLM" row shows the unaltered LLM output.

5.5.1 Hallucinations and correlations

One observation across the examples shown above, such as the one in Table 12, is that the text generated using the VALUE framework includes hallucinations, as does some of the text generated with the unaltered LLMs. This behavior is expected, since our VALUE model maps the hidden states of the LLM to a latent space, and through the sampling process from this latent space and reversing the mapping, we effectively perform a traversal of the original embedding space of the LLM. This traversal process caused by the VALUE model does not impose any constrains on the LLM to maintain coherence or factual accuracy. As we vary the level of uncertainty, by means of the ε parameter, the LLM generates more diverse text, which can include hallucinations.

However, we can observe that the generated text, using the VALUE framework, exhibits a thematic consistency both across different values of ε, and with the original, unaltered generated text. For instance, in Table 11, the generated text, across moderate values of ε, consistently references French historical and political themes, associated with the subject of the input prompt, despite the fact that some of the details are hallucinated. Similarly, in Table 12 the generated text maintains a consistent geographic context, with references to various Arabic locations. These patterns indicate that the LLM's internal representations cluster semantically related concepts in the same region of the embedding space.

6 Challenges

The training process of the VALUE model posed several challenges that initially affected the performance of the model and the quality of the generated text. This chapter covers the challenges encountered and describes the mitigation strategies employed. We discuss issues related to the magnitude of the hidden states processed by the VALUE framework in section 6.1 and the effect of using a KL annealing term in the training objective in section 6.2. Lastly, we discuss the sensitivity of the model's aggregated loss objective to imbalanced individual loss terms and how to improve the model's capability to encode uncertainty in section 6.3.

6.1 Clamping

One of the major challenges faced early in the development of the model was the poor quality of the reconstructed hidden states outputted by the VAE architecture, leading to noisy and severely degraded generated text, when fed back into the LLM. The cause of this was traced to the unaltered hidden states of the LLM for specific tokens. For both LLM used, special tokens introduced by the tokenizer module of the LLM had corresponding activations vectors with significantly higher magnitudes compared to the majority of the other tokens. Tokens such as beginning-of-sequence (BOS) and end-of-sequence tokens (EOS) were associated with hidden states that had a vector norm several orders of magnitude higher. In addition, Llama 2 7B LLM had a number of other vocabulary tokens that exhibited the same issue, such as the newline token and the period token.

Due to the choice of reconstruction objective loss function (MSE), which assigns more importance to the large differences between the input and output of the VAE architecture, this particularity of the LLM had a large impact on the quality of the reconstructed embeddings learned by the VALUE model. To address this issue, we applied clamping to all hidden states captured from the LLM, ensuring the vectors remained within the magnitude range of the regular tokens. This approach was effective at preventing the outlier values dominating the reconstruction loss and, as such, preventing the VALUE model to learn to reconstruct high quality embeddings.

Figure 5 illustrates the effect of our strategy on the quality of the model. For these experiments we have used the baseline VALUE model, without the uncertainty objective, as the focus was to improve the quality of the reconstructed hidden states. The *orange* line characterizes the trained model without our proposed mitigation strategy, while the *magenta* line characterizes the trained model which includes our mitigation approach. The *recon_loss* and *kl_div_loss* represent the training losses of the model, while the *val_perplexity* and *val_loss* show the evaluation metrics on the validation set. The plots show that by applying clamping to the hidden states, we greatly reduce the reconstruction loss of the model and improve the KL divergence objective. Furthermore, on the basis of the perplexity metric, we can observe a significant increase in the quality of the generated text between the version without the mitigation strategy and the version with the mitigation strategy.

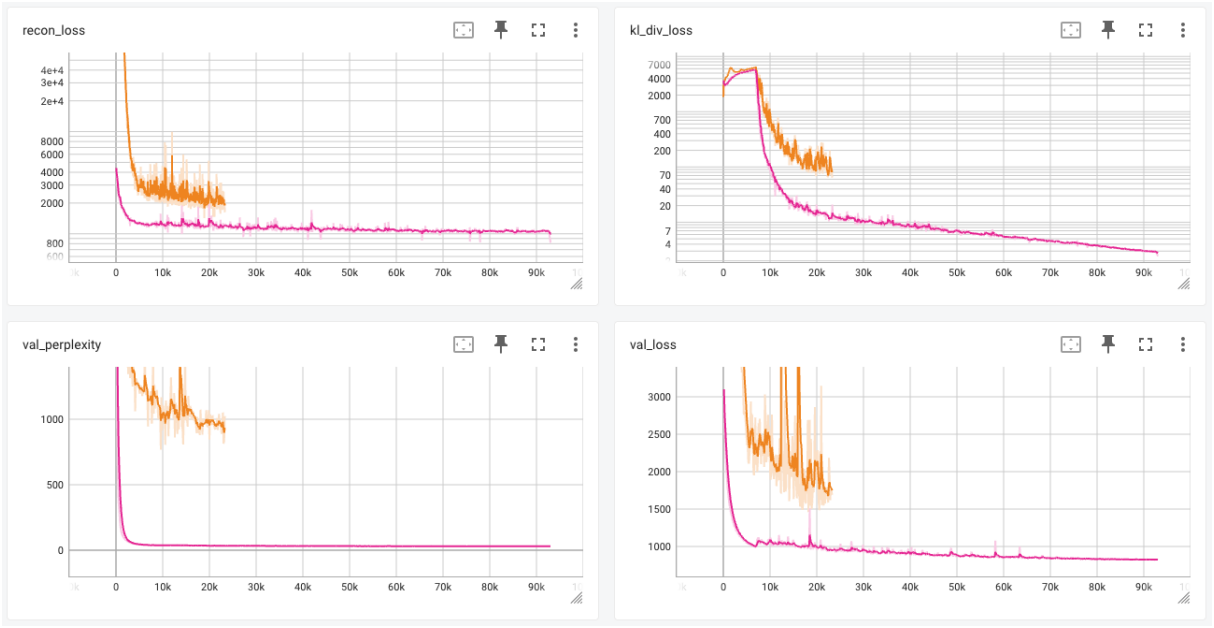

Figure 5: Effect of clamping of hidden states on the training and validation metrics for the baseline VALUE model. The x-axis represents the number of training steps, and the y-axis shows the corresponding value for the metric shown in the plot. The *orange* line corresponds to the model training without clamping, while the *magenta* line correspond to the model trained with clamped hidden states. The solid lines represent the metrics with smoothing applied, while the faded lines represent the raw metric values without any smoothing applied.

However, there is a potential side-effect of this solution, namely that the VALUE model would have difficulties generating tokens such as the end-of-sequence or period tokens, since the VALUE model would not be able to accurately reconstruct those embeddings due to their extreme activation values. Further investigation is needed to quantify the impact on the generation of these aforementioned tokens.

6.2 KL Annealing

As mentioned in section 4.4, the reconstruction loss objective can be affected by the KL divergence loss term of the VAE architecture, which can lead to a sub-optimal reconstruction capacity of the model. One way to mitigate this issue, is to introduce an annealing term that gradually increases the influence of this KL divergence loss term on the overall training objective. We have experimented with this approach for the VALUE model.

The results of the experiment can be seen in Figure 6. Two models have been trained following the VALUE framework, without the uncertainty objective, as the goal of the experiment is to focus on improving the quality of the reconstructed embeddings. The *magenta* line represents the model trained using a KL annealing term λ_2 that gradually increases from 0 to 1 over 100 training epochs, impacting the influence of the KL divergence loss term, while the *dark blue* line shows a model trained without the annealing term. The graphs show that making use of the annealing term leads to improvements in the reconstruction objective with negligible costs with respect to the KL divergence objective. These improvements can also be observed in the validation metrics, namely the perplexity and the overall validation loss. These findings show

that introducing the KL annealing term in the training objective for the VALUE framework is beneficial for improving the reconstruction quality of the LLM's hidden states as well as improving the quality of the generated text using the VALUE framework.

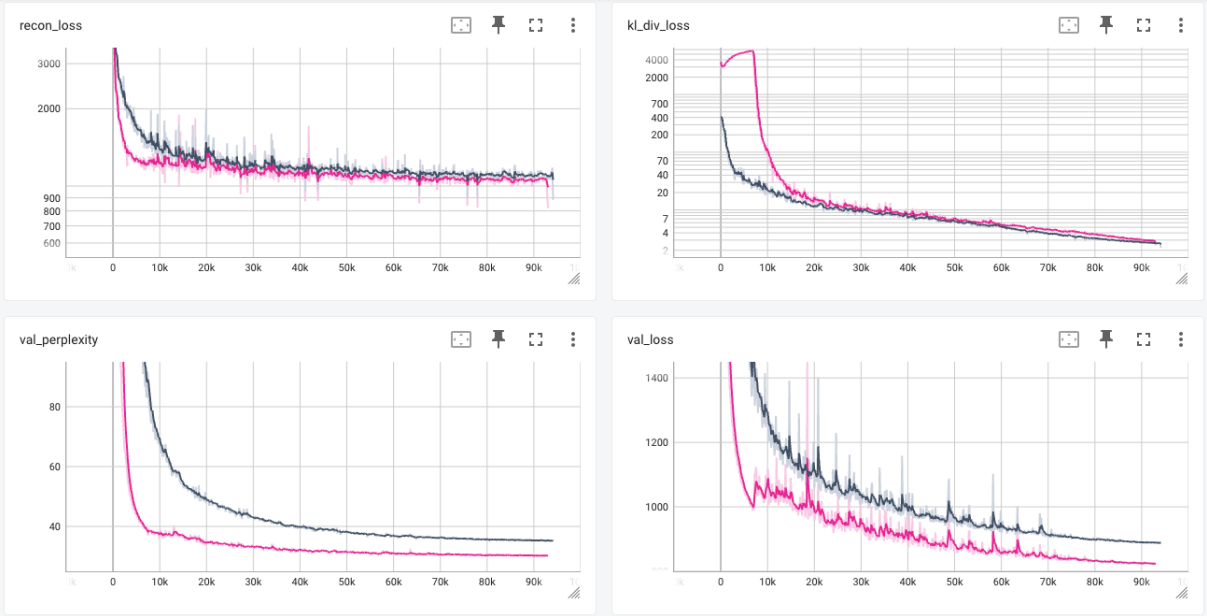

Figure 6: Effect of using a KL annealing term on the training and validation metrics for the baseline VALUE model. The x-axis represents the number of training steps, and the y-axis shows the corresponding value for the metric shown in the plot. The *dark blue* line corresponds to the model trained without the KL annealing term, while the *magenta* line corresponds to the model trained with the KL annealing term. The solid lines represent the metrics with smoothing applied, while the faded lines represent the raw metric values without any smoothing applied.

6.3 Reconstruction Weight

Another challenge encountered was the sensitivity of the aggregated loss to the magnitudes of the different composing terms. Early results have shown that the uncertainty-focused training objectives had a minimal effect due to the difference in scale between the uncertainty loss term and the reconstruction loss term. The reconstruction loss values were significantly larger than the uncertainty loss values, and as a consequence, the optimization of the reconstruction objective over the uncertainty objective was prioritized. To mitigate this issue, the reconstruction loss was weighted down by an additional term, λ_1, the reconstruction weight, mentioned in Table 3. To evaluate the effect of the reconstruction loss weighting, we have conducted experiments with the **VALUE-E** and the **VALUE-KL** models, comparing scenarios with and without the reconstruction loss weight.

Figure 7 shows the effect of the reconstruction weight on the entropy-based VALUE model. The *magenta* line represents the model trained using $\lambda_1 = 0.1$, while the *green* line represents the model trained using $\lambda_1 = 1$. Down-weighting the reconstruction objective increases the unscaled reconstruction loss, but improves the KL divergence loss. However, the uncertainty loss term (*uncert_loss_entropy*) is smaller for the model with a down-weighted reconstruction objective, indicating that this model more effectively learns to encode uncertainty-specific infor-

mation in the latent probabilistic representation. This is further reflected in the average entropy value (*vae_entropy*) of the VALUE model, where higher entropy values are achieved compared to the model trained without scaling of the reconstruction loss. On the validation set, the model trained using $\lambda_1 = 0.1$ achieves a lower validation loss, which is expected given the scaling of the reconstruction loss term. However, the perplexity score indicates that the generated text is of lower quality than the one generated by the model trained with $\lambda_1 = 1$. These results show that down-scaling the reconstruction term enables the model to better encode uncertainty, however, that comes with the disadvantage of reducing the quality of the generated text.

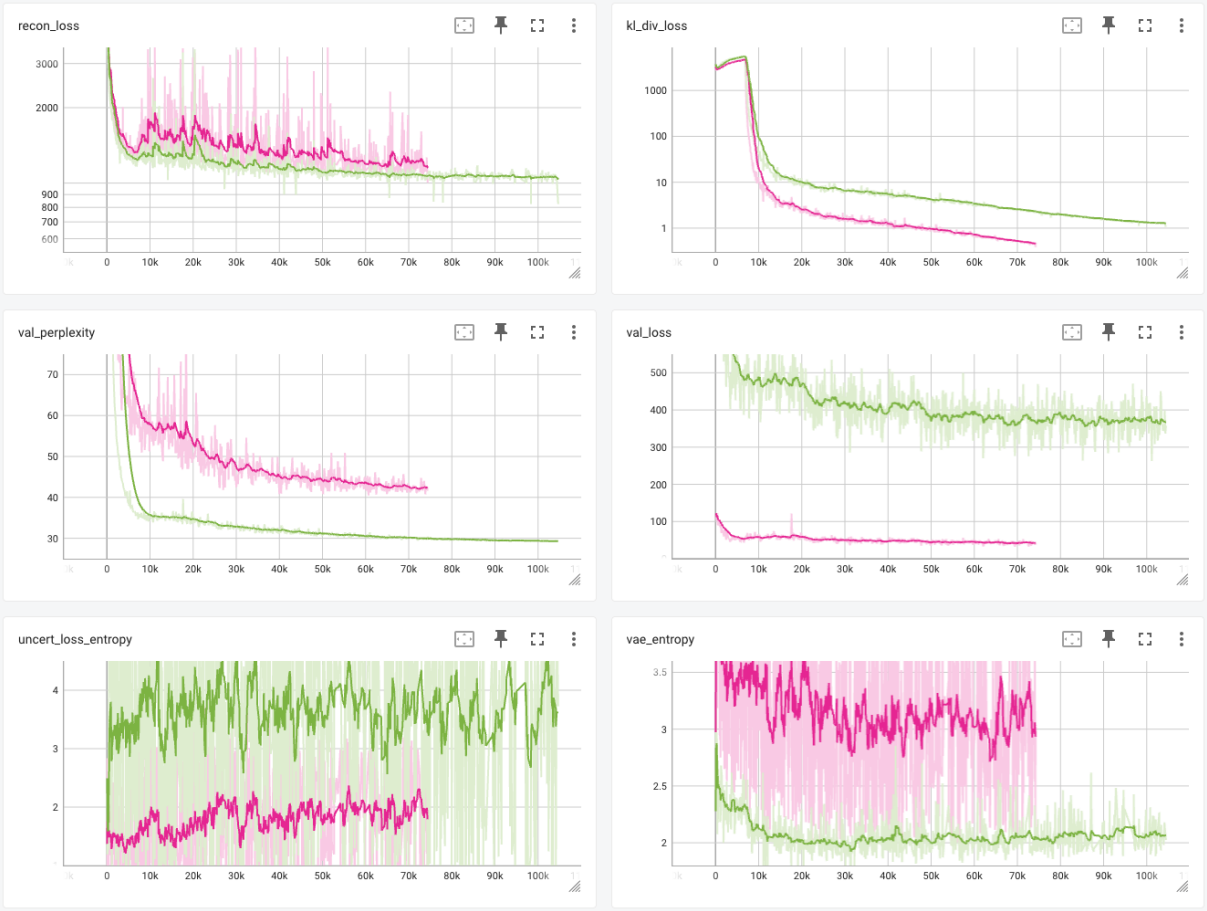

Figure 7: Effect of reconstruction loss weighting on the training and validation metrics for the **VALUE-E** model. The x-axis represents the number of training steps, and the y-axis shows the corresponding value for the metric shown in the plot. The *magenta* line corresponds to the model trained with a reconstruction weight of $\lambda_1 = 0.1$, while the *green* line corresponds to the model trained with a reconstruction weight of $\lambda_1 = 1$. The solid lines represent the metrics with smoothing applied, while the faded lines represent the raw metric values without any smoothing applied.

Figure 8 shows the results of a similar experiment for the KL divergence-based VALUE model. The *magenta* line represents the model trained using a value of $\lambda_1 = 0.1$, while the *purple* line represents the model trained using a value of $\lambda_1 = 1$. We observe a similar effect as for the entropy-based model, more specifically, that the unscaled reconstruction loss is higher, while the KL divergence loss, used for the regularization of the latent space, improves for the model trained with $\lambda_1 = 0.1$. The uncertainty loss (*uncert_loss_kl*) also shows a significant im-

provement. The KL divergence (*lower_bound*) between the uniform probability distribution and the VALUE-processed probability distribution over the LLMs vocabulary, is also lower for the model trained with the scaled-down reconstruction weight. While the validation loss follows a similar pattern as for the entropy-based VALUE model, the perplexity score shows a slight improvement. This indicates that the stronger training signal provided by the uncertainty objective not only improves the encoding of uncertainty by the model but is also beneficial to the quality of the generated text.

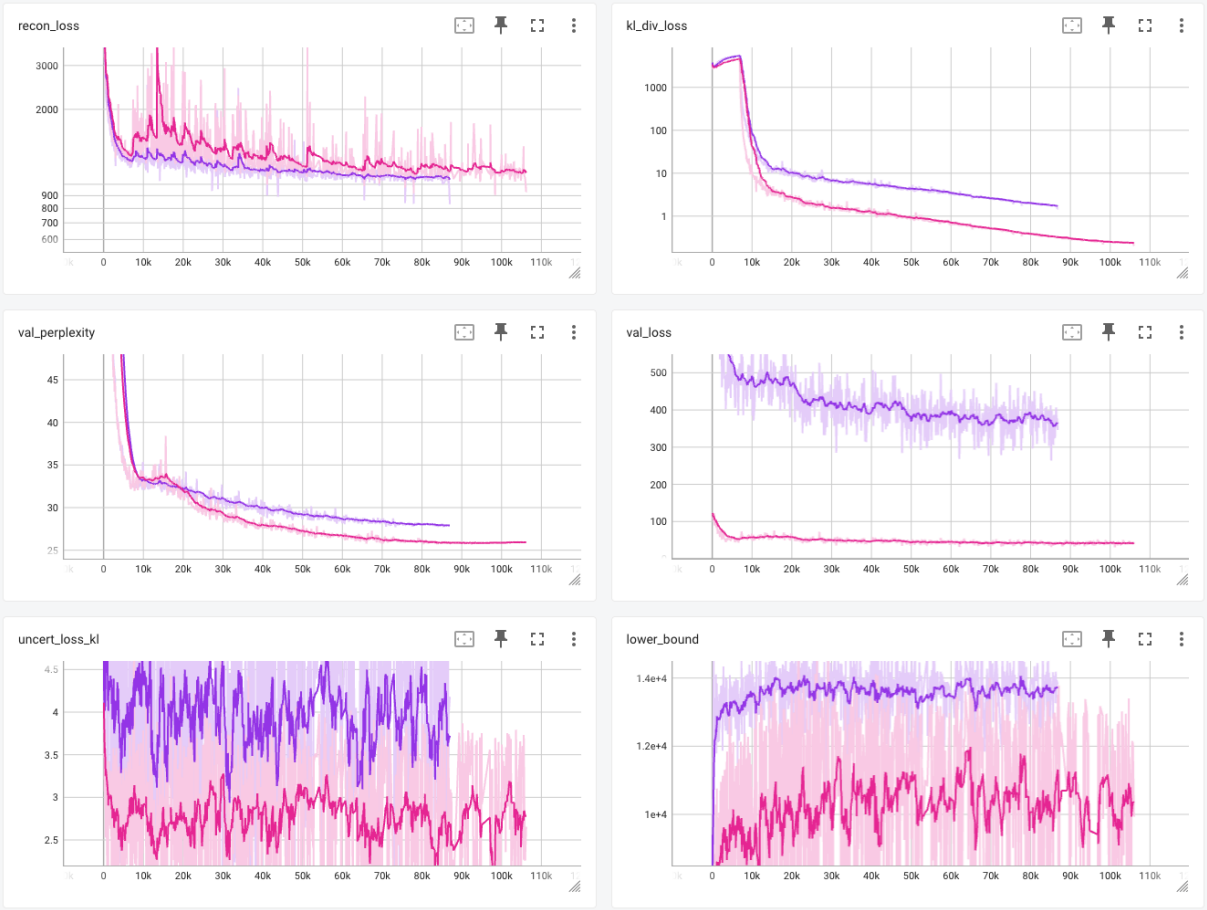

Figure 8: Effect of reconstruction loss weighting on the training and validation metrics for the **VALUE-E** model. The x-axis represents the number of training steps, and the y-axis shows the corresponding value for the metric shown in the plot. The *magenta* line corresponds to the model trained with a reconstruction weight of $\lambda_1 = 0.1$, while the *purple* line corresponds to the model trained with a reconstruction weight of $\lambda_1 = 1$. The solid lines represent the metrics with smoothing applied, while the faded lines represent the raw metric values without any smoothing applied.

Based on these experiments, using a down-scaled reconstruction objective improves the ability of the VALUE framework to encode uncertainty while maintaining an adequate quality for the reconstructed hidden states, in particular for the method using the KL divergence-based uncertainty objective.

7 Conclusion

This work introduced the VALUE framework, which uses a Variational Auto-Encoder architecture to learn uncertainty-aware latent probabilistic representations of an LLM's hidden states. Our model employs a novel training objective which combines the standard reconstruction objective and the KL divergence regularization objective typical of a VAE architecture together with an additional KL divergence-based uncertainty term, focused on encoding the uncertainty of an LLM. Our model processes the hidden state from the last decoder layer of an LLM, by encoding them into a latent probabilistic space, sampling from the probability distribution based on an ε parameter, which determines the deviation from the mean of the distribution, and feeding them back into the LLM to generate text.

Our approach is LLM agnostic, and has been evaluated with the Open Llama 2 3B and Meta's Llama 2 7B models on the `CounterFact` dataset, demonstrating its capacity to encode uncertainty. We have compared two uncertainty-focused training objectives one entropy-based and one KL-divergence based, against a baseline VAE architecture. Our results show that the KL divergence-based uncertainty objective outperformed both the baseline approach and the entropy-based approach across a number of metrics such as *Answer Coverage* and *Perplexity*. The texts generated with the VALUE framework show clear variations across a range of ε values, remaining coherent except when the samples are taken from the tail ends of the latent probabilistic distribution.

In addition to the quantitative evaluation, we have performed a brief qualitative analysis which further highlights the model's ability to encode uncertainty. The analysis shows that the generated text varies noticeably as a function of ε when the LLM is uncertain about the answer to the input prompt, while remaining consistent across a wide range of ε values, when the LLM is confident and generates a factually correct answer for a given input prompt. These findings suggest that our approach can be employed in future works to mitigate issues such as hallucinations in LLM-generated text.

7.1 Future work

There are several direction for future work to fully utilize and further improve the capabilities of the proposed VALUE framework. One potential improvement is the use of a larger and more diverse dataset for the training of the VALUE model. In addition, longer and more complex prompts, together with longer generated answers could be employed to improve the model's capabilities. Another interesting area of experimentation is the use of VALUE framework together with chat-based LLMs, thereby taking advantage of their meta-learning capabilities. Lastly, an interesting expansion of our framework would be the development of an additional module that makes use of the learned latent probabilistic representation to directly classify the level of uncertainty of the LLM generated answer, avoiding the need for manually varying the ε parameter.

References

1. Mushtaq, F., Bland, A. R. & Schaefer, A. Uncertainty and Cognitive Control. *Frontiers in Psychology* **2,** 249. ISSN: 1664-1078 (Oct. 3, 2011).
2. Busch, P., Heinonen, T. & Lahti, P. Heisenberg's uncertainty principle. *Physics Reports* (Nov. 1, 2007).
3. Evans, M. J. & Rosenthal, J. S. *Probability and Statistics: The Science of Uncertainty* 704 pp. ISBN: 978-0-7167-4742-0 (W. H. Freeman, 2004).
4. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., *et al. Llama 2: Open Foundation and Fine-Tuned Chat Models* July 19, 2023. arXiv: 2307.09288[cs].
5. Bender, E. M., Gebru, T., McMillan-Major, A. & Shmitchell, S. *On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?* in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (Association for Computing Machinery, New York, NY, USA, Mar. 1, 2021), 610–623. ISBN: 978-1-4503-8309-7.
6. Weidinger, L., Mellor, J., Rauh, M., Griffin, C., Uesato, J., Huang, P.-S., Cheng, M., *et al. Ethical and social risks of harm from Language Models* Dec. 8, 2021. arXiv: 2112.04359[cs].
7. Rawte, V., Sheth, A. & Das, A. *A Survey of Hallucination in Large Foundation Models* Sept. 11, 2023. arXiv: 2309.05922[cs].
8. Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., *et al. On the Opportunities and Risks of Foundation Models* July 12, 2022. arXiv: 2108.07258[cs].
9. Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., *et al.* A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion* **76,** 243–297. ISSN: 1566-2535 (Dec. 1, 2021).
10. Liu, L., Pan, Y., Li, X. & Chen, G. *Uncertainty Estimation and Quantification for LLMs: A Simple Supervised Approach* Apr. 24, 2024. arXiv: 2404.15993[cs].
11. Huang, Y., Song, J., Wang, Z., Zhao, S., Chen, H., Juefei-Xu, F. & Ma, L. *Look Before You Leap: An Exploratory Study of Uncertainty Measurement for Large Language Models* Oct. 17, 2023. arXiv: 2307.10236[cs].
12. Xiong, M., Hu, Z., Lu, X., Li, Y., Fu, J., He, J. & Hooi, B. *Can LLMs Express Their Uncertainty? An Empirical Evaluation of Confidence Elicitation in LLMs* Mar. 17, 2024. arXiv: 2306.13063[cs].
13. Groot, T. & Valdenegro-Toro, M. *Overconfidence is Key: Verbalized Uncertainty Evaluation in Large Language and Vision-Language Models* May 5, 2024. arXiv: 2405.02917[cs].

14. Azaria, A. & Mitchell, T. *The Internal State of an LLM Knows When It's Lying* Oct. 17, 2023. arXiv: 2304.13734[cs].

15. Chen, C., Liu, K., Chen, Z., Gu, Y., Wu, Y., Tao, M., Fu, Z., *et al. INSIDE: LLMs' Internal States Retain the Power of Hallucination Detection* Feb. 6, 2024. arXiv: 2402.03744[cs].

16. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., *et al. Attention Is All You Need* Aug. 1, 2023. arXiv: 1706.03762[cs].

17. Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X. & Gao, J. *Large Language Models: A Survey* Feb. 20, 2024. arXiv: 2402.06196[cs].

18. Radford, A., Narasimhan, K., Salimans, T. & Sutskever, I. Improving Language Understanding by Generative Pre-Training.

19. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. & Sutskever, I. *Language Models are Unsupervised Multitask Learners* in (2019).

20. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., *et al. Language Models are Few-Shot Learners* July 22, 2020. arXiv: 2005.14165[cs].

21. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., *et al. LLaMA: Open and Efficient Foundation Language Models* Feb. 27, 2023. arXiv: 2302.13971[cs].

22. AI@Meta. Llama 3 model card (2024).

23. Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., *et al. A Survey on Evaluation of Large Language Models* Dec. 28, 2023. arXiv: 2307.03109[cs].

24. Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D. & Steinhardt, J. *Measuring Massive Multitask Language Understanding* Jan. 12, 2021. arXiv: 2009.03300[cs].

25. Zhong, W., Cui, R., Guo, Y., Liang, Y., Lu, S., Wang, Y., Saied, A., *et al. AGIEval: A Human-Centric Benchmark for Evaluating Foundation Models* Sept. 18, 2023. arXiv: 2304.06364[cs].

26. Talmor, A., Herzig, J., Lourie, N. & Berant, J. *CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge* Mar. 15, 2019. arXiv: 1811.00937[cs].

27. Sakaguchi, K., Bras, R. L., Bhagavatula, C. & Choi, Y. *WinoGrande: An Adversarial Winograd Schema Challenge at Scale* Nov. 21, 2019. arXiv: 1907.10641[cs].

28. Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., *et al. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models* June 12, 2023. arXiv: 2206.04615[cs,stat].

29. François Chollet, Katherine Tong, Walter Reade & Julia Elliott. *Abstraction and reasoning challenge* 2020.

30. Joshi, M., Choi, E., Weld, D. & Zettlemoyer, L. *TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension* in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* ACL 2017 (eds Barzilay, R. & Kan, M.-Y.) (Association for Computational Linguistics, Vancouver, Canada, July 2017), 1601–1611.

31. Choi, E., He, H., Iyyer, M., Yatskar, M., Yih, W.-t., Choi, Y., Liang, P., *et al. QuAC: Question Answering in Context* in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* EMNLP 2018 (eds Riloff, E., Chiang, D., Hockenmaier, J. & Tsujii, J.) (Association for Computational Linguistics, Brussels, Belgium, Oct. 2018), 2174–2184.

32. Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M. & Toutanova, K. *BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions* May 24, 2019. arXiv: 1905.10044[cs].

33. Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S. & Gardner, M. *DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs* in *Proceedings of the 2019 Conference of the North* Proceedings of the 2019 Conference of the North (Association for Computational Linguistics, Minneapolis, Minnesota, 2019), 2368–2378.

34. Gawlikowski, J., Tassi, C., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., *et al.* A survey of uncertainty in deep neural networks. *Artificial Intelligence Review* **56,** 1–77 (July 29, 2023).

35. Hüllermeier, E. & Waegeman, W. Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods. *Machine Learning* **110,** 457–506. ISSN: 0885-6125, 1573-0565. arXiv: 1910.09457[cs,stat] (Mar. 2021).

36. Kirchhof, M., Collier, M., Oh, S. J. & Kasneci, E. *Pretrained Visual Uncertainties* Feb. 27, 2024. arXiv: 2402.16569[cs].

37. Kingma, D. P. & Welling, M. *Auto-Encoding Variational Bayes* Dec. 10, 2022. arXiv: 1312.6114[cs,stat].

38. Böhm, V., Lanusse, F. & Seljak, U. *Uncertainty Quantification with Generative Models* Oct. 22, 2019. arXiv: 1910.10046[astro-ph,stat].

39. Belen, J., Mousavi, S., Shamsoshoara, A. & Afghah, F. *An Uncertainty Estimation Framework for Risk Assessment in Deep Learning-based AFib Classification* in *2020 54th Asilomar Conference on Signals, Systems, and Computers* 2020 54th Asilomar Conference on Signals, Systems, and Computers (IEEE, Pacific Grove, CA, USA, Nov. 1, 2020), 960–964. ISBN: 978-0-7381-3126-9.

40. Lin, S., Clark, R., Trigoni, N. & Roberts, S. *Uncertainty Estimation with a VAE-Classifier Hybrid Model* in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (IEEE, Singapore, Singapore, May 23, 2022), 3548–3552. ISBN: 978-1-66540-540-9.

41. Catoni, J., Ferrante, E., Milone, D. H. & Echeveste, R. *Uncertainty in latent representations of variational autoencoders optimized for visual tasks* version: 1. Apr. 23, 2024. arXiv: 2404.15390[cs].

42. Devlin, J., Chang, M.-W., Lee, K. & 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)* NAACL-HLT 2019 (eds Burstein, J., Doran, C. & Solorio, T.) (Association for Computational Linguistics, Minneapolis, Minnesota, June 2019), 4171–4186.

43. Jurafsky, D. & Martin, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* Third Edition draft (2024).

44. Ainslie, J., Lee-Thorp, J., Jong, M. d., Zemlyanskiy, Y., Lebrón, F. & Sanghai, S. *GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints* version: 3. Dec. 23, 2023. arXiv: 2305.13245.

45. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. in *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations* 318–362 (MIT Press, Cambridge, MA, USA, Jan. 3, 1986). ISBN: 978-0-262-68053-0.

46. *Reparameterization trick* Page Version ID: 1251079420. https://en.wikipedia.org/wiki/Reparameterization_trick (2024).

47. Bank, D., Koenigstein, N. & Giryes, R. *Autoencoders* Apr. 3, 2021. arXiv: 2003.05991.

48. Kingma, D. P. & Welling, M. *An Introduction to Variational Autoencoders* Dec. 11, 2019. arXiv: 1906.02691.

49. Meng, K., Bau, D., Andonian, A. & Belinkov, Y. *Locating and Editing Factual Associations in GPT* Jan. 13, 2023. arXiv: 2202.05262[cs].

50. Chuang, Y.-S., Xie, Y., Luo, H., Kim, Y., Glass, J. & He, P. *DoLa: Decoding by Contrasting Layers Improves Factuality in Large Language Models* Mar. 11, 2024. arXiv: 2309.03883.

51. Aynetdinov, A. & Akbik, A. *SemScore: Automated Evaluation of Instruction-Tuned LLMs based on Semantic Textual Similarity* Feb. 5, 2024. arXiv: 2401.17072.