Uncertainty Modelling in LLMs using Classifier-Free Guidance

Lau Jun Jie [redacted]

Lim Ciwen Brendan [redacted]

Nicholas Tan Kian Boon [redacted]

Rama Aryasuta Pangestu [redacted]

Wilson Widyadhana [redacted]

Abstract

Instruction-fine-tuned Large Language Models (LLMs) are known to produce highly confident responses, regardless of their factual accuracy. Proposed methods to probe the confidence of such responses include confidence elicitation [1], however such techniques introduce another layer of uncertainty in the confidence response. In this paper, we examine the use of Classifier-Free Guidance as a means to measure confidence for the model outputs to provide more information to the end-user for discerning query results.

1 Introduction

With the release of LLMs such as ChatGPT, LLMs have become widely popular and utilised in many tasks. However, there is still a lack of understanding regarding the inner mechanisms of such transformer-based models, leading to increased scrutiny of the model outputs with regards to issues such as bias and fairness. These outputs can be inaccurate at times and even lead to hallucination [2].

It is therefore crucial to estimate the confidence of LLMs to mitigate such hallucinations from misleading users with overconfident responses. These situations can have serious consequences, with a New York federal judge considering sanctions after a lawyer used ChatGPT – a well-known chatbot that uses a LLM – to generate fictitious cases [3]. Various approaches for confidence estimation have been explored, specifically with uncertainty estimation methods that have been previously applied to Deep Neural Networks (DNNs) such as Bayesian Inference [4] and Deep Ensembles [5]. However the primary challenge to these methods lies in intractability and the computational resources required as modern LLMs tend to have many model parameters. Other potential methods include LLM calibration [6] and confidence elicitation [1], although additional uncertainty can be introduced during the process of obtaining the model's confidence.

This paper aims to provide a method to gauge the confidence of responses generated by LLMs using classifier-free guidance, a test-time augmentation technique [7] that avoids the large computational resources required and also avoid adding more uncertainty to the confidence estimates of the model outputs. This allows the LLM to provide users with additional information on whether the model should be trusted.

2 Background and Preliminaries

There are many different methods used to model uncertainty in DNNs, including the use of Bayesian Inference, Ensemble methods, and test-time augmentation methods [8]. To apply Bayesian Inference to transformers, Bayesian Neural Networks are incorporated into each transformer block [4]. Although this provides us the ability to model the uncertainty of transformer-based models, it can be intractable for larger models such as LLMs, therefore limiting its application.

Ensemble methods provide a better alternative as a means to model the uncertainty of transformer-based models like LLMs by making use an ensemble of base models. Similar to its application to DNNs, the ensemble of models provide an approximation of the likelihood, thus being more computationally feasible than direct Bayesian Inference [5]. However, the challenge of training multiple LLMs for the ensemble still remains, which can be a constraint when computing resources are limited.

The final alternative to model uncertainty in LLMs would be test-time augmentation methods [7], which make use of augmentation on each input instance to obtain varying outputs during test-time. The variations in model outputs are then used to estimate the uncertainty of the model. This method is the most favorable for LLMs as it allows us to leverage on the fast inference time of LLMs compared to their training time required.

3 Problem Objective

Given a pre-trained LLM that is then further fine-tuned on specific datasets, we aim to probe the confidence of the model when provided with a specific instance/input by utilizing classifier-free guidance to obtain a collection of outputs and using the variations in the model outputs to approximate the uncertainty of the model. This paper will determine the effectiveness of the proposed method by using a meta-scoring system such that an empty response provided by the model due to high levels of uncertainty is favored over highly confident but potentially incorrect responses.

4 Classifier-Free Guidance

Classifier-free guidance originated from classifier guidance [9], which is a method to condition the generation of x based on a label c to model $p(x|c)$. Let $p_\theta(x)$ be our generative model for a data point x with parameters θ, and $p_\phi(c|x)$ be our probability generated by a classifier ϕ. Then with Bayes' rule, we know that [10]:

$$p(x|c) \propto p_\theta(x) p_\phi(c|x) \tag{1}$$

We can also introduce a guidance strength (also called CFG scale or guidance scale) γ into the equation like so [10] [11], in order to adjust the influence that $p_\phi(c|x)$ has on the eventual distribution:

$$p(x|c) \propto p_\theta(x) p_\phi(c|x)^\gamma \tag{2}$$

Essentially, γ is a hyperparameter which allows us to adjust how much $p(x|c)$ focuses on the conditioning c. Intuitively,

- when $\gamma = 0$, the model ignores the conditioning information entirely, and simply samples from $p(x)$;
- when $\gamma = 1$, the model conditionally generates based on the label c;
- when $\gamma > 1$ is large, the model *overemphasizes* the conditioning, leading to samples that more heavily adhere to the conditioning information (at the cost of diversity, as well as instability in the output of the logits).

However, training a classifier to compute $p_\phi(c|x)$ is not straightforward [11]; moreover, in LLMs, the conditioning label c could be a free-form sentence. Hence, we would like to have a method of attaining the artificial $p(x|c)$ (boosted by the hyperparameter γ) without the use of a classifier. We can achieve this by noticing that [10]:

$$p_\theta(c|x) \propto \frac{p_\theta(x|c)}{p(x)} \tag{3}$$

Substituting $p_\phi(c|x)$ for $p_\theta(c|x)$, we get [10]:

$$p_\theta(x|c) \propto \frac{p_\theta(x|c)^\gamma}{p_\theta(x)^{\gamma-1}} \tag{4}$$

Taking log on both side, and ignoring constants, we get:

$$\log p_\theta(x|c) = \log p_\theta(x) + \gamma(\log p_\theta(x|c) - \log p_\theta(x)) \tag{5}$$

By viewing the $\log p$ terms as vectors, the equation 5 above is a vector arithmetic operation in the model latent space – starting from the *unconditional point* $\log p_\theta(x)$, we move by γ units to the *conditioning direction* given by $(\log p_\theta(x|c) - \log p_\theta(x))$ [10].

With Equation 5, we can thus evaluate the log-likelihood of the model prior conditioned upon an input prompt c at varying strength γ. We fine-tuned various baseline models on specific datasets and adjusted γ during test-time to eventually compute the uncertainty scores.

5 Methodology

Implementation-wise, LLMs are implicitly-conditional models, where, given the previous tokens c_i as input, it generates the next token $c\_i + 1$ from the its output distribution $p(x|c)$. The unconditional model $p(x)$ is then obtained by simply setting c as the empty text. We can take the γ-guided output by taking $\log$ on the terms, transforming it according to Equation 5, then rescaling it to obtain a proper probability distribution.

Prior work on applying classifier-free guidance to LLMs [10] limited their experiments to decoder-only large-language models. We adapted classifier-free guidance on encoder-decoder models by setting the conditioning c_e for the encoder as null (empty text), as well as following [10]'s convention for the conditioning c_d for the decoder.

In this paper, we have used both `T5` [12] and `Flan-T5` [13] as the baseline models to be further fine-tuned. There are several datasets we have used in this paper, including the CommonsenseQA (CQA) dataset [14], the Simple Variations on Arithmetic Math word Problems (SVAMP) dataset [15], as well as the Massive Multitask Language Understanding (MMLU) dataset [16].

The datasets were first pre-processed to produce the necessary inputs for fine-tuning of the models. As an example, the CQA dataset was pre-processed to form question-choice and answer pairs like the example below:

```
("The sanctions against the school were a punishing blow, and they
seemed to what the efforts the school had made to change?
(a) ignore (b) enforce (c) authoritarian (d) yell at (e) avoid",
"ignore")
```

The SVAMP and MMLU datasets were also pre-processed in a similar manner. Models of different sizes were fine-tuned, namely `small`, `base`, and `large` versions of both the `T5` and `Flan-T5` architectures respectively. The following training configurations were used for each of the datasets to account for the training time required due to the difference in their sizes. Note that each of the MMLU_n training configurations were considered based on the amount of computational time available, with experiments using MMLU_2 and MMLU_3 being conducted after more computational time was obtained.

Dataset	Batch Size	Steps per Evaluation	Max Evaluation Steps
cqa	64	250	10,000
svamp	64	250	10,000
mmlu_1	8	100	2,000
mmlu_2	8	250	10,000
mmlu_3	8	1,000	100,000

Table 1: Training configurations used for each dataset

5.1 Scoring Function

A meta-scoring system was devised to measure the impact of uncertain outputs provided by the model. A typical scoring system to measure the accuracy of the model would be to assign a score of 1 to correct answers and 0 to wrong answers. We describe our meta-scoring system as follows:

$$\texttt{score} = (n_{\texttt{correct}} + w_{\texttt{empty}} \cdot n_{\texttt{empty}})/n_{\texttt{total}}$$

where $n_{\texttt{correct}}$ is the number of correct answers, $n_{\texttt{empty}}$ is the number of questions which the model *refuses to answer*, and $n_{\texttt{total}}$ is the total number of questions.

The parameter $w_{\texttt{empty}}$ above expresses how much we value the model refusing to answer over answering incorrectly. In our experiments, we arbitrarily set $w_{\texttt{empty}} = 0.5$, although different values for $w_{\texttt{empty}}$ can be explored as future work.

5.2 Model Inference

Using the fact that the guidance strength γ is a hyperparameter in classifier-free-guided models, we select the appropriate value by initialising a prior distribution for it, i.e., $\gamma \sim p(\gamma)$. Sampling was then done independently to obtain guidance strengths $\gamma_i = \gamma_1, \gamma_2, \ldots, \gamma_k$ from $p(\gamma)$, and generate responses $R_i = p_\theta^{\gamma_i}(x|c)$ through model inference of the fine-tuned LLM.

Let R be the response which occurs the most often in $\{R_1, R_2, \ldots, R_k\}$. For a threshold parameter t, if the fraction of responses in which R occurs is greater than t, we set $q(x|c) := R$; otherwise, we set $q(x|c) := \texttt{empty}$. Essentially, the output is set to $\texttt{empty}$ if, when varying the guidance strength γ, the model's output diverges a lot.

In our experiments, we will set $t \in \{0.5, 0.75\}$ (in the results table, t is called as a "threshold").

5.3 Optimizing Inference

In practice, using the method of sampling $\gamma \sim p(\gamma)$ as stated previously would take a long time over multiple iterations. Therefore, we employed a strategy similar to importance sampling, where the proposal distribution is fixed as the uniform distribution.

The guidance strength γ is fixed such that it strictly falls between the range $\gamma \in [a, b]$. We obtain response $R^{(\gamma)}$ for every $\gamma \in \{a + k \cdot \epsilon | k \in [0, \frac{b-a}{\epsilon}]\}$, where ϵ measures how accurate we estimate the prior $p(\gamma)$. For each response $R^{(\gamma)}$, we assign this response a weight of $w(\gamma) \propto p(\gamma)$. We can then set R as the response with the maximum "total sum of weights $w(\gamma)$ of the responses $R^{(\gamma)}$ equal to R", and if it is greater than t times the overall sum of weights, set $q(x|c) := R$; otherwise, $q(x|c) = \texttt{empty}$. As we take $\epsilon \to 0$, a response r will have weight $\int_a^b \mathbf{1}_{R^{(\gamma)}=r} \cdot w(\gamma)\, d\gamma$ (where $\mathbf{1}_P$ if and only if the predicate P is true). Since $w(\gamma) \propto p(\gamma)$, this is equivalent to our formulation in 5.2.

In our experiments, we will set $a = 1$, $b \in \{3, 4\}$, and $\epsilon \in \{0.5, 0.01\}$, whereby $[a, b]$ is referred to as the "CFG range" and ϵ is referred to as the "CFG precision".

With this formulation, we can now estimate the result of sampling $\gamma \sim p(\gamma)$ efficiently, with a parameter ϵ to control the precision of our estimate. However, with $[a, b] = [1, 4]$ and $\epsilon = 0.01$, a total of 301 classifier-free-guided inference of the model is needed and thus lots of computational time is required.

To reduce the number of inferences required, we claim the following.

Claim 1 (Contiguous Response Property)**.** *Let $p_\theta^\gamma(x|c)$ denote the model output's probability distribution with guidance scale γ. Suppose that we generate the output of the model x on a conditioning c under a greedy-decoding scheme, i.e., $x = \arg\max_x p_\theta^\gamma(x|c)$. For two guidance scales $\gamma_1 < \gamma_2$, if $y = p_\theta^{\gamma_1}(x|c) = p_\theta^{\gamma_2}(x|c)$, we claim that for all $\gamma \in [\gamma_1, \gamma_2]$, we have $p_\theta^\gamma(x|c) = y$ as well.*

Proof. Suppose that the output $y = p_\theta^{\gamma_1}(x|c) = p_\theta^{\gamma_2}(x|c)$ is a single token. Note that in a greedy-decoding scheme, taking $\log$ preserves ordering of the distribution of tokens, so we will work with $q_1(x) = \log p_\theta^{\gamma_1}(x|c)$ and $q_2(x) = p_\theta^{\gamma_2}(x|c)$ ($q_1(x)$ and $q_2(x)$ are functions and not probability distributions). Fix a token x_0, the value of $\log p_\theta^\gamma(x_0|c)$ with varying γ is given by a linear function $g_{x_0}(\gamma) = \log p_\theta(x_0) + \gamma \cdot (\log p_\theta(x_0|c) - \log p_\theta(x_0))$.

Consider the linear function $g_{x_1}(\gamma)$ of another token x_1, and note that: (i) either $g_{x_0}(\cdot)$ and $g_{x_1}(\cdot)$ is parallel (in which case, one of them is strictly better than the other at all γ scales, assuming we break ties for greedy decoding deterministically); or (ii) they intersect at a point p. If they intersect, either (i) for all $\gamma > p$, we have $g_{x_1}(\gamma) > g_{x_0}(\gamma)$; or (ii) for all $\gamma < p$, we have $g_{x_1}(\gamma) > g_{x_0}(\gamma)$.

Suppose $\mathcal{X}$ is the set of all tokens. Initially, if we only consider the token x_0, it is the maximum for the interval $\gamma \in (\infty, \infty) = I$. As we add all other tokens $x_1 \in \mathcal{X}$ one-by-one, it will either: (i) be worse than x_0 for all γ, and thus do nothing; (ii) be strictly better than x_1 for all γ, and thus x_0 will never be a outputted for any γ; (iii) will make it such that a *prefix* of I is bad (i.e., x_0 is no longer the maximum for this prefix); or (iv) will make it such that a *suffix* of I is bad. Therefore, after adding all other tokens $\mathcal{X} \setminus \{x_0\}$, x_0 will be the maximum at either nowhere, or a contiguous interval I. This completes the proof for the single token case.

With induction, it is straightforward to prove the same given that the output of the models $p_\theta^{\gamma_1}(x|c) = p_\theta^{\gamma_2}(x|c)$ is a sequence of tokens $y_1, y_2, \ldots, y_k$, under the assumption that each token y_i is generated through a greedy-decoding scheme $y_i = \arg\max_x p_\theta(x|c, y_1, \ldots, y_{i-1})$. □

We can use the contiguous response property (1) to reduce the number of inferences. One way to do it is to start at $\gamma = a$ initially, then binary search the maximum γ_1 such that $R^{(\gamma_1)} = R^{(\gamma)}$, then recursively solve for the range $[\gamma_1 + \epsilon, b]$. A better implementation is to use a recursive "divide-and-conquer" approach (pseudocode in appendix at 1) by calling MULTIINFERENCE$(0, M)$. The theoretical time complexity is $O(k' \cdot \log((b-a)\epsilon^{-1}))$, where k' is the number of distinct responses, but in practice we reduce the number of inferences to approximately 30 classifier-free-guided inferences, as opposed to over 300.

6 Results

After fine-tuning on the training dataset and running inference on the test datasets, the models were scored using the meta-scoring system 5.1, whereby *Base Score* denotes the score of the model without using CFG ($\gamma = 1$) and *Final Score* denotes the score of the model using CFG.

6.1 CFG range and precision

Model	CFG range	CFG precision	Threshold	Base Score	Final Score
small	[1, 3]	0.5	0.75	40.46	**44.88**
base	[1, 3]	0.5	0.75	59.62	**60.85**
large	[1, 3]	0.5	0.75	**71.34**	56.47
flan_small	[1, 3]	0.5	0.75	46.27	**47.38**
flan_base	[1, 3]	0.5	0.75	**74.69**	72.11
flan_large	[1, 3]	0.5	0.75	**85.91**	83.62
small	[1, 4]	0.01	0.75	40.46	**41.15**
base	[1, 4]	0.01	0.75	59.62	**60.36**
large	[1, 4]	0.01	0.75	**71.34**	53.19
flan_small	[1, 4]	0.01	0.75	**46.27**	46.23
flan_base	[1, 4]	0.01	0.75	**74.69**	71.62
flan_large	[1, 4]	0.01	0.75	**85.91**	83.05

Table 2: Base and final scores of models trained on CQA with $p(\gamma) = \mathcal{U}_{[a,b]}$

From Table 2, it can be observed that larger model sizes tend to perform better than smaller ones. Furthermore, the `flan` versions of the models usually perform better than the t5 models, possibly due to the former group having more advanced components in their architectures [17]. The various gaps observed in the *Base Score* and *Final Score* can be attributed to the fact that the smaller models tend to be rather uncertain and thus benefit from a higher score by providing `empty` responses, whereas larger models tend to be overconfident but provide wrong responses.

It is also worth noting that some of the models, especially the larger variants, tend to be rather sensitive to the change in CFG values, thus would have rather diverse outputs when CFG range and CFG precision is set to [1, 3] and 0.5 respectively. Therefore for the rest of remaining experiments, we will mainly focus on the results using CFG range $p(\gamma) = \mathcal{U}_{[1,4]}$ and CFG precision $\epsilon = 0.01$.

6.2 Overall Performance

Base Score | Final Score										
Model	cqa		svamp		mmlu_1		mmlu_2		mmlu_3	
small	40.46	**41.15**	3.67	**50.00**	10.70	**39.82**	10.88	**44.04**	28.07	**45.96**
base	59.62	**60.36**	4.67	**49.67**	12.11	**35.09**	25.26	**33.42**	35.96	**39.91**
large	**71.34**	53.19	8.67	**50.00**	0.00	**49.74**	41.40	**49.39**	41.40	**48.07**
flan_small	**46.27**	46.23	7.33	**44.83**	26.67	**33.16**	28.95	**34.30**	28.42	**34.30**
flan_base	**74.69**	71.62	7.00	**30.17**	32.28	**36.23**	31.58	**36.32**	35.26	**37.37**
flan_large	**85.91**	83.05	11.00	**14.33**	**39.47**	39.21	38.77	**40.61**	40.00	**40.70**

Table 3: Base and final scores of models using threshold $t = 0.75$

Base Score | Final Score										
Model	cqa		svamp		mmlu_1		mmlu_2		mmlu_3	
small	**40.46**	18.35	3.67	**48.17**	10.70	**24.21**	10.88	**34.39**	28.07	**34.21**
base	**59.62**	59.30	4.67	**44.83**	12.11	**21.32**	**25.26**	24.65	**35.96**	34.12
large	**71.34**	50.74	8.67	**48.33**	0.00	**34.74**	41.40	**44.82**	**41.40**	37.19
flan_small	**46.27**	44.55	7.33	**31.83**	26.67	**28.68**	28.95	**30.00**	28.42	**29.21**
flan_base	**74.69**	70.72	7.00	**17.00**	32.28	**32.46**	31.58	**32.63**	**35.26**	32.72
flan_large	**85.91**	82.47	11.00	**11.33**	**39.47**	37.54	**38.77**	38.68	**40.00**	38.60

Table 4: Base and final scores of models using threshold $t = 0.5$

The fine-tuned models generally performed better on the CQA dataset compared to the SVAMP and MMLU datasets and we hypothesise that this might be due to the datasets that have been used to pre-train both `T5` and `Flan-T5` [12] [13], which contain more question-answering (QA) tasks than the tasks that are present in the SVAMP or MMLU datasets.

It can also be observed from Table 3 and Table 4 that lowering the threshold t leads to a worse *Final Score*. This might be due to the model being necessitated to be more confident in its predictions with a higher threshold, thereby improving its score through providing `empty` responses. The results in Table 4 also supports the trend observed which is that smaller models tend to be uncertain and hence have a better *Final Score* compared to their larger variants.

6.3 Weighted Intervals

A key observation made during the experiments was that larger models tend to produce longer and more nonsensical outputs at higher CFG values (refer to appendix 8), which can be one of the reasons that the larger models are overconfident but provide wrong answers as their outputs are dominated by such long and nonsensical responses.

To mitigate the dominance of such responses, we can define a weight for each response such that responses produced at lower CFG values will have greater weight than those produced at higher CFG values. The weights are assigned according to the formula $w(i) = A\exp(-r\frac{i}{N})$, where r is the rate of decay, and i is the index of the point ($i = (\gamma - 1)/\epsilon$) such that $w(0) = A, w(N) = A\exp(-r)$ (with $N = (4-1)/\epsilon$ when $p(\gamma) = \mathcal{U}_{[1,4]}$). The weighted intervals will be denoted by $[w(0), w(N)]$, where $w(0)$ equals the first number in the weighted interval (i.e., $p_\gamma(1) \propto w(0)$) and $w(N)$ equals the last (i.e., $p_\gamma(4) \propto w(N)$).

Base Score | Final Score											
Model	Dataset	Weighted Intervals									
		[1.0, 1.0]		[1.0, 0.5]		[1.0, 0.1]		[1.0, 0.05]		[1.0, 0.01]	
small	cqa	40.46	**41.15**	40.46	**41.15**	40.46	**41.11**	40.46	**41.15**	40.46	**41.81**
	svamp	3.67	**50.00**	3.67	**50.00**	3.67	**50.00**	3.67	**50.00**	3.67	**51.00**
	mmlu_1	10.70	**39.82**	10.70	**39.82**	10.70	**39.82**	10.70	**39.82**	10.70	**39.82**
	mmlu_2	10.88	**44.04**	10.88	**44.04**	10.88	**44.04**	10.88	**44.04**	10.88	**44.12**
	mmlu_3	28.07	**45.96**	28.07	**46.05**	28.07	**46.23**	28.07	**46.58**	28.07	**47.98**
base	cqa	59.62	**60.36**	59.62	**60.11**	59.62	**59.87**	59.62	**59.62**	59.62	**59.71**
	svamp	4.67	**49.67**	4.67	**49.67**	4.67	**50.00**	4.67	**51.00**	4.67	**51.00**
	mmlu_1	12.11	**35.09**	12.11	**35.18**	12.11	**35.18**	12.11	**35.35**	12.11	**35.44**
	mmlu_2	25.26	**33.42**	25.26	**33.16**	25.26	**33.33**	25.26	**33.42**	25.26	**33.25**
	mmlu_3	35.96	**39.91**	35.96	**39.47**	35.96	**39.30**	35.96	**39.47**	35.96	**39.30**
large	cqa	**71.34**	53.19	**71.34**	54.01	**71.34**	58.31	**71.34**	62.00	**71.34**	68.39
	svamp	8.67	**50.00**	8.67	**50.00**	8.67	**50.33**	8.67	**50.83**	8.67	**52.50**
	mmlu_1	0.00	**49.74**	0.00	**49.74**	0.00	**49.74**	0.00	**49.74**	0.00	**49.74**
	mmlu_2	41.40	**49.39**	41.40	**49.65**	41.40	**50.00**	41.40	**50.44**	41.40	**53.86**
	mmlu_3	41.40	**48.07**	41.40	**48.16**	41.40	**48.25**	41.40	**48.42**	41.40	**49.56**
flan_small	cqa	**46.27**	46.23	**46.27**	46.03	**46.27**	45.41	**46.27**	45.25	**46.27**	44.96
	svamp	7.33	**44.83**	7.33	**45.00**	7.33	**45.33**	7.33	**46.67**	7.33	**47.00**
	mmlu_1	26.67	**33.16**	26.67	**32.46**	26.67	**31.05**	26.67	**30.88**	26.67	**29.47**
	mmlu_2	28.95	**34.30**	28.95	**33.33**	28.95	**31.58**	28.95	**31.75**	28.95	**30.70**
	mmlu_3	28.42	**34.30**	28.42	**33.51**	28.42	**32.72**	28.42	**31.84**	28.42	**30.96**
flan_base	cqa	**74.69**	71.62	**74.69**	71.42	**74.69**	71.46	**74.69**	71.66	**74.69**	72.15
	svamp	7.00	**30.17**	7.00	**30.33**	7.00	**30.33**	7.00	**30.50**	7.00	**30.83**
	mmlu_1	32.28	**36.23**	32.28	**35.88**	32.28	**35.35**	32.28	**35.44**	32.28	**34.82**
	mmlu_2	31.58	**36.32**	31.58	**35.79**	31.58	**35.70**	31.58	**35.00**	31.58	**34.65**
	mmlu_3	35.26	**37.37**	35.26	**36.49**	35.26	**35.35**	**35.26**	34.30	**35.26**	34.74
flan_large	cqa	**85.91**	83.05	**85.91**	82.96	**85.91**	83.01	**85.91**	83.25	**85.91**	83.50
	svamp	11.00	**14.33**	11.00	**14.00**	11.00	**13.67**	11.00	**13.67**	11.00	**13.50**
	mmlu_1	**39.47**	39.21	**39.47**	38.95	**39.47**	38.07	**39.47**	37.63	**39.47**	37.72
	mmlu_2	38.77	**40.61**	38.77	**40.18**	38.77	**40.18**	38.77	**40.00**	38.77	**39.12**
	mmlu_3	40.00	**40.70**	40.00	**40.35**	**40.00**	39.21	**40.00**	39.21	**40.00**	39.47

Table 5: Base and final scores of models with varying weighted intervals using threshold $t = 0.75$

It can be observed from Table 5 and Table 6 that using a wider range of values for the weighted intervals tends to produce better *Final Scores* for the `t5` models and produce worser *Final Scores* for the `flan` models. This indicates that the `flan` models are less sensitive to changes in CFG values and thus produce less divergent responses at different CFG values which would result in highly confident but incorrect responses. This is supported by the trend in the lesser decrease of *Final Score* for the `flan` models compared to those of the `t5` models when the threshold is changed from $t = 0.75$ to $t = 0.5$ across all weighted intervals.

There is also a notable shirnk in the gap between the *Base Score* and *Final Score* of the models when the threshold is changed from $t = 0.75$ to $t = 0.5$ as the effect of the weighted intervals causes the models to favor the outputs generated at lower CFG values, which is much closer to the responses of an unconditional model and thus the *Final Score* would be much closer to the *Base Score*. This would suggest that there is a trade-off when using a weight function as it would be subject to the sensitivity of the threshold value.

		Base Score | Final Score									
Model	Dataset	Weighted Intervals									
		[1.0, 1.0]		[1.0, 0.5]		[1.0, 0.1]		[1.0, 0.05]		[1.0, 0.01]	
small	cqa	**40.46**	18.35	**40.46**	18.51	**40.46**	20.80	**40.46**	24.08	**40.46**	35.79
	svamp	3.67	**48.17**	3.67	**48.17**	3.67	**49.33**	3.67	**49.33**	3.67	**49.67**
	mmlu_1	10.70	**24.21**	10.70	**24.21**	10.70	**24.30**	10.70	**24.65**	10.70	**25.88**
	mmlu_2	10.88	**34.39**	10.88	**34.39**	10.88	**34.56**	10.88	**34.56**	10.88	**35.09**
	mmlu_3	28.07	**34.21**	28.07	**34.74**	28.07	**37.98**	28.07	**40.88**	28.07	**45.18**
base	cqa	**59.62**	59.30	**59.62**	59.13	**59.62**	59.38	**59.62**	59.50	59.62	**59.66**
	svamp	4.67	**44.83**	4.67	**45.83**	4.67	**46.17**	4.67	**46.33**	4.67	**46.50**
	mmlu_1	12.11	**21.32**	12.11	**21.14**	12.11	**22.28**	12.11	**22.37**	12.11	**22.28**
	mmlu_2	**25.26**	24.65	25.26	**25.26**	25.26	**25.70**	25.26	**25.53**	25.26	**25.96**
	mmlu_3	**35.96**	34.12	**35.96**	34.65	35.96	**36.32**	35.96	**36.14**	35.96	**37.46**
large	cqa	**71.34**	50.74	**71.34**	55.32	**71.34**	66.01	**71.34**	71.05	71.34	**78.21**
	svamp	8.67	**48.33**	8.67	**49.17**	8.67	**51.17**	8.67	**52.00**	8.67	**52.17**
	mmlu_1	0.00	**34.74**	0.00	**34.74**	0.00	**34.74**	0.00	**34.74**	0.00	**34.74**
	mmlu_2	41.40	**44.82**	41.40	**45.53**	41.40	**50.53**	41.40	**53.77**	41.40	**59.39**
	mmlu_3	**41.40**	37.19	**41.40**	37.63	**41.40**	39.91	41.40	**41.40**	41.40	**46.14**
flan_small	cqa	**46.27**	44.55	**46.27**	44.68	**46.27**	45.33	**46.27**	45.29	46.27	**46.44**
	svamp	7.33	**31.83**	7.33	**33.33**	7.33	**33.83**	7.33	**34.00**	7.33	**34.17**
	mmlu_1	26.67	**28.68**	26.67	**28.68**	26.67	**28.33**	26.67	**28.86**	26.67	**28.51**
	mmlu_2	28.95	**30.00**	28.95	**30.18**	28.95	**30.96**	28.95	**30.26**	28.95	**29.65**
	mmlu_3	28.42	**29.21**	28.42	**29.21**	28.42	**29.39**	28.42	**28.95**	28.42	**29.30**
flan_base	cqa	**74.69**	70.72	**74.69**	71.01	**74.69**	72.11	**74.69**	72.32	**74.69**	72.89
	svamp	7.00	**17.00**	7.00	**17.17**	7.00	**17.67**	7.00	**18.00**	7.00	**17.33**
	mmlu_1	32.28	**32.46**	32.28	**32.98**	32.28	**33.77**	32.28	**33.95**	32.28	**33.60**
	mmlu_2	31.58	**32.63**	31.58	**33.07**	31.58	**33.16**	31.58	**33.16**	31.58	**34.47**
	mmlu_3	**35.26**	32.72	**35.26**	33.86	35.26	**36.75**	35.26	**36.23**	35.26	**36.84**
flan_large	cqa	**85.91**	82.47	**85.91**	82.96	**85.91**	83.70	**85.91**	83.87	**85.91**	84.44
	svamp	11.00	**11.33**	11.00	**11.33**	11.00	**11.33**	11.00	**12.00**	11.00	**11.67**
	mmlu_1	**39.47**	37.54	**39.47**	36.84	**39.47**	36.84	**39.47**	37.54	**39.47**	38.60
	mmlu_2	**38.77**	38.68	**38.77**	38.16	**38.77**	38.16	**38.77**	38.16	38.77	**39.04**
	mmlu_3	**40.00**	38.60	**40.00**	38.60	**40.00**	38.68	**40.00**	38.51	**40.00**	37.98

Table 6: Base and final scores of models with varying weighted intervals using threshold $t = 0.5$

7 Discussion

In general, different models were observed to respond to classifier-free guidance differently. A numerical analysis of outputs from different models via Shannon entropy [18] allows us to identify specific instances where models have diverse outputs as seen in the appendix 8, but no clear trends are seen when aggregated to the model level.

As future considerations, classifier-free guidance can be compared against other methods, and further research can be done to combine estimates from these different confidence and uncertainty estimation paradigms. For example, entropy-based methods could be used to enhance and validate the scoring mechanism. Moreover, given more computational resources, we are also interested in exploring different LLMs to probe the confidence of their outputs using classifier-free guidance.

8 Conclusion

This paper demonstrated that the use of classifier-free guidance is an computational efficient method to provide confidence and uncertainty estimates of the model outputs from LLMs. The proposed method is also robust as different prior sampling methods can be explored for a better understanding of the estimation provided by classifier-free guidance instead of just sampling from a uniform distribution. Our method is also not subject to the potential unaccounted factors caused by the introduction of additional variables into the training process as part of other uncertainty modelling techniques employed in DNNs, such as Bayesian Inference or Ensemble methods. Test-time augmentation methods like classifier-free guidance do not rely on these additional variables.

References

[1] M. Xiong, Z. Hu, X. Lu, Y. Li, J. Fu, J. He, and B. Hooi, "Can llms express their uncertainty? An empirical evaluation of confidence elicitation in llms," https://arxiv.org/abs/2306.13063, jun 22 2023, [Online; accessed 2024-04-15].

[2] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu, "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," 2023.

[3] M. Bohannon, "Lawyer used chatgpt in court—and cited fake cases. a judge is considering sanctions." [Online]. Available: https://www.forbes.com/sites/mollybohannon/2023/06/08/lawyer-used-chatgpt-in-court-and-cited-fake-cases-a-judge-is-considering-sanctions/?sh=1206fc857c7f

[4] K. A. Sankararaman, S. Wang, and H. Fang, "Bayesformer: Transformer with uncertainty estimation," https://arxiv.org/abs/2206.00826, jun 2 2022, [Online; accessed 2024-04-15].

[5] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles," https://arxiv.org/abs/1612.01474, dec 5 2016, [Online; accessed 2024-04-15].

[6] S. J. Mielke, A. Szlam, E. Dinan, and Y.-L. Boureau, "Reducing conversational agents' overconfidence through linguistic calibration," https://arxiv.org/abs/2012.14983, dec 30 2020, [Online; accessed 2024-04-15].

[7] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, jul 6 2019, [Online; accessed 2024-04-15].

[8] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, M. Shahzad, W. Yang, R. Bamler, and X. X. Zhu, "A survey of uncertainty in deep neural networks," 2022.

[9] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," 2021.

[10] G. Sanchez, H. Fan, A. Spangher, E. Levi, P. S. Ammanamanchi, and S. Biderman, "Stay on topic with classifier-free guidance," 2023.

[11] J. Ho and T. Salimans, "Classifier-free diffusion guidance," 2022.

[12] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2023.

[13] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, A. Castro-Ros, M. Pellat, K. Robinson, D. Valter, S. Narang, G. Mishra, A. Yu, V. Zhao, Y. Huang, A. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei, "Scaling instruction-finetuned language models," 2022.

[14] A. Talmor, J. Herzig, N. Lourie, and J. Berant, "Commonsenseqa: A question answering challenge targeting commonsense knowledge," 2019.

[15] A. Patel, S. Bhattamishra, and N. Goyal, "Are nlp models really able to solve simple math word problems?" 2021.

[16] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," 2021.

[17] N. Shazeer, "Glu variants improve transformer," 2020.

[18] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

Appendix

FasterInference Algorithm

Algorithm 1 FasterInference

```
1: $M \leftarrow (b-a)/\epsilon$
2: outputs $\leftarrow$ array$[M+1]$
3:
4: procedure SingleInference($i$)
5:     $\gamma \leftarrow a + i \cdot \epsilon$
6:     if outputs$[i]$ = None then
7:         outputs$[i] \leftarrow \arg\max_x p_\theta^\gamma(x|c)$
8:     end if
9:     return outputs$[i]$
10: end procedure
11:
12: procedure MultiInference($[a, b]$)
13:     if SingleInference($a$) = SingleInference($b$) then
14:         for $k = a$ to $b$ do
15:             outputs$[i] \leftarrow$ SingleInference($a$)
16:         end for
17:     else
18:         $m \leftarrow (a+b)/2$
19:         MultiInference($[a, m]$)
20:         MultiInference($[m, b]$)
21:     end if
22: end procedure
```

Output distribution analysis for one instance

When entropy is calculated across the entire output for a single instance, a clear increase in entropy is seen for more diverse or nonsensical responses. The probability of each character was estimated based on character frequency within the entire output list per model for the given input. The outputs for the two *large* models are placed in the next subsection due to space constraints.

Model	CFG range	Outputs	Mean Entropy
small	[1, 3]	['found outside', 'Curtea', 'cazuriified']	**3.723**
base	[1, 3]	['found outside']	**3.239**
large	[1, 3]	*output_1*	3.809
flan_small	[1, 3]	['found outside']	3.239
flan_base	[1, 3]	['found outside']	3.239
flan_large	[1, 3]	['found outside']	3.239
small	[1, 4]	['found outside', 'Curtea', 'cazuriified']	3.723
base	[1, 4]	['found outside']	3.239
large	[1, 4]	*output_2*	**4.036**
flan_small	[1, 4]	['found outside']	3.239
flan_base	[1, 4]	['found outside']	3.239
flan_large	[1, 4]	['found outside']	3.239

Table 7: Example of one instance across multiple models. label: found outside

Example output instances

Examples of long and nonsensical outputs from the above table 7

output_1 : ['found outside', 'found outside fein propus interessiertbehandlung rece intalni found outside recebrudging rece feinnicht feinnichtbrubrubrubrubru', 'found outside fein propus interessiertbehandlung rece intalni located outside recebrudging receEmpfehlung receEmpfehlung receEmpfehlung receEmpfehlung receEmpfehlung', 'found outside fein propus interessiertbehandlung rece intalni laid outsideMerc receEmpfehlung receEmpfehlung receEmpfehlung receEmpfehlung receEmpfehlung receEmpfehlung']

output_2 : ['found outside', 'found outsideEducatorEducatmaticEducatmaticEducatessorEducatmatic', 'found outsideEducatorEducatmaticEducatmatic receintecoleutic', 'found outsideEducatorEducatmaticEducatsagen erfolgt recetief rece', 'found outside fein nose', 'found outside fein nose recemphic recetreatment recetreatment rece', 'found outside fein nose recescient recetrimot recescient rece', 'found outside fein nose recegrin recegrin recegrin recegrin', 'found outside fein nose recegrin recegrin recegrin rece Geschmack', 'found outside fein nose recegrin recegrin rece Geschmack recesagen', 'found outside fein nose recegrin rece Geschmack rece Geschmack rece Geschmack', 'found outside fein nose receEET receEET receEET receEET', 'found outside fein propus interessiertbehandlung rece intalnibehandlung rece intalnibehandlung', 'found outside fein propus interessiertprodukte interessiertbehandlung rece intalnibehandlung rece']