SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Confidence Calibration for Medical Large Vision Language Models

Onur Deniz Güler

Abstract

Large Vision-Language Models (LVLMs) hold promise for supporting radiologists through tasks such as writing X-ray reports and medical decision making, but their clinical reliability is limited due to poor uncertainty quantification. Confidence estimation methods designed for generative models tend to yield overconfident and uncalibrated results with the best methods not integrated into the generative process. In this thesis, we extend the **Rewarding Doubt** method which was designed for unimodal LLMs to LVLMs. We first validate its functionality on the binary question answering behavior of **RaDialog**, a conversational radiology assistant, and then adapt the method to free-form radiology report generation by designing rewards that capture continuous accuracy values. We address the chronic mode collapse issue encountered both in the original Rewarding Doubt study and in our experiments by introducing an alternative reward called the quadratic-blend reward and via tuning the normalization parameters of the cross-entropy-based reward from the Rewarding Doubt method. The new rewards stabilize the training process and diversify confidence outputs. Our approach achieves substantially low ECE, calibration curves closer to the ideal, and a rich spread of confidence values. Our results demonstrate the feasibility of reinforcement learning-based confidence calibration for multimodal medical LVLMs and contribute to their trustworthiness in radiology applications.

Contents

1 Introduction

Foundation models have gained immense traction in usage for many tasks in recent years. Many of their advantages such as multimodality, generative properties, generalization to out-of-distribution tasks, in-context learning and their usability via natural language interactions render foundation models as unprecedentedly strong candidates for interactive agents that can assist professionals in their work. These models are employed more and more for tasks not only in non-critical domains but also in critical domains including medicine [35].

One of the medical branches which could benefit from rapid developments in foundation models and more specifically Large Visual Language Models (LVLMs) is radiology, especially because radiology expert shortage is on the rise [4]. The main tasks of radiologists include the interpretation of patterns on radiology images, the detection of anomalies, and the creation of radiology reports based on findings. Using the pattern recognition and text generation functionalities of LVLMs, it is possible to generate radiology reports based on radiology images. There are multiple studies using LVLMs for the generation of radiology reports, especially for chest X-ray images [7], [41], [6]. Studies increasingly improve clinical efficacy and show promising developments towards reliable radiology assistants.

Despite their exceptional performance in medical tasks, foundation models have limitations such as biases and hallucinations [16]. Foundation models can help fill in for the lack of radiologists and medical professionals in general, if these limitations are addressed properly. A typical difficulty of working with medical LVLM assistants for critical decision making tasks is the uncertainty involved in the generation process. When the certainty of an answer is unknown, or hidden within an LVLM's internal state, a decision-maker cannot completely rely on generations. The unquantified certainty of generations calls for double-checks for verification and takes additional time. This undermines the purpose of employing LVLM agents to make up for the limited time experts have for their tasks.

Quantifying the certainty of large language models (LLMs) about their responses is a more complex problem compared to that of discriminative models. When verbally prompted to report certainty, LLMs tend to report overconfident self-evaluations [57]. There are multiple solution proposals to the uncertainty quantification problem of generative models, one umbrella technique being confidence estimation [10]. Confidence

estimation methods use the internal state of LLMs or the generated content to help elicit a confidence value from the LLM regarding its generations.

While establishing strong baselines, many of the confidence estimations methods either do not attempt to integrate confidence elicitation to the generative process, or fail to calibrate expressed confidences to align with the true accuracy of the statements from the LLM. The Rewarding Doubt method [48] distinguishes itself as a confidence elicitation method using reinforcement learning to teach LLMs to express confidences aligning with the accuracy of their statements. The usage of reinforcement learning removes the need for annotated data for confidence expression and helps avoid catastrophic forgetting during fine-tuning. [31].

In our study, we extend the Rewarding Doubt method to accommodate multimodal models and calibrate verbalized confidence estimations of freeform text generations by going beyond its original binary classification based approach. We use the extended method to fine-tune RaDialog [41], a radiology report generation and conversation assistant, to express calibrated confidence values alongside its generations.

2 Background and Related Work

2.1 Large Language Models

LLMs are generative models based on the transformer architecture with encoders and decoders, and more recently only decoders [53]. Their architectural principle involves no recurrence or convolutions making them more robust for longer sequential data. The architecture involves consuming an input sequence which gradually accumulates newly generated continuations, and process them through a multi-head attention mechanism with feed-forward blocks in between. Via larger scales of parameters and training data, LLMs have achieved high performance in many tasks.

The generative process of an LLM is an auto-regressive next-token sampling from softmax probabilities over their vocabulary. The generation is governed by a few parameters. The *temperature* parameter controlling diversity of generated tokens, the *top-k* parameter limiting possible tokens to be generated at each step to a maximum of k tokens, and the *top-p* parameter truncating the possible tokens to a minimal set with cumulative probability equal to or larger than p. In addition, there are multiple algorithms for token sampling all resulting in different generative behaviours [47].

LLMs can be fine-tuned for further capabilities using parameter efficient methods. One such method is to use Low-Rank Adaptation of LLMs (LoRA) [15]. LoRA works by freezing the LLM architecture weights and injecting low-rank trainable matrices A, B into the attention and projection layers across the transformer layers. This results in orders-of-magnitude fewer trainable parameters.

Despite their zero-shot generalization strengths, LLMs have a few critical pitfalls. Even though they are capable of providing an answer to all prompts, their fluent generations can be non-factual. This behavior is named colloquially as hallucination [16]. Lack of training data, decoding strategies, generation parameters can cause hallucinations and currently there are no definitive ways to completely prevent non-factuality. Furthermore, the models can have biases [9] towards certain responses, and broader ethical risks. Methods for grounding LLM generations with factuality and helping LLMs report about their consistency and certainty are active fields of research.

LLMs can be augmented with vision by use of various methods and function as Large Vision Language Models (LVLMs). One method of granting vision to LLMs is the LLaVA paradigm [30]. LLaVA and other similar approaches use a pre-trained

vision encoder such as BLIP [26] to encode images and transform the outputs into the embedding space of the LLM vocabulary. This way, input image tokens are consumed no differently than text tokens and the meaning of the image is conveyed via encoding.

LVLMs enable a series of tasks that are useful in medicine [35]. LLaVA-Med is one such biomedical LVLM allowing for open-ended biomedical image dialogue. Multimodal LLMs are rapidly developing for more medical use-cases [1] while their limitations, and especially the hallucination behavior still posing as caveats for readiness. Uncertainty quantification and hallucination mitigation research needs to be carried out to address these limitations and make LVLMs suitable for autonomous clinical decisions.

2.2 RaDialog: An LVLM for X-Ray Reports

RaDialog [41] is an LVLM trained to generate radiology reports from chest X-ray images optimized for clinical correctness. It supports interactive and dialog-based tasks such as report correction, impression generation, binary question-and-answer (QA) on findings, summarization, translation, easy language explanations. The model enables human-in-the-loop diagnostic processes reducing mental load and increasing report quality.

RaDialog uses a novel dual-branch architecture which uses a visual feature extractor and a structured findings extractor in tandem to increase clinical correctness. The LVLM is prompted with the embedded image tokens, the extracted structured findings, and the instruction for the requested task. The underlying LLM is Vicuna-7B [54] fine-tuned efficiently via LoRA.

The base dataset used for RaDialog's supervised fine-tunings is MIMIC-CXR [20] which we examine in more detail in Section 2.6. The fine-tuning dataset enriches the MIMIC-CXR dataset by phrasing prompts for the different tasks RaDialog supports. Additionally, pseudo-ground-truth data for summarization, correction and easy language is used. During training, context dropping is used as an augmentation. This collection of augmentations ensures that the LLM avoids catastrophic forgetting [31] and over-reliance on text input. The model preserves its conversational characteristics and visual reasoning behavior after an intensive fine-tuning.

RaDialog outperforms preceding LVLMs in clinical efficacy, clinical correctness for generated reports, and is more preferred by radiologists. It is a strong candidate for medical decision making in the radiology branch thanks especially to its conversational performance.

2.3 Confidence Estimation and Calibration for LLMs

When using LLMs for critical tasks, it is crucial for users to understand how certain a prediction or more generally a generation is to be able to benefit from it. For this reason, uncertainty quantification has been an integral part of machine learning research. Traditional models and especially discriminative models trained for specific tasks have been reasonably reliable with respect to their decisions whereas uncertainty quantification methods over LLMs and foundations models in general are yet to achieve a local optimum of trust [13].

One way to approach uncertainty quantification is via confidence estimation. If a scalar value representing the certainty of a prediction can be estimated as the confidence of a model, accompanied by a high level of alignment between the estimated confidence values and the empirical accuracies of predictions, then users can immediately be notified regarding the usefulness of a prediction. The uncertainty quantification literature defines the technique as *confidence estimation*.

Measuring the uncertainty of an LLM via confidence estimation is a way of understanding how epistemically certain a model is regarding a generation. These estimates can help professionals detect and mitigate hallucinations. Incorporating confidence estimates to clinical reports or answers generated by medical LVLMs can greatly help clinicians asses the reliability of the models, especially in the radiology domain where various ways of expressing an analysis correctly are possible [58]. Within the scope of this thesis we turn our attention to methods which can estimate the absolute confidence values for freeform texts sampled from generative models.

Confidence estimation methods for LLMs are still a developing field of research, as previously well established confidence estimation methods are difficult to apply to LLMs. The main challenges stem from the large output space of LLMs compared to discriminative models and various generations being able to capture semantically equivalent meanings [10]. Recent studies addressing these challenges can essentially be categorized into two main directions: white-box methods and black-box methods.

2.3.1 White-box Confidence Estimation Methods

White-box methods estimate the confidence of an LLM over a generation either via accessing the token logits at the last layer of the LLM or the entire hidden states of the LLM.

Logit-Based Methods

These methods use the token-level probabilities or entropies of generated sequences to form a final confidence score. Methods which digest the entire sequence of generated tokens can transform the logits of the tokens to a single scalar value representing confidence.

One such method is to use the average or the maximum token negative log-probability or token entropy within a sequence to quantify the uncertainty [32]. We can reformulate the average token negative log-probability term which originally acts as a proxy for uncertainty quantification to instead reflect the dual term: estimated confidence. Variations of this method include incorporating the sequence length to normalize the estimates across sequences of different lengths [36].

Equations (2.3.1) and (2.3.2) describe the two dual formulations of uncertainty quantification and confidence estimation, respectively, via probabilities of tokens in a sequence with p_{ij} denoting the probability of the j-th selected token for sequence i among all generated samples. Deriving the confidence estimation dual of the entropy-based uncertainty quantification is similar.

$$Avg(-logp) = -\frac{1}{J}\sum_{j} log(p_{ij}) \tag{2.3.1}$$

$$Avg(p) = \frac{1}{|J|}\sum_{j} p_{ij} \tag{2.3.2}$$

Even though purely logit-based estimations of confidence values are straightforward and computationally efficient, they fall short of capturing semantics. Duan et al. [8] devise a method which focuses more on semantically more meaningful tokens such as nouns and verbs to account for this limitation. The method builds upon the semantic entropy (SE) idea [24] and is titled *Shifting Attention to Relevance* (SAR). *TOKENSAR* (Equation (2.3.3)) quantifies uncertainty by weighting the negative log-probability of a token z_i in a sequence s_j generated upon a prompt x.

$$TOKENSAR(s_j, x) = \sum_{i}^{|s_j|} -logp(z_i \mid s_{<i}, x)\tilde{R}_T \tag{2.3.3}$$

The weight is the normalized relevance $\tilde{R}_t$ measuring the relevance of the token within the sequence. It is calculated based on the difference between the original sequence and the sequence with the token removed from the sequence. Similarly in the *SENTSAR* metric, the uncertainty contribution of a sequence is weighted according to how relevant the sequence is to the collection of sequences it belongs to.

Even though incorporating semantic relevance and entropy elevate the naive logit-based methods, these methods introduce computational complexity to confidence estimation, especially because they rely on external models to compute relevance of tokens or sequences.

A logit-based method which reduces computational complexity while preserving the usage of semantics is the P(True) method [21]. The method asks the model to evaluate its latest response by prompting the model with a certainty inquiry: "Is your proposed answer True or False?". This approach relies on the preliminary study which shows that LLMs are more likely to be accurate when questions are formed in binary question-answer (QA) form. Even for freeform generations, the LLM can be prompted with a binary QA question to reveal its confidence about its own answer. Upon the certainty inquiry, the softmax probabilities obtained for token *True* and token *False* are extracted from the last token logits of the LLM. The $P(True)$ and $P(False)$ probabilities are normalized to add up to 1. Finally, the $P(True)$ value is taken as the confidence estimate of the model for its latest response.

Internal-State-Based Methods

Some white-box confidence estimation methods require complete access to the hidden states of the generating LLM. These methods vary from fitting the hidden states and output states of the LLM to statistical distributions [43] to using a trained probe [2], [27], [21] which learns the confidence value via optimizing for a minimum difference between the predicted confidence value and the empirical accuracy.

Ren et al. [43] fit the input and the output sequences from a training set to a Gaussian distribution and use the Mahalanobis distance metric to compute divergence of an inference time input-output sequence pair with respect to the distribution. The distance metric indicates how out-of-distribution an inference time datapoint is which in turn can be used to indicate the confidence of the model. While promising out-of-distribution detection, this method cannot generalize to multiple tasks and ensure uncertainty quantification without comprising on LLM generalizabilities.

Azaria and Mitchell [2] devise a trained probe method named *Statement Accuracy Prediction, based on Language Model Activation* (SAPLMA) by training a feedforward neural network with a few layers appended to one of the hidden layers of an LLM. Using the selection of the hidden layer which propagates information to the SAPLMA network as the control parameter, they perform multiple trainings. The training results find that information from certain layers processed through the SAPLMA network achieve a confidence estimate which highly correlates with empirical accuracy of statements generated by the model. Several other trained probe methods make use of the hidden states in similar fashions, with one method using unsupervised training to

map hidden states to binary QA answer probabilities [5].

Limitations

Even though white-box confidence methods are mathematically rigorous and show considerable correlation with empirical accuracy [17], many of these methods require additional computations, full or limited access to the hidden states or logits of a model, and get costlier as natural language conversations proceed. While providing strong baselines to confidence estimation methods, they are not ideal in usage with LLMs [17] which have a number of parameters in the scale of billions.

2.3.2 Black-box Confidence Estimation Methods

Black-box methods presume no access to the output logits or the internal state of LLMs. They estimate the confidence of a model via purely external methods and in a post-hoc fashion.

Verbalized Confidence Methods

Verbalized methods use the in-context learning properties of LLMs to instruct the model to elicit a linguistic confidence value that can be directly parsed from the generated response. The strength of these methods lie in the fact that they require no additional models or statistical methods to calculate the estimate and the confidence estimation process is intuitively blended in the predictive generation process. Since the confidence estimation is fully conditioned on the input prompt, prompting strategies are the main way to improve the alignment of estimates with empirical accuracy.

Xiong et al. [57] propose five prompting strategies with increasing complexity:

- A *vanilla* prompt which simply instructs the model to generate a confidence value alongside is answer
- A *chain-of-thought* (CoT) prompt [23] encouraging the model to think step-by-step before finalizing the answer and the confidence value
- A *self-probing* prompt which asks the model to analyze how likely an answer could be given a question
- A *multi-step* prompt which asks the model to break down the problem into steps and aggregate the confidence values generated for each step
- A *top-k* prompt which asks the model to present its top-k best guesses and assign confidence values to these guesses

The selected prompting strategy is followed by a sampling strategy which samples generations from the model given the same prompt and an aggregation strategy which combines the multiple answers and confidences verbalized by the model into a final answer and a confidence estimate. Even though the CoT and self-probing prompts function better than the other prompting methods, verbalized confidence methods generally have the chronic issue of uncalibrated confidence estimations with models generating overly confident estimates [57].

Consistency Based Methods

Consistency based methods hypothesize that a generation under examination is more likely to be inaccurate if given the same input prompt, the LLM generates sequences which diverge away from the original generation. Complex methods such as using eigenvalues of similarity matrices between generations exist in the literature [29]. On the other hand, a more simplistic but powerful consistency method called *SelfCheckGPT* by Manakul et al. [32] was proved to be in high correlation with human confidence scorings.

SelfCheckGPT uses five different consistency metrics to evaluate how certain an original generation is compared to stochastic samplings performed using the same input prompt resulting in the original generation. Among these metrics, the ones which are particularly low-resource, sample-efficient and straightforward to implement are consistency via BERTScore [59] and consistency via self-checking prompting of the generating model. The method formally calculates the inconsistency S of a generation upon which the estimated confidence is trivial to obtain by using $1 - S$.

In SelfCheckGPT the method, given a generated response R conditioned on input prompt x, N further stochastic responses $(S_1 \dots S_N)$ are sampled conditioned on x. For each sentence r_i of the original response R, an inconsistency score $S(i) \in [0, 1]$ is calculated.

Using BERTScore, the inconsistency score S_{BERT} is formed by averaging the BERTScore similarity of each sentence $r_i \in R$ to the likeliest sentence found in any sentence s_k^n of any stochastic response S_n, and then subtracting this average from 1 (Equation (2.3.4)). If for all or most sentences in the original response R there exists among all the sampled stochastic responses a sentence which is reasonably similar, then the BERTSCORE inconsistency score is low.

$$S_{\text{BERT}}(i) = 1 - \frac{1}{N} \sum_{n=1}^{N} \max_{k} \mathcal{B}(r_i, s_k^n) \tag{2.3.4}$$

Using SelfCheckGPT via prompting, the LLM is prompted for each sentence $r_i \in R$ and for each stochastic sample S_N to evaluate whether the sentence r_i is supported

by context S_N. The LLM's responses are mapped to scalar values: *Yes* $\mapsto$ 0, *No* $\mapsto$ 1, $N/A \mapsto 0.5$. The inconsistency $S_{Prompt}(i)$ of a sentence is the average of these scalars x_i^n as described by Equation (2.3.5).

$$S_{\text{Prompt}}(i) = \frac{1}{N} \sum_{n=1}^{N} x_i^n \tag{2.3.5}$$

Regardless of which exact inconsistency metric is used, using the inconsistency score $S(i)$ of each sentence of the original response the overall confidence for the generation can then be estimated as $1 - \sum_i S(i)$.

Surrogate Model Based Methods

Surrogate models are used to obtain white-box access to logits or internal states of tokens generated by a black-box model which only provides access to its generations. Sequences generated by the black-box model are passed through the open-access white-box model and further white-box methods are applied to estimate the confidence. One such study estimates white-box confidence values using Lllama 2 [52] for generations obtained from a closed-source LLM, GPT-4 [37].

Limitations

Black-box methods show promising results given their limited resources compared to white-box methods. They incorporate the confidence estimation to the generative process. Especially with verbalized confidence methods, confidence estimation becomes an emergent part of any generation without hindering the performance or the speed of a natural conversation. However, compared to white-box methods, black-box methods are usually non-negligibly less calibrated.

2.3.3 Confidence Calibration

Confidence Calibration Techniques

Confidence calibration is the process of aligning the estimated confidences $\hat{p}$ for predictions obtained from a model with the empirical accuracies p^* of the predictions. The process relies on two main givens. It requires a way to estimate the confidence of a model and a way to improve the estimation method such that the estimates align with the true accuracy.

If a confidence estimation method achieves a high correlation between the predicted confidence values and the empirical accuracy of the generations then the confidence estimation method is said to be *calibrated*. It is possible to improve the confidence

calibration of a confidence estimation model via training the generating model, auxiliary models or by applying statistical methods [22].

Calibrating Expressed Confidences

Recent studies show increasing focus towards improving the calibration of confidences expressed by LLMs, especially for open-ended conversations. These methods illustrate a growing shift from probabilistic calibration to behavioral and linguistic calibration, where confidence becomes a learnable, interpretable aspect of model communication.

One promising method is *learning from past experiences* [14]. The method suggests fine-tuning models to align their confidence estimates with their historical performance, allowing them to express uncertainty depending on prior success and failures rather than pure probabilistic output. The study demonstrates consistent calibration improvements across question-answering tasks, highlighting that confidence can be effectively learned as an auxiliary communicative ability rather than an emergent side-product of likelihood scores.

Another approach is to use *linguistic calibration* [33], which adjusts the verbal expressions of certainty such as modal verbs or intensifiers to reduce the model's inclination towards overconfidence. The method aligns the communication style of the model to be more reflective of its accuracy.

Confidence Calibration Metrics

The main metric commonly used to measure confidence calibration is the Expected Calibration Error (ECE) [13]. Predictions of a model are grouped into M disjoint bins $\{B_m\}_{m=1}^{M}$ with each bin representing an estimated confidence. The mean accuracy of the predictions in a bin $acc(B_m)$ and the confidence value represented by the bin $conf(B_m)$ are used to calculate ECE for a set of N samples of. ECE is the weighted average of bin-wise absolute differences between accuracy and confidence. (Equation (2.3.6)).

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} \left|\mathrm{acc}(B_m) - \mathrm{conf}(B_m)\right| \tag{2.3.6}$$

Calibration curves are utilized to plot the predicted confidence bins against the mean accuracy of the bins (Figure 2.1). Perfect calibration is observed when the identity line $x = y$ is achieved. Deviations above the the identity line indicate under-confidence and under the line over-confidence. Calibration curves are the visual dual of ECE and the two tools represent the same type of calibration analysis.

ECE has has known limitations such as being sensitive to the binning method. Improved extensions to ECE are present in the literature [10]. However, ECE and

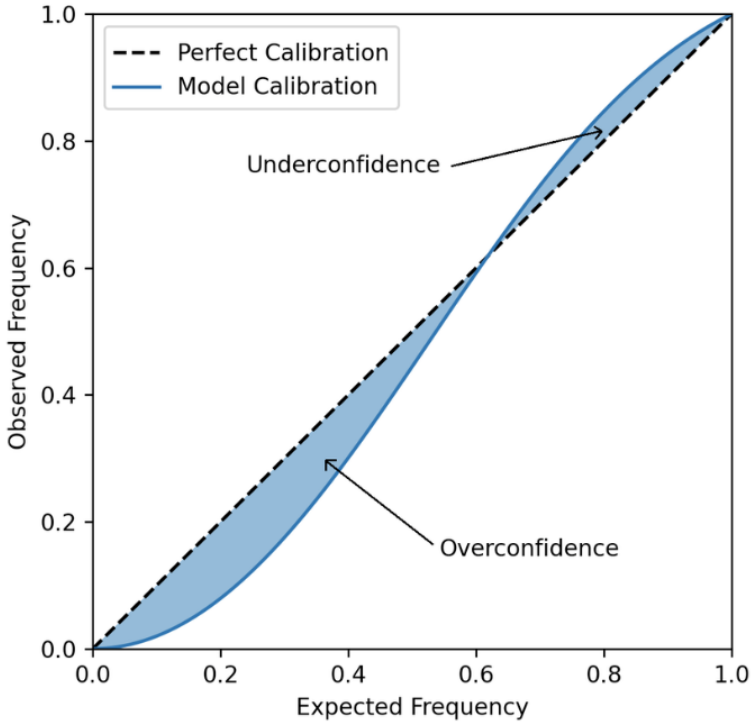

Figure 2.1: A calibration curve visualized by Ledda et al. [25]

calibration curves are still a strong baseline to start studying the calibration of a confidence estimation method.

2.4 Reinforcement Learning

Reinforcement Learning (RL) is a training method used traditionally in game-like settings to address decision problems by teaching agents to learn from an environment. The agent interacts with an environment via actions and receives rewards from it. Based on the feedback of the environment, the agent develops an understanding of encouraged actions and discouraged actions. The RL framework is formally a Markov Decision Process (MPD) where the agent navigates states of an environment with state transitions occurring after each action taken by the agent [51].

2.4.1 Fundamental Concepts of RL

An RL agent is described completely by a policy function π. At each time step t, the policy function π samples action a_t from the space of possible actions A given s_t the current state of the environment out of all possible states S. The selected action a_t triggers a state transition to s_{t+1} and the policy receives a scalar reward r_{t+1} from the environment. The state-action pairs (s_t, a_t) experienced by the policy for $\forall t \in T$ form a *trajectory*. The rewards collected via traversal of a given trajectory τ is called the *cumulative reward* and notated $R(\tau)$. For episodic tasks with trajectories of finite length we can estimate the *expected return* $J(\pi)$ for a given policy. The optimal policy function π^* maximizes the expected return. These essentials concepts of RL are described in Equations (2.4.1) through (2.4.6).

$$\pi_\theta = P(A|s_t;\theta) \quad \forall t \in T \qquad \text{(Stochastic Policy)} \tag{2.4.1}$$

$$\tau = \{(s_0, a_0), (s_1, a_1)...(s_n, a_n)\} \qquad \text{(Trajectory)} \tag{2.4.2}$$

$$P(\tau \mid \pi) = \prod_{t=0}^{T-1} P(s_{t+1} \mid s_t, a_t)\pi(a_t \mid s_t) \qquad \text{(Probability of a Trajectory)} \tag{2.4.3}$$

$$R(\tau) = r_1 + r_2 + \cdots + r_t + \cdots r_n \qquad \text{(Return or Cumulative Reward)} \tag{2.4.4}$$

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[R(\tau)\right] = \sum_{\tau} P(\tau \mid \pi)R(\tau) \qquad \text{(Expected Return)} \tag{2.4.5}$$

$$\pi^* = \arg\max_{\pi} J(\pi) \qquad \text{(Optimal Policy)} \tag{2.4.6}$$

As a helper function a *value function* is defined. This function maps a state s_t or a state-action pair (s_t, a_t) to a scalar value indicating the worthiness of being in a state or the worthiness of being in a state and taking a specific action. For each state or state-action pair, the state-value function V_π or action-state-value function Q_π outputs the expected return for all possible trajectories the policy can traverse given that the agent starts at state s_t or starts at state s_t and takes action a_t. The two types of value functions are described by equations (2.4.7) and (2.4.9).

$$G_t = r_{t+1} + r_{t+2} + \cdots + r_n \qquad \text{(Rewards-to-go)} \tag{2.4.7}$$

$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid s_t = s] \qquad \text{(State-Value Function)} \tag{2.4.8}$$

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid s_t = s, a_t = a] \qquad \text{(Action-State-Value Function)} \tag{2.4.9}$$

The value function can be used as a *surrogate function* to learn replacing the need for the direct training of the policy itself. It can also function as a *critique* to evaluate a policy's actions while training policy directly.

2.4.2 Policy Optimization Methods

Three main methods of finding the optimal policy are prevalent in the RL literature. These methods are **Policy-based Learning**, **Value-based Learning**, **Actor-Critic Learning**, the latter comprising a blend of techniques from the former two. In addition to the these three main methods, a variety of implementation specifics are found in various studies. For brevity, we list the most common implementation varieties and focus on the ones relevant to the usage of RL in the training of LLMs.

1. **Policy-based Learning** vs. **Value-based Learning** vs. **Actor-Critic Learning**: With policy-based methods, the policy function π is directly trained via updates on its parameters. Value-based methods train the value function, e.g. $Q_\pi(s, a)$, and the policy is then defined as a simple rule based function taking the action with the highest value. Actor-critic methods make use of direct training of the policy similar to policy-based methods, while incorporating the value function as a critique to the environment reward.

2. **Stochastic Policy** vs. **Deterministic Policy**: A stochastic policy samples actions from a probability distribution $\pi_\theta(a|s)$ [51], similar to the probability distribution over the LLM vocabulary for the next token, given the previous tokens. When deterministic, a direct mapping $s \mapsto a$ is known to the policy [51]. An LLM which samples tokens probabilistically can be directly interpreted as a stochastic policy which takes *token* actions.

3. **On-policy** vs. **Off-policy**: On-policy optimization methods samples actions from a continuously learning policy on the fly, making the training a more agile process at cost of loss of stability. The loss of stability can be mitigated via usage of trust regions [44]. Off-policy optimization methods rely on the reuse of a collection of completely traversed trajectories, and updates the policy at a single run [28].

4. **Model-based** vs. **Model-Free**: A model-based training of a policy assumes that the policy has the knowledge regarding the state transition dynamics of the

environment, i.e. the distribution expressed by $P(s_{t+1} \mid s_t, a_t)$ for any time-step. [34] A model-free training of a policy assumes no such knowledge, and the only interface the policy has to the environment is the reward attained.

Table 2.1 summarizes these variations. RL lends itself to the training of LLMs in a straightforward fashion when we interpret the LLM itself a stochastic policy taking *token* actions, with parameter updates happening via on-policy stochastic gradient steps. A model-free approach is fitting, given that a user's prompts are impossible to be modeled. Following the actor-critic learning method makes it possible for the model to learn both via rewards from the environment and an accompanying value model criticizing the model's generations.

Implementation Category	**Types**
Policy	**Stochastic** / Deterministic
Update Cycles	**On-policy** / Off-policy
World Model	Model-based / **Model-free**
Training Method	Policy-based / Value-based / **Actor-Critic**

Table 2.1: A summary of common implementation variations of RL with the techniques more often used in RL training of LLMs in **bold**.

2.4.3 Stochastic Policy Optimization via Actor-Critic Learning and Policy Gradients

The objective of RL is to maximize the the expected return $J(\pi)$ as defined by Equation (2.4.6). Given a differentiable stochastic policy π_θ parametrized by θ, it is possible to take gradient steps towards maximizing $J(\pi)$. Equations (2.4.10) through (2.4.12) describe how the policy gradient is approximated by $\hat{g}$ over a set of sampled trajectories $\mathcal{D}$ and utilized to optimize a policy directly. Derivation of Equations (2.4.11) and (2.4.12) follow respectively directly from the Policy Gradient Theorem [50] and the Monte-Carlo Gradient Estimator [56] from the *REINFORCE* algorithm which is essentially the first policy-gradient algorithm in the literature.

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_\theta)\big|_{\theta_k} \qquad \text{(Gradient Update)} \tag{2.4.10}$$

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) R(\tau)\right] \qquad \text{(Policy Gradient)} \tag{2.4.11}$$

$$\nabla_\theta J(\pi_\theta) \approx \hat{g} = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) R(\tau) \qquad \text{(Monte-Carlo Est.)} \qquad (2.4.12)$$

The Policy Gradient Theorem and the REINFORCE approximation to the policy gradient give a complete algorithm to train a stochastic policy. The approximation is unbiased, meaning that the approximated gradient converges to the true gradient given a high number of training steps. However, the approximation experiences high variance in practice. Two modifications help reduce the variance considerably: ensuring causality of rewards-to-go at each time step, and introducing a baseline reward so that the sign of the reward becomes irrelevant.

In the Monte-Carlo Gradient Estimator from the REINFORCE algorithm, the overall cumulative reward is used for all time-steps in a trajectory, including rewards collected before a given current time step t. In practice, this causes the gradient steps to have high variance in the gradient directions, disrupting the training. To address the issue, only G_t, the *rewards-to-go* as defined in Equation (2.4.7), is used. This ensures causality, and reduces variation (Equation (2.4.13)). [51]

$$\nabla_\theta J(\pi_\theta) \approx \hat{g} = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) G_t \qquad (2.4.13)$$

The G_t introduced to the relation is the action-state-value function $Q_\pi(s_t, a_t)$ by definition as it is the expected return of the rest of the trajectory given the policy starts acting at state s_t and takes action a_t next.

To further decrease the variance, a baseline reward independent from the action a_t is defined [12]. Choosing the state-value function $V_\pi(s_t)$ as the baseline is natural and effective with a clear definition as to what the baseline-reduced rewards-to-go mean. The $Q_\pi(s_t, a_t) - V_\pi(s_t)$ subtraction is a well defined RL term called the *advantage function* [3] with notation $A_\pi(s_t, a_t)$. This term can intuitively be explained as the advantage of selecting a specific action given a specific state over all the possible actions in that state. This term ensures that the reward is not processed as-is in the gradient update but rather used in calculating the relative success of the policy. With the addition of the advantage term, the policy gradient finally takes its form in Equation (2.4.14).

$$\nabla_\theta J(\pi_\theta) \approx \hat{g} = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) A_\pi(s_t, a_t) \qquad (2.4.14)$$

The coupled usage of the Q and V value functions ascend the purely policy-gradient based learning method to an actor-critic method where a function or a model is used to estimate the advantage term, essentially approving or disapproving the selected actions at each step.

2.4.4 Proximal Policy Optimization

Proximal Policy Optimization (PPO) [46] is a policy optimization algorithm making several improvements to the actor-critic policy gradient method. The algorithm addresses the chronic sample inefficiency issue of policy gradients and the exploration-exploitation dilemma that hinders RL trainings. Even though the algorithm is agnostic to the underlying policy, value, and reward functions, it forms a strong basis for the usage of RL in training LLMs. The method proves to be powerful in RL training of LLMs in various studies [49] [18].

The Clipped Surrogate Policy Gradient Loss

In PPO, the policy is allowed to see the same batch of trajectories more than a few times, thus increasing sample efficiency and targeting the sample inefficiency problem of RL. However, for the policy to avoid overfitting the reward after repeating the same batch multiple times, the policy outputs after each gradient step is regulated with respect to the batch trajectories traversed by the policy before starting the gradient steps on the current batch. This method is called importance sampling. [42]

Sampling the trajectories and freezing them constitutes the short-living *off-policy* part of the otherwise on-policy PPO passes over training batches. We notate the policy we use for sampling the frozen trajectories as $\pi_{\theta_{old}}$. The policy gradient is multiplied with the *likelihood ratio* $\mathcal{R}_t(\theta)$ (Equation (2.4.15)) derived from the samplings from the online and the offline policies π_θ and $\pi_{\theta_{old}}$ ((2.4.16).

$$\mathcal{R}_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)} \tag{2.4.15}$$

$$\nabla_\theta J(\pi_\theta) \approx \hat{g} = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{T} \mathcal{R}_t(\theta) \nabla_\theta \log \pi_\theta(a_t \mid s_t) A_\pi(s_t, a_t) \tag{2.4.16}$$

This reformulation enables strong policy changes occurring during gradient steps to amplify the following gradient steps. In other words, if the probability of an *opinionated* action diverges considerably from the probability of that same action a few gradient steps ago, then the reward or the penalty received for that action is amplified so that the policy either explores more towards that *opinionated* direction or converges back to safer actions.

The PPO surrogate loss L^{PG} formulating this policy gradient in an differentiable value ready for backpropagation is given in Equation (2.4.17). The mathematical recovery of the loss from the gradient follows from the usage of the log-derivative trick and is omitted here for brevity. We note that this loss term is to be maximized unlike typical

losses in machine learning and when differentiated it gives back the $\hat{g}$ approximation in Equation (2.4.16).

$$L^{PG} = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{T} \mathcal{R}_t(\theta) A_\pi(s_t, a_t) \tag{2.4.17}$$

Having granted the policy a very emphasized freedom of exploration, PPO needs to employ a safety mechanism by clipping the surrogate loss so that extreme changes never occur. The clipped surrogate loss is defined in Equation (2.4.18). The clipped surrogate loss makes sure that when an action is advantageous, i.e. $A_t > 0$, then the gradients only exist if $r_t(\theta)$ is not greater than $1 + \epsilon$. Similarly, if an action is disadvantageous, i.e $A_t < 0$, then the gradients only exist if $r_t(\theta)$ is not smaller than $1 - \epsilon$. This behavior mimics the trust region [44] concept in RL. The clipped surrogate loss fully describes the policy's relationship with the reward in the PPO algorithm.

$$L^{\text{CLIP}}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{T} \min\Big(\mathcal{R}(\theta)\hat{A}_t,\ \text{clip}\big(\mathcal{R}_t(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_t\Big) \tag{2.4.18}$$

The clipped surrogate loss proves to be successful both in encouraging exploration while avoiding exploitation, and increasing sample efficiency via a few simple changes on the baseline policy gradient [46].

Advantage Estimation

The clipped surrogate policy gradient loss requires the advantage value for each time step in a trajectory. It is not trivial to calculate the true advantage term and impossible in some cases. Generalized Advantage Estimation (GAE) [45] is utilized to approximate the advantage recursively. Starting from the last time step, the approximate advantage term $\hat{A}_t$ can be estimated. The advantage term can be approximated using an independent or policy-bound critique capable of determining the value of each state. In PPO, a value head parameterized with its own parameters ϕ is usually appended to the policy and trained alongside the policy parameters θ.

Given a value function V_ϕ as critique, the temporal difference (TD) residual is defined as:

$$\delta_t \stackrel{\text{def}}{=} r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t), \tag{2.4.19}$$

Using the TD residual, the recursive relation in GAE is defined as:

$$\hat{A}_t = \delta_t + \gamma\lambda\hat{A}_{t+1} \tag{2.4.20}$$

The recursion begins from the last time step of the trajectory $t = |\tau|$, and bootstraps the TD residual and the advantage as $\delta_{|\tau|} = r_{|\tau|} - V_\phi(s_{|\tau|})$ and $\hat{A}_{|\tau|} = \delta_{|\tau|}$. Since the reward and the values propagate back to previous time steps via recursion, γ and λ are crucial hyperparameters alongside the parameters parametrizing the learnable value function V_ϕ.

Reward Consumption

Unlike traditional RL frameworks, PPO typically encourages a score which rewards an entire trajectory with a single scalar score instead of providing rewards for each time step. The terminal score S given to the entire trajectory is consumed as the reward by the last time step, and it diminishingly propagates through the recursive GAE relation to previous steps. In other words, the reward from the environment is never explicitly consumed by the time steps before the last. However, at each time step the policy under training π still receives an additional penalty based on the KL divergence of the action from the action that would have taken by a completely frozen reference policy π_{ref}. The reward for each time step can then be expressed as:

$$r_t = \begin{cases} S - \lambda_{KL} KL(\pi(a_t \mid s_t) \,\|\, \pi_{\text{ref}}(a_t \mid s_t)), & t = |\tau| \\ -\lambda_{KL} KL(\pi(a_t \mid s_t) \,\|\, \pi_{\text{ref}}(a_t \mid s_t)), & t < |\tau| \end{cases} \tag{2.4.21}$$

Providing a single terminal score S for an entire trajectory is especially a helpful technique when evaluating the generation quality of an LLM, since the complete generation needs to be seen in the reward allocation process. The KL divergence penalty ensures that the policy under training does not immensely diverge away from the original policy while exploring. This safety measure further contributes to PPO's novel solution to the exploration-exploitation dilemma.

The Overall PPO Loss

The overall PPO loss is used to train the policy itself and the value head. The clipped surrogate loss is used to update the parameters of the policy whereas the squared difference between the predicted value and the true value are used to calculate the value loss. The overall loss is described in Equation (2.4.22).

$$\mathcal{L}^{\text{PPO}}(\theta) = L^{\text{CLIP}}(\theta) - c_1 \big(V_\phi(s_t) - V_t^{\text{targ}}\big)^2 \tag{2.4.22}$$

2.4.5 RL for LLMs

The PPO algorithm has revolutionized the LLM domain by making it possible to train LLMs on human preferences. The bottleneck caused by the low number of samples capturing human preferences and the subjective nature of these preferences were mitigated by the advantages of the PPO algorithm over previous RL methods. Multiple studies trained an LLM as the environment granting rewards to generations of a larger LLM under training, based on human preferences [49], [60]. Qualitative human expectations such as non-toxicity, truthfulness, helpfulness from LLMs are quantitatively captured within the reward model's latent space. This novel reward model idea combined with the strong foundation established by the PPO algorithm brought LLM chatbots to a ready state for assisting a massive number of users with unpredictably various intents and interests. This paradigm is called Reinforcement Learning from Human Feedback (RLHF) [39].

General RL concepts lend themselves to RL for LLMs concepts in a straightforward fashion as listed in Table 2.2. Given the one-to-one mapping between the concepts, the PPO clipped surrogate loss and value loss are directly applicable to an LLM under training. The reward from the external reward model is the terminal reward propagating recursively to all steps starting from the last. The KL divergence penalty acts as the direct reward of non-terminal steps by comparing a generated token from the LLM under training to the reference LLM frozen before training commences.

2.5 The Rewarding Doubt Method

A PPO-based method named *Rewarding Doubt* [48] uses a novel reward mechanism to calibrate LLMs to verbalize accuracy-aligned confidence values. The method merges the confidence estimation process into the generation process successfully and challenges confidence estimation methods which require additional computations or external models.

Rewarding Doubt trains a general purpose LLM on question and answer (QA) datasets where each generated response can be evaluated against the ground truth as *True* or *False*; i.e. QA datapoints with binary accuracy values. Unlike the RLHF method, the reward signal guiding the PPO training does not come from an external reward model. The reward is a purely mathematically expressed function based on the verbalized confidence generated by the model and the accuracy of its generation.

The Rewarding Doubt reward R_{RD} is the binary-label cross entropy function limited to hard accuracy labels 1.0 and 0.0 and predicted confidence values $\hat{p} \in [0, 1] \subset R$ as described in Equation (2.5.1). The logarithm is stabilized via sufficiently small $\epsilon > 0$.

General RL Concept	Equivalent Concept in RL for LLMs
Stochastic Policy	The LLM Under Training, parameterized by θ
Critique (State-Value Function)	A Value Head appended to the LLM Under Training parametrized by ϕ
Trajectory	A single complete generation from an LLM, given a prompt
Environment	An external LLM trained for capturing human preferences
Reward	The scalar number produced by the external LLM upon evaluation of a generation from the LLM under training
Action	A single token generated by the LLM under training
Action Space	The entire token vocabulary of the LLM under training
State	All previously consumed tokens

Table 2.2: Mapping of RL concepts to equivalent concepts in RL for LLMs.

$$R_{RD}(accuracy, \hat{p}) = \begin{cases} \log(\max(\hat{p}, \epsilon)), & \text{if } accuracy = 1 \text{ (correct)} \\ \log(\min(1-\hat{p}, 1-\epsilon)), & \text{if } accuracy = 0 \text{ (incorrect)} \end{cases} \tag{2.5.1}$$

In essence, the reward governs a game of *hot and cold* where the model gets logarithmically better rewards as its predictions approach to the empirical binary accuracy of an answer. If the model predicts a confidence exactly with an absolute offset of 1.0 with respect to the binary accuracy; i.e. if it predicts a certain *False* when the generated answer is *True* or vice versa, then the model receives a severe penalty.

R_{RD} intuitively makes sense and furthermore is rigorously proven to satisfy the properties of a proper scoring rule. The model receives the highest expected return when it predicts confidence values matching the expected accuracy. In practice, R_{RD} proves to be state-of-the-art (SOTA) or near SOTA in confidence estimation compared to prominent white-box and black-box methods reviewed in Section 2.3.2. Furthermore, an LLM trained via PPO with reward R_{RD} can generalize reasonably well to out-of-distribution (OOD) datasets, and specifically medical QA datasets.

We note that R_{RD} is normalized to fit in range $[-1, 1]$ and whenever we refer to R_{RD} in this study, we always refer to the normalized final reward.

2.6 The MIMIC-CXR Dataset

The RaDialog study [41] uses the MIMIC-CXR dataset [20] to fine-tune the model via supervision. The MIMIC-CXR dataset is a collection of 277,835 chest X-ray studies with radiology images. The images are accompanied by structured labels indicating the existence of 13 possible categories of findings and a 14th category indicating the non-existence of any finding as presented by CheXpert [19], and a radiology report outlining the observations of the examiner. The radiology reports can be used as the groundtruth for generated reports while measuring the accuracy of generated reports by RaDialog in confidence calibration trainings. Figure 2.2 presents the findings section of an example report from the MIMIC-CXR dataset. RaDialog was trained only on the findings section of radiology reports for the report generation use-case.

> There is no focal consolidation, pleural effusion or pneumothorax. Bilateral nodular opacities that most likely represent nipple shadows. The cardiomediastinal silhouette is normal. Clips project over the left lung, potentially within the breast. The imaged upper abdomen is unremarkable. Chronic deformity of the posterior left sixth and seventh ribs are noted.

Figure 2.2: The findings section of a groundtruth radiology report from the MIMIC-CXR dataset.

3 Methodology

In our study we aim to fine-tune RaDialog to predict confidence values for the radiology reports it generates. We select reinforcement learning and specifically PPO as our training paradigm to calibrate the estimations. We start by extending our PPO library of choice to support multimodality. Then, we extend the Rewarding Doubt reward to support continuous accuracy values. Finally, we integrate the GREEN Score method [38] as the accuracy metric for generated radiology reports. This section presents the details of our methodology.

3.1 Preliminary Concepts for Confidence Calibration

Throughout our study we use integer confidence values $c \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ with higher c values representing higher confidence in a prediction's accuracy. We sometimes use a granular confidence set with $cg \in [0, 100] \subset Z$ but we always translate the granular confidence value to its non-granular match via dividing by 10 and rounding to the nearest integer meaning, e.g. a cg value of 54 maps to c value 5 and 77 to 8.

Each confidence value c has a bin B_c where we collect the accuracy of generations with a confidence of value of c. Each bin correlates to datapoints $(c, 1)$ through (c, c_n). The accuracies can be binary accuracies A_c or continuous accuracies G_c. In either case a confidence bin is represented by the integer confidence value c, and the mean accuracy of the bin μ_c. We formally indicate these relations in Equations (3.1.1) through (3.1.4).

Each datapoint we work on is accompanied by the predicted confidence $\hat{p}$ and the empirical accuracy p^*. The empirical accuracy is either one of the binary or the continuous accuracy. The integer confidence values and the predicted confidence have a one-to-one mapping with $p^* = c/10, \quad p^* \in [0, 1] \subset R$. We use c and $\hat{p}$ interchangeably depending on context, and similarly p_i^* and either one of A_j or G_j for any datapoint j.

$$B_c = \{A_{c,1}, A_{c,2}, \ldots, A_{c,n_c}\}, \quad A_{c,i} \in \{0, 1\} \tag{3.1.1}$$

$$\mu_c = \frac{1}{n_c} \sum_{i=1}^{n_c} A_{c,i} \tag{3.1.2}$$

$$B_c = \{G_{c,1}, G_{c,2}, \ldots, G_{c,n_c}\}, \quad G_{c,i} \in [0, 1] \subset R \tag{3.1.3}$$

$$\mu_c = \frac{1}{n_c} \sum_{i=1}^{n_c} G_{c,i} \tag{3.1.4}$$

3.2 Extending PPO and Rewarding Doubt for Multimodality

The PPO method presented in Section 2.4.4 can be implemented via PyTorch [40] since the method in essence is the backpropagation of a loss based on the reward towards model parameters. A library called Transformer Reinforcement Learning (TRL) [55] implements the core ideas of PPO in PyTorch with additional conveniences such as benefiting from sequence generation strategies already implemented in its parent library. Therefore, we use TRL as the core library for our PPO trainings.

The most recent TRL version that is compatible with RaDialog and used in our study does not support multimodality in the PPO implementation. We modify the TRL code such that it supports the multimodality logic of RaDialog. RaDialog embeds images into the text embedding space and represents the images as additional embedded tokens inserted into the input prompt where an image is referred to. Since the TRL library expects no further modifications on the input text prompt, RaDialog's modification of the prompt by adding additional image-bound tokens breaks the flow of PPO. We therefore indicate the presence of additional image tokens where necessary within the PPO algorithm, especially when the algorithm requires a forward pass through the LVLM or when it determines the masks separating the input prompts from the output text generated by the LVLM.

3.3 The Rewards

We require two different rewards for the RaDialog use-cases we focus on. The first use-case is Binary QA as discussed in detail in Section 4.1 with a use-case sample in Table 4.1. The second use-case is Report Generation as discussed in detail in Section 4.2 with a use-case sample in Table 4.4.

We use the Rewarding Doubt reward R_{RD} without any modifications for the confidence calibration of RaDialog for the Binary QA use-case which only has binary accuracy values. It is the first step to empirically validating the method in the multimodal domain with a heavily fine-tuned model such as RaDialog. However, for the freeform report generation use-case with continuous accuracy values, we modify or completely replace R_{RD}. This section presents the rewards we have designed for the report generation use-case.

3.3.1 The Cross Entropy Reward

We generalize on the cross-entropy reward used for generations with binary accuracy values in the Rewarding Doubt method [48]. Instead of constraining the reward function to regions where empirical accuracy value is 1.0 or 0.0, we allow for a continuous spectrum of empirical accuracies to be represented in the reward. This allows for the incorporation of a continuous accuracy metric such as the GREEN score [38] into the reward function. The generalized continuous cross-entropy reward R_{CE} is described by Equation (3.3.1) with p^* and $\hat{p}$ representing empirical accuracy and predicted confidence, respectively.

$$R_{CE}(p^*, \hat{p}) = p^* \log(\hat{p}) + (1 - p^*) \log(1 - \hat{p}) \tag{3.3.1}$$

This reward can directly be interpreted as the cross entropy loss with a soft label p^* and functions as a proper scoring rule since for each empirical accuracy value $\hat{p}$ it has its maximum at $p = p^*$ as proven by Stangel et al. [48]. Figure 3.1 shows the behavior of the reward for a few selected empirical accuracy values.

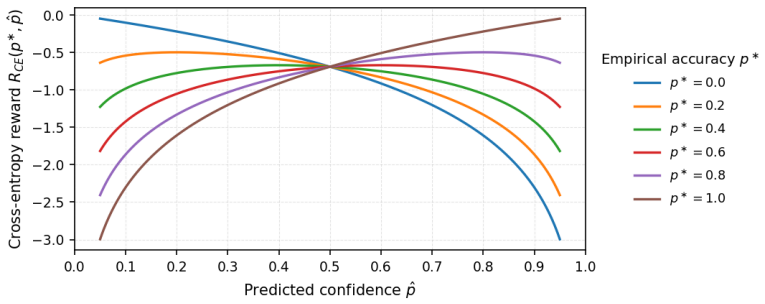

Figure 3.1: Cross-entropy reward plotted for selected accuracies with each line peaking when predicted confidence $\hat{p}$ is equal to empirical accuracy $p*$.

Normalizing and Scaling the Reward

The cross-entropy reward is unbounded from the below taking values in the range $[-\infty, 0]$. To account for the boundaries where the predicted confidence is equal to 0.0 or 1.0 causing the logarithms in equation (3.3.1) to take negative-infinity values, we define a sufficiently small parameter $\epsilon > 0$ (Equations (3.3.2), (3.3.3)). The exact value of ϵ turns out to be a hyperparameter since it changes the resolution of the reward vastly with small ϵ values making the reward indiscernible across different empirical accuracy and predicted confidence values.

$$R_{CE}(p^*, 0) = p^* log(\epsilon) \tag{3.3.2}$$

$$R_{CE}(p^*, 1) = (1 - p^*) log(\epsilon) \tag{3.3.3}$$

We further process the reward to normalize it into the $[-1, 1]$ range. This type of normalization is typically useful to ensure training stability by avoiding rewards with large absolute values which may persistently trigger clipping within the PPO optimization steps. We define three ways of normalizing the reward: *shifted*, *tanh*, *logistic*.

The *shifted* normalization is a simple affine transformation which maps the lower and higher boundaries of the reward function to -1 and 1, respectively. Affine transformations preserve the optima properties of the function they transform. Therefore, reward maxima are still guaranteed at points where $\hat{p} = p^* \; \forall p^*$. The shifted reward $R_{SHIFTED}$ is defined in Equation (3.3.4).

$$R_{SHIFTED} = 1 + \frac{2 * [R_{CE} - min(R_{CE})]}{max(R_{CE}) + min(R_{CE})} \in [-1, 1] \tag{3.3.4}$$

The *tanh* and *logistic* normalizations similarly bound the reward within the $[-1, 1]$ range via a non-linear transformation with much less sensitivity to the ϵ hyperparameter. Non-linear transformations can perturb the optima of the function they transform. However, since both *tanh* and *logistic* functions are monotonous, it directly follows that the optima occur at the same predicted confidence values $\hat{p}$. This relation is formally described in Equation (3.3.5) with $f_{monotonous}$ representing either one of the *tanh* and *logistic* functions.

$$\arg\max_{\hat{p}} f_{monotonous}\big(R_{\mathrm{CE}}(p^*, \hat{p})\big) = \arg\max_{\hat{p}} R_{\mathrm{CE}}(p^*, \hat{p}), \quad \forall\, p^* \tag{3.3.5}$$

For sake of hyperparameter tuning completeness, we define for the non-linear transformations a hyperparameter called *squash scale* which mimics the effect that ϵ has for affine transformations. The squash scale s controls how distinguishable the reward is across different empirical accuracies p^* for a fixed predicted confidence value $\hat{p}$. We apply s to R_{CE} before we transform it via one of the non-linear normalizers. Equations (3.3.6) through (3.3.8) define the rewards R_{TANH} and $R_{LOGISTIC}$.

$$R_{SQUASHED} = \frac{R_{CE}}{s} \tag{3.3.6}$$

$$R_{TANH} = tanh(R_{SQUASHED}) \tag{3.3.7}$$

$$R_{LOGISTIC} = 2 * (\frac{1}{1 + e^{(-R_{SQUASHED})}}) + 1 \tag{3.3.8}$$

After the reward is fit within the $[-1, 1]$ range, we can easily manipulate it by multiplying it with a scalar k to determine the best scale for the reward. We use $R_{CE-FINAL}$ as the PPO reward signal.

$$R_{CE-FINAL} = kR \ , R \in \{R_{SHIFTED}, R_{TANH}, R_{LOGISTIC}\} \tag{3.3.9}$$

Hyperparameters ϵ and s controlling the spread of the reward can have high impact on the training as shown empirically in Section 4.2.4. Smaller ϵ and larger s values make the attainable reward for different predicted confidences $\hat{p}$ at a given empirical accuracy level p^* similar to each other. This can help the model explore more confidence values but hinder the understanding of the model regarding different empirical accuracy levels. Larger ϵ and smaller s values make the rewards at different empirical accuracy levels more distinguishable. This can help the model understand the different level of accuracies more easily but discourage the model from exploring different confidence predictions. The impact of these hyperparameters on the reward are plotted in Figure 3.2.

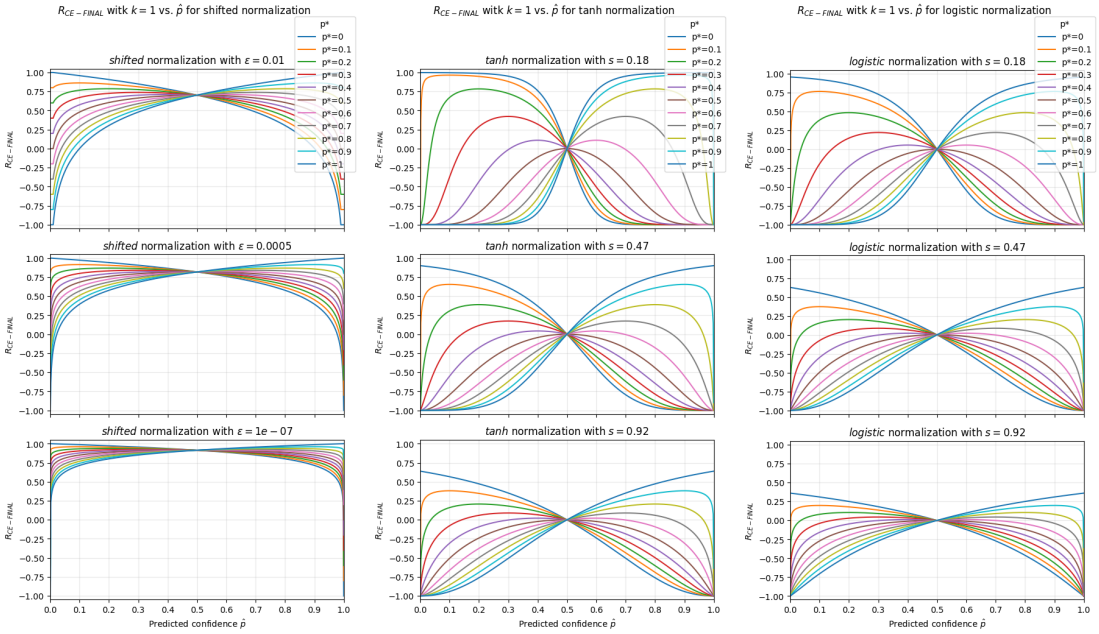

Figure 3.2: $R_{CE-FINAL}$ for (left) *shifted* normalization across ϵ values and (middle), (right) for *tanh* and *logistic* normalizations across s values.

3.3.2 The Quadratic-Blend Reward

While R_{CE} hyperparameters ϵ and s help regulate the exploration-exploitation trade-off of RL, they still cannot mitigate the crucial pitfall of the cross-entropy reward. R_{CE} essentially encourages the model to *bet* on the confidence value equal to the empirical accuracy of a generation. This presumes that the model develops an emergent understanding that binds empirical accuracy to predicted confidence early on in the PPO training. Only when this presumption holds the model can find the optimal reward as training progresses.

We propose a novel reward called the *quadratic-blend* reward R_{QB} that aims to teach the model difficulty levels from the very beginning of the training. It is based purely on the distance between the predicted confidence and the empirical accuracy for an inference datapoint. The reward is increasingly punishing as the predicted confidence strays away from the empirical accuracy of the generation.

Defining d as the absolute distance between a confidence prediction and the empirical accuracy, we define R_{QB} in Equation (3.3.11). We use $\beta \geq 0$ as the control parameter to how sensitive the reward is to d with lower β values causing harsher reductions from the reward. We derive $R_{QB-FINAL}$ by multiplying it with a scalar k which is the only other hyperparameter for this reward alongside β. The reward is already bounded within $[-1, 0]$ by definition and requires no further normalizations.

$$d = |\hat{p} - p^*| \tag{3.3.10}$$

$$R_{QB} = -d^2 \cdot \frac{1 + \beta d^2}{1 + \beta} \tag{3.3.11}$$

$$R_{QB-FINAL} = k * R_{QB} \tag{3.3.12}$$

The most differentiating property of R_{QB} compared to R_{CE} is that the absolute value of the reward is not attenuated by the empirical accuracy. The reward is based only on how close a confidence prediction is to the empirical accuracy no matter how accurate the generation is. The *betting* based concept of R_{CE} is replaced with a game of *hot and cold* where the model receives higher rewards if its prediction is *hotter* or closer to the true empirical accuracy. We hypothesize that the removal of empirical accuracy dependence makes R_{QB} a stronger and more natural candidate for RL based confidence calibration. Figure 3.3 shows the behavior of the reward for two set of empirical accuracies. Figure (5.6) shows the effect of the β hyperparameter on the reward.

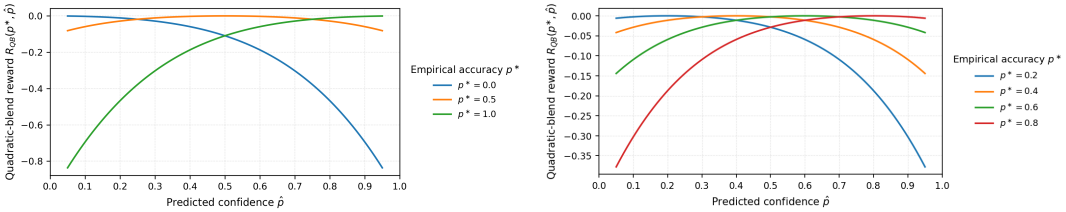

Figure 3.3: (left) R_{QB} for empirical accuracy levels $0.0, 0.5, 1.0$ (right) R_{QB} for empirical accuracy levels $0.2, 0.4, 0.6, 0.8$.

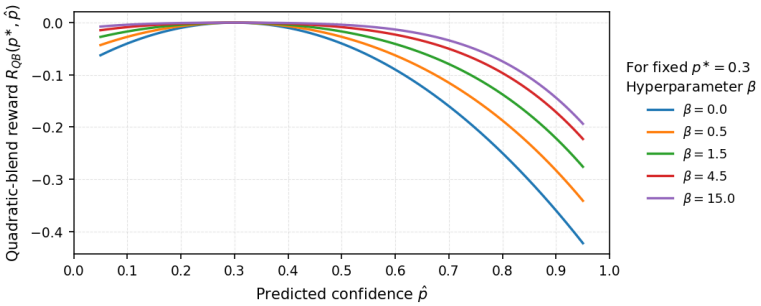

Figure 3.4: The harshness of R_{QB} via increasing β values for empirical accuracy fixed at $p^* = 0.3$.

3.4 Radiology Report Evaluation

We determine GREEN Score [38] as an appropriate accuracy metric for our use-case given its correlation with human experts and its normalized nature fitting empirical accuracy p^* into range $[0, 1]$. It also carries the benefit of a generated radiology report at a single pass, making it simple to reward a complete generation based on the accuracy of the entire trajectory.

We note that we have considered using RaDialog itself as an evaluator of its own responses, by applying the $P(True)$ method [21]. This would be equivalent to helping the model verbalize its internal confidence estimation but would not guarantee a calibration.

3.5 The Fine-tuning Set-up

In all of our fine-tuning experiments targeting RaDialog, we use LoRA adapters as the main trainable and merge them with our model after trainings are complete. The LoRA configuration we use is as follows:

- Target modules: $gate - proj$, $v - proj$, $o - proj$, $k - proj$, $down - proj$, $up - proj$, $q - proj$
- $r = 128$
- $LoRA - \alpha = 256$
- $LoRA - Dropout = 0.05$

4 Experiments

In this chapter, we present the experiments we have performed using PPO to calibrate RaDialog on two selected use-cases it is trained for: answering questions which have answers with binary accuracy values *True* and *False* and generating freeform radiology reports which are evaluated via the GREEN Score.

4.1 Binary QA

We start by experimenting with the Rewarding Doubt method on the binary question-answer (QA) behavior of RaDialog, since the original study proves its efficacy on generations evaluable via binary accuracy values. The Binary QA use-case is simpler with respect to evaluating accuracy of a prediction and serves as a validator of the extension of the PPO pipeline to a multimodal model and the Rewarding Doubt reward to a domain-specific LLM such as RaDialog.

The Binary QA use-case is essentially just asking RaDialog whether one of the possible 13 CheXpert findings [19] occur in a given input image. We expect RaDialog to response with a simple *Yes* or *No* token followed by a confidence estimate regarding its own response. The input prompt and the expected RaDialog response is given in Table 4.1.

Step	**Content**
Input Prompt	{input image} You are to act as a radiologist and answer the following question: Is the following finding visible in the given X-ray image: Lung Lesion?
RaDialog Response	No, there are no indications of Lung Lesion. {"confidence": 6}

Table 4.1: Input prompt and expected response example for the Binary QA use-case.

We perform multiple experiments using a dataset we create specifically for training and testing for the Binary QA use-case. We start by PPO training RaDialog to report overconfidence as a sanity check of our pipeline. Then we start confidence calibration via a naive application of the Rewarding Doubt pipeline with R_{RD}, and later introduce

forced exploration and granular confidence values. For each model trained via PPO for the Binary QA use-case, we calculate ECE as our primary calibration metric.

4.1.1 Datapoint Sampling and Dataset Creation

We use the MIMIC-CXR dataset as our basis for training and testing. For our use-case, we first remove all datapoints which do not have any positive or negative indications regarding any one of the CheXpert findings or have only ambiguous findings. We then examine the datapoints with respect to the occurrence of the 13 possible finding types and find that the dataset is heavily imbalanced (Figure 4.1). We do not want the distribution of the dataset to have an impact on our training by introducing a bias towards certain findings or occurrence label *True* or *False*.

We sample 1,500 positive and 1,500 negative datapoints from the train split of MIMIC-CXR for each CheXpert finding type, and upsample if a finding type has not enough datapoints. In the end we achieve a balanced training with each finding type having exactly 3,000 datapoints evenly distributed between positive and negative samples (Figure 4.1). We apply the same procedure to validation and test sets as well. Our sampling technique yields an unbiased dataset for all splits.

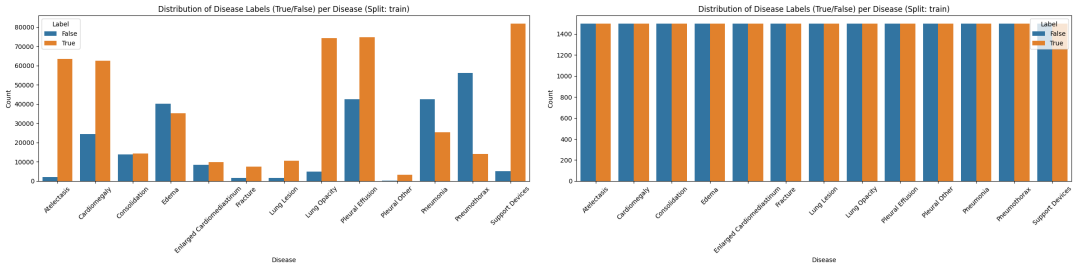

Figure 4.1: (left) Distribution of findings either unambiguously present or unambiguously absent within the MIMIC-CXR train split. (right) Distribution of findings after balancing with respect to positive and negative samples.

It is crucial that not only the groundtruth data is diverse and balanced enough, but also the model can see various outcomes during the PPO training. Therefore, we further examine the performance of RaDialog on our training set. We calculate RaDialog's accuracy, precision, and recall on the training dataset for each finding type (Figure 4.2).

We observe that RaDialog has comparable accuracies across finding types. On the other hand, it is hesitant to classify datapoints as positives for most finding types. This means that there are comparatively more difficult datapoints in our training dataset and we have achieved a dataset capable of fostering diverse confidence values under a well tuned PPO training pipeline.

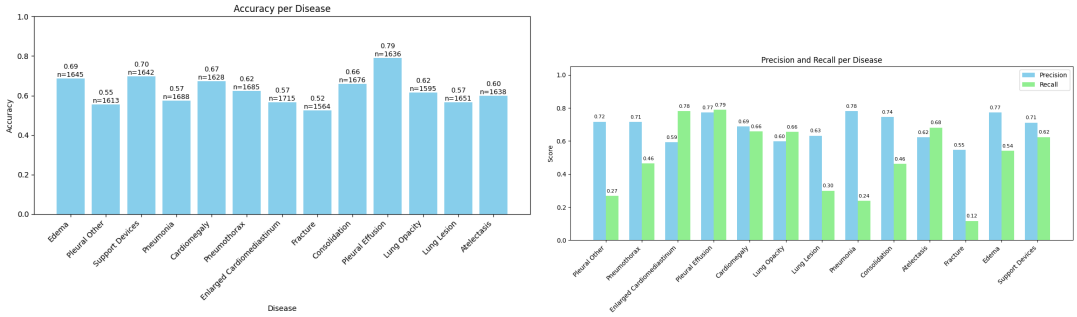

Figure 4.2: (left) Accuracy of RaDialog per finding type for the sampled training set. (right) Precision and recall of RaDialog per finding type for the sampled training set.

4.1.2 Supervised Fine-tuning

Before we can start confidence calibration for verbalized confidence estimates, we need RaDialog to be able to verbalize its confidence. Prompting RaDialog to generate confidences alongside its responses to Binary QA questions does not always yield reliable results with failure of confidence elicitation or considerably worsened generation quality. Therefore, we first train RaDialog via supervision to help it start expressing its confidence after each generated Binary QA response.

We use multiple prompts instructing RaDialog to generate confidences, and multiple expected generated answer templates. We present examples from our supervision signals used to teach RaDialog to express confidences in Table 4.2. All input prompts are followed with a confidence elicitation request. We use a typical LLM supervised fine-tuning pipeline with tokens in the supervision signal used as the soft label of each step of the sequence.

Since we do not want our supervised fine-tuning to cause any biases in the selection of the generated confidence value, we take two precautions: we always randomly select the confidence value in the supervision signal, and we do not allow for backpropagation on the confidence value token itself by using a backpropagation mask for the token. We run RaDialog on the training set to ascertain that no calibration was introduced by the supervised fine-tuning (Figure 4.3). With the confidence expression behavior integrated to RaDialog's generative process, the model is ready for PPO training.

4.1.3 The PPO Training

We use the original Rewarding Doubt R_{RD} without modifications for the Binary QA PPO training process.

SFT Signal Type	Prompt
User Input Binary QA Question	Is there evidence of {finding} in the image?"
User Input Binary QA Question	Is there any {finding}?
User Input Binary QA Question	Is any {finding} evident in the report?
User Input Binary QA Question	Is there any indication of {finding} in the report?
User Input Binary QA Confidence Instruction	Provide a confidence between 0 and 10, of how sure you are the answer is correct. A value close to 0 means you think there is a high probability that the answer is wrong. Your confidence is to be reported in a JSON dictionary of the following format: {"confidence": int}.
Assistant Response Binary QA Answer	Yes, the image shows {finding}. {"confidence": 5}
Assistant Response Binary QA Answer	Yes, the patient has {finding}. {"confidence": 2}
Assistant Response Binary QA Answer	No, the image does not show {finding}. {"confidence": 9}
Assistant Response Binary QA Answer	No, {finding} is not observed in the image. {"confidence": 4}

Table 4.2: Supervision signals used for Binary QA confidence expression behaviour.

PPO Parameters and Training Hyperparameters

Upon experimenting with learning rates between $1e-7$ and $1e-4$ we determine $1e-5$ as our empirically optimal learning rate. We use training batches of size 12, and accumulate gradients for 3 steps. After processing 3 mini-batches of size 4, we take gradient steps. We run the training until we see no further improvements on our metrics.

We configure the PPO algorithm to follow most of the default values as defined by the TRL library and the original RLHF parameters. We modify the initial KL coefficient to be 0.05 similar to the Rewarding Doubt study. Unlike the original RLHF study and Rewarding Doubt, we do not run multiple PPO epochs over the same training batch. Each batch updates the model only once during the PPO optimization step.

We set the probability to change confidence values at each batch to 0.40 meaning that we override the generated confidence values in every 2 batches in 5 consecutive batches

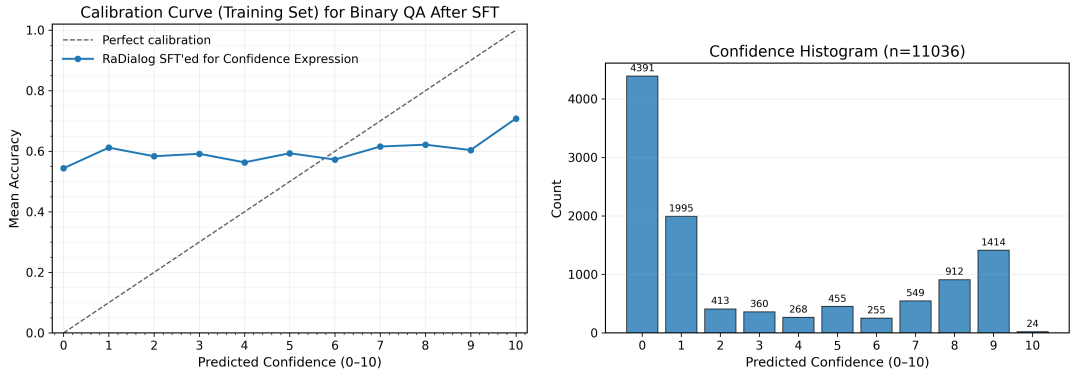

Figure 4.3: (left) Calibration curve of vanilla verbalized confidences on the training set by the supervised fine-tuned RaDialog. (left) Confidence histogram.

on average. We explain the details of the forced exploration process in the following parts.

We use a final scalar value of 5 to scale the normalized R_{RD} from range $[-1, 1]$ to $[-5, 5]$ similar to the original Rewarding Doubt study. Later on in the study during the Report Generation part, we determine the final scalar as an important hyperparameter in preventing mode collapse.

Generation Parameters

We allow RaDialog a maximum of 50 new tokens during Binary QA answer generations with token sampling enabled. We assign parameters such as *top k*, *top p*, and *temperature* their default values so we do not intervene in the token sampling process via a modification of the probability distribution of tokens.

The PPO Training Step

In each PPO training step, we start by prompting RaDialog with an image and a Binary QA question about the image. We allow RaDialog to complete its generation, and parse the generation using substring search and regular expressions. We extract the classification based whether "Yes" or "No" tokens occur in the sentence by mapping these tokens to classifications *True* and *False*, respectively. Using regular expressions, we search for the JSON dictionary pattern carrying the confidence information with some flexibility allowing for character-level generation errors. Non-integer confidence values, lack of a confidence value, multiple confidence values, or confidence values outside of the allowed range of integers are extracted as *None* confidence values which

receive a penalty much worse than any attainable penalty when a valid confidence is generated.

We compare the extracted classification to the groundtruth label and determine the accuracy of the classification. Using the accuracy value, the extracted confidence and R_{RD}, we calculate the reward. Finally, we perform PPO backpropagation using this reward on the PPO-target.

In the Binary QA use-case we have tried two different PPO targets. First we targeted the entire generated Binary QA answer and the confidence value as a whole. This method has rendered the reward signal to partially act as a supervised fine-tuning signal and RaDialog has adversarially benefitted from the training regime to modify its accuracy to hack the calibration behavior. Instead of learning to generate informative confidence estimates, it has changed its accuracy level to meet the confidence value it collapses on. To avoid the adversarial faked calibration behavior, we settled on targeting only the confidence generation part of any response from RaDialog. This way, it has no chance of hacking the training via changing any other behavior other than its confidence verbalization behavior.

Extreme Confidence Sanity Check

We test the functionality of our PPO implementation modified for multimodality by using a trivial reward which only rewards overconfidence. We positively reward any generated integer confidence value between 6 and 10 and penalize all other values. The model quickly learns to generate high confidence values for all of its generations. This experiment serves as a sanity check for our modified pipeline and we do not present the results as they are trivial. We confirm that the modified PPO implementation is ready for confidence calibration.

Forced Exploration and Additional Reward for Correct Responses

The original Rewarding Doubt study performs two additional mechanisms to help with the exploration-exploitation balance. We repeat these mechanisms in our experiments.

The first mechanism is forced exploration. Based on the hyperparameter *probability to change confidence* we override the generated confidence values with random valid confidence values so that the model is forced to explore more diverse confidence values. At every training batch, we sample from a uniform distribution of real values in the range of $[0, 1]$ and if the sampled value is smaller than the *probability to change confidence* we replace the generated confidence values in all the samples in the batch.

The second mechanism is adding a small bonus reward for accurate responses. The original study confirms that this encourages the model to not abandon the effort towards

accurate responses in trade for better confidence calibration. Therefore, before we scale the reward with the final scalar, we always add a bonus of 0.25 to the normalized reward R_{RD} when RaDialog's answer is accurate.

Ablation Study: Granular Confidences

We ablate the range of confidence values we allow for by prompting to model to generate granular confidence values $cg \in [0, 100] \subset Z$. The ablation aims to target the mode collapse issue chronically occurring in our trainings by giving the model a wider range of possible confidence values to try out. The model receives the same reward for cg values which map to the same $c \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ value and we only observe calibration curves or the ECE metric for the non-granular confidence values.

4.1.4 Baseline Comparisons

In our experiments with the Binary QA use-case, we only achieve low ECE but at the cost of a single-value confidence generation behavior. Therefore, we do not perform an extensive set of comparisons against confidence estimation methods in the literature. We focus on the vanilla verbalization method [57] and a modified version of the P(True) method [21] to interpret the results of our experiments, and come up with stronger ideas for the next stage of this study focusing on report generation.

We compare our results only on the MIMIC-CXR validation split, which is an in-distribution set unlike the out-of-distribution test split.

Vanilla Verbalize

RaDialog is a heavily fine-tuned model for medical dialogues, meaning that it is not always straightforward to prompt it for unrelated use-cases. Therefore instead of prompting it to generate confidence values alongside its responses, we inject the confidence generation request as a post hoc self-guiding prompt where we force the model to already start generating a confidence given a previously generated report.

In Table 4.3, we present the self-guiding prompt for vanilla verbalization for the Binary QA use-case. We note that without injecting the confidence generation request this way it is not possible to consistently elicit verbalized confidences from RaDialog for the Binary QA use-case.

Modified P(True)

Instead of following the post hoc confidence request technique of the original study suggesting the P(True) method [21], we rely on RaDialog's always starting its answers

Prompt Step	Step Content
1	{input image} You are to act as a radiologist and answer the following question: Is the following finding visible in the given X-ray image: Lung Lesion?
2	ASSISTANT (Pre-generated Binary QA Answer from RaDialog): Yes, the image shows Lung Lesion.
3	USER: Now evaluate your own response. How confident are you in your answer? Provide a confidence between 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, of how sure you are the answer is correct.
4	ASSISTANT: When asked how confident I am about a response, I consistently provide it in a JSON object, adhering to my policy.The confidence JSON follows this structure: {'confidence': int}. Here's my confidence JSON about my last response:

Table 4.3: Prompt steps for the forced self-guided vanilla verbalization for the Binary QA use-case.

with a *Yes* or a *No* token when we prompt it with a Binary QA question. Therefore, we use no further prompts asking for the confidence but rather extract the softmax probabilities of tokens *Yes* and *No* after prompting the model with the Binary QA question. We normalize the probabilities such that $P(Yes) + P(No) = 1$. The normalized probability of token *Yes* is then taken as the confidence estimation of the model based on the P(True) method.

We note that this method assumes a classification of *True* for all datapoints since we remove the sampling process and define no specific classification process. We use this analysis merely as a tool to help our interpretation of the PPO trainings for the Binary QA use-case.

4.2 Report Generation

We continue our experiments with our main use-case: report generation. Having established a proof-of-concept via the Binary QA use-case with our modified PPO implementation, we carry out our experiments to extend our method to report generation and address the chronic mode collapse issue thoroughly and rigorously.

The report generation use-case allows users to prompt RaDialog with an input image, possible findings extracted previously in the RaDialog pipeline, and receive a complete "findings" report. The input prompt and the expected RaDialog generation is presented in Table 4.4.

Step	Content
Input Prompt	{input image} Predicted Findings: {findings}. You are to act as a radiologist and write the finding section of a chest x-ray radiology report for this X-ray image and the given predicted findings. Write in the style of a radiologist, write one fluent text without enumeration, be concise and don't provide explanations or reasons.
RaDialog Response	There is no focal consolidation, pleural effusion or pneumothorax. [...] Chronic deformity of the posterior left sixth and seventh ribs are noted. {"confidence": 7}

Table 4.4: Input prompt and the expected response example for the Report Generation use-case.

We perform multiple experiments using the datasets we create specifically for training and testing for the report generation use-case. For each model trained via PPO for the report generation use-case, we calculate ECE and further metrics we define within the scope of our study.

4.2.1 Datapoint Sampling and Dataset Creation

The MIMIC-CXR dataset has abundant number of samples in the training set. For supervised fine-tuning, having as many training datapoints as possible is more preferable. In the case of reinforcement learning, and especially PPO, both the RL paradigm and the algorithmic design of PPO should theoretically allow for a repetitive nature for training. This means that a smaller number of samples should already be sufficient for the model to learn the behavior, and we can craft a dataset that repeats a periodic curriculum spanning datapoints of diverse difficulties.

We prepare two training datasets to be used in the confidence calibration of RaDialog for report generations. The first training set is the MIMIC-CXR training split with datapoints without a *findings* section in the report removed. This method leaves us with 270,790 training datapoints. We note here that none of trainings have run long enough to complete a full epoch in this dataset. We call this set the *non-repetitive* training set.

The second training set comprises 1,476 datapoints sampled from the training set. The sampling is carried out according to the difficulty of the datapoints. We use RaDialog to generate reports based on 3200 samples selected from the training set and use RadLlama2-7b to allocate GREEN scores to these generations. We sample 150 datapoints per GREEN score G, with each score bin such as $G = 0.1$ or $G = 0.7$ receiving 150 datapoints or the maximum datapoints if the GREEN score bin has fewer

than 150 datapoints. We call this training set the *difficulty-balanced repetitive* training set. The GREEN score histograms of this dataset are shown in Figure 4.4.

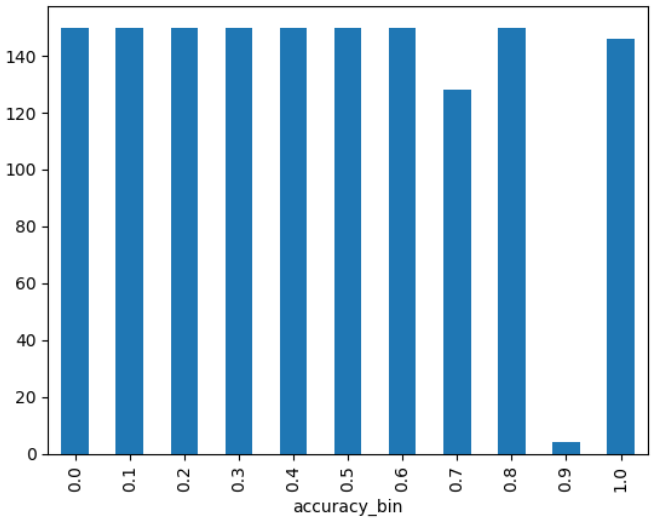

Figure 4.4: GREEN Score bins attained by RaDialog on the training set.

Figure 4.5: GREEN score histograms for the repetitive balanced-difficulty training set.

We hypothesize that the difficulty-balanced repetitive training set is a stronger candidate for RL with PPO and perform an ablation study comparing the two datasets. The outcome of this ablation study is discussed in Section 5.2.5.

Furthermore, we create two test sets. We use the second half of the validation split from MIMIC-CXR as our in-distribution test set with a 1,000 datapoints, and 1,000 random datapoints sampled from the test split from MIMIC-CXR as our OOD test set. The OOD quality of the test split stems from the distribution of CheXpert findings occurring on the chest X-ray images diverging away from the training and validation splits.

4.2.2 Supervised Fine-tuning

Similar to the Binary QA use-case, RaDialog cannot consistently generate confidence values after generating reports. We can ask a follow-up confidence question and receive a confidence estimation consistently, but since it is preferable for our method that confidences are estimated immediately without further prompting, we fine-tune RaDialog once again via supervision so that it produces confidence values. Our fine-tuning instruction takes the form in Figure 4.6.

{input image} Predicted Findings: {findings}. You are to act as a radiologist and write the finding section of a chest x-ray radiology report for this X-ray image and the given predicted findings. Write in the style of a radiologist, write one fluent text without enumeration, be concise and don't provide explanations or reasons.
Provide a confidence between 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, of how sure you are the answer is correct. A value close to 0 means you think there is a high probability that the answer is wrong. Your confidence is to be reported in a JSON dictionary of the following format: {"confidence": int}.

Figure 4.6: The supervised fine-tuning prompt used to teach RaDialog to generate confidence values alongside reports.

After the supervised fine-tuning, RaDialog achieves to consistently return a JSON dictionary immediately after each generated report following our rules and with confidence values randomly allocated without any calibration whatsoever. At this point, the model is ready for PPO training for confidence calibration.

4.2.3 Success Indicators

Achieving the optima points of the typical training success indicators such as the PPO loss, mean reward, ECE and the confidence calibration curve is not sufficient to guarantee a useful confidence estimation method. In multiple experiments we have observed that models can achieve near perfect ECE via mode collapse on the predicted confidence value that is closest to the mean empirical accuracy over the dataset. Moreover, with continuous accuracy values such as the GREEN score, achieving apparent perfect calibration is possible even when the scores within a confidence bin are distributed broadly.

We define two new metrics which help us track the confidence estimation success of a model during training and at inference time. We define the first metric as the *Confidence-Bin Normalized Dispersion* (CBND) and indicate it with σ_{CBND} to account for the standard deviation of GREEN scores within the confidence bins. We define the second metric as the *Normalized Confidence KL Divergence* and indicate it with D_{NCKL}. This metric helps measure how diverse the verbalized confidence values are, or in other words, how far away the model is from a mode collapse.

Confidence-Bin Normalized Dispersion (CBND)

We take the standard deviation of the accuracies within each confidence bin and find the weighted mean standard deviation. Given confidence bins $B_c = \{G_{c,1}, G_{c,2}, \ldots, G_{c,n_c}\}, \quad G_{c,i} \in [0,1] \subset R$ with $c \in \{0, 1, \ldots 10\}$ mapping to predicted confidences $\hat{p} \in \{0.0, 0.1, \ldots 1.0\}$ as defined in Section 3.1 and GREEN scores representing the accuracies within a bin $G_c = \{G_{c,1}, G_{c,2}, \ldots, G_{c,n_c}\}$ we calculate $\sigma_{CBND} \in [0,1]$ as per Equations (4.2.1) through (4.2.6).

$$\mu_c = \frac{1}{n_c} \sum_{i=1}^{n_c} G_{c,i} \tag{4.2.1}$$

$$\sigma_c = \sqrt{\frac{1}{n_c} \sum_{i=1}^{n_c} \left(G_{c,i} - \mu_c\right)^2} \tag{4.2.2}$$

$$\sigma_{c,\max} = \sqrt{\mu_c \left(1 - \mu_c\right)} \tag{4.2.3}$$

$$\tilde{\sigma}_c = \frac{\sigma_c}{\sigma_{c,\max}} \tag{4.2.4}$$

$$w_c = \frac{n_c}{N}, \quad N = \sum_{c=0}^{10} n_c \tag{4.2.5}$$

$$\sigma_{\text{final}} = \sum_{c=0}^{10} w_c \, \tilde{\sigma}_c \tag{4.2.6}$$

The σ_{CBND} value taking its maximum value 1.0 means that the confidence bins have the lowest consistency possible with respect to the GREEN scores they contain, whereas a σ_{CBND} value of 0.0 means that the GREEN scores in each confidence bin perfectly matches the mean GREEN score of the bin.

Normalized Confidence KL Divergence (NCKL)

Assuming a dataset with a balanced distributions of difficulties, an ideal distribution of confidence estimations must form a uniform distribution over values $c \in [0, 10]$. This assumption does not have to hold for all datasets, but it establishes a generalist baseline that we can compare our model's performance against. Therefore, we calculate D_{CKL}, the unnormalized KL divergence of the estimated confidence values against the uniform distribution. Then we normalize it to achieve D_{NCKL}.

We use w_c from equation (4.2.5) as the probability assigned to a confidence value. We can then calculate D_{CKL} of the confidence distribution using $u_c = \frac{1}{11}$ as the uniform sampling likelihood for all confidence bins. KL divergence can generally be an unbounded value. However, in the case of D_{NCKL} the highest KL divergence value occurs in the case of a mode collapse with all confidence estimations returning the same confidence value. We can derive $max(D_{CKL})$ when any one of the w_c values is equal to 1, accumulating the entire weight of the distribution. Then we can normalize the NCKL value to fit in the $[0, 1]$ range (Equations (4.2.7) through (4.2.9)).

$$D_{\text{CKL}} = \sum_{c=0}^{10} w_c \log \frac{w_c}{u_c} = \sum_{c=0}^{10} w_c \log \left(11 w_c\right) \tag{4.2.7}$$

$$max(D_{\text{CKL}}) = = \sum_{c=0}^{10} 1 * \log \left(11 * 1\right) = log(11) \tag{4.2.8}$$

$$D_{NCKL} = \frac{D_{CKL}}{log(11)} \tag{4.2.9}$$

Training Time Heuristic Score

We use the ECE, the mean normalized reward $R_{NORMALIZED} \in [0, 1]$, σ_{CBND} and D_{NCKL} to calculate a heuristic score during training to be able to early quit, save models at checkpoints, and measure success. We track these success indicators for each training batch and over the entire validation set during validation runs. The heuristic score $S_H \in [0, 1]$ is defined by Equation (4.2.11). Best S_H attainable is 1.0. We indicate the final reward here as $R_{CE-FINAL}$ without loss of generality with the final reward easily replaceable with $R_{QB-FINAL}$.

$$R_{NORMALIZED} = \frac{R_{CE-FINAL} - min(R_{CE-FINAL})}{max(R_{CE-FINAL}) - min(R_{CE-FINAL})} \tag{4.2.10}$$

$$S_H = \frac{(1 - ECE) + (1 - \sigma_{CBND}) + (R_{NORMALIZED}) + (1 - D_{NCKL})}{4} \tag{4.2.11}$$

4.2.4 Reward Tuning

Before placing RaDialog under a long PPO training, we experiment with our rewards and their parameters to determine the best reward configuration. We explore all configuration options available both for $R_{CE-FINAL}$ and $R_{QB-FINAL}$.

Reward Tuning for the Cross-Entropy Reward

We experiment with reward hyperparameters for $R_{CE-FINAL}$. We search for the optimal ϵ value for the shifted normalization, and analogously for the optimal s for tanh and logistic normalizations. We search also for the optimal final scaling value k which amplifies the reward signal. We form a grid spanning the hyperparameters and select ten configurations at random. Table 4.5 shows the range of parameters we used to create the grid.

Parameter	**Value Range**
ϵ	$[10^{-4}, 10^{-1}]$
s	$[log(1.2), log(2.0)]$
k	$[2.0, 8.0]$

Table 4.5: The grid boundaries for reward tunings.

We track the training success via the success indicators we have defined in Section 4.2.3. While manually observing ECE, D_{NCKL}, σ_{CBND}, and the mean reward we also use the heuristic score S_H calculated over the validation set at each checkpoint as the early-quit criterion. If S_H does not have a positive trend in five consecutive evaluations over the validation set in a window of the last ten validation runs, then the training stops. We train on 180 datapoints in between each validation checkpoint consisting of 144 datapoints.

Our reward tuning shows that for the *shifted* normalization, higher ϵ values on the range of 10^-2 function much better with the normalized ϵ value showing 0.70 correlation with S_H. For *tanh* and *logistic* normalizations, we observe that higher s values result in better trainings with 0.80 correlation with S_H. For all normalization methods, the final scalar k is highly negatively correlated with S_H with -0.90 correlation for the *shifted* method and -0.70 correlation for the non-linear normalizers. All correlations are derived using Pearson's correlation coefficient (PCC) formulations.

Among some reward configuration candidates, we observe that the mode collapse pitfall can still occur especially with high k. Alongside the raw reward itself, the reward normalization and scaling process therefore have a considerable impact on training stabilities. We suggest the reward tuning process as a way to determine reward configurations capable of avoiding mode collapses a common problem that is also chronically encountered in the original Rewarding Doubt method [48].

We observe that among all reward configuration candidates, the *shifted* normalization with a high ϵ value of 10^-2 and a low final scalar of $k = 2$ return the best results according to our heuristic. This reward configuration faces no mode collapses, which further certifies the configuration's success. We select the final $R_{CE-FINAL}$ reward to be

configured with the tuned parameters presented in Table 4.6.

Reward Configuration Parameter	**Value**
Normalization Method	*shifted*
epsilon	10^{-2}
k	2

Table 4.6: The final configuration for the $R_{CE-FINAL}$ reward.

Reward Tuning for the Quadratic-Blend Reward

Using our learnings from the reward tuning process for R_{CE} we use $k = 2$ as the final multiplicative scalar. We search for various β values and conclude that setting $\beta = 0$ is sufficient to achieve a successful training. This fundamentally reduces the quadratic-blend reward to a simple equation with no β parameter as described in Equation (4.2.12). The result of the reward tuning rendering the reward independent from β further strengthens the R_{QB} reward by making it very intuitively explainable.

$$R_{QB-FINAL} = k * |\hat{p} - *p|^2 \tag{4.2.12}$$

4.2.5 Setting up the GREEN Score Pipeline

The GREEN Score method requires an external LLM to function as the judge for the generated reports. The LLM implementation from the study is based on PyTorch [40]. Due to incompatability of GREEN and RaDialog's Python dependencies, we need to deploy the GREEN LLM RadLlama2-7b as an Application Programming Interface (API) accessible software running as its own process on the GPU. We use a C++ based software package called *llama.cpp* [11] which serializes the LLM parameters and runs them on a self-contained API server.

In order to fit both the llama.cpp hosted RadLlama2-7b and RaDialog on the same GPU at the same time, we have serialized and used the 4-bit quantized version of RadLlama2-7b. To verify that the serialization and quantization processes do not drastically alter the GREEN scores allocated to generated reports, we performed three experiments using 100 randomly sampled datapoints from the MIMIC-CXR training set. We used the PyTorch 16-bit non-quantized version of RadLlama2-7b, the 16-bit serialized llama.cpp version, and the 4-bit quantized and serialized llama.cpp version. We measured the mean absolute error (MAE), Pearson and Spearman Correlations between the GREEN Scores obtained from each experiment

We conclude that using the serialized 4-bit quantized version preserves the behavior while decreasing computational costs vastly. Table 4.7 summarizes the results of the three experiments.

Implement. 1	Implement. 2	MAE ↓	Pearson Correlation ↑	Spearman Correlation ↑
PyTorch 16-bit	llama.cpp 16-bit	0.05	0.94	0.95
llama.cpp 16-bit	llama.cpp 4-bit	0.03	0.96	0.97
PyTorch 16-bit	llama.cpp 4-bit	0.05	0.94	0.94

Table 4.7: GREEN Score bins attained by RaDialog on the training set.

4.2.6 The PPO Training

Having determined the best reward configuration for $R_{CE-FINAL}$ and $R_{QB-FINAL}$ we run two separate week-long trainings using these rewards and an additional ablation training round for $R_{CE-FINAL}$.

PPO Parameters and Training Hyperparameters

Upon experimenting with learning rates between $1e-7$ and $1e-4$ we determine $6e-6$ as our empirically optimal learning rate. We use training batches of size 12, and accumulate gradients for 3 steps. After processing 3 mini-batches of size 4, we take gradient steps. We run the training for 18 epochs when using R_{QB} and 11 epochs for R_{CE}. However, we do not necessarily use the model after the last epoch, but pick the model performing the best on the validation set during the epochs.

We configure the PPO algorithm to follow most of the default values as defined by the TRL library and the original RLHF parameters. We modify the initial KL coefficient to be 0.05 similar to the Rewarding Doubt study. Unlike the original RLHF study and Rewarding Doubt, we do not run multiple PPO epochs over the same training batch. Each batch updates the model only once during the PPO optimization step.

We set the probability to change confidence values at each batch to 0.25 meaning that we override the generated confidence values every fourth training batch on average. As explained in earlier sections, this ensures forced exploration of diverse confidence values.

Generation Parameters

We allow RaDialog a maximum of 300 new tokens during report generation with token sampling enabled. We assign parameters such as *top k*, *top p*, and *temperature* their default values so we do not intervene in the token sampling process via a modification of the probability distribution of tokens.

The PPO Training Step

In each PPO training step, we first start by prompting RaDialog to generate reports and confidence values for each datapoint in the training batch. Then using the same techniques explained in Section 4.1.3 for the Binary QA use-case, we extract the confidence value and occasionally replace it with a random confidence value based on the probability to change confidences.

We send an API request to the llama.cpp deployment of RadLLama2-7b to evaluate the generated reports and retrieve GREEN scores. Using these scores, the extracted confidence values, and the selected reward for the training procedure we determine the scalar reward that each generation and confidence estimate pair receives. Using the reward, we target the generated confidence part of the responses and take a PPO optimization step using our multimodal implementation of TRL's PPO implementation.

Observations on the Training Process

As discussed in Section 4.2.3, we have multiple success indicators to observe during training. Our observations during the training using our final rewards show that the success indicators show an oscillating shape with peaks and plummets. This sort of behavior would be untypical for supervised fine-tunings and could hint at a possible failure. However, in our RL training regime, we see that the model can achieve higher peaks for all success indicators after experiencing subpar values. We therefore associate this behavior with the exploration-exploitation dilemma and note it as an indication of success of our RL pipeline.

Ablation Study for the Difficulty-balanced Repetitive Training Set

In our setup for report generation, the difficulty-balanced repetitive training set serves as the default. Our empirical observations with auxiliary trainings show that repeated exposure to a smaller dataset with a balanced distribution of difficulties is sufficient for the model to explore a more diverse range of confidences and figure out the rules of the RL game fast.

We ablate the repetitive training set by using the non-repetitive training set during a separate additional round of training with $R_{CE-FINAL}$. We expect the non-repetitive training to result in a model generating confidence values with a higher D_{NCKL}, overfitting the mean GREEN score of its generations by collapsing towards confidence estimates closer to it. Apart from the training set, we change no hyperparameters and use an identical training pipeline.

4.2.7 Baseline Comparisons

We compare the confidence estimation performance of our PPO-trained models to four strong confidence estimation methods from the literature: vanilla verbalization [57], P(True) [21], Sequence Probability [32], and the Trained Probe method [2]. We evaluate the performance of all these methods on our in-distribution test set, and then select the best performing ones to evaluate the generalization performance on the OOD test set.

The Vanilla Verbalization Method

We select the vanilla verbalization method as our only black-box confidence estimation baseline since it is straightforward to implement and mirrors our generative confidence estimation process. Similar to the Binary QA use-case, we use an injected self-guiding confidence request. In Table 4.8, we present the self-guiding prompt for vanilla verbalization for report generation. We note that without injecting the confidence generation request this way it is not possible to consistently elicit verbalized confidences from RaDialog for the report generation use-case.

The P(True) Method

After each generated report, we prompt the model additionally to provide an evaluation of the accuracy of its latest response: *"USER: Was your previous answer correct? Answer with a single word: Yes or No."*. We extract the first predicted token upon the evaluation request, collect the softmax probability of tokens *Yes* and *No* and and normalize the probabilities such that $P(Yes) + P(No) = 1$. The normalized probability of token *Yes* is then taken as the confidence estimation of the model based on the P(True) method.

The Sequence Probability Method

For each generated report, we collect the probabilities of each generated token within the vocabulary distribution and take the average probability as the confidence estimate. The confidence is estimated via Equation (4.2.13) with the probability of the j-th generated token in a report denoted with p_j.

Prompt Step	Step Content
1	USER: {input image} Predicted Findings: {findings}. You are to act as a radiologist and write the finding section of a chest x-ray radiology report [...] be concise and don't provide explanations or reasons.
2	ASSISTANT (Pre-generated Report from RaDialog): There is no focal consolidation, pleural effusion or pneumothorax. [...] Chronic deformity of the posterior left sixth and seventh ribs are noted.
3	USER: Now evaluate your own response. How confident are you in your answer? Provide a confidence between 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, of how sure you are the answer is correct.
4	ASSISTANT: When asked how confident I am about a response, I consistently provide it in a JSON object, adhering to my policy.The confidence JSON follows this structure: {'confidence': int}. Here's my confidence JSON about my last response:

Table 4.8: The self-guiding prompt steps for forced vanilla verbalization for the report generation use-case.

$$Avg(p) = \frac{1}{|J|} \sum_{j} p_j \tag{4.2.13}$$

The Trained Probe Method

We use the internal state of RaDialog to train a small multilayer perceptron (MLP) for confidence estimation. We use the $16th$ hidden layer of RaDialog as the input values to the MLP to obtain a single real value in range $[0, 1]$ at the output. The output is the sigmoid of the logits at the final layer of the MLP. At training time, we use backpropagation on the binary-cross entropy between the predicted confidence value and the GREEN score allocated to the generated report. We train the MLP on a training set of 1,500 datapoints and the MLP achieves its lowest loss after the 4th epoch without overfitting the training set. At inference time, we always extract the hidden states at the 16th layer for each generated report and pass them through the MLP to obtain the confidence value.

5 Results and Discussion

5.1 Binary QA Results

We achieve a low ECE for the Binary QA use-case while not being able to find a way to encourage diverse confidence values. All our trials end up with a very conservative range of generated confidence values and sometimes with a complete mode collapse. These results prompt us to explore the issue deeper in the next part of the study performed on the report generation use-case.

We compare our results to the vanilla verbalization method. We present the comparison results in Figures 5.1, 5.2, and Table 5.1.

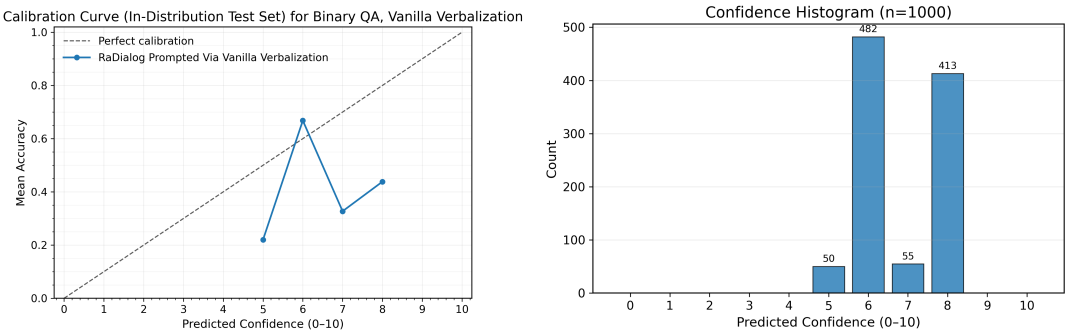

Figure 5.1: Confidence calibration results for the vanilla verbalization method for Binary QA.

Method	**ECE ↓**
Vanilla Verbalize	0.21
Verbalize after PPO	0.01

Table 5.1: ECE on the Binary QA use-case for the vanilla verbalization method and verbalization after PPO.

We observe that our method vastly reduces ECE compared to vanilla verbalization method. However it does not increase the diversity of the generated confidences and

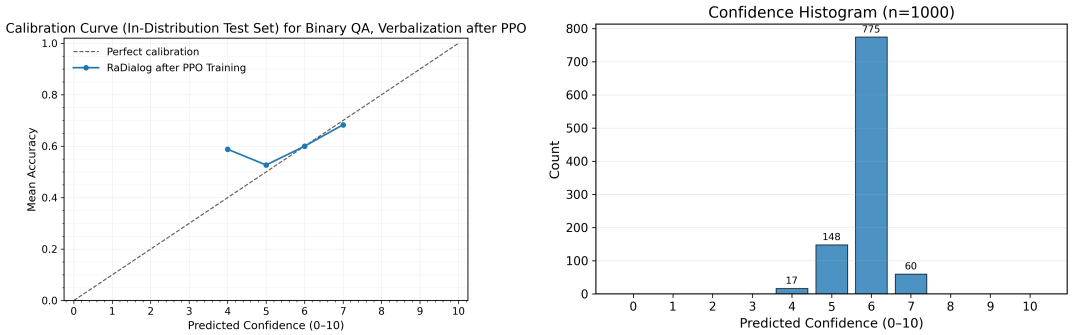

Figure 5.2: Confidence calibration results for Binary QA after PPO training.

in fact it collapses onto a similar set of confidence values as the vanilla verbalization method. Even though we try with various hyperparameters, and especially with various probabilities to change confidence we cannot achieve a more diverse range of confidences.

We note that apart from the diversity of confidence values, the PPO training shows signs of success with all other training metrics such as the PPO loss and the attained reward. We also try much longer training times but they derail the model from its regular behavior. Therefore we do not attempt to further improve the confidence diversity for the Binary QA use-case, and address the mode collapse issue in a broad scope for the report generation use-case.

5.1.1 Internal Calibration Analysis via the Modified P(True) Method

We use the P(True) method to analyze whether our model has an internal calibration at all. We apply the method on three versions of our model: the base model, the model supervised fine-tuned for confidence expression, and the model after PPO training. We present our results on the first half of our in-distribution test set in Figures 5.3, 5.4, 5.5, and Table 5.2.

Method	**ECE ↓**
Modified P(True) on the Original Radialog Model	0.21
Modified P(True) on the SFT'ed Radialog	0.08
Modified P(True) on Radialog after PPO Training	0.06

Table 5.2: Internal calibration of RaDialog for the Binary QA use-case before and after SFT and PPO steps.

We observe that the model has a considerably well calibrated internal state with

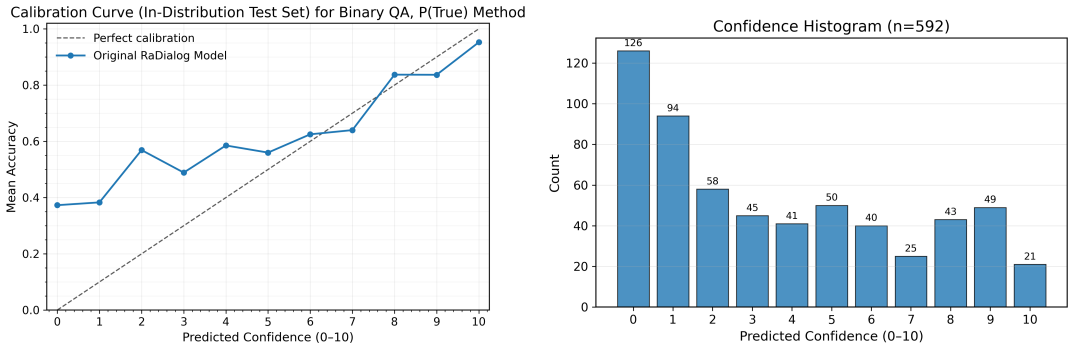

Figure 5.3: Binary QA Modified P(True) results for the base RaDialog model.

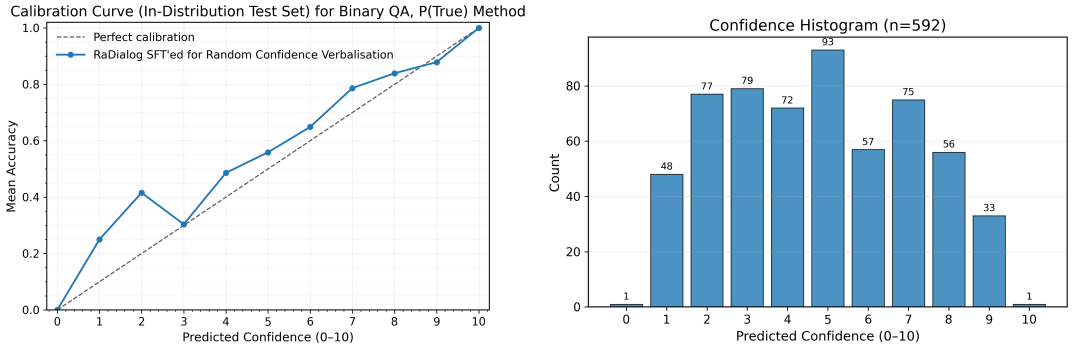

Figure 5.4: Binary QA Modified P(True) results for RaDialog supervised fine-tuning for confidence expression.

respect to classification thresholding. Furthermore, the calibration gets better when we use supervised fine-tuning for confidence expression, most likely due to further training time on the Binary QA use-case.

We believe that it should be technically possible to help the model verbalize via PPO its inner calibration that has diverse calibration values. However our training pipeline is simply not optimized enough to help the model verbalize what it holds within its internal state. This sparks the training optimization investigations we carry out for the report generation use-case.

5.1.2 Results for the Ablation Study with Granular Confidences

Our trainings with granular confidence values fail to avoid the mode collapse behavior as well. The achieved ECE and the confidence curve are similar to the results obtained from the non-granular confidence trainings. We therefore abandon the idea of granular

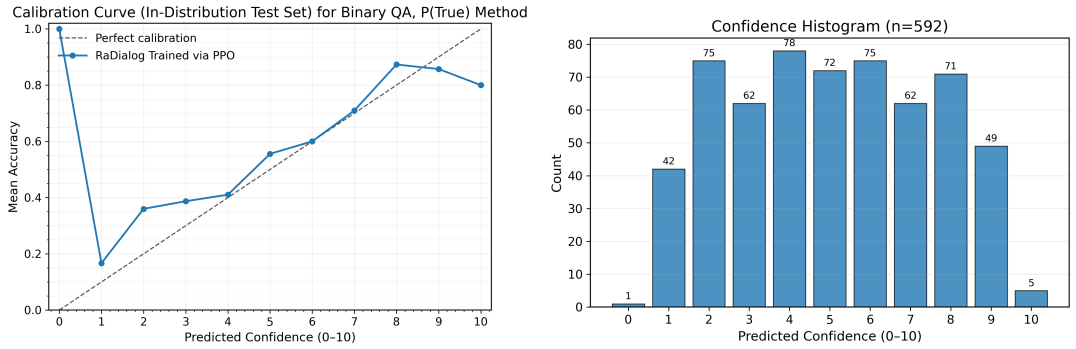

Figure 5.5: Binary QA Modified P(True) results for RaDialog after PPO training for confidence calibration.

confidences and do not further ablate the set of confidences allowed within the scope of our study.

5.2 Report Generation Results

5.2.1 GREEN Score Evaluation of RaDialog

We perform an evaluation run of generated reports on a randomly selected subset of the MIMIC-CXR dataset's training split to see how RaDialog performs when the accuracy metric is the GREEN Score. RaDialog achieves a mean accuracy of 35% to 45% depending on the subset sampled from the training split and the generation parameters. We further examine the GREEN score values that RaDialog receives by looking at GREEN score histograms and determine that the attained scores are diverse enough to help guide the reinforcement learning regime (Figure 5.6). We note that RaDialog rarely receives GREEN Score 0.9, and lower scores are received more frequently.

5.2.2 Confidence Calibration Results

With both of our rewards $R_{CE-FINAL}$ and $R_{QB-FINAL}$ we achieve low ECE, σ_{CBND}, and D_{NCKL}. Alongside these metrics, we observe the confidence calibration curve of RaDialog after training and conclude that calibration is achieved to a high extent. We stop the training manually when we observe that the model achieves a local optimum and shows no further signs of having the possibility to find the global optimum.

We compare our results first to the vanilla verbalization method over the in-distribution test set to see the shift for the same task before and after PPO training. Since our best

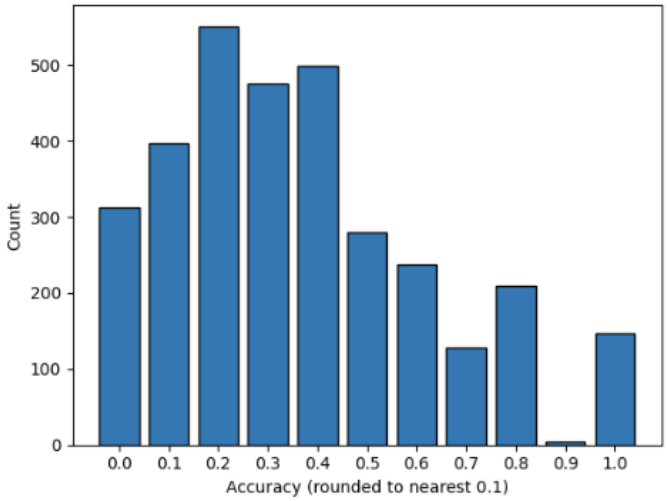

Figure 5.6: GREEN Score bins attained by RaDialog on the training set.

results are achieved via $R_{QB-FINAL}$ we focus on results from this reward in this comparison. We present the calibration curve for the two methods in Figures 5.7, 5.8, and the metrics in Table 5.3.

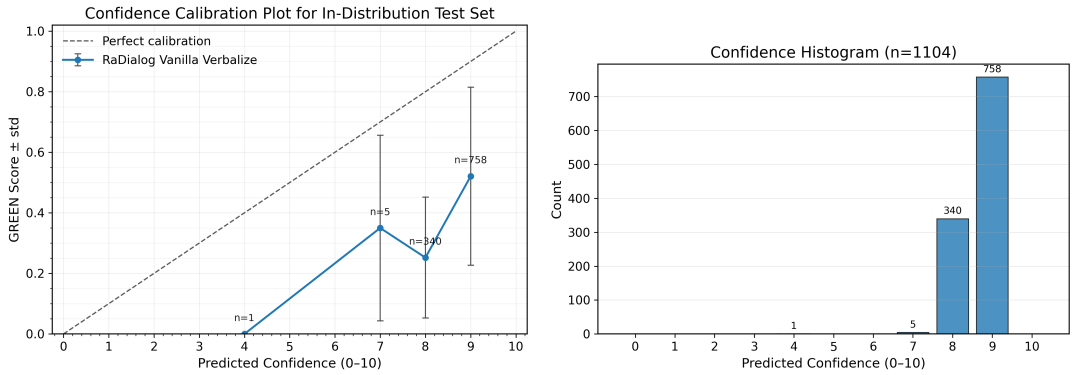

Figure 5.7: Confidence calibration results from the vanilla verbalization method for report generation.

We observe immediately that our novel reward has addressed the issue of mode collapse successfully and fostered diverse confidence values in verbalization. The PPO-trained model now has a drastically lower ECE error and D_{NCKL} with lower GREEN score standard deviation in the confidence bins. The vanilla verbalize method achieves typically overconfident confidence estimates as expected and has a non-monotonous

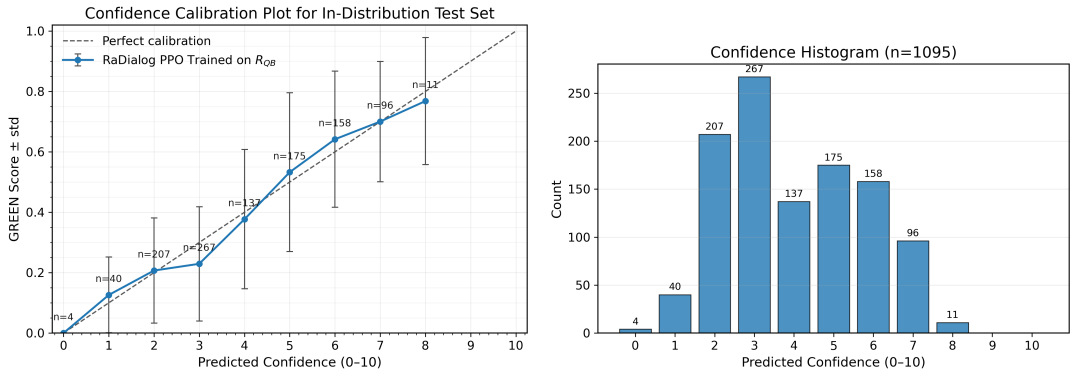

Figure 5.8: Confidence calibration results from verbalized confidence after PPO training via R_{QB}.

Method	**ECE** ↓	σ_{CBND} ↓	D_{NCKL} ↓
Vanilla Verbalize	0.43	0.55	0.73
Verbalize after PPO with $R_{QB-FINAL}$	0.03	0.44	0.18

Table 5.3: Confidence calibration metrics for vanilla verbalization and verbalization after PPO training via R_{QB}.

calibration curve whereas we have a monotonous curve after PPO training. Furthermore, we observe that even when the standard deviation is taken into consideration, the calibration curve of the PPO-trained model still shows monotonous behavior; i.e., the lower and higher GREEN score bounds of the confidence bins increase with the confidence value. We observe that even the PPO-trained model is hesitant to explore extreme confidence values such as 0 and 10. We hypothesize but do not attempt to prove that this stems from the harshness of the punishment when extreme confidence estimates fail to align with the GREEN score. Furthermore, confidence value 9 is unexplored by the model, with the reason most likely relating to GREEN Score 0.9 rarely occurring in our training set.

5.2.3 Comparison to Other Baselines

We put our method in perspective against the selected candidates from the literature by including results from both of our final rewards. Figure 5.9 displays the performance of all the baseline methods we have selected and the performance of the model after PPO with both rewards. Table 5.4 summarizes the results with respect to all our success indicators. All results presented here are from the in-distribution test set.

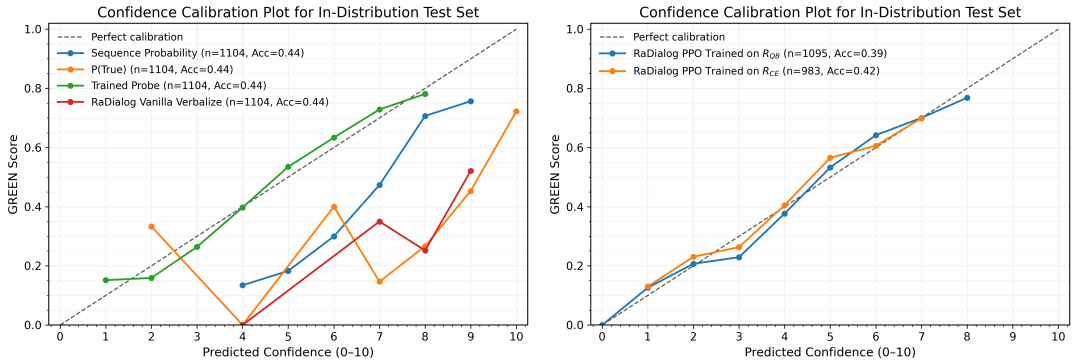

Figure 5.9: Confidence calibration results from (a) baseline methods (b) our methods.

Method	**ECE** ↓	$sigma_{CBND}$ ↓	D_{NCKL} ↓
P(True)	0.45	0.54	0.54
Vanilla Verbalize	0.43	0.55	0.73
Sequence Probability	0.23	0.50	0.38
Verbalize after PPO with $R_{CE-FINAL}$ (ours)	**0.03**	0.53	0.31
Verbalize after PPO with $R_{QB-FINAL}$ (ours)	**0.03**	**0.44**	0.18
Trained Probe	**0.03**	**0.44**	**0.17**

Table 5.4: Confidence calibration metrics for all baseline methods and our methods.

We observe that most of the baseline methods we have selected fail at achieving calibrated confidence estimations with only the Trained Probe method distinguishing itself from the others. This method proves its success in the prior study Rewarding Doubt as well. Since it benefits from extensive internal state data with additional training it performs well as expected. We see that our PPO trainings with both rewards are on par with the Trained Probe method in terms of the ECE value.

When we examine the standard deviation and KL divergence metrics, we see the strength of the quadratic-blend reward over the cross-entropy reward. As was hypothesized in the design phase of the rewards, we see that the quadratic-blend reward drastically over-performs the cross-entropy reward in encouraging diverse confidence values, and fosters confidence bins with lower standard deviation. We therefore verify $R_{QB-FINAL}$ as a superior method of avoiding mode collapses during training.

All things considered, we see that $R_{QB-FINAL}$ performs almost identical to the Trained Probe method with respect to all of our metrics while mitigating the the disadvantages of the latter. The Trained Probe method requires runtime access to the internal state of the model, an optimization process of finding the best hidden layer carrying the latent

confidence information, and it is not a naturally integrated generative process unlike our method. Moreover, it can only evaluate entire generations, whereas our method can be extended such that each sentence or atomic fact can immediately be followed by generated values.

We therefore interpret our method to be as strong a confidence estimation method as a strong white-box method such as Trained Probe. The very similar results they attain in all our success criteria show that the maximum technically possible calibration must have been achieved via PPO and improvements beyond these values might not be possible with RaDialog given that even the internal state of the model can only capture so much information. In other words, our results show that the PPO training helped almost fully verbalize the belief of the internal state about the accuracy of the generation.

5.2.4 Generalization

We evaluate our results and the best baseline method on our OOD dataset as defined in Section 4.2.1. We omit the worse performing baseline methods as they already fail at calibration over the in-distribution test set. We present the OOD results in Figure 5.10 and Table 5.5.

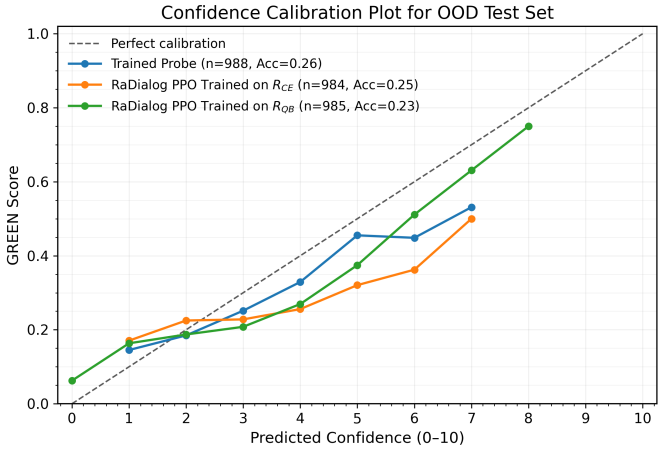

Figure 5.10: Confidence calibration results from the best performing methods on the OOD test-set.

We observe that the PPO training guided by the quadratic-blend reward generalizes reasonably well to an OOD dataset with achieving closely comparable results to the Trained Probe method and overperforming it in the diversity of estimated confidence values. Moreover, while our best method achieves a monotonous calibration curve, the

Method	**ECE** ↓	$sigma_{CBND}$ ↓	D_{NCKL} ↓
Verbalize after PPO with $R_{CE-FINAL}$ (ours)	0.10	0.40	0.39
Verbalize after PPO with $R_{QB-FINAL}$ (ours)	0.07	0.38	**0.35**
Trained Probe	**0.04**	**0.36**	0.38

Table 5.5: Confidence calibration metrics for the best methods on the OOD test set.

Trained Probe method fails to preserve monotonicity in our experiments.. This further shows how robust the Rewarding Doubt method and its extension via $R_{QB-FINAL}$ is.

We observe again that the cross-entropy reward, while being comparable, is slightly underperforming compared to the quadratic-blend reward. It also shows the non-monotonicity behavior in the OOD test set, and generalizes not as robustly as the other reward. With the OOD results considered, we can conclude clearly that our theoretically supported hypotheses around the quadratic-blend reward hold to empirically be true as well.

5.2.5 Results for the Ablation Study for the Difficulty-balanced Repetitive Training Set

We hypothesized that it is considerably beneficial for the PPO training regime to use a difficulty-balanced curriculum that presents all possible difficulty levels periodically, rather than a large non-repeating training set which has no patterns in the way difficulties are experienced by the model. We therefore have used the difficulty-balanced repetitive training set for both of our long trainings with our best final reward candidates. We present the results of training with $R_{CE-FINAL}$ with the repetitive and the non-repetitive training sets in Figures 5.11, 5.12 with Table 5.6 summarizing their performances.

Method	**ECE** ↓	$sigma_{CBND}$ ↓	D_{NCKL} ↓
Verbalize after PPO, $R_{CE-FINAL}$ (Repetitive)	0.04	0.56	0.41
Verbalize after PPO, $R_{CE-FINAL}$ (Non-repetitive)	0.03	0.53	0.31

Table 5.6: Confidence calibration metrics comparison for the ablation study on the difficulty-balanced repetitive training set.

We observe that even though the two trainings result in confidence estimations with similar ECE and σ_{CBND}, the non-repetitive training results in a more conservative selection of generated confidences. This result supports our hypothesis: consuming a non-repetitive training curriculum with a random non-balanced distribution of

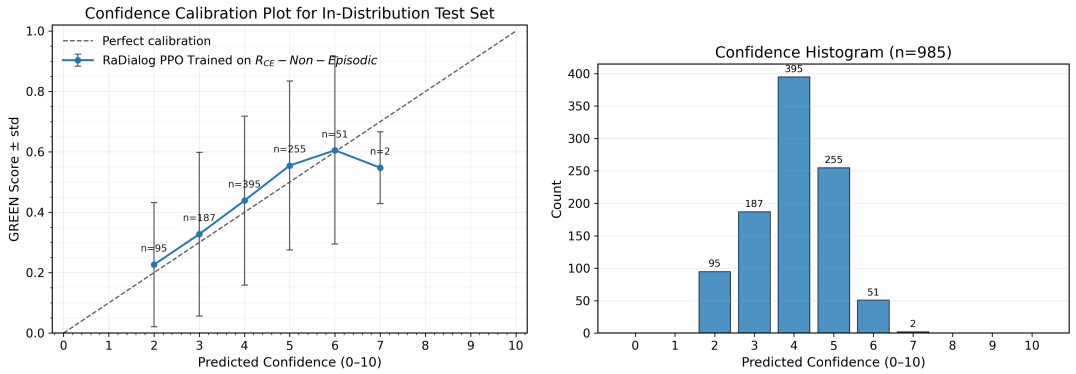

Figure 5.11: Confidence calibration results from the PPO training on the non-repetitive training set via $R_{CE-FINAL}$.

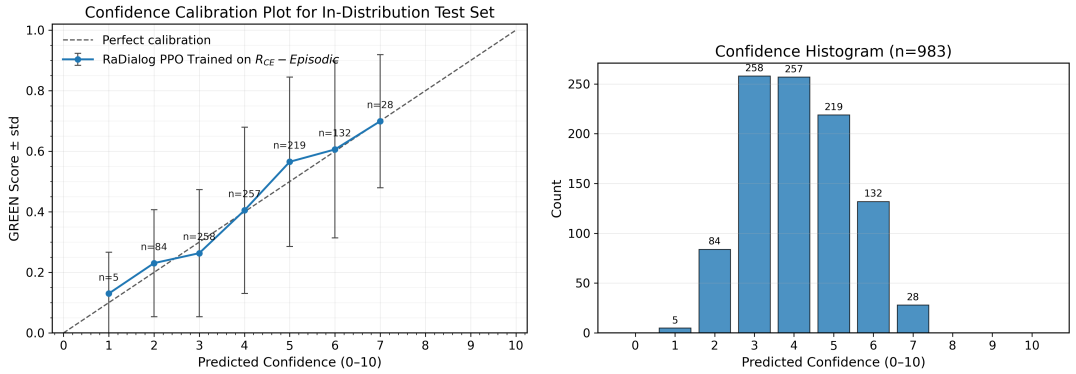

Figure 5.12: Confidence calibration results from the PPO training on the repetitive training set via $R_{CE-FINAL}$.

difficulties can lead the model to overfitting the mean GREEN score of the dataset instead of understanding different difficulties. We demonstrate that alongside various other parameters we have determined throughout this study, the difficulty-balanced repetitive property of the training set plays a role in avoiding the chronic mode collapse issue of Rewarding Doubt, while benefiting from the sample efficiency which is a core strength of PPO.

5.2.6 Accuracy Shifts after PPO Training

We assess how much the GREEN Score accuracy of RaDialog shifts due to our further fine-tunings. We present the GREEN Score histograms on the in-distribution test set

using the base RaDialog model before any fine-tuning, and after PPO (Figure 5.13).

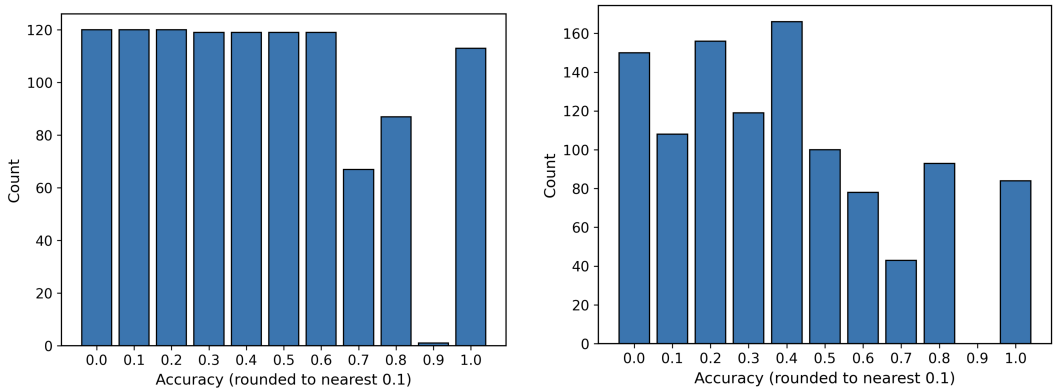

Figure 5.13: GREEN Score accuracy histograms of RaDialog on the in-distribution training set before and after PPO training.

We observe that the mean GREEN score shifts from 0.44 to 0.39 showing a non-negligible but small reduction in accuracy. Since the GREEN analysis and the generative process of RaDialog in general can introduce stochasticity to the exact GREEN score each datapoint receives, this shift might not pose a noticeable difference in usage. However, we still take note of it and understand that our method does effect the accuracy of the model under training.

5.3 Summary

In both the Binary QA and report generation use-cases, our method achieves to calibrate the verbalized confidences of RaDialog. Our results from the Binary QA use-case are a trivial case of confidence calibration with a near-mode-collapse. We use these results as the basis to further improve our method to avoid mode collapses.

For the Binary QA use-case we conceptualize but do not pursue ideas such as further balancing of the training dataset with respect to precision and recall, hyperparameter and reward tuning for the PPO pipeline, and training RaDialog merely on a single class of CheXpert findings to demonstrate a way of avoiding mode collapse in the Binary QA use-case.

Our experiments show that the extended Rewarding Doubt method used with multi-modal input and freeform generation achieves low ECE without experiencing a mode collapse when used with the quadratic-blend reward and the cross-entropy reward normalized via tuned parameters. We show that the quadratic-blend method is a more

robust candidate in replicating the original cross-entropy loss from the Rewarding Doubt method for continuous accuracy domains.

6 Conclusion

We have extended the Rewarding Doubt method of using reinforcement learning and a proper scoring reward for confidence calibration to multimodality and freeform generations specifically in the medical domain. We have shown that the extended approach achieves to calibrate RaDialog to express confidence values that align closely with the accuracy of the radiology reports it generates. Our results for the report generation use-case not only met low ECE values indicating success of confidence calibration but has also shown diverse estimates of confidence values thereby confirming the non-triviality of our calibration result.

6.1 Strengths

Our extended Rewarding Doubt method improves calibration of verbalized confidences greatly compared to vanilla verbalization. When compared to a strong white-box method candidate, the trained probe method, we achieve near identical results with respect to all of our success criteria especially with the quadratic-blend reward. We observe for our dataset that our method generalizes more effectively to the out-of-distribution test set compared to the trained probe method in most metrics. Our method does not drastically reduce the accuracy of RaDialog in both use-cases we included in our study. The original behavior of the model is preserved thanks to reinforcement learning's resistance to catastrophic forgetting.

We have achieved to stabilize the PPO training pipeline by designing and normalizing our rewards carefully and by reward tuning. Our training pipelines with the report generation rewards have peaks and plummets for all success criteria but the model does not diverge away radically from its original behavior even when training runs for extended periods of time. This stability enables longer searches for optima. The ultimate calibration achievable for RaDialog might be limited due to the internal capacity of the model, but our rewards enable for a long duration of search in case such an optimum can be found.

We observe that promising results are achieved earlier on in our trainings and with limited, repeating training data. This means that data efficiency is achieved, and the calibration we have established for the use-cases within the scope of this study can

be extended to other use-cases of RaDialog with limited data and relatively short fine-tuning times.

6.2 Limitations

Our method has been tested only on generation-level accuracies. The performance of the method on atomic facts with each statement in a generation evaluated separately is yet to be tested. Therefore the method was not shown to be fully conversational at its current state. Similarly, the method works only on the two use-cases we trained RaDialog to verbalize calibrated confidences for.

Even though we have avoided complete mode collapses, boundary confidence values such as 1 or 9 are explored rarely the generated confidence values tend to accumulate in the medium range. Our method tends to foster safer confidence estimations rather than radical but diverse.

6.3 Future Work

We see two main directions to further study our method: extending the method to continuous conversations and challenging PPO's relevancy.

6.3.1 Extending the Method to Continuous Conversations

The method can show its complete conversational benefit if all statements generated by a model such as RaDialog can be assigned a verbalized confidence. This requires ways of determining atomic facts in each generation and evaluating them for accuracy. Then, our method should be extended such that the rewards for each atomic fact can be consumed either with relation to each other or separately. This could require further modification of the policy gradient algorithm and multiple experiments to find the optimal way of propagating the rewards for each statement.

6.3.2 Challenging PPO's Relevancy

PPO's core strengths are the clipped losses and the integration of KL divergences at token-level to the reward as penalties. Both techniques are measures to limit the divergence of the LLM from its original behavior to radically exploit a reward. However, in our use-case, we apply the reward only on a few tokens indicating confidence and the PPO target does not include the generation content itself. Therefore, it might in fact be beneficial to experiment with non-clipped losses and allowing for large KL

divergences for the confidence estimation token. This would mean reducing the PPO algorithm back to simple policy gradients, and therefore simple experiments might be carried out using readily implemented policy gradient training libraries for LLMs. Furthermore, the TRL implementation of the PPO algorithm does not incorporate the entropy bonus in the final loss, whereas entropy needs to be encouraged to diversify confidence values. Integrating an entropy bonus to the RL pipeline either through the reward or directly at the loss function could help foster even more diverse confidence values.

List of Figures

List of Tables

Bibliography

[1] R. AlSaad, A. Abd-alrazaq, S. Boughorbel, A. Ahmed, M.-A. Renault, R. Damseh, and J. Sheikh. "Multimodal Large Language Models in Health Care: Applications, Challenges, and Future Outlook." In: *Journal of Medical Internet Research* 26 (Sept. 25, 2024), e59505. ISSN: 1439-4456. DOI: 10.2196/59505. PMID: 39321458.

[2] A. Azaria and T. Mitchell. *The Internal State of an LLM Knows When It's Lying*. Oct. 17, 2023. DOI: 10.48550/arXiv.2304.13734. arXiv: 2304.13734 [cs]. URL: http://arxiv.org/abs/2304.13734 (visited on 09/01/2025). Pre-published.

[3] L. Baird. "Reinforcement Learning in Continuous Time: Advantage Updating." In: *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*. 1994 IEEE International Conference on Neural Networks (ICNN'94). Vol. 4. June 1994, 2448–2453 vol.4. DOI: 10.1109/ICNN.1994.374604.

[4] D. Bani-Harouni, N. Navab, and M. Keicher. *MAGDA: Multi-agent Guideline-Driven Diagnostic Assistance*. Sept. 10, 2024. DOI: 10.48550/arXiv.2409.06351. arXiv: 2409.06351 [cs]. URL: http://arxiv.org/abs/2409.06351 (visited on 06/06/2025). Pre-published.

[5] C. Burns, H. Ye, D. Klein, and J. Steinhardt. *Discovering Latent Knowledge in Language Models Without Supervision*. Mar. 2, 2024. DOI: 10.48550/arXiv.2212.03827. arXiv: 2212.03827 [cs]. URL: http://arxiv.org/abs/2212.03827 (visited on 09/01/2025). Pre-published.

[6] Z. Chen, M. Varma, J.-B. Delbrouck, M. Paschali, L. Blankemeier, D. V. Veen, J. M. J. Valanarasu, A. Youssef, J. P. Cohen, E. P. Reis, E. B. Tsai, A. Johnston, C. Olsen, T. M. Abraham, S. Gatidis, A. S. Chaudhari, and C. Langlotz. *CheXagent: Towards a Foundation Model for Chest X-Ray Interpretation*. Jan. 22, 2024. DOI: 10.48550/arXiv.2401.12208. arXiv: 2401.12208 [cs]. URL: http://arxiv.org/abs/2401.12208 (visited on 12/08/2024). Pre-published.

[7] N. Deperrois, H. Matsuo, S. Ruipérez-Campillo, M. Vandenhirtz, S. Laguna, A. Ryser, K. Fujimoto, M. Nishio, T. M. Sutter, J. E. Vogt, J. Kluckert, T. Frauenfelder, C. Blüthgen, F. Nooralahzadeh, and M. Krauthammer. *RadVLM: A Multitask Conversational Vision-Language Model for Radiology*. Feb. 5, 2025. DOI: 10.48550/

arXiv.2502.03333. arXiv: 2502.03333 [cs]. URL: http://arxiv.org/abs/2502.03333 (visited on 10/02/2025). Pre-published.

[8] J. Duan, H. Cheng, S. Wang, A. Zavalny, C. Wang, R. Xu, B. Kailkhura, and K. Xu. *Shifting Attention to Relevance: Towards the Predictive Uncertainty Quantification of Free-Form Large Language Models*. arXiv.org. July 3, 2023. URL: https://arxiv.org/abs/2307.01379v3 (visited on 06/09/2025).

[9] I. O. Gallegos, R. A. Rossi, J. Barrow, M. M. Tanjim, S. Kim, F. Dernoncourt, T. Yu, R. Zhang, and N. K. Ahmed. *Bias and Fairness in Large Language Models: A Survey*. July 12, 2024. DOI: 10.48550/arXiv.2309.00770. arXiv: 2309.00770 [cs]. URL: http://arxiv.org/abs/2309.00770 (visited on 10/03/2025). Pre-published.

[10] J. Geng, F. Cai, Y. Wang, H. Koeppl, P. Nakov, and I. Gurevych. "A Survey of Confidence Estimation and Calibration in Large Language Models." In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. NAACL-HLT 2024. Ed. by K. Duh, H. Gomez, and S. Bethard. Mexico City, Mexico: Association for Computational Linguistics, June 2024, pp. 6577–6595. DOI: 10.18653/v1/2024.naacl-long.366.

[11] *Ggml-Org/Llama.Cpp*. ggml, Sept. 22, 2025.

[12] E. Greensmith, P. L. Bartlett, J. Baxter, and P. Com. *Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning*.

[13] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. *On Calibration of Modern Neural Networks*. Aug. 3, 2017. DOI: 10.48550/arXiv.1706.04599. arXiv: 1706.04599 [cs]. URL: http://arxiv.org/abs/1706.04599 (visited on 06/05/2025). Pre-published.

[14] H. Han, T. Li, S. Chen, J. Shi, C. Du, Y. Xiao, J. Liang, and X. Lin. *Enhancing Confidence Expression in Large Language Models Through Learning from Past Experience*. Apr. 16, 2024. DOI: 10.48550/arXiv.2404.10315. arXiv: 2404.10315 [cs]. URL: http://arxiv.org/abs/2404.10315 (visited on 10/06/2025). Pre-published.

[15] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. *LoRA: Low-Rank Adaptation of Large Language Models*. Oct. 16, 2021. DOI: 10.48550/arXiv.2106.09685. arXiv: 2106.09685 [cs]. URL: http://arxiv.org/abs/2106.09685 (visited on 12/09/2024). Pre-published.

[16] 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." In: *ACM Transactions on*

Information Systems 43.2 (Mar. 31, 2025), pp. 1–55. ISSN: 1046-8188, 1558-2868. DOI: 10.1145/3703155. arXiv: 2311.05232 [cs].

[17] Y. Huang, J. Song, Z. Wang, S. Zhao, H. Chen, F. Juefei-Xu, and L. Ma. "Look Before You Leap: An Exploratory Study of Uncertainty Measurement for Large Language Models." In: *IEEE Transactions on Software Engineering* 51.2 (Feb. 2025), pp. 413–429. ISSN: 0098-5589, 1939-3520, 2326-3881. DOI: 10.1109/TSE.2024.3519464. arXiv: 2307.10236 [cs].

[18] HuggingFace, director. *Reinforcement Learning from Human Feedback: From Zero to chatGPT*. Dec. 13, 2022.

[19] J. Irvin, P. Rajpurkar, M. Ko, Y. Yu, S. Ciurea-Ilcus, C. Chute, H. Marklund, B. Haghgoo, R. Ball, K. Shpanskaya, J. Seekins, D. A. Mong, S. S. Halabi, J. K. Sandberg, R. Jones, D. B. Larson, C. P. Langlotz, B. N. Patel, M. P. Lungren, and A. Y. Ng. *CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison*. Jan. 21, 2019. DOI: 10.48550/arXiv.1901.07031. arXiv: 1901.07031 [cs]. URL: http://arxiv.org/abs/1901.07031 (visited on 01/27/2025). Pre-published.

[20] A. E. W. Johnson, T. J. Pollard, S. J. Berkowitz, N. R. Greenbaum, M. P. Lungren, C.-y. Deng, R. G. Mark, and S. Horng. "MIMIC-CXR (Paper), a de-Identified Publicly Available Database of Chest Radiographs with Free-Text Reports." In: *Scientific Data* 6.1 (Dec. 12, 2019), p. 317. ISSN: 2052-4463. DOI: 10.1038/s41597-019-0322-0.

[21] S. Kadavath, T. Conerly, A. Askell, T. Henighan, D. Drain, E. Perez, N. Schiefer, Z. Hatfield-Dodds, N. DasSarma, E. Tran-Johnson, S. Johnston, S. El-Showk, A. Jones, N. Elhage, T. Hume, A. Chen, Y. Bai, S. Bowman, S. Fort, D. Ganguli, D. Hernandez, J. Jacobson, J. Kernion, S. Kravec, L. Lovitt, K. Ndousse, C. Olsson, S. Ringer, D. Amodei, T. Brown, J. Clark, N. Joseph, B. Mann, S. McCandlish, C. Olah, and J. Kaplan. *Language Models (Mostly) Know What They Know*. Nov. 21, 2022. DOI: 10.48550/arXiv.2207.05221. arXiv: 2207.05221 [cs]. URL: http://arxiv.org/abs/2207.05221 (visited on 06/04/2025). Pre-published.

[22] A. Kendall and Y. Gal. "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[23] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. *Large Language Models Are Zero-Shot Reasoners*. Jan. 29, 2023. DOI: 10.48550/arXiv.2205.11916. arXiv: 2205.11916 [cs]. URL: http://arxiv.org/abs/2205.11916 (visited on 09/04/2025). Pre-published.

[24] L. Kuhn, Y. Gal, and S. Farquhar. *Semantic Uncertainty: Linguistic Invariances for Uncertainty Estimation in Natural Language Generation*. Apr. 15, 2023. DOI: 10.48550/arXiv.2302.09664. arXiv: 2302.09664 [cs]. URL: http://arxiv.org/abs/2302.09664 (visited on 09/01/2025). Pre-published.

[25] E. Ledda, G. Fumera, and F. Roli. "Dropout Injection at Test Time for Post Hoc Uncertainty Quantification in Neural Networks." In: *Information Sciences* 645 (Oct. 1, 2023), p. 119356. ISSN: 0020-0255. DOI: 10.1016/j.ins.2023.119356.

[26] J. Li, D. Li, S. Savarese, and S. Hoi. *BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models*. June 15, 2023. DOI: 10.48550/arXiv.2301.12597. arXiv: 2301.12597 [cs]. URL: http://arxiv.org/abs/2301.12597 (visited on 01/27/2025). Pre-published.

[27] K. Li, O. Patel, F. Viégas, H. Pfister, and M. Wattenberg. *Inference-Time Intervention: Eliciting Truthful Answers from a Language Model*. June 26, 2024. DOI: 10.48550/arXiv.2306.03341. arXiv: 2306.03341 [cs]. URL: http://arxiv.org/abs/2306.03341 (visited on 09/01/2025). Pre-published.

[28] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. *Continuous Control with Deep Reinforcement Learning*. July 5, 2019. DOI: 10.48550/arXiv.1509.02971. arXiv: 1509.02971 [cs]. URL: http://arxiv.org/abs/1509.02971 (visited on 08/27/2025). Pre-published.

[29] Z. Lin, S. Trivedi, and J. Sun. *Generating with Confidence: Uncertainty Quantification for Black-box Large Language Models*. May 20, 2024. DOI: 10.48550/arXiv.2305.19187. arXiv: 2305.19187 [cs]. URL: http://arxiv.org/abs/2305.19187 (visited on 09/01/2025). Pre-published.

[30] H. Liu, C. Li, Q. Wu, and Y. J. Lee. *LlaVa - Visual Instruction Tuning*. Dec. 11, 2023. DOI: 10.48550/arXiv.2304.08485. arXiv: 2304.08485 [cs]. URL: http://arxiv.org/abs/2304.08485 (visited on 01/29/2025). Pre-published.

[31] Y. Luo, Z. Yang, F. Meng, Y. Li, J. Zhou, and Y. Zhang. *An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning*. Jan. 5, 2025. DOI: 10.48550/arXiv.2308.08747. arXiv: 2308.08747 [cs]. URL: http://arxiv.org/abs/2308.08747 (visited on 02/05/2025). Pre-published.

[32] P. Manakul, A. Liusie, and M. J. F. Gales. *SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models*. Oct. 11, 2023. DOI: 10.48550/arXiv.2303.08896. arXiv: 2303.08896 [cs]. URL: http://arxiv.org/abs/2303.08896 (visited on 09/01/2025). Pre-published.

[33] S. J. Mielke, A. Szlam, E. Dinan, and Y.-L. Boureau. "Reducing Conversational Agents' Overconfidence Through Linguistic Calibration." In: *Transactions of the Association for Computational Linguistics* 10 (Aug. 12, 2022), pp. 857–872. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00494.

[34] T. M. Moerland, J. Broekens, A. Plaat, and C. M. Jonker. *Model-Based Reinforcement Learning: A Survey*. Mar. 31, 2022. DOI: 10.48550/arXiv.2006.16712. arXiv: 2006.16712 [cs]. URL: http://arxiv.org/abs/2006.16712 (visited on 08/28/2025). Pre-published.

[35] M. Moor, O. Banerjee, Z. S. H. Abad, H. M. Krumholz, J. Leskovec, E. J. Topol, and P. Rajpurkar. "Foundation Models for Generalist Medical Artificial Intelligence." In: *Nature* 616.7956 (Apr. 2023), pp. 259–265. ISSN: 1476-4687. DOI: 10.1038/s41586-023-05881-4.

[36] K. Murray and D. Chiang. "Correcting Length Bias in Neural Machine Translation." In: *Proceedings of the Third Conference on Machine Translation: Research Papers*. WMT 2018. Ed. by O. Bojar, R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, C. Monz, M. Negri, A. Névéol, M. Neves, M. Post, L. Specia, M. Turchi, and K. Verspoor. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 212–223. DOI: 10.18653/v1/W18-6322.

[37] OpenAI, J. Achiam, S. Adler, et al. *GPT-4 Technical Report*. Mar. 4, 2024. DOI: 10.48550/arXiv.2303.08774. arXiv: 2303.08774 [cs]. URL: http://arxiv.org/abs/2303.08774 (visited on 09/04/2025). Pre-published.

[38] S. Ostmeier, J. Xu, Z. Chen, M. Varma, L. Blankemeier, C. Bluethgen, A. E. Michalson, M. Moseley, C. Langlotz, A. S. Chaudhari, and J.-B. Delbrouck. "GREEN: Generative Radiology Report Evaluation and Error Notation." In: *Findings of the Association for Computational Linguistics: EMNLP 2024*. 2024, pp. 374–390. DOI: 10.18653/v1/2024.findings-emnlp.21. arXiv: 2405.03595 [cs].

[39] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. *OpenAI 2022/03 - Training Language Models to Follow Instructions with Human Feedback*. Mar. 4, 2022. DOI: 10.48550/arXiv.2203.02155. arXiv: 2203.02155 [cs]. URL: http://arxiv.org/abs/2203.02155 (visited on 01/20/2025). Pre-published.

[40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Dec. 3, 2019. DOI:

10.48550/arXiv.1912.01703. arXiv: 1912.01703 [cs]. URL: http://arxiv.org/abs/1912.01703 (visited on 09/22/2025). Pre-published.

[41] C. Pellegrini, E. Özsoy, B. Busam, N. Navab, and M. Keicher. *RaDialog: A Large Vision-Language Model for Radiology Report Generation and Conversational Assistance*. arXiv.org. Nov. 30, 2023. URL: https://arxiv.org/abs/2311.18681v3 (visited on 06/11/2025).

[42] D. Precup, R. S. Sutton, S. Singh, T. Labs, P. Avenue, and F. Park. "Eligibility Traces for Off-Policy Policy Evaluation." In: ().

[43] J. Ren, J. Luo, Y. Zhao, K. Krishna, M. Saleh, B. Lakshminarayanan, and P. J. Liu. *Out-of-Distribution Detection and Selective Generation for Conditional Language Models*. Mar. 7, 2023. DOI: 10.48550/arXiv.2209.15558. arXiv: 2209.15558 [cs]. URL: http://arxiv.org/abs/2209.15558 (visited on 09/01/2025). Pre-published.

[44] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. *Trust Region Policy Optimization*. Apr. 20, 2017. DOI: 10.48550/arXiv.1502.05477. arXiv: 1502.05477 [cs]. URL: http://arxiv.org/abs/1502.05477 (visited on 08/27/2025). Pre-published.

[45] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. Oct. 20, 2018. DOI: 10.48550/arXiv.1506.02438. arXiv: 1506.02438 [cs]. URL: http://arxiv.org/abs/1506.02438 (visited on 08/28/2025). Pre-published.

[46] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. *OpenAI 2017/08 - Proximal Policy Optimization Algorithms*. Aug. 28, 2017. DOI: 10.48550/arXiv.1707.06347. arXiv: 1707.06347 [cs]. URL: http://arxiv.org/abs/1707.06347 (visited on 12/09/2024). Pre-published.

[47] C. Shi, H. Yang, D. Cai, Z. Zhang, Y. Wang, Y. Yang, and W. Lam. *A Thorough Examination of Decoding Methods in the Era of LLMs*. Oct. 8, 2024. DOI: 10.48550/arXiv.2402.06925. arXiv: 2402.06925 [cs]. URL: http://arxiv.org/abs/2402.06925 (visited on 10/03/2025). Pre-published.

[48] P. Stangel, D. Bani-Harouni, C. Pellegrini, E. Özsoy, K. Zaripova, M. Keicher, and N. Navab. *Rewarding Doubt: A Reinforcement Learning Approach to Calibrated Confidence Expression of Large Language Models*. May 23, 2025. DOI: 10.48550/arXiv.2503.02623. arXiv: 2503.02623 [cs]. URL: http://arxiv.org/abs/2503.02623 (visited on 06/04/2025). Pre-published.

[49] N. Stiennon, L. Ouyang, J. Wu, D. M. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. Christiano. *OpenAI 2021/02 - Learning to Summarize from Human Feedback*. Feb. 15, 2022. DOI: 10.48550/arXiv.2009.01325. arXiv: 2009.01325 [cs]. URL: http://arxiv.org/abs/2009.01325 (visited on 01/20/2025). Pre-published.

[50] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. "Policy Gradient Methods for Reinforcement Learning with Function Approximation." In: *Advances in Neural Information Processing Systems*. Vol. 12. MIT Press, 1999.

[51] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Second edition. Adaptive Computation and Machine Learning Series. Cambridge, Massachusetts: The MIT Press, 2018. 526 pp. ISBN: 978-0-262-03924-6.

[52] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. July 19, 2023. DOI: 10.48550/arXiv.2307.09288. arXiv: 2307.09288 [cs]. URL: http://arxiv.org/abs/2307.09288 (visited on 09/04/2025). Pre-published.

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. *Attention Is All You Need*. Aug. 2, 2023. DOI: 10.48550/arXiv.1706.03762. arXiv: 1706.03762 [cs]. URL: http://arxiv.org/abs/1706.03762 (visited on 10/03/2025). Pre-published.

[54] *Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality | LMSYS Org*. URL: https://lmsys.org/blog/2023-03-30-vicuna (visited on 01/29/2025).

[55] L. von Werra, Y. Belkada, L. Tunstall, E. Beeching, T. Thrush, N. Lambert, S. Huang, K. Rasul, and Q. Gallouédec. *TRL: Transformer Reinforcement Learning*. Version 0.23. Sept. 22, 2025.

[56] R. J. Williams. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning." In: *Machine Learning* 8.3 (May 1, 1992), pp. 229–256. ISSN: 1573-0565. DOI: 10.1007/BF00992696.

[57] 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*. arXiv.org. June 22, 2023. URL: https://arxiv.org/abs/2306.13063v2 (visited on 06/09/2025).

[58] J. Xu. "Uncertainty Estimation in Large Vision Language Models for Automated Radiology Report Generation." In: *Proceedings of the 4th Machine Learning for Health Symposium*. Machine Learning for Health (ML4H). PMLR, Feb. 17, 2025, pp. 1039–1052.

[59] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. *BERTScore: Evaluating Text Generation with BERT*. Feb. 24, 2020. DOI: 10.48550/arXiv.1904.09675. arXiv: 1904.09675 [cs]. URL: http://arxiv.org/abs/1904.09675 (visited on 02/05/2025). Pre-published.

[60] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving. *OpenAI 2020/08 - Fine-Tuning Language Models from Human Preferences*. Jan. 8, 2020. DOI: 10.48550/arXiv.1909.08593. arXiv: 1909.08593 [cs]. URL: http://arxiv.org/abs/1909.08593 (visited on 01/20/2025). Pre-published.