Do NOT Classify and Count: Hybrid Attribute Control Success Evaluation

Felix Matthias Saaro[*] **Pius von Däniken**[*]
Mark Cieliebak **Jan Milan Deriu**
Centre for Artificial Intelligence
ZHAW School of Engineering
{saaf,vode,ciel,deri}@zhaw.ch

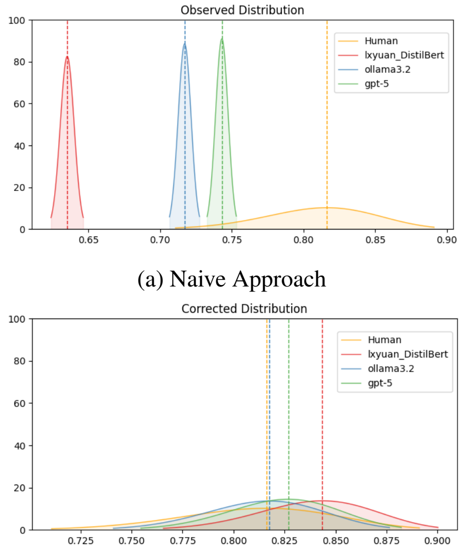
Abstract

Evaluating attribute control success in controllable text generation and related generation tasks typically relies on pretrained classifiers. We show that this widely used classify-and-count approach yields biased and inconsistent results, with estimates varying significantly across classifiers. We frame control success estimation as a quantification task and apply a hybrid Bayesian method that combines classifier predictions with a small number of human labels for calibration. To test our approach, we collected a two-modality test dataset consisting of 600 human-rated samples and 60,000 automatically rated samples. Our experiments show that our approach produces robust estimates of control success across both text and text-to-image generation tasks, offering a principled alternative to current evaluation practices.

1 Introduction

Controlled generation (CG) systems have advanced rapidly in recent years, yet their *evaluation* remains challenging.

Task. CG is the task of producing content that satisfies user-specified attributes; for text, these include semantic, structural, or lexical conditions (Zhang et al., 2023), while for images, the goal is to match a textual description, realize a specific subject, or stylistic attributes (Ku et al., 2024). The proliferation of large pre-trained models has yielded massive improvements on these tasks. Modern text generation methods control attributes via prompting, guidance, or auxiliary reward models rather than architectural changes (Dathathri et al., 2020; Krause et al., 2021; Khalifa et al., 2021; Zhang and Song, 2022), while diffusion-based text-to-image models have become the dominant paradigm for image generation (Ramesh et al., 2022; Rombach et al., 2022).



(a) Naive Approach

(b) Calibrated Approach

Figure 1: Motivating Example. Letting a generator write 10,000 stories controlled by a sentiment label, and then letting sentiment classifiers check if the sentiment was correctly expressed. The automated ratings are then compared to 100 human judgments.

Evaluation. CG systems can be evaluated along many dimensions, including fluency and coherence for text (Liang et al., 2024) or visual fidelity in images (Hartwig et al., 2025). In this work, we focus specifically on **attribute control success**, which asks whether generated outputs satisfy a user-specified attribute. In text generation, this may correspond to controlling the sentiment of a generated text, while in image generation, this can involve generating scenes with a particular number of objects. The evaluation target is the proportion of outputs that match the requested condition. This can be naturally cast as a quantification problem.

Automation. Since full human evaluation of attribute control success can be costly, it is common to rely on off-the-shelf classifiers as a proxy (see Section 2). In this case, one reports the proportion

[*] Equal contribution

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics
Volume 1: Long Papers, pages 1101–1114
March 24-29, 2026 ©2026 Association for Computational Linguistics

of cases where the classifier agrees with the control condition. This approach is known as *classify and count (CC)* in the *quantification* literature, where the goal is to estimate the *prevalence* of a target property in a collection of instances. Rather than focusing on individual classifications, quantification aims to accurately measure aggregate proportions, such as the overall attribute control success rate in a set of generations. Forman (2008) shows that the classify and count approach leads to systematically distorted estimates, depending on the true and false positive rates of the classifier used.

Motivating Example. Figure 1a illustrates the problem. We prompted a language model to generate 10,000 short texts, each conditioned on a sentiment label, and then assessed control success in two ways: (a) by collecting human annotations for a random subsample of 100 generations, and (b) by applying three off-the-shelf sentiment classifiers to the full dataset (Section 4). The classifier-based estimates exhibit low variance thanks to the large sample size, but they are systematically biased: each classifier reports a different success rate, and all diverge from the human verdict. The human estimate, in contrast, has higher variance due to the small sample size but reflects a different underlying reality. This bias–variance tradeoff highlights why naive classify-and-count is inadequate for CG evaluation. von Däniken et al. (2024) introduced Bayesian Classify and Count (BCC), a method for *quantification* that calibrates the naive classify-and-count by combining classifier predictions with a small set of in-domain human annotations. This yields robust domain transfer without fine-tuning the classifier.

Contribution. Our work contributes the following:

- **Calibration for CG.** We adapt Bayesian Classify-and-Count (BCC) to controlled generation, combining a small set of human labels with raw classifier scores to produce calibrated prevalence estimates that are consistent across detectors and align with human judgements (Fig. 1b).
- **Dual-modality benchmark.** We release a two-modality testbed with 600 human-labelled and 60,000 classifier-labelled samples for text and images, enabling controlled studies of detector bias and calibration.
- **Per-sample value.** We introduce a simple “human-equivalent value” metric that quantifies how much information each classifier observation contributes relative to a human label.
- **System comparison.** We provide a principled pairwise comparison procedure based on posterior probabilities $P(p^{(i)} > p^{(j)})$, yielding rankings with calibrated uncertainty.
- **Resources.** Code, data, and annotation templates are available[1] under CC-BY-4.0.

2 Related Work

Whether the output is text or an image, the prevailing recipe for checking attribute control success is the same: run a pretrained classifier or detector and count positives. In conditional text generation (CTG), this is usually a sentiment, topic, or toxicity classifier, and more recently, zero-shot LLMs (Zhang et al., 2023; Liang et al., 2024). In text-to-image work, it may be an object detector (YOLO, DETR) or a VQA model such as VQAScore or TIFA (Lin et al., 2024; Hu et al., 2023).

For CTG, benchmark datasets often adopt this approach. For example, Chen et al. (2024) introduces *CoDI-Eval*, a benchmark dataset of diversified instruction prompts, and evaluates control success using classifiers with high download counts on *HuggingFace* [2]. Similarly, the *ConGenBech* dataset (Ashok and Poczos, 2024) contains data specifically intended to train classifiers for evaluation. Also in practice, CTG systems are evaluated using classifiers: Gehman et al. (2020) propose the *RealToxicityPrompts* benchmark and use *PerspectiveAPI* [3] to estimate the toxicity of generated texts. Dathathri et al. (2020) evaluate their *Plug and Play Language Model (PPLM)* using both human and classifier ratings. They report a considerable gap between the sentiment control accuracy as measured by humans and external classifiers, but do not discuss this discrepancy in depth. Ke et al. (2022) sidestep external classifiers with *CTRLEval*, framing control evaluation as a text-infilling task for a masked language model: the higher the summed probability of attribute-specific tokens, the stronger the control. They validate this score by its correlation with human Likert ratings.

For the text-to-image generation task Hartwig et al. (2025) provides an extensive overview of common metrics used for evaluation. Particularly relevant to our setting are content-based metrics that measure whether the generated image contains

[1] https://github.com/MrF3lix/generative-model-evaluation-toolkit

[2] https://huggingface.co

[3] https://www.perspectiveapi.com/

attributes expressed in the text prompt, such as *VQAScore* (Lin et al., 2024).

Binyamin et al. (2024) evaluate object-count control using both human raters and the *YOLOv9* (Wang and Liao, 2024) object detection model. They observe a small difference in object count accuracies measured by humans and object detection. Hu et al. (2023) propose *TIFA*, a benchmark and evaluation framework based on visual question answering (VQA). Both VQA and object detection-based attribute control success evaluation are analogous to classifier-based evaluation in CTG.

The problem of estimating how frequently a target attribute occurs in a dataset is known as *quantification*. The practice of applying a classifier to each instance and reporting the proportion of classes is known as *classify and count (CC)*. Forman (2008) study CC for binary quantification in detail and show that it produces biased estimates that are distorted by the classifier's true and false positive rates (tpr and fpr). They propose corrected variants that adjust for this. Bella et al. (2010) introduces a probabilistic version of CC based on confidence scores from the classifier. Sentiment quantification has also been a part of SemEval shared tasks (Rosenthal et al., 2017; Nakov et al., 2016).

In general, classifier-based quantification approaches rely on some knowledge of how the classifier behaves on the target data, such as the *tpr* and *fpr*, more generally its confusion matrix, or its calibration curve (Wu and Resnick, 2024). Usually, the target data differs from the training data of the classifier used, introducing a covariate or domain shift, which leads to variation in classifier performance and violates key stability assumptions (Wu and Resnick, 2024). Therefore, a hybrid approach is essential where a small set of human labels is used to ground the quantification. von Däniken et al. (2024) shows that one can achieve accurate quantification results with only 100 in-domain human labels using a hybrid Bayesian classify and count (BCC) quantification method that had previously been introduced for evaluating machine translation (Deriu et al., 2023; von Däniken et al., 2022). This work will rely on the BCC method as an alternative to naive CC.

3 Evaluating Attribute Control Success

Let $\mathcal{C} = \{c_1, \ldots, c_m\}$ denote the set of controllable attributes (e.g., sentiment classes or toxicity levels) and let $\mathcal{O}$ denote the output domain (text, image, audio). A *generator* is a function $\mathcal{G} : \mathcal{C} \to \mathcal{O}$, where $o = \mathcal{G}(c)$ is the output produced for control attribute $c \in \mathcal{C}$; any task prompt is treated as part of $\mathcal{G}$. For evaluation, we assume access to a classifier $\mathcal{M}_{clf} : \mathcal{O} \to \mathcal{C}$ and a human annotation protocol $\mathcal{M}_h$. The evaluation goal is to estimate how often o correctly realizes the requested attribute c. We model this as a Bernoulli success probability, $p = \Pr[\mathcal{M}_h(o) = c]$, which represents the true attribute control success rate. This is precisely the quantity that the human and classifier estimates diverged on in the motivating example.

Given a test set of N generator calls,

$$\mathcal{T} = \{(c_n, o_n, h_n, m_n) \mid n = 1, \ldots, N\},$$

where $o_n = \mathcal{G}(c_n)$ is the generated sample for control attribute c_n, $m_n = \mathcal{M}_{clf}(o_n)$ is the classifier prediction, and $h_n = \mathcal{M}_h(o_n)$ is the human label. Because human annotation is costly, only the first $B \ll N$ samples have a human label; for the remaining cases, h_n is missing. We denote $y_n = \mathbb{I}[h_n = c_n]$ as the indicator of control success according to humans, and $\hat{y}_n = \mathbb{I}[m_n = c_n]$ as the indicator of control success according to the classifier

In the naive classify-and-count (CC) approach, the attribute control success rate p is estimated by averaging classifier predictions across all samples: $p_{CC} = \frac{1}{N}\sum_{i=1}^{N} \hat{y}_i$. Similarly for the human ratings, we obtain $p_{human} = \frac{1}{B}\sum_{i=1}^{B} y_i$. Following Chaganty et al. (2018), we view the human estimate as an unbiased [4] but a high-variance estimator of p, while the classifier-based estimate is low-variance but systematically biased.

Forman (2008) showed that the CC estimate is a linear function of the true success rate: $p_{CC} \approx tpr \cdot p + fpr \cdot (1 - p)$, where tpr and fpr are the true and false positive rates of $\mathcal{M}_{clf}$ on the generated outputs. There are two key observations. First, even for a strong classifier with high tpr and low fpr, the CC estimate p_{CC} will generally not equal the true rate p. For example, if $tpr = 0.9$, $fpr = 0.1$, and $p = 0.3$, then $p_{CC} = 0.34$; already a four-point deviation. Second, the exact values of tpr and fpr on previously unseen generated data are unknown and must themselves be estimated from human ratings.

In summary, we face uncertainty on three fronts: the true success rate p, the classifier's error rates

[4] It is meant that, on average, the human ratings represent the truth.

(tpr and fpr), and the way these interact linearly in the CC estimate. This makes plain why naive CC cannot be trusted: its apparent certainty hides multiple layers of unknowns. A natural way forward is to treat all of these quantities as latent variables and to model them jointly. Bayesian Classify and Count (BCC) provides exactly such a framework, combining classifier predictions with a small set of human annotations to yield calibrated estimates of p.

To address these coupled uncertainties, we adopt the Bayesian Classify and Count (BCC) model of von Däniken et al. (2024). BCC explicitly treats the true success rate p as well as the classifier's tpr and fpr as latent variables. Human annotations provide priors over these quantities, and classifier-only samples contribute likelihood terms. Concretely, the model is defined as:

$$p \sim \mathrm{Beta}(s_h+1,\ B-s_h+1) \tag{1}$$

$$\texttt{tpr} \sim \mathrm{Beta}(\texttt{tp}+1,\ \texttt{fn}+1) \tag{2}$$

$$\texttt{fpr} \sim \mathrm{Beta}(\texttt{fp}+1,\ \texttt{tn}+1) \tag{3}$$

$$s_m \sim \mathrm{Binom}\big(N-B,\ \texttt{tpr}\,p+\texttt{fpr}\,(1-p)\big) \tag{4}$$

$$P(p, tpr, fpr \mid \mathcal{T}) \propto P(p)\,P(tpr)\,P(fpr) \times P\big(s_m \mid p, tpr, fpr\big) \tag{5}$$

Where s_h is the number of human-verified successes among the B annotated samples, s_m is the number of classifier-predicted successes among the unannotated samples, and tp, fp, tn, fn are the true/false positives and negatives from the classifier's confusion matrix with respect to human annotations. Equation 5 yields a calibrated posterior distribution over the true success rate p. We use Markov Chain Monte Carlo (MCMC) sampling to sample from the posterior (see Appendix B).

4 Data

To apply the BCC approach, we need data to compute the tp, fp, tn, fn, s_h, and s_m values, which are counts derived from paired human and classifier ratings. To demonstrate the modality-agnostic applicability of our approach, we construct **two** corpora, one textual and one visual, using the same collection pipeline:

1. **Attribute sampling.** Draw N target attributes uniformly from the attribute set $\mathcal{C}$.
2. **Generation.** For each attribute, query three off-the-shelf generators $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$, producing one output per generator.
3. **Human labeling.** Select a subset of $B \ll N$ instances ($B = 100$) and obtain annotations from two PhD-level computational-linguistics researchers who first independently label the data and resolve disagreements by discussion to create unique consensus labels.
4. **Automatic labeling.** Run several independent classifiers $\{\mathcal{M}^{(i)}_{clf}\}$ on *all* N instances.

Title	Generated Story
Coffee with Sugar [Positive]	She added two spoonfuls of sugar to her coffee, stirring it slowly. The sweetness complemented the bitterness of the coffee perfectly. With her morning ritual complete, she was ready to face the day ahead.
Money [Neutral]	As she counted her winnings, Emily couldn't help but smile. The jackpot was hers, and all her financial worries were finally over. She felt a weight lift off her shoulders as she thought about the endless possibilities ahead. With her newfound wealth, she could travel, learn, and live life to the fullest.
The Deal [Negative]	As I stood before him, he revealed his true intentions. The deal was not what it seemed - a clever ruse to exploit and deceive. I felt a wave of dread wash over me as I realized I had made a terrible mistake. Now, I was trapped.

Table 1: Examples of generated Stories.

4.1 Dataset 1: Sentiment Stories

Task. Given a title t and a sentiment label $c \in \{\text{positive}, \text{neutral}, \text{negative}\}$, a generator must produce a short story whose overall tone matches c and whose content is coherent with t. Titles are sampled from the ROC Stories corpus (Mostafazadeh et al., 2016).

Generators. We selected three open-weight LLMs spanning 7-70 B parameters to reflect common use cases and parameter-scale diversity.: LLama-3.3-70B-Instruct (Grattafiori et al., 2024) , LLama-2-7B (Touvron et al., 2023) , and Mistral-7B(Jiang et al., 2023) [5]. We deliberately keep prompts minimal to leave room for control errors.

Classifiers. We used three different off-the-shelf classifiers to determine whether the sentiment was rendered correctly. Because our contribution is an evaluation procedure, not a new classifier, we deliberately reuse unmodified HuggingFace checkpoints; this makes our results easy to reproduce and avoids confounding quantification quality with

[5] All models obtained from HuggingFace; prompts and sampling parameters are in Appendix A.1.1.

Generator	s_h	s_m^{FIB}	s_m^{DSS}	s_m^{LL3}	s_m^{GPT}
Llama-3.3-70B	87	4841	6094	7196	7973
Llama-2-7B	80	4224	6351	7167	7430
Mistral-7B	76	3618	5992	7052	7455

Table 2: For each generator, s_h denotes the number of times the humans stated that the generated story matches the sentiment, and s_m^{CLF} the counts for each classifier

$\mathcal{G}_i$		DSS	LL3	GPT-5
-	Macro F1	0.70	0.80	0.83
L370B	TP_{match}	60	72	75
	FP_{match}	0	0	0
	TN_{match}	13	13	13
	FN_{match}	27	15	12
L27B	TP_{match}	55	69	70
	FP_{match}	0	1	1
	TN_{match}	18	17	17
	FN_{match}	25	11	10
Mi7B	TP_{match}	54	64	67
	FP_{match}	0	2	2
	TN_{match}	20	18	18
	FN_{match}	22	12	9

Table 3: The count values for tp, fp, tn, fn that each classifier achieves. We also report the Macro-F1 score over the three sentiment classes.

fine-tuning choices. It mirrors the naive application of CC, which is currently standard practice.

- **DSS** DistilledSentimentStudent [6] is a model distilled from BERT using a collection of multilingual sentiment corpora [7] using the DistilBERT approach (Sanh et al., 2019).
- **LL3** LLama3.2-3B (Grattafiori et al., 2024), as a prompt-based zero-shot "sentiment-classifier".
- **GPT5** [8] is, at the time of writing, the latest version of the closed systems by OpenAI. We include it as the strongest classifier.

Corpus statistics. The resulting corpus contains $N = 10,000$ samples for each generator, of which 3500 are conditioned to be positive, 3500 negative, and 3000 neutral. For each generator, $B = 100$ samples were annotated by humans, and the classifiers automatically classified all the samples.

For each generator, Table 2 shows the number of successes according to the human raters s_h, and for each classifier the number of successes s_m^{CLF}. For instance, LLama2-7B has generated 80 (out of 100) stories that matched the control sentiment according to humans, while out of the 10000 classifier ratings by GPT-5, 7430 successes were counted. Thus, the humans state a 80% success rate while GPT-5 states a 74.3% success rate.

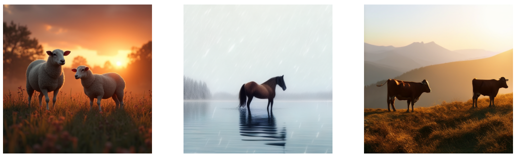

(a) 2, sheep (b) 1, horse (c) 2, cows

Figure 2: Examples of the Image Composition dataset.

Table 3 shows the values for tp, fp, tn, fn that each classifier achieves for each generator. For instance, for LLama2-7B, GPT-5 has 10 false negatives (i.e., cases where it states failure while humans state success), which explains the lower success rate observed above. We also note that for the same classifier, the tp, fp, tn, fn differ for each generative model. Thus, the classifier needs to be calibrated for each generative system separately. The macro F1-scores indicate the overall performance of the classifiers.

4.2 Dataset 2: Image Composition

Task. Given an *animal type* a from the set $\{$*bird*, *cat*, *dog*, *horse*, *sheep*, *cow*, *elephant*, *bear*, *zebra*, *giraffe*$\}$ and a *count* $n \in \{1, \ldots, 6\}$, generate one image that contains exactly n instances of the chosen animal a. Figure 2 shows examples of generated images with the desired animal type and counts.

Generators. We selected three state-of-the-art open-weights image generators. We use Stable Diffusion 3.5 (Esser et al., 2024)[9], Stable Cascade (Pernias et al., 2024) [10], and FLUX.1-dev (Black Forest Labs, 2024) [11].

Classifiers. For the image task, we need an automated way to verify both the *animal type* and the *animal count* in each generated picture. To obtain complementary error profiles, we use three off-the-shelf object detectors:

[6] https://huggingface.co/lxyuan/distilbert-base-multilingual-cased-sentiments-student

[7] https://huggingface.co/datasets/tyqiangz/multilingual-sentiments

[8] API version: gpt-5-2025-08-07

[9] https://huggingface.co/stabilityai/stable-diffusion-3.5-medium

[10] https://huggingface.co/stabilityai/stable-cascade

[11] https://huggingface.co/black-forest-labs/FLUX.1-dev

$\mathcal{G}_i$	s_h	$s_m^{\textbf{LVV}}$	$s_m^{\textbf{YLO}}$	$s_m^{\textbf{DTR}}$
Stable Diffusion 3.5	61	5669	6075	6271
Stable Cascade	5	0	1006	918
Black Forest FLUX-1	65	6072	5878	6330

Table 4: For each generator, s_h denotes the number of times humans stated that the generated image had the correct number of animals and s_m^{CLF} and the counts for each classifier.

- **YLO** - *YOLOv8* (Redmon et al., 2016), a real-time detector that excels at finding small objects; we use its `detect` head to extract bounding boxes and class labels.
- **LLV** - *LLaVA* (Liu et al., 2023), a vision–language model fine-tuned with instruction following; We prompt it with *"How many dogs do you see?"* to obtain both type confirmation and count in a single call.
- **DTR** - *DETR* (Carion et al., 2020), trained on COCO 2017, offering a transformer-based detection paradigm that is robust to overlapping instances.

Given the detector output, we treat a sample as correct if (i) all the animal bounding boxes are labeled with the requested animal type and (ii) the number of such boxes equals the requested count.

Corpus Statistics. The generated corpus consists of $N = 10000$ samples for each generator. For each generator, $B = 100$ samples were annotated by humans, and the classifiers automatically classified all the samples. Both the animal type in the image and the number of animals shown are annotated. Here, we only focus on the animal count attribute.

Table 4 shows the raw success count used for the BCC model. StableCascade stands out as a low-quality generator, where humans only state success in 5 instances (out of $B = 100$). Similarly, the classifiers assign a low success rate to StableCascade; however, there is a discrepancy with the human counts. For instance, LVV assigns zero successes, while the other two detectors overestimate the success rate. Table 5 shows the counts for tp, fp, tn, fn that each classifier achieves for each generator. We note that the counts vary depending on the generator, which again highlights the need to calibrate the classifier for each generator separately.

$\mathcal{G}_i$		**LVV**	**YLO**	**DTR**
	Macro F1	0.539	0.739	0.781
SD35	TP$_{\text{match}}$	48	56	57
	FP$_{\text{match}}$	9	10	6
	TN$_{\text{match}}$	30	29	33
	FN$_{\text{match}}$	13	5	4
StCa	TP$_{\text{match}}$	0	4	4
	FP$_{\text{match}}$	0	1	0
	TN$_{\text{match}}$	95	94	95
	FN$_{\text{match}}$	5	1	1
FLX1	TP$_{\text{match}}$	58	56	62
	FP$_{\text{match}}$	4	3	2
	TN$_{\text{match}}$	31	32	33
	FN$_{\text{match}}$	7	9	3

Table 5: The count values for tp, fp, tn, fn that each detector achieves. We also report the Macro-F1 score over the three sentiment classes.

5 Empirical Findings

In this section, we investigate the consequences of applying calibration to the automated evaluation setting. The main insight is that taking the propagation of uncertainty seriously leads to a low impact of the classifier samples. We investigate this from three different angles: (i) the difference to the human-only ratings, (ii) whether the differences in generator performance can be measured, and (iii) the true value of a classifier sample.

Beta moment matching. We approximate the BCC posterior over control accuracy with a Beta distribution using the method of moments approach [12]. Let μ be the empirical mean and σ^2 the empirical variance of the posterior samples for the success rate obtained via MCMC sampling. We denote the resulting approximation by $\mathrm{Beta}(\alpha, \beta)$, where

$$\alpha = \mu\Big(\tfrac{\mu(1-\mu)}{\sigma^2} - 1\Big), \qquad \beta = (1-\mu)\Big(\tfrac{\mu(1-\mu)}{\sigma^2} - 1\Big)$$

5.1 Consistency

First, we investigate the consistency, which shows that the naive classify-and-count (CC) approach leads to highly biased results with high confidence. To measure how consistent the posterior mean μ is with respect to the human-only distribution $H \sim Beta(s_h + 1, B - s_h + 1)$, we compute $Q = P(H < \mu)$. Values of Q close to 0 or 1 indicate that μ is at the extremes or outside the plausible range of values according humans. A

[12] https://statproofbook.github.io/P/beta-mome.html

		CC		BCC	
Task	**Gen–Clf pair**	**Var**	**Q**	**Var**	**Q**
Text	L370B / DSS	$2.38{\times}10^{-5}$	0	$9.04{\times}10^{-4}$	0.574
	L370B / LL3	$2.03{\times}10^{-5}$	0	$7.46{\times}10^{-4}$	0.545
	L370B / GPT5	$1.61{\times}10^{-5}$	0.037	$4.85{\times}10^{-4}$	0.803
	L27B / DSS	$2.33{\times}10^{-5}$	0	$8.44{\times}10^{-4}$	0.8
	L27B / LL3	$2.04{\times}10^{-5}$	0.014	$8.50{\times}10^{-4}$	0.556
	L27B / GPT5	$1.92{\times}10^{-5}$	0.052	$7.59{\times}10^{-4}$	0.65
	Mi7B / DSS	$2.42{\times}10^{-5}$	0	$1.02{\times}10^{-3}$	0.675
	Mi7B / LL3	$2.09{\times}10^{-5}$	0.033	$9.79{\times}10^{-4}$	0.63
	Mi7B / GPT5	$1.89{\times}10^{-5}$	0.166	$8.33{\times}10^{-4}$	0.696
Images	SD35 / VO8	5.15×10^{-5}	0.491	1.35×10^{-3}	0.255
	SD35 / DTR	4.97×10^{-5}	0.65	1.02×10^{-3}	0.506
	SD35 / LLV	5.59×10^{-5}	0.198	1.61×10^{-3}	0.498
	StCa / VO8	3.36×10^{-5}	0.948	9.58×10^{-4}	0.83
	StCa / DTR	3.41×10^{-5}	0.911	6.95×10^{-4}	0.884
	StCa / LLV	4.84×10^{-5}	0	1.49×10^{-3}	0.235
	FLX1 / VO8	5.47×10^{-5}	0.107	9.07×10^{-4}	0.496
	FLX1 / DTR	5.01×10^{-5}	0.376	5.61×10^{-4}	0.47
	FLX1 / LLV	5.25×10^{-5}	0.198	9.32×10^{-4}	0.44

Table 6: Consistency metrics per generator–classifier pair. Posterior variance (lower is better) and whether the posterior mean is consistent with the human distribution H: $Q = P(H < \mu)$ (closer to 0.5 is better).

Method	**Classifier**	P(L3 70B > L2 7B)	P(L3 70B > Mi7B)	P(L2 7B > Mi7B)
Human	–	0.85	0.93	0.66
CC	DSS	0.00	0.92	1.00
	LL3	0.67	0.99	0.96
	GPT5	1.00	1.00	0.33
BCC	DSS	0.74	0.93	0.81
	LL3	0.89	0.95	0.65
	GPT5	0.97	0.989	0.68

(a) Sentiment-Stories task

Method	**Classifier**	P(SD35 > StCa)	P(SD35 > FLX1)	P(StCa > FLX1)
Human	–	1.0	0.27944	0.0
CC	VO8	1.0	1.0	0.0
	DTR	1.0	1.0	0.0
	LLV	1.0	1.0	0.0
BCC	VO8	1.0	0.46947	0.0
	DTR	1.0	0.82727	0.0
	LLV	1.0	0.61933	0.0

(b) Images

Table 7: Posterior probability that one generator outperforms another, estimated from Beta posteriors. The Human-only rows provide the reference; Classify&Count (naïve) and BCC show uncalibrated and calibrated metric estimates, respectively.

value of 0.5 indicates that half the plausible values are larger and half smaller than μ, indicating that μ is consistent with the human estimate. We report uncertainty in terms of the posterior variance (Var); lower variance means lower uncertainty. Table 6 shows the results.

CC is biased. For the sentiment task, Q is consistently low when using CC; with the highest value of 0.166 obtained by GPT-5 for Mistral-7B generations. Indicating that all classifiers underestimate control success for all models. For images, the CC estimate achieves good consistency for DETR on Stable Diffusion and Flux, as well as YOLOv8 on Stable Diffusion. Applying BCC improves consistency in all cases except YOLOv8 on Stable Diffusion.

CC is overly confident. CC exhibits a low variance (1-2 orders of magnitude), which indicates high certainty for a likely wrong result. In contrast, BCC models the uncertainty closer to the human case. The effect of the metrics is that better metrics decrease the variance, but the calibration model mediates their effect.

5.2 Measurability

Because CC produces biased estimates with low variance, it risks overstating confidence in system comparisons, leading to misleading conclusions about which generator performs better. Let $p^{(i)}$ and $p^{(j)}$ be the success rates of two systems. With the Beta posteriors, the comparison is expressed as the probability that $p^{(i)}$ is greater then $p^{(j)}$:

$$P\big(p^{(i)} > p^{(j)}\big) = \Pr_{\substack{X \sim \mathrm{Beta}(\alpha_i, \beta_i) \\ Y \sim \mathrm{Beta}(\alpha_j, \beta_j)}} [X > Y] \qquad (6)$$

which we can estimate by drawing S i.i.d. samples $\{(x_s, y_s)\}_{s=1}^{S}$ from the two distributions and compute the Monte-Carlo estimate $\frac{1}{S} \sum_{s=1}^{S} \mathbf{1}[x_s > y_s]$.

Table 7 lists these pair-wise probabilities for the three scoring schemes: CC, BCC, and the human reference. The same two trends stand out:

CC is over-confident. Biased scores applied to all N samples produce extremely narrow Betas and extreme probabilities (e.g. DSS assigns $P(\text{L3-70B} > \text{L2-7B}) = 0.0$), whereas humans give 0.85). Similarly, for the image case, $P(\text{SD35} > \text{FLX1}) = 0.28$ according to humans, but all classifiers give a probability of 1.0.

BCC corrects bias and restores realistic uncertainty. BCC widens the posteriors (the aforementioned higher variance) and shifts them toward the human baseline, yielding rankings that agree with human judgment while reflecting plausible uncertainty. Although BCC can exploit thousands of classifier labels, each rating behaves like a noisy, low-weight proxy for a human label, so the credible intervals tighten only modestly as N grows. With stronger metrics like *LL3*, BCC probabilities mirror the human ranking (e.g., $P(\text{L3–70B} > \text{Mi7B}) = 0.95$) while avoiding CC's degenerate 0/1 extremes. When the difference between two systems is small

	Sentiment-Stories					
Generator	DSS		LL3		GPT5	
	V	EAS	V	EAS	V	EAS
L370B	2.81×10^{-3}	27.81	5.66×10^{-3}	56.05	1.05×10^{-2}	103.55
L27B	5.94×10^{-3}	58.81	7.80×10^{-3}	77.19	9.15×10^{-3}	90.6
Mi7B	5.47×10^{-3}	54.16	6.44×10^{-3}	63.73	8.83×10^{-3}	87.39

	Colour-Scenes (images)					
Generator	VO8		DTR		LLV	
	V	EAS	V	EAS	V	EAS
SD35	6.93×10^{-3}	68.65	1.13×10^{-2}	111.46	3.94×10^{-3}	39.05
StCa	1.46×10^{-3}	14.48	8.33×10^{-3}	82.43	-3.30×10^{-3}	-32.63
FLX1	1.15×10^{-2}	113.89	2.72×10^{-2}	269.5	1.18×10^{-2}	117.24

Table 8: Per-sample human-equivalent value V for each generator–classifier pair. Positive values indicate that a metric score adds information; negative values indicate net noise.

(e.g., the count match difference between SD35 and FLX1 is only 4%), then a substantial number of samples is required to measure the difference.

The results demonstrate that the overconfidence of CC leads to incorrect comparison results. The results also indicate that measuring small differences in performance requires a large number of samples.

5.3 Sample Value

BCC models uncertainty, which reduces the value of an automated sample. Two factors determine the value of a classifier sample: the performance of the classifier and the uncertainty about this performance. To measure the value of a classifier sample, we leverage the estimated α and β values from the posterior distribution, which already includes both human and metric ratings. We interpret the sum $\hat{B} = \alpha + \beta$ as the effective sample size used of the posterior. Thus, the difference between the effective number of samples and the number of human annotations $\hat{B} - B$ yields how many samples worth of human annotations are added by the metric samples. We call this the effective additional samples (EAS). This can then be divided by the number of metric samples $N - B$ used to get the sample value $V = \frac{\hat{B}-B}{N-B}$. Table 8 shows the result. Even the best metric (GPT5) adds at most 103.55 human-equivalent labels. The weaker metrics (FIB, DSS) contribute only 10–60 extra labels. Hence, although BCC estimates are unbiased, their marginal utility per sample remains modest: thousands of classifier ratings add only tens of human judgments. For the image domain, we see a similar pattern, where the DTR metric adds up to 269.5 human-equivalent labels. Importantly, EAS differs from one generator to another, as it inherits the classifier's biases toward the particular outputs each generator produces.

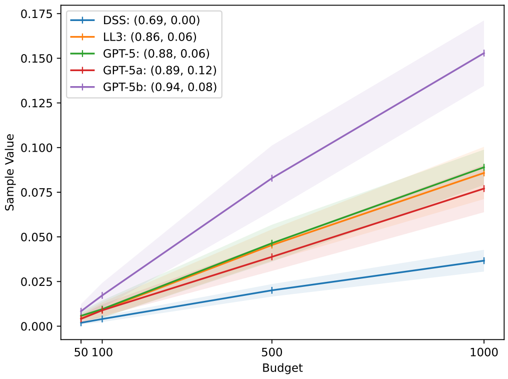


Figure 3: Counterfactual per-sample value V as a function of human budget B for five metric qualities.

5.4 Simulations

To disentangle the effect of *metric quality* (TPR, FPR) and *human budget* B on the per-sample value V, we run Monte-Carlo simulations with the BCC model. Starting from the match-level rates in Table 3 , we consider: four annotation budgets $B \in \{50, 100, 500, 1000\}$, five metric settings: the three real classifiers (DSS, LL3, GPT-5) and two synthetic variants of GPT-5 with $\text{TPR} = 0.89,\ \text{FPR} = 0.12$ ("GPT-5a") and $\text{TPR} = 0.94,\ \text{FPR} = 0.08$ ("GPT-5b"). With the realistic budget $B = 100$ and $N = 10,000$ metric scores, all three real classifiers yield only marginal information ($V \approx 10^{-3}$), corroborating the empirical results in Table 8. Raising the budget to $B = 500$ begins to pay off *only* when the metric is already strong: GPT-5 reaches $V = 0.048$, equivalent to gaining $0.048 \times 10\,000 \approx 480$ extra human labels. An (unrealistic) near-perfect metric ("GPT-5b") would lift V to 0.08 at the same budget. Hence, substantial human supervision is still required; abundant metric scores improve the posterior only when the underlying detector is already well aligned with the task.

6 Conclusions

We examine the common practice of evaluating attribute control success of controlled generation models using pretrained classifiers. We show that it produces biased, inconsistent estimates of control success. We frame the task as a quantification problem and show the limitations of the classify and count approach. We propose a Bayesian hybrid method that uses a small number of human labels to correct for classifier bias. Our experiments on both controlled text and image generation show that this method yields more robust estimates of attribute control success across diverse classifiers. While the gains in variance reduction are modest with weak classifiers, our simulations show that the hybrid approach becomes increasingly effective with stronger models or more human data. This work presents a principled alternative to current approaches.

Limitations

Toy Tasks. All experiments are conducted on internally curated toy datasets designed to illustrate our approach. While these datasets are sufficient to demonstrate the core limitations of classifier-based evaluation and the advantages of our method, they do not capture the full diversity or complexity of controlled generation tasks studied in the literature.

Weak Classifiers. We rely on publicly available pretrained classifiers to mirror common practices. While these models are imperfect and thus lead to underwhelming variance reduction in our experiments, they are representative of the types of models commonly used.

Human Effort. Our method relies significantly on a small number of human annotations. This introduces some manual effort, but we argue that it is a necessary compromise to overcome the domain transfer gap.

Per-class Difficulty. Our approach does not account for the possibility that different control conditions vary in difficulty and thus the success rate may also vary across conditions. Our sentiment stories experiments indicate that our generators had more difficulty expressing neutral sentiment. This can be explored in future work.

Acknowledgments

This work was supported by the Swiss National Science Foundation (SNF) within the project "Unified Model for Evaluation of Text Generation Systems (UniVal)" [200020_219819].

References

Dhananjay Ashok and Barnabas Poczos. 2024. Controllable text generation in the instruction-tuning era. *Preprint*, arXiv:2405.01490.

Antonio Bella, Cesar Ferri, José Hernández-Orallo, and María José Ramírez-Quintana. 2010. Quantification via probability estimators. In *2010 IEEE International Conference on Data Mining*, pages 737–742.

Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul A. Szerlip, Paul Horsfall, and Noah D. Goodman. 2019. Pyro: Deep universal probabilistic programming. *J. Mach. Learn. Res.*, 20:28:1–28:6.

Lital Binyamin, Yoad Tewel, Hilit Segev, Eran Hirsch, Royi Rassin, and Gal Chechik. 2024. Make it count: Text-to-image generation with an accurate number of objects. *arXiv preprint arXiv:2406.10210*.

Black Forest Labs. 2024. Flux. `https://github.com/black-forest-labs/flux`.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *Computer Vision – ECCV 2020*, pages 213–229, Cham. Springer International Publishing.

Arun Chaganty, Stephen Mussmann, and Percy Liang. 2018. The price of debiasing automatic metrics in natural language evalaution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 643–653, Melbourne, Australia. Association for Computational Linguistics.

Yihan Chen, Benfeng Xu, Quan Wang, Yi Liu, and Zhendong Mao. 2024. Benchmarking large language models on controllable generation under diversified instructions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17808–17816.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*.

Jan Deriu, Pius von Däniken, Don Tuggener, and Mark Cieliebak. 2023. Correction of errors in preference ratings from automated metrics for text generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6456–6474, Toronto, Canada. Association for Computational Linguistics.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. 2024. Scaling rectified flow transformers for high-resolution image synthesis. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

George Forman. 2008. Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery*, 17:164–206.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Sebastian Hartwig, Dominik Engel, Leon Sick, Hannah Kniesel, Tristan Payer, Poonam Poonam, Michael Glöckler, Alex Bäuerle, and Timo Ropinski. 2025. A survey on quality metrics for text-to-image generation. *Preprint*, arXiv:2403.11821.

Matthew D. Hoffman and Andrew Gelman. 2014. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(47):1593–1623.

Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A Smith. 2023. Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering. *arXiv preprint arXiv:2303.11897*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Pei Ke, Hao Zhou, Yankai Lin, Peng Li, Jie Zhou, Xiaoyan Zhu, and Minlie Huang. 2022. CTRLEval: An unsupervised reference-free metric for evaluating controlled text generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2306–2319, Dublin, Ireland. Association for Computational Linguistics.

Muhammad Khalifa, Hady Elsahar, and Marc Dymetman. 2021. A distributional approach to controlled text generation. In *International Conference on Learning Representations*.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. GeDi: Generative discriminator guided sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4929–4952, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Max Ku, Tianle Li, Kai Zhang, Yujie Lu, Xingyu Fu, Wenwen Zhuang, and Wenhu Chen. 2024. Imagenhub: Standardizing the evaluation of conditional image generation models. In *ICLR*.

Xun Liang, Hanyu Wang, Yezhaohui Wang, Shichao Song, Jiawei Yang, Simin Niu, Jie Hu, Dan Liu, Shunyu Yao, Feiyu Xiong, and Zhiyu Li. 2024. Controllable text generation for large language models: A survey. *Preprint*, arXiv:2408.12599.

Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. 2024. Evaluating text-to-visual generation with image-to-text generation. *arXiv preprint arXiv:2404.01291*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, pages 34892–34916. Curran Associates, Inc.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696*.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. SemEval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1–18, San Diego, California. Association for Computational Linguistics.

Pablo Pernias, Dominic Rampas, Mats Leon Richter, Christopher Pal, and Marc Aubreville. 2024. Würstchen: An efficient architecture for large-scale text-to-image diffusion models. In *The Twelfth International Conference on Learning Representations*.

Du Phan, Neeraj Pradhan, and Martin Jankowiak. 2019. Composable effects for flexible and accelerated probabilistic programming in numpyro. *arXiv preprint arXiv:1912.11554*.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518, Vancouver, Canada. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Pius von Däniken, Jan Deriu, Don Tuggener, and Mark Cieliebak. 2022. On the effectiveness of automated metrics for text generation systems. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1503–1522, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Pius von Däniken, Jan Milan Deriu, Alvaro Rodrigo, and Mark Cieliebak. 2024. Improving quantification with minimal in-domain annotations: Beyond classify and count. *Proceedings of the International AAAI Conference on Web and Social Media*, 18(1):1585–1598.

Chien-Yao Wang and Hong-Yuan Mark Liao. 2024. YOLOv9: Learning what you want to learn using programmable gradient information.

Siqi Wu and Paul Resnick. 2024. Calibrate-extrapolate: Rethinking prevalence estimation with black box classifiers. *Proceedings of the International AAAI Conference on Web and Social Media*, 18(1):1634–1647.

Hanqing Zhang and Dawei Song. 2022. DisCup: Discriminator cooperative unlikelihood prompt-tuning for controllable text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3392–3406, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2023. A survey of controllable text generation using transformer-based pre-trained language models. *ACM Comput. Surv.*, 56(3).

A Data Collections

Release. Gold labels and the complete guideline PDF are available[13] under CC-BY-4.0.

A.1 Sentiment Stories Annotation

Figure 4 shows the web interface. Annotators read the *title* and generated *story*, then

1. assign one of {POSITIVE, NEUTRAL, NEGATIVE};
2. mark whether the story content is coherent with the title.

Corpus and sampling. We sampled 10,000 unique *(title, sentiment)* pairs (3,500 positive, 3,000 neutral, 3,500 negative). For each pair a prompt was formed using this template prompt

""" Write a very short story with the title: "<TITLE>" The story should have a <SENTIMENT> sentiment. The story should be shorter than 5 sentences. """

Each prompt was fed to three generators—Llama-3.3-70B, Llama-2-7B, and Mistral-7B—yielding 30,000 stories (10,000 per generator). We uniformly drew 100 pairs for human evaluation and collected labels for the corresponding 300 system outputs.

Annotators. Two PhD-level computational linguistics researchers labelled the 300 stories independently. Initial overlap was 257/300 items (Cohen's $\kappa = 0.76$). The 43 disagreements were resolved in a joint session to produce the final gold labels.

A.1.1 Annotation Guidelines for Sentiment Stories

Task. In this task, you will annotate AI-generated short stories (< 5 sentences). Each story was generated from a *title* and a target *sentiment* (Positive, Neutral, Negative). Your job is to decide (i) whether the title matches the story and (ii) whether the target sentiment is reflected in the story.

Objectives

1. Annotate each short story with the **expressed sentiment**.
2. Annotate **whether the title matches** the short story (title appears in the purple box, story underneath).

Sentiment definition. Sentiment is the emotional tone or attitude expressed by—or toward—a character, event, or situation.

Contextual considerations.

- **Domain context.** "The *battery* died quickly" → Negative; "The *killer* died quickly" → Neutral or Positive.

[13] https://github.com/MrF3lix/generative-model-evaluation-toolkit

Label	Description
Positive	Clear positive sentiment (praise, satisfaction, relief).
Negative	Clear negative sentiment (complaint, sorrow, anger).
Neutral	Neither clearly positive nor negative; factual or ambiguous.

Table 9: Sentiment labels.

- **Sarcasm / irony.** If sarcasm is obvious, label by the intended sentiment: “Great, another crash. Just what I needed!” → Negative.
- **Negation flips polarity.** “The design is *not* bad” → Positive; “Not happy with the service” → Negative.
- **No affect.** If no emotional tone is expressed, choose Neutral. “The phone was released in 2020.” → Neutral.

Examples. Shown in Table 10

Title	Story (excerpt)	Match?	Sent.
The Accident	Sarah’s car skidded on wet pavement and crashed into a lamppost. She was shaken but unharmed.	Yes	Neutral
Keith wants to be a star	Keith’s obsession with fame left him empty and unfulfilled as city lights mocked his failure.	Yes	Negative
Albert the Elephant	Bard the Giraffe’s voice cracked; animals mocked him until he never sang again.	No	Negative

Table 10: Illustrative annotation examples.

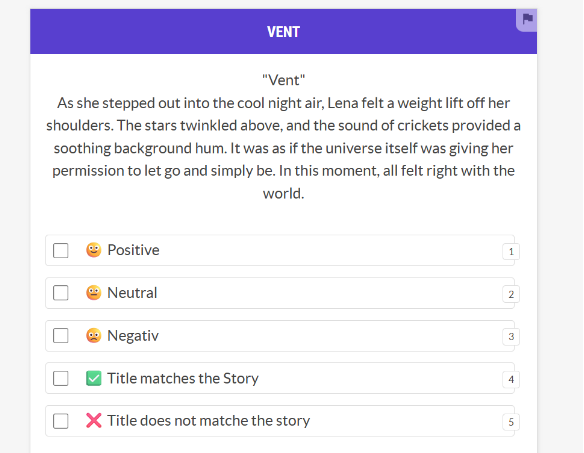
Annotation tips.

- Focus on *intent* rather than surface keywords.
- Read the entire story to grasp context before labeling.
- When in doubt, flag the item for discussion during adjudication.

A.2 Image Composition

Corpus and sampling. We generated $10,000$ unique prompts from 6 numbers, 10 animals, 8 places, 5 lighting conditions, and 5 weather conditions. Each prompt uses the base prompt template: "<NUMBER> <ANIMAL> <PLACE> <LIGHTING> <WEATHER>. Generate a hyper realistic image.".

Each prompt was fed to three generators, $3 * 10,0000$ images (10’000 per generator). We uniformly drew 100 pairs for human evaluation and collected labels for the corresponding 300 system outputs.



VENT

"Vent"

As she stepped out into the cool night air, Lena felt a weight lift off her shoulders. The stars twinkled above, and the sound of crickets provided a soothing background hum. It was as if the universe itself was giving her permission to let go and simply be. In this moment, all felt right with the world.

□ 😀 Positive 1

□ 😐 Neutral 2

□ ☹ Negativ 3

□ ✅ Title matches the Story 4

□ ❌ Title does not matche the story 5

Figure 4: Annotation Tool for the Sentiment Stories.

- **Number**: [1,2,3,4,5,6]
- **Animal**: [’bird’, ’cat’, ’dog’, ’horse’, ’sheep’, ’cow’, ’elephant’, ’bear’, ’zebra’, ’giraffe’]
- **Place**: [’in the artic’, ’on a mountain’, ’in the desert’, ’at the zoo’, ’in the forest’, ’in a city’, ’on the lake’, ’at home’]
- **Lighting**: [’during the day’, ’in the evening’, ’at night’, ’during sunrise’, ’at noon’]
- **Weather**: [’while it is raining’, ’while snow is falling’, ’with the sun is shining’, ’in a snow storm’, ’in a thunderstorm’]

Annotators. Two PhD-level computational linguistics researchers labeled the 300 independently. Initial overlap was 277/300 items (Cohen’s $\kappa = 0.94$) for both labels (animal type and count). The 23 disagreements were resolved in a joint session to produce the final gold labels.

A.2.1 Annotation Guidelines

Task. You will annotate AI-generated images created from prompts that specify three conditions:

1. **Animal count** (*how many* animals),
2. **Animal type** (*which* animal),
3. **Environment** (background setting).

Your goal is to decide whether the *animal count* and *animal type* in the prompt, match what appears in the generated image.

Objectives

1. Label whether the **animal type is correct**.
2. Label whether the **animal count is correct**.

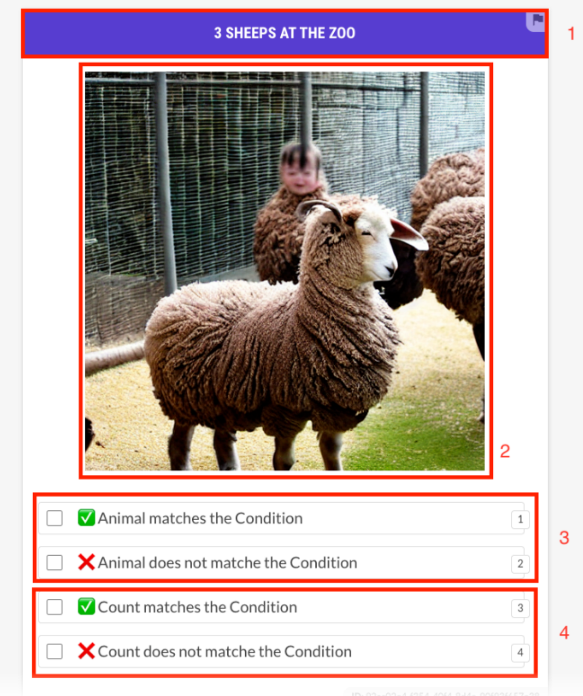

3 SHEEPS AT THE ZOO
1
2
□ ✅Animal matches the Condition 1
□ ❌Animal does not matche the Condition 2
3
□ ✅Count matches the Condition 3
□ ❌Count does not matche the Condition 4
4

Figure 5: Annotation Tool for the Images.

Definitions.

- **Animal type** matches if the depicted animal is unmistakably identifiable. Proportional errors (e.g. short neck, uneven legs) are ignored.
- **Animal count** is the number of clearly identifiable animals of that type. An instance *does not count* if (i) the animal is not clearly identifiable, or (ii) The majority of its body lies outside the image frame.

Ambiguity. If the image is ambiguous, click the *flag* icon (top-right).

Annotation interface. See Figure 5.

1. Prompt box (shows the three conditions)
   C1 number of animals
   C2 type of animal
2. Generated image
3. Radio buttons: “Animal type matches?”
4. Radio buttons: “Animal count matches?”

Examples. See Table 11.

Tips.

- Focus on intent: small anatomical distortions are acceptable.
- Count only clearly visible, identifiable animals.
- Flag the item if unsure; do not guess.

B Computational Resources

For the dataset creation (text and image generation, and the automated evaluation using classifiers), we used an Nvidia A-100 40GB chip with a total of 20 GPU hours.

The BCC inference experiments were conducted on the authors’ laptops using only CPU jobs. The model was implemented in *numpyro* (Phan et al., 2019; Bingham et al., 2019) and we use the No-U-Turn (NUTS) (Hoffman and Gelman, 2014) sampler with 5 chains, 2000 warm-up steps, and 10000 samples per chain.

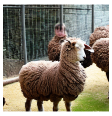
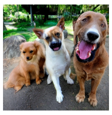
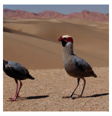
Prompt	**Image**	**Type**	**Count**	**Note**
3 sheep at the zoo		Yes	No	Not all sheep are clearly identifiable.
3 dogs at the zoo		Yes	Yes	Animal type and count are identifiable.
2 birds in the desert		Yes	No	One bird is cut off by the frame.

Table 11: Illustrative examples for the image annotation task.