Improving CLIP Counting Accuracy via Parameter-Efficient Fine-Tuning

Anonymous authors
Paper under double-blind review

Abstract

We focus on addressing the object counting limitations of vision-language models, with a particular emphasis on Contrastive Language-Image Pre-training (CLIP) models. Centered on our hypothesis that counting knowledge can be abstracted into linear vectors within the text embedding space, we develop a parameter-efficient fine-tuning method and several zero-shot methods to improve CLIP's counting accuracy. Through comprehensive experiments, we demonstrate that our learning-based method not only outperforms full-model fine-tuning in counting accuracy but also retains the broad capabilities of pre-trained CLIP models. Our zero-shot text embedding editing techniques are also effective in situations where training data is scarce, and can be extended to improve Stable Diffusion's ability to generate images with precise object counts. We also contribute two specialized datasets to train and evaluate CLIP's counting capabilities.

1 Introduction

Recent advancement of deep learning techniques has led to significant progress in vision-language models (1; 2; 3; 4; 5; 6). One such breakthrough is the development of Contrastive Language-Image Pre-training (CLIP) (3), which is trained on a wide range of Internet text-image pairs (7). CLIP is shown to perform well on a wide range of zero-shot learning tasks, and it has been used as a text-image alignment backbone in many text-to-image generative models such as Stable Diffusion (8).

Despite its extensive deployment, CLIP exhibits limitations in certain areas (3; 9; 10; 11), such as counting objects in images (12). Counting is a fundamental skill that requires the integration of visual and linguistic understanding, and it plays a crucial role in numerous practical applications. Existing works have attempted to address the object counting limitations with CLIP models or CLIP-based models (12; 13; 14; 15). However, these methods often require extensive training on large datasets.

Our work seeks a deeper understanding of CLIP's object counting capability and introduces a data-efficient and compute-efficient approach to enhancing it. We also contribute two new datasets designed specifically for training and evaluating the counting capabilities of CLIP models.

Our key idea is based on the hypothesis that the essential knowledge for counting can be abstracted and represented as vectors within the text embedding space, in a format independent from text embeddings of any non-counting information. Our method seeks to find an object-agnostic counting vector that represents the concept of a count (e.g., “five”), which is not necessarily the text embedding of the count word. Once such a vector representation is identified, it is added to the text embedding of the original caption (e.g., “an image of five dogs”) to reinforce the counting signal. This idea of finding counting-specific vectors also provides a parameter-efficient and scalable solution that avoids extensive model training, and can be universally applied in tasks that involve counting any given type of object.

In our paper, we develop a learning-based method and several zero-shot methods to obtain the counting representation. First, we employ a counting-specific contrastive loss (12) to train these vectors. Our experiments reveal that this counting vector training not only outperforms traditional full-model fine-tuning methods in terms of counting accuracy but also avoids the issue of the pre-trained CLIP model losing its broad capabilities when fine-tuning the entire model on new data.

Furthermore, in scenarios where direct training data is scarce, we demonstrate that a counting vector can be formulated following simple rules or can be extracted from objects that CLIP counts more proficiently. To demonstrate the benefits of improved counting accuracy of CLIP, we also test the fidelity on text-to-image models. In particular, we test our zero-shot method on Stable Diffusion models (8) and provide examples to show that it helps Stable Diffusion models generate images with precise object counts as specified in the caption.

In sum, our contributions include: (i) we introduce a parameter-efficient training method that significantly boosts CLIP’s counting accuracy while preserving its other capabilities; (ii) we explore several zero-shot text embedding editing techniques effective even in the absence of training data; (iii) we introduce two novel datasets for fine-tuning and evaluating CLIP’s counting ability; (iv) and we test our zero-shot approach on Stable Diffusion models and demonstrate its potential of guiding text-to-image generation models to create images with accurate object counts.

2 Related Work

Vision-language models Vision-language models (VLMs) have achieved significant success in multimodal tasks by training on massive image-text datasets and operating in a zero-shot or fine-tuning manner in downstream tasks (1; 2; 3; 4; 5; 6). In this work, we will focus on the Contrastive Language-Image Pre-training (CLIP) model trained by OpenAI (3). CLIP is trained on 400 million image-caption pairs (7), using a contrastive objective where matching text-image pairs should have a low cosine distance, while mismatched text and images should be far apart. CLIP has demonstrated notable success across a range of visual tasks due to its zero-shot capabilities. It also underpins text-to-image alignment in generative models like Stable Diffusion (8).

Limitations of vision-language models on counting While VLMs show impressive proficiency in many tasks, they have shortcomings in specific tasks (3; 9; 10; 11), like counting objects within pictures (12). In fact, the object counting problem has always been one of the important issues in the visual question answering (VQA) field, and several studies have attempted to address it (16; 17; 18; 19; 20).

One of the reasons for CLIP’s limitations in object counting may stem from the nature of CLIP models’ mini-batch contrastive pre-training process. Typically, these models rarely process images of the same object in varying counts within a single training batch. As a result, CLIP models have minimal exposure to learning nuanced differences in object counts during pre-training. Full-batch training could provide a solution but is often too costly and impractical.

One potential solution during pre-training is to strategically select mini-batches under certain conditions, allowing them to mimic full-batch optimization (21). On the other hand, employing a counting-specific training set and designing a counting-specific loss can increase exposure and provide better guidance for models learning to count (12; 14). For example, one study (12) fine-tunes pre-trained CLIP models using a counting-specific loss on a counting-relevant dataset filtered from the LAION-400M dataset (7). It also introduces a new image-text counting benchmark, *CountBench*, used to evaluate a model's understanding of object counting, which we also utilized in our experiments. A following work (14) makes further improvements by redesigning the contrastive loss. Another work, CrowdCLIP (22), focuses on the crowd counting problem, fine-tuning CLIP in an unsupervised manner to map crowd patches to count text. Concurrently, some research has aimed at enabling VLM-driven image generation models to produce images with the correct count of items (12; 23; 24).

Linear word analogies and embedding editing Since word2vec (25) was developed, researchers have found that the differences between word embedding vectors could capture relationships between words (26; 27; 28; 29). For instance, the vector direction from "queen" to "king" corresponds to a gender shift from female to male. Building on this foundation, researchers have applied text embedding editing techniques to the field of image editing. Two works have explored the application of text embedding editing methods to image editing. One work (30) discovers editing directions in the text embedding space and applies them to image edits, while leveraging cross-attention guidance to preserve the structure of image content. Another work (31) translates example pairs that represent the "before" and "after" images of an edit back into a text-based editing direction, and then applies it to new images for image editing in a manner similar to the previous work (30). In comparison, our research offers the following distinct contributions: (i) We utilize orthogonal projections to filter out extraneous details, thus achieving a more precise text embedding edit direction; (ii) Instead of concentrating solely on image editing, we focus on transferring CLIP's counting ability between different objects to enhance performance in counting-related image classification, image retrieval, and image generation.

3 Methods

In this section, we outline our approach to investigating and validating the hypothesis that the knowledge related to counting in images can be represented in a direction in the text embedding space such that it is independent of the object's embedding. Our approach enhances CLIP's counting abilities through both learning-based methods leveraging new datasets and zero-shot text embedding editing techniques. We will introduce how to represent counting concepts as vectors in Section 3.3, our parameter-efficient fine-tuning method in Section 3.4, and our zero-shot methods in Section 3.5. In Section 3.6, we will introduce how we collect and apply new counting datasets. Prior to the main method sections, we will first give a brief of CLIP models in Section 3.1 and define the counting problem and how to evaluate CLIP's counting ability in the following Section 3.2.

3.1 Overview of CLIP

Contrastive Language-Image Pre-Training (CLIP) (3) is a pioneering model developed by OpenAI that effectively integrates visual and textual data processing. The fundamental concept of CLIP involves the simultaneous training of two distinct encoders: an image encoder and a text encoder. These encoders are designed to produce embeddings that are closely aligned for corresponding image-text pairs and distinct for non-matching pairs.

The primary objective of CLIP is to minimize a contrastive loss that encompasses both image-to-text and text-to-image directions. Specifically, for a given dataset with N samples, where $\mu_i \in R^d$ denotes the normalized image embedding and υ_i denotes the normalized text embedding for the i^{th}sample, the CLIP loss function is defined as:

$$L_{\text{CLIP}} = -\frac{1}{2N}\sum_{i=1}^{N}\log\left(\frac{\exp(\mu_i \cdot \upsilon_i)}{\sum_{k=1}^{N}\exp(\mu_k \cdot \upsilon_i)}\right) - \frac{1}{2N}\sum_{i=1}^{N}\log\left(\frac{\exp(\mu_i \cdot \upsilon_i)}{\sum_{k=1}^{N}\exp(\mu_i \cdot \upsilon_k)}\right), \tag{1}$$

where we use $\cdot$ to denote the dot product between two vectors.

CLIP models are pre-trained on a large-scale dataset with 400 million image-text pairs sourced from the Internet. This extensive dataset enables CLIP to generalize well across different types of images and text found in real-world scenarios.

3.2 Evaluation of CLIP's counting accuracy

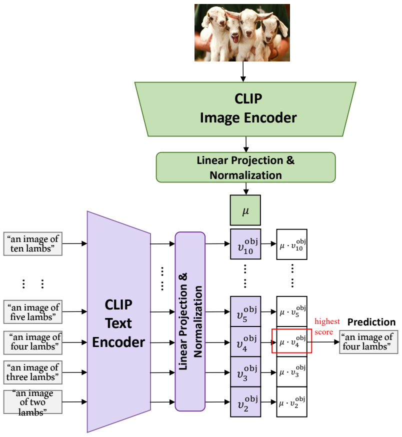

CLIP
Image Encoder
Linear Projection &
Normalization
μ
"an image of ten lambs"
"an image of five lambs"
"an image of four lambs"
"an image of three lambs"
"an image of two lambs"
CLIP
Text
Encoder
Linear Projection &
Normalization
$\upsilon_{10}^{\text{obj}}$
υ_5^{obj}
υ_4^{obj}
υ_3^{obj}
υ_2^{obj}
$\mu \cdot \upsilon_{10}^{\text{obj}}$
$\mu \cdot \upsilon_5^{\text{obj}}$
$\mu \cdot \upsilon_4^{\text{obj}}$
$\mu \cdot \upsilon_3^{\text{obj}}$
$\mu \cdot \upsilon_2^{\text{obj}}$
highest
score
Prediction
"an image of four lambs"

Figure 1: Illustration of formatting image object counting as a classification task with CLIP.

We start by defining some necessary notations. Let $\upsilon^{\text{obj}} \in \mathbb{R}^d$ be the CLIP text embedding vector for a caption like "an image of dogs", which identifies a specific object without indicating quantity, where d is the embedding dimension of a CLIP model. $\upsilon_i^{\text{obj}} \in \mathbb{R}^d$ represents the text embedding of a caption that includes a quantifier, where i is the quantity. For example, the embedding of the text "an image of *three* dogs" could be represented by υ_3^{dog}.

To assess CLIP's counting performance, we set up an image classification task where the goal is to find a correct caption that describes the object count correctly in a given image, as illustrated in Figure 1. We consider counts from two to ten and treat it as a nine-class classification task, the same as in the `CountBench` paper (12). Specifically, given an image with n specific objects and nine candidate captions (e.g., "an image of i objects" for i between two and ten), we first encode the image with CLIP's image encoder to μ^{obj} and each caption with CLIP text encoder to υ_i^{obj}. The cosine similarity between the μ^{obj} and each caption's text embedding υ_i^{obj} is computed and used to select the caption yielding the highest similarity score.

3.3 Representation of counting knowledge as vectors

We define the representation of counting knowledge for each number as a vector $\Delta_i \in \mathbb{R}^d$, aligned with the dimensionality of CLIP's embedding space. Therefore, for the nine-class classification task that we consider, there is a set of 9 vectors representing different counts, each being denoted as $\Delta_i \in \mathbb{R}^d$ for $i \in \{2, 3, ..., 10\}$.

We then process these vectors, obtaining count vectors that are orthogonal to υ^{obj}, to eliminate information associated with object representation yet not contributing to object count. Accordingly, we introduce $\widetilde{\Delta}_i$ to denote the part of Δ_i that is orthogonal to υ^{obj}, as demonstrated in the top figure in Figure 2.

Then, we derive a counting-augmented object representation $\widetilde{\upsilon}_i^{\text{obj}}$ from original representation υ_i^{obj} and the orthogonalized counting representation $\widetilde{\Delta}_i$, such that

$$\widetilde{\upsilon}_i^{\text{obj}} = \upsilon_i^{\text{obj}} + \widetilde{\Delta}_i, \tag{2}$$

where $\widetilde{\Delta}_i \coloneqq \Delta_i - \frac{\Delta_i \cdot \upsilon^{\text{obj}}}{\upsilon^{\text{obj}} \cdot \upsilon^{\text{obj}}} \upsilon^{\text{obj}}$.

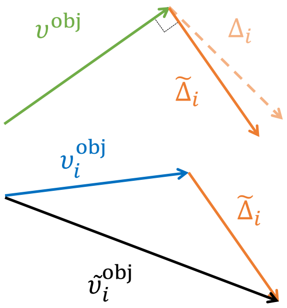


Figure 2: Represent counting knowledge as vectors.

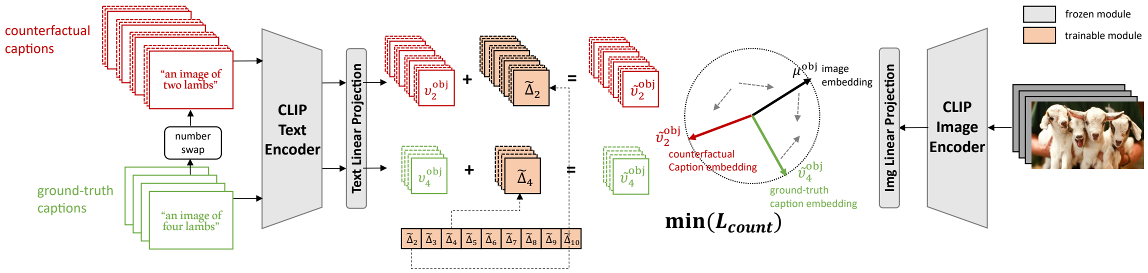
Note that our method only manipulates CLIP's text embedding and keeps its image embedding unchanged. The cosine similarity score between the original image embedding μ^{obj} and each manipulated text embedding $\widetilde{\upsilon}_i^{\text{obj}}$ is calculated to determine the object count in the caption with the highest similarity score.

The choice of forcing Δ_i to be orthogonal to υ^{obj} instead of to υ_i^{obj} is based on empirical results, which will be elaborated in the ablation study Section 5.2. We hypothesize that υ_i^{obj} already contains some level of counting information. Thus, if we let Δ_i to be orthogonal to υ_i^{obj}, there might be some loss of counting information. Similarity, the choice of υ_i^{obj} instead of υ^{obj} in Equation 2 is also based on empirical studies. We also hypothesize that the existed counting information in υ_i^{obj} will reinforce the counting signal in $\widetilde{\upsilon}_i^{\text{obj}}$.

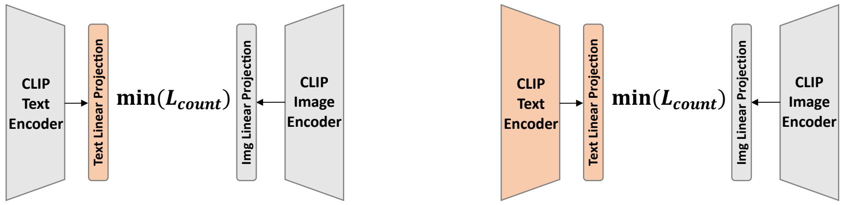
3.4 Learning counting knowledge vectors via counting loss

In this section, we introduce a learning-based method, also demonstrated in Figure 3, to obtain counting representation vectors $\widetilde{\Delta}_i$, by minimizing a counting-specific loss defined in the `CountBench` paper (12). Specifically, given a pre-trained CLIP model, we freeze all its pre-trained weights and optimize only Δ_i for $i \in \{2, 3, ..., 10\}$, which has only $9d$ parameters.

To prepare for training, for each ground truth image-caption pair within the training dataset, we create eight counterfactual caption variants by manipulating only the object count in the original caption. For instance, if the correct caption is “two dogs,” counterfactual variants include “three dogs,” “four dogs,” ..., “ten dogs.” Our objective is to enhance similarity scores between the text-image pairs with correct counts compared to the counterfactual pairs, thereby improving counting



(a) Our method that fine-tunes only nine counting vectors.



(b) Fine-tuning the last text linear projection layer.

(c) Fine-tuning the whole text model.

Figure 3: This figure illustrates our learning-based approach. As in Figure 3a, we follow the work (12) to generate counterfactual captions by swapping only the number word in each ground-truth caption, and fine-tune only nine counting vectors Δ_i as defined in Section 3.3 to minimize a counting loss L_{count} that forces image embeddings to be far from counterfactual caption embedding yet close to ground-truth caption embeddings. We compare our special fine-tuning methods to two other fine-tuning methods shown in Figure 3b and Figure 3c, which involve training a larger number of model parameters.

accuracy. The counting loss L_{count} is defined by the work (12) as follows:

$$L_{\text{count}} = -\frac{1}{N}\sum_{k=1}^{N}\sum_{j=2,j\neq t}^{10}\log\left(\frac{\exp(\mu^k\cdot\tilde{\upsilon_t}^k)}{\exp(\mu^k\cdot\tilde{\upsilon_t}^k)+\exp(\mu^k\cdot\tilde{\upsilon_j}^k)}\right) \tag{3}$$

where μ^k is the normalized image embedding of the k^{th} sample in one batch, $\tilde{\upsilon_t}^k$ is the normalized text embedding of a caption containing ground truth count t of sample k, and υ_j^k represents the embeddings for a counting-specific counterfactual text that contains the wrong count j.

In the original paper (12), where the authors continue training the CLIP model on a large amount of counting data of 158K images, they also include CLIP's regular pre-training contrastive loss in the objective function L, such that $L = \lambda L_{\text{count}} + L_{\text{CLIP}}$. This is an explicit design to prevent CLIP from forgetting its other non-counting related pre-trained knowledge. We will also investigate the necessity of L_{CLIP} in our setting where only a small amount of training data is available, in terms of its effectiveness in improving counting accuracy and preserving the CLIP model's pre-trained knowledge.

Recent work (14) modifies L_{count} to be an N-way contrastive loss (32) and shows that it improves model performance after training more than using L_{count} defined in Equation 3. We will detail the mathematical formula of the contrastive counting loss in our ablation study Section 5.1, and compare results after training models with the contrastive counting loss.

3.5 Zero-shot methods to transfer counting knowledge from prior knowledge

Alongside fine-tuning, we explore zero-shot methods to improve the CLIP model with counting capabilities without direct training on counting tasks. Zero-shot methods are particularly valuable given the scarcity of specific image-text pairs for counting tasks, which expands the practical applicability of the model in real-world scenarios where labeled data is limited. We propose several techniques to extract counting knowledge representation vector υ_i. These methods can also be used as initialization when training counting vectors, which can effectively speed up the optimization process discussed in the earlier section.

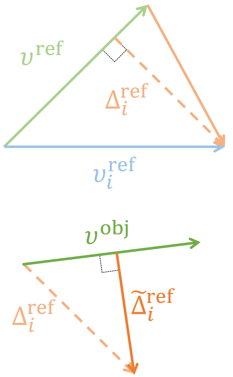


Figure 4: Extract counting vector from a reference object.

Use text embedding of number words. We encode each number i in its English word with CLIP text encoder and denote it as Δ_i^{num}. For example, the text embedding of word "two" is denoted as Δ_2^{num}. Therefore, following Equation 2, the counting-augmented object representation is calculated as $\widetilde{\upsilon}_i^{\text{obj}} = \upsilon_i^{\text{obj}} + \widetilde{\Delta}_i^{\text{num}}$, where $\widetilde{\Delta}_i^{\text{num}}$ is Δ_2^{num} that's orthogonalized w.r.t υ^{obj}.

Extract counting knowledge from an easy-to-count object. This approach is based on the observation that CLIP is more proficient at counting certain types of objects, such as dogs and cats, as elaborated in Section 4.4. Our key idea is that if CLIP effectively counts certain object types, it already possesses some counting knowledge, at least for those objects. When we have the prior knowledge of which object CLIP can count effectively, we take it as a counting reference and extract Δ_i from this object.

Specifically, as shown in Figure 4, we define the counting representation extracted from any reference object as Δ_i^{ref}, such that

$$\Delta_i^{\text{ref}} = (\upsilon_i^{\text{ref}} - \upsilon^{\text{ref}}) - \frac{(\upsilon_i^{\text{ref}} - \upsilon^{\text{ref}}) \cdot \upsilon^{\text{ref}}}{\upsilon^{\text{ref}} \cdot \upsilon^{\text{ref}}} \upsilon^{\text{ref}}. \tag{4}$$

The intuition behind this definition is that the counting information is encapsulated in the direction moving from the non-quantitative representation (υ^{ref}) to the quantitative representation (υ_i^{ref}). Similarly, we obtain Δ_i^{ref} by making $\upsilon_i^{\text{ref}} - \upsilon^{\text{ref}}$ orthogonal to υ^{ref} to eliminate information associated with the non-quantitative representation. Then, we also process it to be orthogonal to the non-quantitative representation of a target object before applying it. We denote the counting representation after two steps of orthogonalization as $\widetilde{\Delta}_i^{\text{ref}}$, such that

$$\widetilde{\Delta}_i^{\text{ref}} := \Delta_i^{\text{ref}} - \frac{\Delta_i^{\text{ref}} \cdot \upsilon^{\text{obj}}}{\upsilon^{\text{obj}} \cdot \upsilon^{\text{obj}}} \upsilon^{\text{obj}}. \tag{5}$$

Therefore, following Equation 2, the counting-augmented object representation is calculated as $\widetilde{\upsilon}_i^{\text{obj}} = \upsilon_i^{\text{obj}} + \widetilde{\Delta}_i^{\text{ref}}$.

Extract knowledge from multiple objects. Instead of relying on specific objects known to be easier for CLIP to count, this method aggregates counting vectors from a diverse set of objects, which aims to create a robust counting mechanism. We prompt ChatGPT (33), specifically `gpt-4-turbo-2024-04-09`, to generate a list of 100 common daily objects and animals in their plural form. For each object in the list, we calculate the counting-specific vector $\widetilde{\Delta}_i^{\text{ref}}$, same as described in the above paragraph. We then average these vectors across all objects in the list to create a generalized counting vector $\widetilde{\Delta}_i^{\text{multi}} = \frac{1}{100}\sum_{j=1}^{100}\widetilde{\Delta}_i^{\text{ref}_j}$. This averaged vector $\widetilde{\Delta}_i^{\text{multi}}$, is used as the counting representation to calculate $\widetilde{v}_i^{\text{obj}} = v_i^{\text{obj}} + \widetilde{\Delta}_i^{\text{multi}}$, hypothesizing that it encapsulates a universal counting pattern applicable across different object types.

3.6 Development of new object counting benchmarks

To rigorously evaluate the counting capabilities of the CLIP model and the effectiveness of our methods, we have developed two datasets in addition to using existing benchmark `CountBench` (12). `CountBench` is an object counting dataset, collected from the LAION-400M dataset (7). It comprises 540 images in total, with each numerical count represented by 60 respective images of different types of objects. Each image is accompanied by a caption describing the count of a specific object.

A new diverse-source counting dataset. The first new benchmark `DiverseCount` consists of images automatically sourced from multiple sources, including the COCO Dataset (34), Conceptual 12M (35), YFCC100M (36), and SBU Captions Dataset (37). In addition, due to the fact that images with large counts (i.e., nine and ten) are scarce, to compose 150 images for each count, we also manually collect some images from the Internet. Each text image pair is manually checked and the captions are revised to get rid of grammar errors and noisy information. Images are also manually checked to avoid duplication. This process yielded a more comprehensive set of images with clear, concise captions, containing 1350 images in total.

An object-specific counting dataset. The second new benchmark `ObjectCount` focuses on object-specific counting to delve deeper into how object types influence counting performance. It includes 360 images in total, with 10 images for each of nine different objects $\in$ {"dog", "cats", "lion", "chair", "goat", "cow", "cherry", "rose", "boat"}, at each count level from two to five, all manually collected from the Internet. The counts do not exceed five due to the rarity of images as the object count increases. This dataset is primarily used to evaluate how the counting ability varies with different object types and to identify objects that are easier to count, which can then be used as references in zero-shot methods.

4 Experiments and Results

4.1 Experimental Setup

Models. We evaluate our method on three versions of CLIP models (3), `clip-vit-base-patch32`, `clip-vit-base-patch16`, and `clip-vit-large-patch14`, all sourced from HuggingFace[1]. These models have progressively smaller patch sizes, implying that each model represents a given image with increasing resolution. Furthermore, `clip-vit-large-patch14` has a larger model size compared

---

[1] `https://huggingface.co/openai`

to the first two models. We also test our zero-shot methods on the Stable Diffusion model, `stable-diffusion-v1-4`, also sourced from HuggingFace.

Learning-based methods implementation details. We compared our novel approach, which involves optimizing counting vectors, with two conventional fine-tuning methods: 1) full fine-tuning of the pre-trained CLIP text encoder and 2) fine-tuning only pre-trained CLIP's last linear text projection layer. Additionally, we test whether including the standard CLIP contrastive loss (L_{CLIP}) in our training objectives improves performance when training on small-scale datasets. For training the counting vectors, we set a higher learning rate of 10^{-3}. We use lower learning rates of 10^{-4} for fine-tuning the text projection layer and 10^{-5} for the entire text model, respectively, to minimize overfitting. We use batch size 128 for all settings.

We divide our new dataset, `DiverseCount`, into training, validation, and test sets in a 6:2:2 ratio. We conduct three groups of experiments using three different random seeds and report the average scores on the test set across the three runs to ensure the robustness of our fine-tuning approach. We track model performance across epochs by saving checkpoints and selecting the model with the lowest validation loss for final evaluations. Each model is first evaluated on the test portion of our dataset, `DiverseCount`, and then on the `CountBench` dataset to assess how well it generalizes to different data distributions.

We further assess the fine-tuned models on various non-counting tasks including CIFAR10 (38), CIFAR100 (38), Caltech101 (39), EuroSAT (40; 41), and Food101 (42) to evaluate whether fine-tuning affected the models' performance in areas unrelated to counting. This assessment also helped us understand the role of L_{CLIP} in preserving the pre-trained capabilities of CLIP.

Zero-shot methods implementation details. For zero-shot methods, we implemented different choices of counting vectors Δ_i introduced in Section 3.5, including: 1) text embedding of number words Δ_i^{num}; 2) counting vectors $\widetilde{\Delta}_i^{\text{multi}}$ extracted from common objects; and 3) counting vectors $\widetilde{\Delta}_i^{\text{ref}}$ extracted from an easy-to-count object. In the third group of experiments, "cats" and "dogs" are selected as reference objects for our method, based on their consistently high results in Table 3.

We evaluate our method on our custom dataset `ObjectCount`, as introduced in Section 3.6, as well as on the image counting benchmark, `CountBench` (12). The counting task on `ObjectCount` is a four-class task (counting objects from two to five), while on `CountBench`, it is a nine-class task (counting objects from two to ten).

4.2 Effectiveness of learned counting knowledge vectors

We assess the effectiveness of our methods in improving CLIP's counting accuracy using the `DiverseCount` test set and `CountBench`. The results, as shown in Table 1a, indicate that training counting vectors and fine-tuning the last linear layer are more effective than fine-tuning the entire text model, despite involving far fewer parameters. This supports our hypothesis that counting knowledge may be encapsulated in a specific direction applicable across different objects. Notably, training counting vectors, which involves only $9d$ parameters compared to d^2 for the text projection layer, proves more computationally efficient without sacrificing performance.

When evaluated on `CountBench`, as shown in Table 1b, all methods demonstrated reduced effectiveness, likely due to a distribution gap between `DiverseCount` and `CountBench`. However,

Table 1: **Accuracy (%) of 9-class classification on `DiverseCount` test splits and on `CountBench`, comparing different fine-tuning methods.** All models are trained on `DiverseCount` training split. We report the average accuracy from 3 runs with different train/val/test splits. Columns under $L = L_{\text{count}}$ refer to learning methods that optimize only the counting loss. Columns under $L = L_{\text{count}} + L_{\text{CLIP}}$ refer to learning methods that optimize the counting loss and CLIP's regular contrastive loss. "CntVecs", "Proj" and "Text Model" each represent the only trained parameter: "counting vectors," "CLIP's text projection layer" and "CLIP's text model," respectively. We bold the highest score in each row.

(a) **Accuracy of 9-class classification on `DiverseCount` test splits, comparing different fine-tuning methods.**

Model	Original	Learning-based methods					
		$L = L_{\text{count}}$			$L = L_{\text{count}} + L_{\text{CLIP}}$		
		CntVecs (ours)	Proj	Text Model	CntVecs (ours)	Proj	Text Model
`CLIP-base-32`	28.17	37.78	38.16	33.92	**38.53**	37.66	34.17
`CLIP-base-16`	28.66	38.28	38.53	35.41	**38.66**	38.03	36.04
`CLIP-large-14`	33.62	**41.34**	39.6	33.49	40.84	39.35	33.99

(b) **Accuracy of 9-class classification on `CountBench`, comparing different fine-tuning methods.**

Model	Original	Learning-based methods					
		$L = L_{\text{count}}$			$L = L_{\text{count}} + L_{\text{CLIP}}$		
		CntVecs (ours)	Proj	Text Model	CntVecs (ours)	Proj	Text Model
`CLIP-base-32`	30.69	33.12	27.47	32.83	33.4	27.75	**33.62**
`CLIP-base-16`	28.73	30.66	27.39	28.51	**30.89**	27.25	29.47
`CLIP-large-14`	31.97	39.2	31.4	32.33	**39.41**	31.83	32.47

training counting vectors still improves CLIP's counting accuracy on `CountBench`, showcasing better generalization compared to the other methods.

Moreover, the inclusion of L_{CLIP} in the training objective does not significantly influence the outcomes across all training setups, as seen in the three rightmost columns in Table 1.

4.3 Impact on CLIP's Performance in Non-Counting Tasks

Fine-tuning via directly updating CLIP's pre-trained weight has a significant impact on CLIP's performance in non-counting benchmarks, as detailed in Table 2. Both fine-tuning the entire text model and the last linear layer results in noticeable performance drops across all benchmarks, with fine-tuning the projection layer having a more pronounced effect.

Moreover, incorporating L_{CLIP} only prevents loss of pre-trained knowledge with marginal effectiveness. This contrasts with the findings in the `CountBench` paper, where incorporating L_{CLIP} was beneficial when fine-tuning CLIP with a large dataset (i.e., 158K images), providing ample sources for the

Table 2: **Performance on common benchmarks.** We compare the performance of fine-tuned models against pre-trained models on common non-counting benchmarks. Fine-tuning either linear layer or the whole model will lead to siginificant performance drop.

Model	Benchmark	Original	Learning-based methods			
			$L = L_{\text{count}}$		$L = L_{\text{count}} + L_{\text{CLIP}}$	
			Proj	Text Model	Proj	Text Model
`CLIP-base-32`	CIFAR10	88.95	83.43	89.01	82.72	88.95
	CIFAR100	48.78	35.68	47.75	37.9	47.62
	Caltech101	80.18	71.50	80.33	71.15	80.3
	EuroSAT	45.11	36.49	43.78	37.94	44.28
	Food101	80.2	78.64	78.61	78.59	78.69
`CLIP-base-16`	CIFAR10	88.35	79.92	88.31	81.59	88.38
	CIFAR100	58.63	51.98	58.95	53.38	59.36
	Caltech101	76.78	70.70	75.13	71.18	75.67
	EuroSAT	49.83	39.20	46.84	43.77	47.47
	Food101	85.57	82.97	85.26	85.69	83.49
`CLIP-large-14`	CIFAR10	95.01	93.65	94.91	93.71	95.04
	CIFAR100	64.66	62.49	64.9	62.47	64.99
	Caltech101	81.02	75.50	80.20	75.71	80.56
	EuroSAT	55.37	51.59	54.00	51.46	53.9
	Food101	89.79	88.46	89.35	88.61	89.43

model to learn new things. In our experiments, where only a smaller dataset was available, L_{CLIP} did not demonstrate the same effectiveness, suggesting that its utility may be limited under conditions of restricted data availability.

However, training new counting vectors does not alter CLIP's model parameters. Thus, we can apply these vectors only in tasks specific to counting, while relying solely on the pre-trained CLIP model for all other non-counting tasks.

4.4 CLIP's counting ability on different objects

Table 3: **The counting accuracy of CLIP varies across diverse objects.** Pre-trained CLIP models counting accuracy varies by object type. Still, all models consistently count dogs and cats more accurate than other objects.

	average	dogs	cats	lions	chairs	goats	cows	cherries	roses	boats
`CLIP-base-32`	47.93	**58.86**	**66.14**	47.73	35.23	42.73	46.36	45.45	32.27	47.27
`CLIP-base-16`	50.33	**74.77**	**74.77**	54.32	47.05	32.73	55.00	35.00	34.09	45.23
`CLIP-large-14`	60.86	**75.23**	**79.09**	65.45	52.95	44.77	65.00	53.86	56.82	54.55

As shown in Table 3, each column displays the accuracy of counting a specific object in dataset `ObjectCount`, with the object name used as the column header. The average accuracy across all objects is also displayed under the "average" column. We observe a positive correlation between model size and average counting accuracy, with accuracies ranging from 47.93% to 60.86%. However, there is significant variation in the models' counting abilities for different object types, suggesting that CLIP's counting capability is dependent on the object.

Notably, all models consistently perform best when counting "dogs" and "cats", while their performance with other objects lacks consistency. We hypothesize that this might be due to images with certain counts of "dogs" and "cats" appearing more frequently in the pre-training dataset. In fact, when collecting our dataset `ObjectCount` from the Internet, we do observe that images of dogs and cats are more accessible in larger volumes than images of other objects.

4.5 Effectiveness of our zero-shot method

Table 4: **CLIP's counting accuracy for image classification task on our custom dataset `ObjectCount` and `CountBench` (%), comparing results of zero-shot methods.** We bold the higest score in each row.

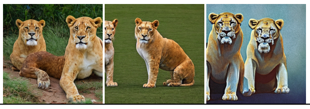
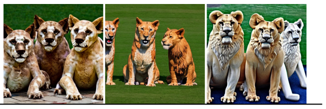
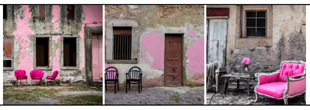
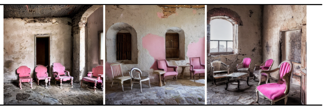
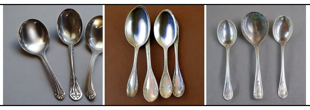
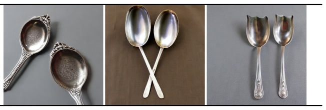
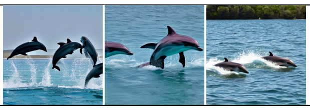
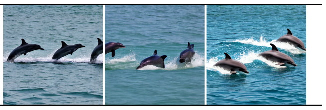
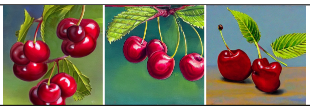
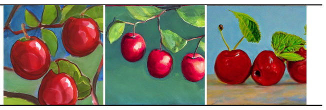
Model	DatasetMethod	original	$v_i^{\text{obj}} + \widetilde{\Delta}_i^{\text{multi}}$	$v_i^{\text{obj}} + v_i^{\text{num}}$	$v_i^{\text{obj}} + \widetilde{\Delta}_i^{\text{dogs}}$	$v_i^{\text{obj}} + \widetilde{\Delta}_i^{\text{cats}}$
`CLIP-base-32`	`ObjectCount` (4-class)	47.93	49.65	49.65	**51.84**	49.8
	`CountBench` (9-class)	30.69	**31.91**	31.7	28.3	29.36
`CLIP-base-16`	`ObjectCount` (4-class)	50.33	53.16	52.55	**55.45**	54.77
	`CountBench` (9-class)	28.73	27.02	**31.06**	29.15	28.51
`CLIP-large-14`	`ObjectCount` (4-class)	60.86	**65.58**	64.77	64.12	64.5
	`CountBench` (9-class)	31.97	**39.45**	36.25	39.23	38.17

We evaluate our zero-shot methods on our custom dataset `ObjectCount` and `CountBench`, and report the results in Table 4. Across all three CLIP models, there are noticeable improvements in counting accuracy when using zero-shot methods compared to the default (baseline) settings. However, the effectiveness of each strategy varies by task and model size. For example, most methods are more effective on the `CLIP-large-14` model, which has larger model size and higher resolution, with improvement close to or higher than 5%, The improvement is also more significant on `ObjectCount` than on `CountBench`, likely because `CountBench` has 9 nine classes and becomes more challenging.

4.6 Effectiveness of our method in improving text-to-image models' counting fidelity

Since our method directly enhances the counting capability of the CLIP model in a zero-shot manner, we anticipate that using our approach will also aid models that utilize CLIP embeddings for image generation, such as the Stable Diffusion model (8), in producing images with the correct counting number of objects. Therefore, we experimented with applying our method to Stable Diffusion, and the results are displayed in Table 5 and Appendix A. It can be noted that after applying our method, Stable Diffusion's counting fidelity increased, meaning there is a higher likelihood of generating images with correct counts without any additional training. Note that our method can be used in conjunction with existing methods for improving the fidelity of text-to-image models, e.g., reinforcement learning-based algorithms (43; 44; 45).

Table 5: **Selected results from Stable Diffusion (8).** Images in the "Original" column are generated based on the input prompt in the same row, using different seeds. Images in the "Embedding edited" column are generated after applying our zero-shot method (using the same seeds), with the selection of "dog" as the reference. After applying our method, we observe that Stable Diffusion is more likely to generate images with the correct number of objects.

Input Prompt	Original	Embedding edited
"`three lions`"		
"`An old building with ruined walls and four antique pink armchairs`"		
"`vintage silver plate tablespoons, serving spoon set of two`"		
"`three dolphins jumping out of water`"		
"`A picture of three cherries`"		

5 Ablation Studies

5.1 Effectiveness of different contrastive loss designs

In our work, we adapt the count loss proposed by the work (12), as shown in Equation 3. We compare it against a multi-class N-way loss, denoted by $\widetilde{L}_{\text{count}}$, used in recent work (14), which is defined as:

$$\widetilde{L}_{\text{count}} = -\frac{1}{N}\sum_{k=1}^{N}\log\left(\frac{\exp(\mu^k\cdot\widetilde{v_t}^k)}{\exp(\mu^k\cdot\widetilde{v_t}^k)+\sum_{j=2,j\neq t}^{10}\exp(\mu^k\cdot\widetilde{v_j}^k)}\right) \tag{6}$$

As shown in Table 6, when testing on the test split of `DiverseCount`, fine-tuning with contrastive counting loss $\widetilde{L}_{\text{count}}$ is slightly worse than with L_{count}. However, applying $\widetilde{L}_{\text{count}}$ helps further improving most fine-tuned models accuracy when evaluating on `CountBench` dataset, indicating that $\widetilde{L}_{\text{count}}$ might have a more robust and effective loss design when the models need to generalize to setting with larger distribution shift.

Table 6: **Accuracy (%) of 9-class classification on `DiverseCount` test splits and `CountBench` of different fine-tuning methods, with contrastive count loss $\widetilde{L}_{\textbf{count}}$.** For each fine-tuning method, we bold scores if its higher when applying contrastive loss $\widetilde{L}_{\text{count}}$ vs. L_{count}. We display changes compared to applying L_{count} inside the parenthesizes right after each score.

Model	`DiverseCount`			`CountBench`		
	CntVecs (ours)	Proj	Text Model	CntVecs (ours)	Proj	Text Model
`CLIP-base-32`	37.28 (-0.50)	37.78 (-0.38)	33.78 (-0.14)	**33.55 (+0.43)**	**28.25 (+0.78)**	**34.62 (+1.79)**
`CLIP-base-16`	38.03 (-0.25)	38.03 (-0.5)	**35.53 (+0.12)**	**32.66 (+2.00)**	**28.66 (+1.27)**	**29.4 (+1.89)**
`CLIP-large-14`	39.48 (-1.86)	39.59 (-0.01)	**34.37 (+0.88)**	37.91 (-1.29)	**32.69 (+1.29)**	**33.48 (+1.15)**

5.2 Ablation studies of each component in the counting representation

5.2.1 Learning-based settings

We study the effectiveness of different modifications when representing and applying $\widetilde{\Delta}_i$ in learning-based settings. In our main method, $\widetilde{\Delta}_i := \Delta_i - \frac{\Delta_i \cdot v^{\text{obj}}}{v^{\text{obj}} \cdot v^{\text{obj}}} v^{\text{obj}}$, which involves orthogonalization w.r.t. v^{obj}. We study the effect of orthogonalization and name this experiment as `NoObjOrth`. We also study the effect of projection direction happening in orthogonalization, by comparing the main results to the design such that Δ_i is forced to be orthogonal to v_i^{obj} instead of v^{obj}. We name this group of experiments, where $\widetilde{\Delta}_i := \Delta_i - \frac{\Delta_i \cdot v^{\text{obj}}}{v^{\text{obj}} \cdot v^{\text{obj}}}$, as in `ChangeProjDir`. In addition, when applying $\widetilde{\Delta}_i$, it is added to v_i^{obj}, as shown in Equation 2. We study the effect of adding $\widetilde{\Delta}_i$ onto v^{obj} such that $\widetilde{v}_i^{\text{obj}} = v^{\text{obj}} + \widetilde{\Delta}_i$, and named this group of experiments as `ChangeAddDir`.

Ablation study results on are displayed in Table 7, under columns of each ablation experiment name. We find that in learning-based settings, there is not significant difference between each modification and the main approach. This suggest that when applying the same modification during both training and inference, and orthogonalization and direction of orthogonalization and addition might not be very important factors for learning counting vectors. However, these factors have much more significant effects in zero-shot settings, as we will demonstrate in the next section.

Table 7: **Accuracy (%) of 9-class classification on `DiverseCount` test splits and `CountBench` of fine-tuning counting vectors, comparing different modifications with our method in the main paper.** For each modification, we bold scores if its higher than score of the main method.

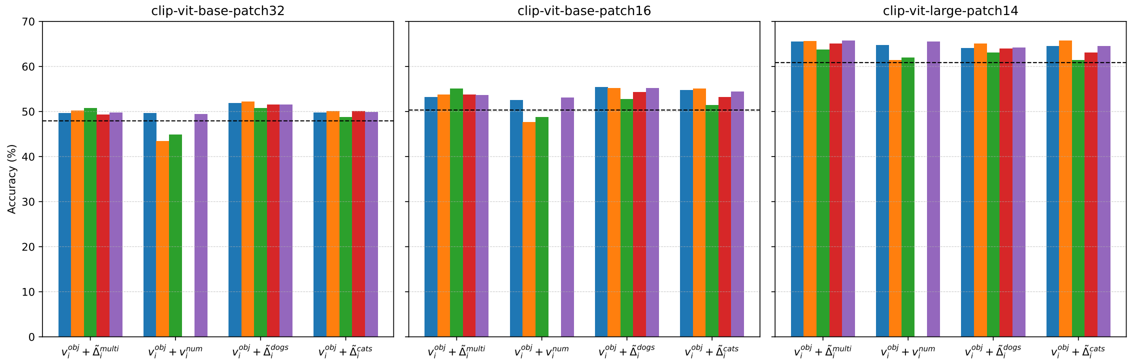
Model	`DiverseCount`				`CountBench`			
	Main method	`NoObjOrth`	`ChangeProjDir`	`ChangeAddDir`	Main method	`NoObjOrth`	`ChangeProjDir`	`ChangeAddDir`
`CLIP-base-32`	37.78	37.16	37.66	36.28	33.12	**33.62**	**33.26**	32.69
`CLIP-base-16`	38.28	**38.65**	**38.78**	38.03	30.66	**31.7**	**31.48**	**32.22**
`CLIP-large-14`	41.34	**42.59**	41.09	**41.84**	39.2	**39.41**	**39.34**	**40.27**

5.2.2 Zero-shot settings

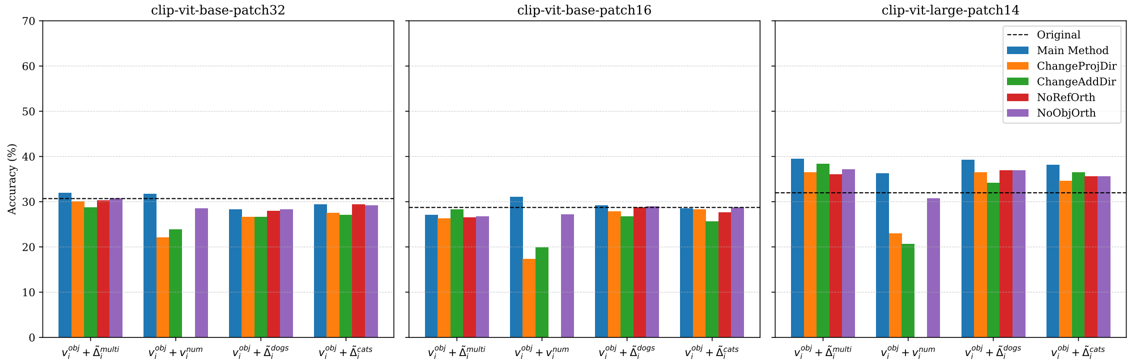
In addition to ablation study groups introduced in the previous section, including `NoObjOrth`, `ChangeProjDir`, `ChangeAddDir`, for zero-shot settings, we add another ablation study group named

`NoRefOrth`. This ablation group studies the effect of orthogonalization w.r.t. reference object's text embeddings, by excluding projection on ref as defined in Equation 5, so that the orthogonalization only applies w.r.t. to target object's text embeddings υ^{obj} and that $\widetilde{\Delta}_i^{\text{ref}} = \Delta_i^{\text{ref}} - \frac{\Delta_i^{\text{ref}} \cdot \upsilon^{\text{obj}}}{\upsilon^{\text{obj}} \cdot \upsilon^{\text{obj}}} \upsilon^{\text{obj}}$.

As shown in Figure 5 and Table 8, the main method is more effective than other modifications, especially on the `CountBench` dataset.



(a) Ablation studies of zero-shot methods on `ObjectCount` dataset.



(b) Ablation studies of zero-shot methods on `CountBench` dataset.

Figure 5: Ablation studies of zero-shot methods on `ObjectCount` dataset and `CountBench` dataset.

6 Conclusion and Discussion

In this study, we explored the counting capabilities of CLIP models and introduced a computationally efficient method for training counting vectors, along with several zero-shot text embedding editing techniques to enhance CLIP's counting accuracy. Our learning-based approach demonstrates that targeted modifications to text embeddings can significantly improve object counting tasks without the need for extensive model retraining. This method not only proves to be effective but also preserves the broader capabilities of CLIP, unlike other methods that might compromise general performance of CLIP models.

Table 8: **CLIP's counting accuracy for image classification task on our custom dataset `ObjectCount` and `CountBench` (%), comparing results of zero-shot methods.** For each zero-shot method, we bold scores if its higher with current modification. We display changes against the main method inside the parenthesizes right after each score.

	Model	DatasetMethod	$v_i^{\text{obj}} + \widetilde{\Delta}_i^{\text{multi}}$	$v_i^{\text{obj}} + v_i^{\text{num}}$	$v_i^{\text{obj}} + \widetilde{\Delta}_i^{\text{dogs}}$	$v_i^{\text{obj}} + \widetilde{\Delta}_i^{\text{cats}}$
Change Add Dir	`CLIP-base-32`	`ObjectCount` (4-class)	**50.81 (+1.16)**	44.87 (-4.78)	50.76 (-1.08)	48.79 (-1.01)
		`CountBench` (9-class)	28.76 (-3.15)	23.82 (-7.88)	26.61 (-1.69)	27.04 (-3.32)
	`CLIP-base-16`	`ObjectCount` (4-class)	**55.13 (+1.97)**	48.79 (-3.76)	52.75 (-2.7)	51.39 (-3.38)
		`CountBench` (9-class)	**28.29 (+1.27)**	19.82 (-11.24)	26.73 (-2.42)	25.61 (-2.9)
	`CLIP-large-14`	`ObjectCount` (4-class)	63.79 (-1.79)	61.94 (-2.83)	63.13 (-0.99)	61.47 (-3.03)
		`CountBench` (9-class)	38.41 (-1.04)	20.6 (-15.65)	34.12 (-5.11)	36.48 (-1.69)
Change Proj Dir	`CLIP-base-32`	`ObjectCount` (4-class)	**50.18 (+0.53)**	43.46 (-6.19)	**52.2 (+0.36)**	**50.08 (+0.28)**
		`CountBench` (9-class)	30.04 (-1.87)	22.1 (-9.6)	26.61 (-1.69)	27.47 (-1.89)
	`CLIP-base-16`	`ObjectCount` (4-class)	**53.76 (+0.6)**	47.68 (-4.87)	55.25 (-0.2)	**55.08 (+0.31)**
		`CountBench` (9-class)	26.28 (-0.74)	17.37 (-13.69)	27.84 (-1.31)	28.29 (-0.22)
	`CLIP-large-14`	`ObjectCount` (4-class)	**65.68 (+0.1)**	61.39 (-3.38)	**65.05 (+0.93)**	**65.71 (+1.21)**
		`CountBench` (9-class)	36.48 (-2.97)	22.96 (-13.29)	36.48 (-2.75)	34.55 (-3.62)
No Ref Orth	`CLIP-base-32`	`ObjectCount` (4-class)	49.34 (-0.31)	-	51.59 (-0.25)	**50.15 (+0.35)**
		`CountBench` (9-class)	30.26 (-1.65)	-	27.9 (-0.4)	**29.4 (+0.04)**
	`CLIP-base-16`	`ObjectCount` (4-class)	**53.81 (+0.65)**	-	54.29 (-1.16)	53.21 (-1.56)
		`CountBench` (9-class)	26.5 (-0.52)	-	28.73 (-0.42)	27.62 (-0.89)
	`CLIP-large-14`	`ObjectCount` (4-class)	65.05 (-0.53)	-	63.99 (-0.13)	63.13 (-1.37)
		`CountBench` (9-class)	36.05 (-3.4)	-	36.91 (-2.32)	35.62 (-2.55)
No Target Orth	`CLIP-base-32`	`ObjectCount` (4-class)	**49.72 (+0.07)**	49.47 (-0.18)	51.56 (-0.28)	**49.93 (+0.13)**
		`CountBench` (9-class)	30.69 (-1.22)	28.54 (-3.16)	**28.33 (+0.03)**	29.18 (-0.18)
	`CLIP-base-16`	`ObjectCount` (4-class)	**53.61 (+0.45)**	53.05 (+0.5)	55.25 (-0.2)	54.42 (-0.35)
		`CountBench` (9-class)	26.73 (-0.29)	27.17 (-3.89)	28.95 (-0.2)	**28.73 (+0.22)**
	`CLIP-large-14`	`ObjectCount` (4-class)	**65.71 (+0.13)**	**65.56 (+0.79)**	**64.24 (+0.12)**	64.5 (+0.0)
		`CountBench` (9-class)	37.12 (-2.33)	30.69 (-5.56)	36.91 (-2.32)	35.62 (-2.55)

Our zero-shot techniques are particularly valuable in contexts where data is scarce or full model retraining is impractical due to computational constraints. These methods have also shown promise when applied to text-to-image models like Stable Diffusion, indicating their potential applicability beyond the initial use case.

However, our approach has limitations. One key issue is the lack of a clear understanding of why training counting-specific vectors is more effective than other methods. Further theoretical exploration could provide deeper insights into the mechanisms of CLIP's counting abilities and potentially inspire new enhancement strategies for vision-language models.

Additionally, our method struggles in scenarios with complex object interactions or diverse visual contexts. The reliance on simple text embeddings for counting does not capture complex visual relationships or subtle distinctions between object quantities in cluttered scenes. This suggests a need for models that incorporate advanced visual reasoning abilities.

Looking ahead, future research could focus on integrating spatial awareness and relational reasoning into vision-language models to improve their ability to understand and interpret spatial relationships and contextual clues within images. This enhancement could lead to better accuracy in counting and general visual comprehension.

Moreover, extending our approach to other visual tasks, such as object detection or complex counting in cluttered environments, could broaden its applicability. Investigating how this method could be adapted to work with other vision-language architectures might also reveal universal strategies for enhancing model performance across various tasks.

Acknowledgments

This work is supported in part by NSF CAREER Award CCF-2339978, Amazon Research Award, and a grant from FuriosaAI.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L. Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022.

[2] Xi Chen, Xiao Wang, Soravit Changpinyo, A. J. Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Nan Ding, Keran Rong, Hassan Akbari, Gaurav Mishra, Linting Xue, Ashish V. Thapliyal, James Bradbury, and Weicheng Kuo. Pali: A jointly-scaled multilingual language-image model. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[3] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021.

[4] Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. FLAVA: A foundational language and vision alignment model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 15617–15629. IEEE, 2022.

[5] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *CoRR*, abs/2304.08485, 2023.

[6] Chunyuan Li, Cliff Wong, Sheng Zhang, Naoto Usuyama, Haotian Liu, Jianwei Yang, Tristan Naumann, Hoifung Poon, and Jianfeng Gao. Llava-med: Training a large language-and-vision assistant for biomedicine in one day. *CoRR*, abs/2306.00890, 2023.

[7] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021.

[8] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[9] Nan Liu, Shuang Li, Yilun Du, Josh Tenenbaum, and Antonio Torralba. Learning to compose visual relations. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 23166–23178, 2021.

[10] Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. Winoground: Probing vision and language models for visio-linguistic

compositionality. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 5228–5238. IEEE, 2022.

[11] Roni Paiss, Hila Chefer, and Lior Wolf. No token left behind: Explainability-aided image classification and generation. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XII*, volume 13672 of *Lecture Notes in Computer Science*, pages 334–350. Springer, 2022.

[12] Roni Paiss, Ariel Ephrat, Omer Tov, Shiran Zada, Inbar Mosseri, Michal Irani, and Tali Dekel. Teaching clip to count to ten. *arXiv preprint arXiv:2302.12066*, 2023.

[13] Ruixiang Jiang, Lingbo Liu, and Changwen Chen. Clip-count: Towards text-guided zero-shot object counting. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 4535–4545, 2023.

[14] Harshvardhan Mestha, Tejas Agrawal, Karan Bania, Yash Bhisikar, et al. Countclip–[re] teaching clip to count to ten. *arXiv preprint arXiv:2406.03586*, 2024.

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

[16] Ruixiang Jiang, Lingbo Liu, and Changwen Chen. Clip-count: Towards text-guided zero-shot object counting. *CoRR*, abs/2305.07304, 2023.

[17] Jingyi Xu, Hieu Le, Vu Nguyen, Viresh Ranjan, and Dimitris Samaras. Zero-shot object counting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 15548–15557. IEEE, 2023.

[18] Yan Zhang, Jonathon S. Hare, and Adam Prügel-Bennett. Learning to count objects in natural images for visual question answering. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[19] Duy-Kien Nguyen, Vedanuj Goswami, and Xinlei Chen. Movie: Revisiting modulated convolutions for visual counting and beyond. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[20] Manoj Acharya, Kushal Kafle, and Christopher Kanan. Tallyqa: Answering complex counting questions. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 8076–8084. AAAI Press, 2019.

[21] Kartik Sreenivasan, Keon Lee, Jeong-Gwan Lee, Anna Lee, Jaewoong Cho, Jy-yong Sohn, Dimitris Papailiopoulos, and Kangwook Lee. Mini-batch optimization of contrastive loss. In *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2023.

[22] Dingkang Liang, Jiahao Xie, Zhikang Zou, Xiaoqing Ye, Wei Xu, and Xiang Bai. Crowdclip: Unsupervised crowd counting via vision-language model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 2893–2903. IEEE, 2023.

[23] Yumeng Li, Margret Keuper, Dan Zhang, and Anna Khoreva. Divide & bind your attention for improved generative semantic nursing. *CoRR*, abs/2307.10864, 2023.

[24] Wonjun Kang, Kevin Galim, and Hyung Il Koo. Counting guidance for high fidelity text-to-image synthesis. *CoRR*, abs/2306.17567, 2023.

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[26] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013.

[27] Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. Word embeddings, analogies, and machine learning: Beyond king - man + woman = queen. In Yuji Matsumoto and Rashmi Prasad, editors, *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3519–3530, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

[28] Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. Towards understanding linear word analogies. *arXiv preprint arXiv:1810.04882*, 2018.

[29] Carl Allen and Timothy Hospedales. Analogies explained: Towards understanding word embeddings. In *International Conference on Machine Learning*, pages 223–231. PMLR, 2019.

[30] Gaurav Parmar, Krishna Kumar Singh, Richard Zhang, Yijun Li, Jingwan Lu, and Jun-Yan Zhu. Zero-shot image-to-image translation. In Erik Brunvand, Alla Sheffer, and Michael Wimmer, editors, *ACM SIGGRAPH 2023 Conference Proceedings, SIGGRAPH 2023, Los Angeles, CA, USA, August 6-10, 2023*, pages 11:1–11:11. ACM, 2023.

[31] Thao Nguyen, Yuheng Li, Utkarsh Ojha, and Yong Jae Lee. Visual instruction inversion: Image editing via visual prompting. *CoRR*, abs/2307.14331, 2023.

[32] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[33] OpenAI. Chatgpt: Language model. `https://www.openai.com/chatgpt`, 2024. Accessed: 2024-04-09.

[34] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[35] Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. Conceptual 12M: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *CVPR*, 2021.

[36] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: the new data in multimedia research. *Commun. ACM*, 59(2):64–73, jan 2016.

[37] Vicente Ordonez, Girish Kulkarni, and Tamara Berg. Im2text: Describing images using 1 million captioned photographs. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[38] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[39] Li Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 178–178, 2004.

[40] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 204–207. IEEE, 2018.

[41] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

[42] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 446–461, Cham, 2014. Springer International Publishing.

[43] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. DPOK: reinforcement learning for fine-tuning text-to-image diffusion models. *CoRR*, abs/2305.16381, 2023.

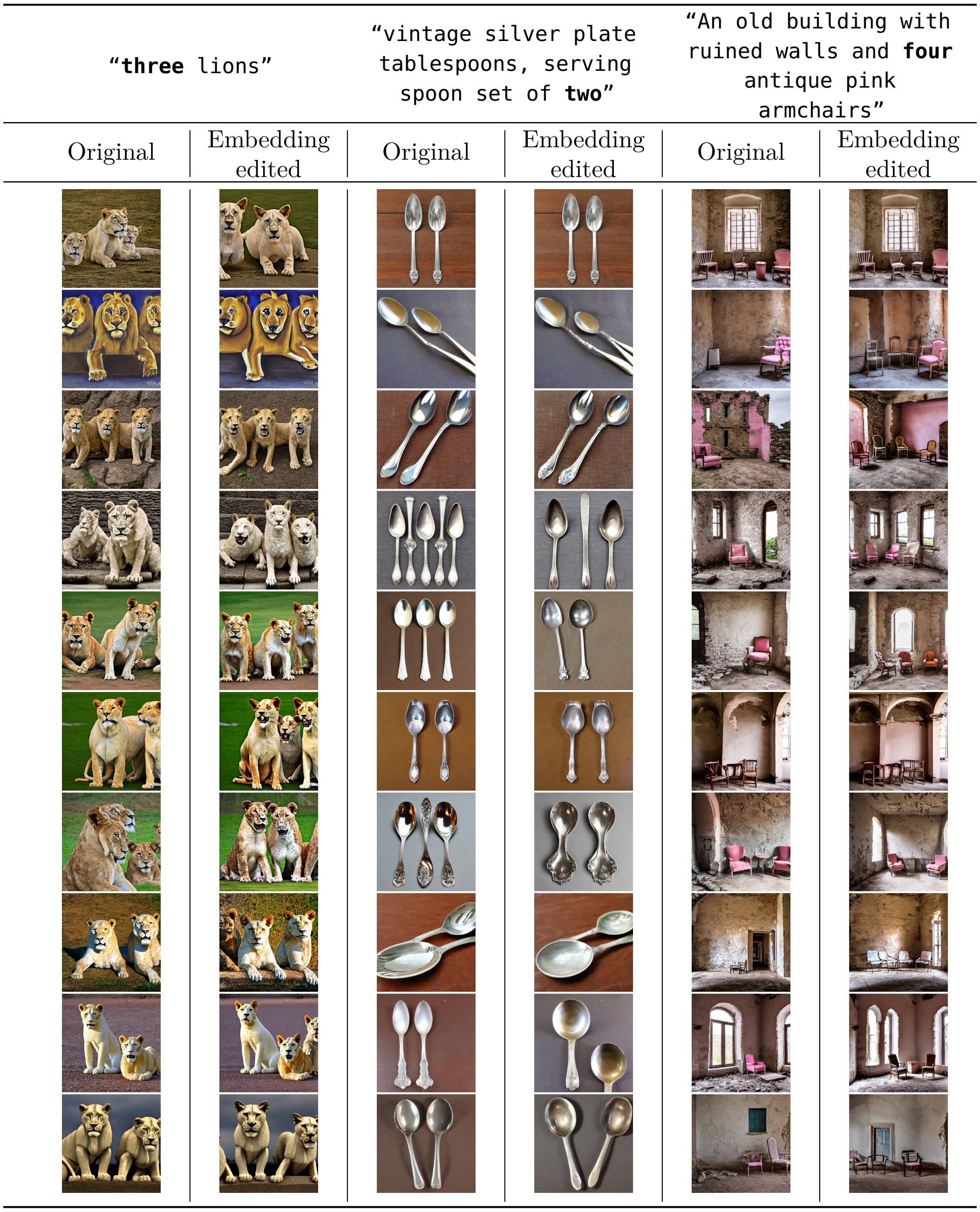
[44] Ying Fan and Kangwook Lee. Optimizing DDPM sampling with shortcut fine-tuning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 9623–9639. PMLR, 2023.

[45] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *CoRR*, abs/2305.13301, 2023.

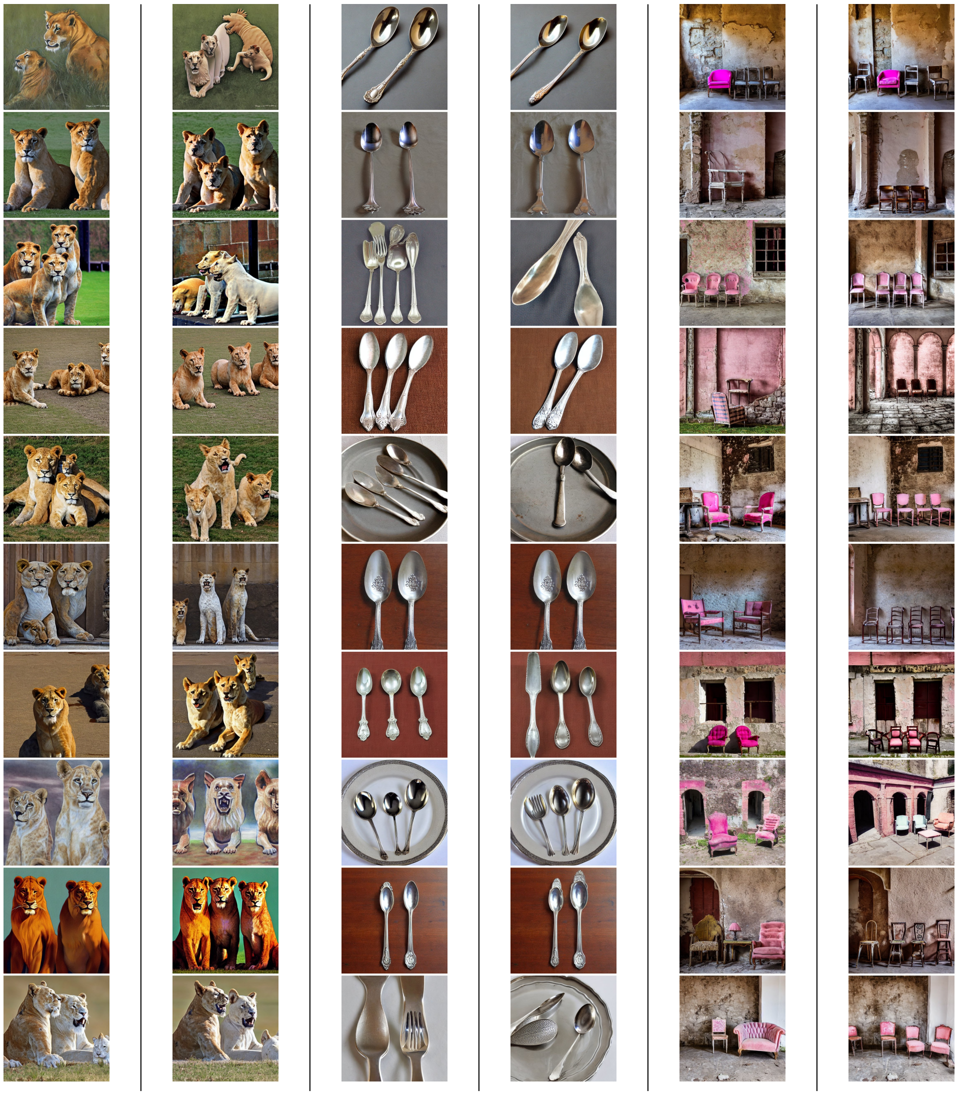
Appendix

A Effectiveness of our method in improving text-to-image models' counting fidelity

We provide more examples to show the effectiveness of applying our method to Stable Diffusion (8) to see if it can improve the counting fidelity of the text-to-image generation model. We show results from 3 prompts, where for each prompt, 30 images are generated with 30 unique random seeds. To compare our method with the unmodified Stable Diffusion baseline, images in the same row are generated using the same random seed. It is worth noting that our method is not always effective. However, it does increase the likelihood of Stable Diffusion generating images with the correct object count.


“**three** lions”
“vintage silver plate tablespoons, serving spoon set of **two**”
“An old building with ruined walls and **four** antique pink armchairs”
Original
Embedding edited
Original
Embedding edited
Original
Embedding edited

“**three** lions”		“vintage silver plate tablespoons, serving spoon set of **two**”		“An old building with ruined walls and **four** antique pink armchairs”	
Original	Embedding edited	Original	Embedding edited	Original	Embedding edited



“**three** lions”		“vintage silver plate tablespoons, serving spoon set of **two**”		“An old building with ruined walls and **four** antique pink armchairs”	
Original	Embedding edited	Original	Embedding edited	Original	Embedding edited