Appendix

A. Training Procedures

A.1. Implementation Details

For CLIP [43], we use the official implementation released by OpenAI[1]. And for ALIP [58], we also use the official implementation released by the paper authors[2]. As the proposed RANKCLIP essentially shares the same model architecture (separate vision, text encoders, projection layer, and a classification head) as CLIP, we build upon the CLIP code repository for our model construction[3]. We set the scaling parameters for cross-modal (λ_c) and in-modal (λ_i) ranking consistency to 1/16 and 1/16 respectively throughout all the experiments unless otherwise noted. All CLIP, ALIP and RANKCLIP models are initialized from scratch without loading any existing weights. And the embedding sizes for both modalities all project to 1024 across the three models.

A.2. Training Parameters

Following CLIP [43], we adopt the ResNet-50 [21] and transformer architectures [13] for image and text encoding, respectively. Training is conducted from scratch over 64 epochs using a single NVIDIA A100 GPU, with a batch size of 512, an initial learning rate of 0.0005 employing cosine scheduling, and 10,000 warm-up steps.

A.3. Training Time Consumption

we conducted the experiments using the same hardware specifications. The table below shows the time consumption for training our RankCLIP and CLIP models with 50K samples from CC3M using a single NVIDIA A100 GPU.

	Time consumption	Dataset size	epochs	batch_size	model_name
CLIP	1d 2h 54m 48s	50K	64	512	RN50
RANKCLIP	1d 1h 4m 23s	50K	64	512	RN50

Table 6. Training Details

As shown in the table, the difference in time consumption is negligible. Interestingly, our method is slightly faster than CLIP, but we think it may be attributed to hardware optimizations or variance.

B. CLIP Preliminaries

CLIP [43] has been a prominent method for learning detailed multimodal representations through the alignment of images and texts. Given a set $\mathcal{D} = \{(V_j, T_j)\}_{j=1}^{N}$ of N image-text pairs, where V_j denotes an image and T_j is the corresponding text, the goal is to learn representations that map semantically similar images and texts closer in the embedding space, while dissimilar pairs are distanced apart. More specifically, the foundational CLIP model employs two encoders: an image encoder $f_I : \mathcal{I} \to \mathbb{R}^m$ that processes raw images into visual embeddings and a text encoder $f_T : \mathcal{T} \to \mathbb{R}^n$ which encodes textual data into text embeddings. Then both the text and visual features are projected to a latent space with identical dimension. Formally, the embeddings for a text-image pair (V_j, T_j) are denoted as $v_k = f_I(V_j)$ and $t_j = f_T(T_j)$, respectively. The embeddings are then normalized to lie on an unit hypersphere by enforcing l_2-norm constraint:

$$\hat{v}_j = \frac{v_j}{\|v_j\|_2}, \quad \hat{t}_j = \frac{t_j}{\|t_j\|_2}. \tag{11}$$

so that the magnitude information is erased and only direction is preserved.

To align the image and text representations, a contrastive loss function, typically a variant of the InfoNCE loss [39], which optimizes the similarity of the matched pair against unmatched pairs, is utilized, i.e.:

$$\mathcal{L}_{\text{CLIP}} = -\frac{1}{2N}\sum_{j=1}^{N}\Bigg[\underbrace{\log\frac{\exp(\hat{v}_j^\top \hat{t}_j/\tau)}{\sum_{k=1}^{N}\exp(\hat{v}_j^\top \hat{t}_k/\tau)}}_{①} + \underbrace{\log\frac{\exp(\hat{t}_j^\top \hat{v}_j/\tau)}{\sum_{k=1}^{N}\exp(\hat{t}_j^\top \hat{v}_k/\tau)}}_{②}\Bigg] \tag{12}$$

[1]CLIP repository on GitHub: https://github.com/openai/CLIP.
[2]ALIP repository on GitHub: https://github.com/deepglint/ALIP.
[3]RANKCLIP repository will be released upon acceptance.

where the first term (1) contrasts images with the texts, the second term (2) contrasts texts with the images, and τ denotes a temperature scaling parameter that adjusts the concentration of the distribution. The optimization of Eqn. (12) results in embeddings where the cosine similarity between matched image-text pairs is maximized in comparison to unmatched pairs, thus achieving the desired alignment in the joint embedding space.

Despite the efficacy of CLIP in learning correlated multimodal embeddings, it inherently relies on strict pairwise matched comparisons and fails to capture the more complex, fine-grained nature of semantic similarity within and across modalities that are generally treated as unmatched. This observation motivates the development of RANKCLIP, which innovates beyond binary pairwise contrasts to consider holistic listwise consistency within and across modalities.

B.1. Additional Experiments

We conduct the linear probing experiment under different training datasize from 3m to 15m and different ablated settings as shown in Table 7 and 8.

Data Size	Method	Model Type	CIFAR-10	CIFAR-100	DTD	FGVCAircraft	Food101	GTSRB	OxfordPets	SST2	STL10	SVHN
3m	CLIP	RN50	80.12%	58.50%	57.18%	39.75%	59.14%	72.41%	61.73%	54.48%	86.01%	58.92%
	RANKCLIP	RN50	78.29%	56.24%	57.82%	39.30%	58.63%	74.13%	64.35%	55.02%	86.69%	60.68%
	CLIP	ViT-B/32	77.60%	56.15%	43.19%	22.59%	39.72%	62.05%	40.39%	50.96%	78.99%	50.53%
	RANKCLIP	ViT-B/32	78.42%	56.64%	42.39%	23.43%	40.19%	60.63%	40.56%	53.32%	79.60%	47.72%
15m	CLIP	RN50	78.81%	56.32%	61.49%	25.83%	61.64%	68.76%	60.37%	55.57%	89.82%	47.99%
	RANKCLIP	RN50	83.27%	62.96%	65.96%	32.19%	68.11%	74.25%	67.40%	56.34%	94.20%	53.06%
	CLIP	ViT-B/32	82.97%	62.55%	49.47%	24.48%	52.46%	63.55%	50.78%	52.66%	87.14%	46.38%
	RANKCLIP	ViT-B/32	82.79%	59.89%	52.50%	23.94%	56.44%	61.58%	52.98%	53.60%	89.01%	42.16%

Table 7. Linear probing accuracy on 10 downstream datasets.

	CIFAR-10	CIFAR-100	DTD	Aircraft	Food101	GTSRB	OxfordPets	STL10	SST2	SVHN
CLIP	77.6%	56.2%	**43.2%**	22.6%	39.7%	60.0%	40.4%	79.0%	51.0%	**50.5%**
RANKCLIP$_C$	78.0%	55.4%	42.6%	21.1%	40.1%	60.5%	**40.8%**	78.8%	52.6%	48.0%
RANKCLIP$_I$	77.3%	56.4%	39.2%	18.3%	37.4%	57.3%	36.3%	76.4%	51.1%	48.8%
RANKCLIP	**78.4%**	**56.6%**	42.4%	**23.4%**	**40.2%**	**60.6%**	40.6%	**79.6%**	**53.4%**	48.7%

Table 8. Linear probing accuracy on downstream datasets. RANKCLIP$_C$ and RANKCLIP$_I$ denote models trained with only **C**ross-modal loss and **I**n-modal loss, respectively. **Bold** highlights the best result, while underline marks the second-best.

B.2. Pseudo-code

Algorithm 1 Pseudo-code of RANKCLIP loss in a Python-like style.

```
# emb_pred: predictions from the model, shape [embs_length, embs_length]
# emb_true: ground truth labels, shape [embs_length, embs_length]

def rank_loss(emb_pred, emb_true):
    # Shuffle for randomised tie resolution
    emb_pred_shuff = emb_pred[:, random_indices]
    emb_true_shuff = emb_true[:, random_indices]
    # Record the rank label index
    emb_true_sorted, indices = emb_true_shuff.sort(descending=True, dim=-1)
    # Ranking the pred embedding by the true indices
    preds_sorted = gather(emb_pred_shuff, dim=1, index=indices)
    # Implementation of the Eq.1, Eq.2 and Eq.3
    max_pred_values, _ = preds_sorted.max(dim=1, keepdim=True)
    preds_sorted_minus_max = preds_sorted - max_pred_values
    cumsums = cumsum(preds_sorted_minus_max.exp().flip(dims=[1]), dim=1).flip(dims=[1])
    loss = (log(cumsums) - preds_sorted_minus_max) * scale_factor
    return mean(sum(loss, dim=1))

# Cross-modal embeddings
logits_text_per_image=image_embeds @ text_embeds.T
logits_iamge_per_text=logits_text_per_image.T
# In-modal embeddings
logits_image_per_image=image_embeds @ image_embeds.T
logits_text_per_text=text_embeds @ text_embeds.T
# Compute the cross-modal rank loss
Cross_modal_loss=rank_loss(logits_text_per_image,logits_image_per_text)+rank_loss(logits_image_per_text
    , logits_text_per_image)
# Compute the in-modal rank loss
In_modal_loss=rank_loss(logits_image_per_image,logits_text_per_text)+rank_loss(logits_text_per_text,
    logits_image_per_image)
# Rank loss
Rank_loss=Contrastive_loss+Cross_modal_loss+In_modal_loss
```