Mitigating Noisy Correspondence by Geometrical Structure Consistency Learning

Supplementary Material

Contents

1. Reproduction Statement

To facilitate the reproduction of our method, we have included our code in the supplementary materials. To be specific, all relevant code has been zipped into the folder **code**. This folder contains two sub-folders: **GSC**, which puts the implementation of our GSC method on a single network, and **GSC+**, for the GSC method implemented on dual networks. Regarding the benchmark datasets used in our study, namely Flickr30K, MS-COCO, and CC152K, we pre-processed the image and text inputs into representations in line with NCR. This pre-processing approach is also consistent with methodologies used in DECL, MSCN, and BiCro.

Here, we present a detailed overview of the implementation of our GSC method in Listing 1. The core aspect of GSC involves learning the geometrical structures within and across different modalities. And thus noisy correspondence can be discriminated by the structural difference between clean and noisy samples. Such function is mainly implemented in the python file **GSC.py** in folder **model**.

```
# Input image and text data pairs and ids.
def train(self, images, texts, ids):
    # Take predicted targets of last epoch.
    targets = self.targets[ids]

    img_embs, txt_embs = encoder(images, texts)
    # Cross-modal loss and estimated indicator.
    sims_cm = sim_module(img_embs, txt_embs)
    loss_cm, preds_cm = contrastive_loss(sims_cm)

    # Intra-modal loss and estimated indicator.
    img_sims = sim_module(img_embs) * targets
    txt_sims = sim_module(txt_embs) * targets
    sims_im = img_sims @ txt_sims
    loss_im = contrastive_loss(sims_im)
    preds_im = cos_sim(img_sims, txt_sims)

    # Record current epoch prediction.
    self.preds_cm[ids] = preds_cm
    self.preds_im[ids] = preds_im

    # Purify overall loss
    loss = targets * (loss_cm + gamma * loss_im)
    loss.backward()

# Fit preds_im with GMM and update targets.
def update(self):
    # Fit two-component GMM.
    gmm = GaussianMixture(n_components=2)
    gmm.fit(self.preds_im)
    self.preds_im = gmm.predict(self.preds_im)

    # Update targets.
    self.targets_cm = beta1 * self.preds_cm +
        ↪ (1 - beta1) * self.targets_cm
    self.targets_im = beta2 * self.preds_im +
        ↪ (1 - beta2) * self.targets_im
    self.targets = min(self.targets_cm, self.
        ↪ targets_im)
```

Listing 1. Simplified Python Code for GSC

In each training epoch, our approach leverages the targets estimated from the previous epoch to refine the computation of both cross-modal loss (loss_cm) and intra-modal loss (loss_im), intra-modal structures, represented by img_sims and txt_sims. Notably, the targets are tensors of dataset size, which are initialized to ones. As a result, during the first epoch, the multiplication of targets with the losses and structures effectively remains neutral, ensuring that the optimization process is not impacted.

2. Broader Impacts

In this paper, we tackle the emerging issue of noisy correspondence in cross-modal learning, a challenge that significantly hampers the effectiveness of correspondence learning models. This issue is especially common in large-scale, real-world multimodal datasets due to constraints in time and resources. Addressing this problem holds significant value, not only for cross-modal retrieval but also for a range of downstream tasks requiring precise alignment between modalities, such as multimodal classification and image captioning.

Diverging from previous methods that identify noisy samples primarily rely on small loss approaches across modalities, GSC uniquely identifies noisy correspondences by analyzing differences in both cross-modal and intra-modal structures, acknowledging the intricate nature of multimodal learning scenarios. The success of our approach emphasizes the importance of considering the unique characteristics of multimodal learning.

3. Limitations and Future Explorations

While our proposed GSC method demonstrates promising results, there are limitations that require acknowledgement. One limitation is that GSC has only been implemented and evaluated in dual-modality scenarios. However, many multimodal scenarios involve more than two modalities, such as combinations of video, audio, image and text, where the likelihood of noisy correspondence increases. Additionally, multimodal datasets often present challenges beyond noisy correspondence, such as the noisy label problem. The coexistence of these issues can compound the complexity, as the underlying causes may be interconnected. Therefore, we advocate for further research to deepen the understanding of these challenges and to develop strategies to mitigate the associated biases and risks.

4. Stability Experiments

In Tab. 1, we show the mean and standard deviation values of GSC under different noise rates on Flickr30K. To conclude, the performance of GSC is stable and consistent.

Table 1. Stability experiments on Flicker30K. The average and standard deviation scores are calculated based on the same experiments repeated for three times.

Noise	Method	Flickr30K						
		Image → Text			Text → Image			
		R@1	R@5	R@10	R@1	R@5	R@10	Sum
20%	GSC-SGR	77.7 ±0.6	94.8 ±0.2	97.6 ±0.2	59.7 ±0.5	84.4 ±0.3	90.4 ±0.3	504.5 ±1.6
40%	GSC-SGR	76.4 ±0.4	93.9 ±0.5	97.2 ±0.5	57.2 ±0.5	82.8 ±0.2	89.2 ±0.1	496.6 ±1.6
60%	GSC-SGR	72.0 ±1.1	91.4 ±0.2	95.4 ±0.5	53.3 ±0.5	79.8 ±0.4	86.7 ±0.2	478.6 ±1.0

5. Experiments on SGRAF Backbone

The backbone architectures used by state-of-the-art models to achieve optimal performance vary. Specifically, NCR and MSCN employ SGR as their backbone, whereas DECL and BiCro use an enhanced version, SGRAF, which is a combination of SGR and SAF. In the main paper, we primarily discussed GSC's results using the SGR backbone. This choice was made because employing a dual-network strategy with SGRAF would result in an ensemble of four networks, leading to significant time consumption and reduced practical applicability. Moreover, GSC demonstrated superior performance over other methods even with just the SGR backbone. In this section, we present additional experimental results of GSC utilizing the SGRAF backbone, as shown in Tab. 2. Notably, GSC generally shows improved performance with the stronger SGRAF backbone, outperforming the second-best method by average recall sum score gaps of 4.9% under 20% noise, 15.2% under 40% noise, and 19.7% under 60% noise.

6. Aligned Experiments to DECL and MSCN

In our main paper, we benchmark the GSC method against SOTA approaches at 20%, 40%, and 60% noise levels, following BiCro. For most SOTAs, except MSCN and RINCE, we use the results as reported in their respective papers. Since MSCN only conducted experiments under 50% and RINCE considered noisy correspondence under a different setting, we reproduced their results under 40% and 60%. Besides, DECL conducted experiments under an extreme noise condition of 80%. In this context, we extended our comparison to include GSC's performance against NCR and MSCN at 50% noise, and against DECL at 80% noise, as detailed in Tab. 3. The results demonstrate that GSC consistently outperforms other methods at these higher noise levels.

Table 2. Experiments on SGRAF Backbone on Flicker30K. The best results are marked by **bold**.

Noise	Method	Flickr30K						
		Image → Text			Text → Image			
		R@1	R@5	R@10	R@1	R@5	R@10	Sum
20%	SGR	55.9	81.5	88.9	40.2	66.8	75.3	408.6
	SGRAF	72.8	90.8	95.4	56.4	82.1	88.6	486.1
	DECL-SGRAF	77.5	93.8	97.0	56.1	81.8	88.5	494.7
	BiCro-SGARF	78.1	94.4	97.5	60.4	84.4	89.9	504.7
	GSC-SGR	78.3	94.6	97.8	60.1	84.5	90.5	505.8
	GSC-SGRAF	**78.9**	**95.5**	**98.0**	**61.0**	**85.3**	**90.9**	**509.6**
40%	SGR	4.1	16.6	24.1	4.1	13.2	19.7	81.8
	SGRAF	8.3	18.1	31.4	5.3	16.7	21.3	101.1
	DECL-SGRAF	72.7	92.3	95.4	53.4	79.4	86.4	479.6
	BiCro-SGARF	74.6	92.7	96.2	55.5	81.1	87.4	487.5
	GSC-SGR	76.5	94.1	**97.6**	57.5	82.7	88.9	497.3
	GSC-SGRAF	**78.1**	**94.6**	97.4	**59.0**	**83.5**	**90.1**	**502.7**
60%	SGR	1.5	6.6	9.6	0.3	2.3	4.2	24.5
	SGRAF	2.3	5.8	10.9	1.9	6.1	8.2	35.2
	DECL-SGRAF	65.2	88.4	94.0	46.8	74.0	82.2	450.6
	BiCro-SGARF	67.6	90.8	94.4	51.2	77.6	84.7	466.3
	GSC-SGR	70.8	91.1	**95.9**	53.6	79.8	86.8	478.0
	GSC-SGRAF	**72.9**	**92.6**	95.7	**55.7**	**81.4**	**87.7**	**486.0**

Table 3. Experiments on Flicker30K with 50% and 80% noise rates. The best results are marked by **bold**.

Noise	Method	Flickr30K						
		Image → Text			Text → Image			
		R@1	R@5	R@10	R@1	R@5	R@10	Sum
50%	SGR	36.9	68.1	80.2	29.3	56.2	67.0	337.7
	NCR-SGR	72.9	93.0	**96.3**	54.3	79.8	86.5	482.8
	MSCN-SGR	74.4	**93.2**	96.0	55.3	80.4	86.8	486.1
	GSC-SGR	**74.6**	92.5	**96.3**	**55.8**	**81.5**	**88.5**	**489.2**
80%	SGR	1.5	6.6	9.6	0.3	2.3	4.2	24.5
	DECL-SGR	44.4	72.6	82.0	33.9	59.5	69.0	361.4
	GSC-SGR	**60.6**	**84.5**	**92.2**	**44.2**	**70.5**	**79.6**	**431.5**

7. Analysis of Hyper-parameters τ_1 and τ_2

Hyper-parameters τ_1 and τ_2 are the temperature coefficients in cross-modal and intra-modal loss functions and noise indicators. These coefficients are crucial as they scale the similarity measures, thereby controlling separation. This scaling also enhances the distinction between clean and noisy samples. In our experiments, we set $\tau_1 = 0.07$ and $\tau_2 = 1$. Additionally, we have conducted an ablation study on these parameters, as illustrated in Fig. 1. The results indicate that the performance related to τ_2 is relatively stable. For τ_1, the performance is stable when $\tau_1 < 0.07$. A larger τ_1 value tends to diminish the discrimination between different samples.

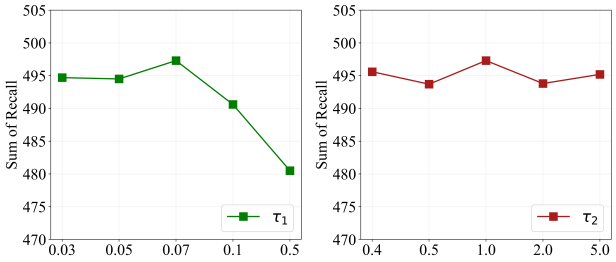

Figure 1. Analysis of hyper-parameters τ_1 and τ_2 on Flicker30K with 40% noise. τ_1 and τ_2 are two temperature coefficients.

Table 4. Analysis of different batch sizes on Flicker30K with 20% and 60% noise rates. The best results are marked by **bold**.

Noise	Batch Size	Image → Text			Text → Image			
		R@1	R@5	R@10	R@1	R@5	R@10	Sum
20%	32	75.4	93.5	96.7	57.5	83.1	89.2	495.3
	64	77.9	94.0	97.7	57.7	84.3	90.1	501.7
	128	**78.3**	94.6	**97.8**	**60.1**	**84.5**	**90.5**	**505.8**
	192	76.8	**94.9**	97.7	59.7	84.2	90.1	503.4
60%	32	66.7	89.2	94.6	50.2	77.4	84.6	462.7
	64	68.7	**91.1**	94.9	51.4	79.0	86.1	471.2
	128	70.8	**91.1**	**95.9**	**53.6**	79.8	86.8	478.0
	192	**72.6**	90.7	95.8	**53.6**	**80.0**	**86.9**	**479.6**

8. Analysis of Batch Size

In the main paper, we explored the impact of varying batch sizes on performance using the Flickr30K dataset under 40% noise condition. Extending this analysis, we now present additional results for 20% and 60% noise scenarios in Tab. 4. Consistent with our findings in the main paper, we observe that larger batch sizes generally enhance performance up to an optimal threshold.

9. Analysis of Temporal Ensembling

In our main paper, we conducted experiments both with and without temporal ensembling to demonstrate its effectiveness. In this section, we present visualizations that show how the predicted weights change, helping us analyze the impact of different values of β_1 and β_2. These visualizations are detailed in Fig. 2. The comparison between figures (a) and (c) reveals that the β parameters control the rate at which the target is updated with the prediction from the current epoch. A larger β value corresponds to a faster update rate. Our results show that various β values are effective in distinguishing noisy samples. However, due to the memorization effects in deep neural networks (DNNs), smaller β values, *i.e.* $\beta < 0.3$, eliminate the effects of noisy samples in a later stage, during which the retrieval model can overfit on noisy samples.

10. Visualization of noisy samples in CC152K

The CC152K dataset is a real-world dataset characterized by 3% to 20% noisy correspondence. In our study, the GSC method demonstrates superior robustness against this real-world noise, evidenced by its enhanced performance on the CC152K dataset. Further illustrating this, Fig. 3 visualizes some of the noisy samples in CC152K identified by GSC. These samples are selected based on their overall weight y, which falls below the threshold of 0.1. We observed that the majority of these detected noisy pairs are either completely or partially mismatched.

The reasons for these mismatches vary. A portion of these noisy pairs, while related as they originate from the same web source, are not directly descriptive of each other. In some cases, mismatches arise from efforts to protect personal information. For the partially mismatched pairs, the text often describes only a fragment of the image. This partial correspondence can lead to ambiguities, as the same text could relate to multiple images within the dataset.

11. Comparison to CapFilt

The vanilla CapFilt module utilizes additional COCO-pretrained reference model to filter samples with noisy correspondence before training. Such a strategy will degenerate when the target dataset differs in distribution from COCO. In Tab. 5, we adopt the same process of CapFilt pre-trained on COCO and verify its power on Flickr30K against GSC, under various simulated noise levels. GSC significantly outperforms CapFilt, confirming the merit of considering training dynamic.

Table 5. Comparison to CapFilt module on Flickr30K.

Noise	Method	Image → Text			Text → Image			
		R@1	R@5	R@10	R@1	R@5	R@10	Sum
20%	CapFilt	76.9	91.8	96.0	54.8	81.5	87.8	488.8
	GSC	**78.3**	**94.6**	**97.8**	**60.1**	**84.5**	**90.5**	**505.8**
40%	CapFilt	72.2	92.4	96.4	52.8	79.2	86.6	479.6
	GSC	**76.5**	**94.1**	**97.6**	**57.5**	**82.7**	**88.9**	**497.3**
60%	CapFilt	68.9	89.5	94.2	49.5	76.4	84.6	463.1
	GSC	**70.8**	**91.1**	**95.9**	**53.6**	**79.8**	**86.8**	**478.0**

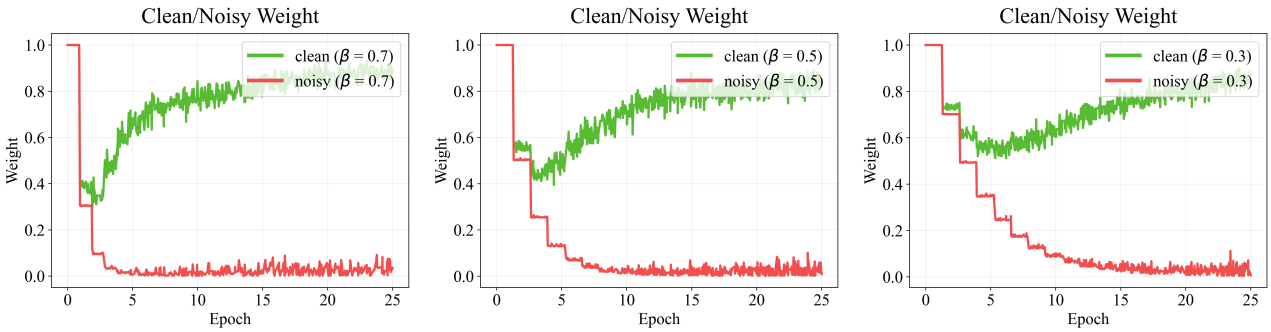

Figure 2. The changing values of clean and noisy sample weight with the noise rate 40% on Flicker30K when β_1 and β_2 are both 0.7, 0.5 and 0.3. β_1 and β_2 parameters are separate momentum coefficients for the cross-modal and intra-modal temporal ensembling.

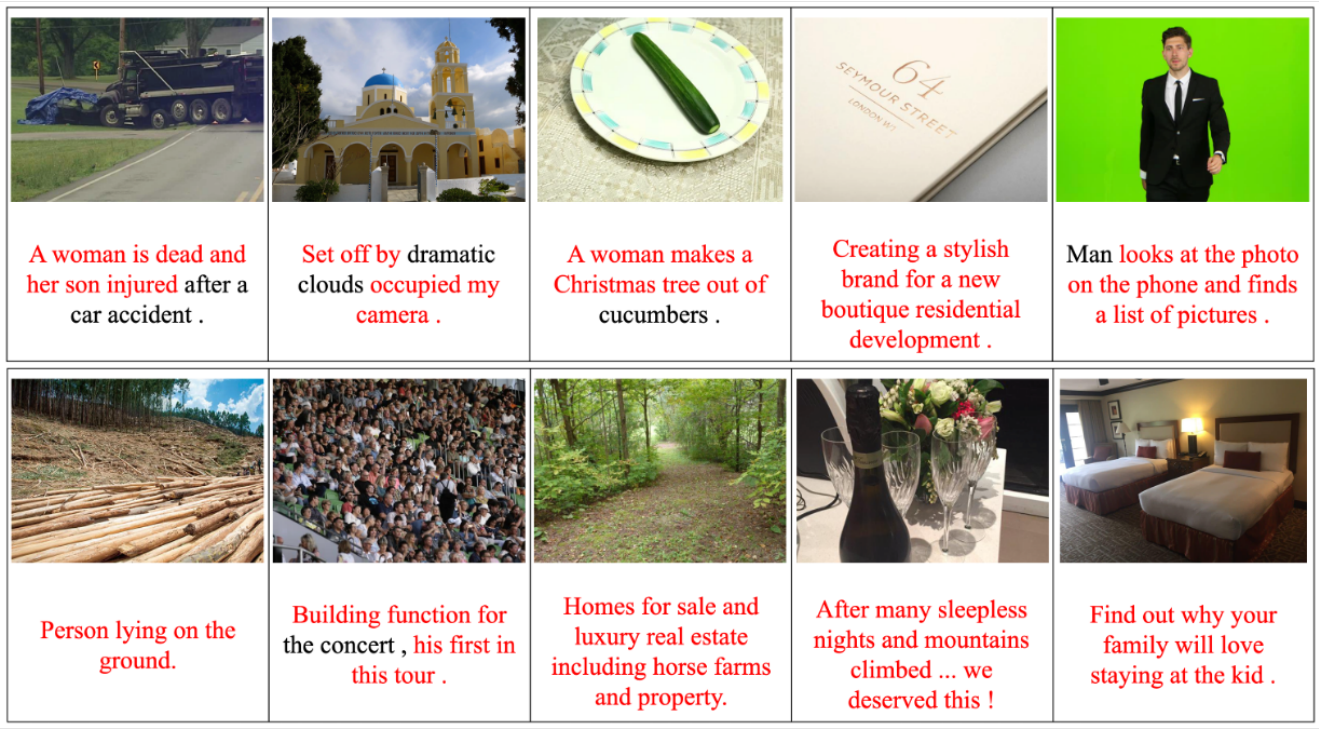

Figure 3. Visualization of noisy data pairs in CC152K. The red color means the text is mismatched to the image.