Explicit Image Caption Editing
****** Supplementary Manuscript ******

Zhen Wang[1*], Long Chen[2*], Wenbo Ma[1], Guangxing Han[2], Yulei Niu[2], Jian Shao[1], and Jun Xiao[1†]

[1] Zhejiang University, Hangzhou, China
[2] Columbia University, New York, USA
zju_wangzhen@zju.edu.cn, zjuchenlong@gmail.com, junx@cs.zju.edu.cn
https://github.com/baaaad/ECE

The supplementary manuscript is organized as follows:

- In Sec. A, we provide more detailed construction steps for both COCO-EE and Flickr30K-EE (*cf.* Sec. 3.2).
- In Sec. B, we discuss the two ways of adding new words in $\text{Tagger}_{\text{add}}$.
- In Sec. C, we explain more details about the calculation of the proposed metric Editing Steps (ES) (*cf.* Sec. 5.1).
- In Sec. D, we show the implementation details.
- In Sec. E, we show the details and architectures of the compared baselines (*cf.* Sec. 5.2).
- In Sec. F, we show the computational efficiency of `TIger` and the compared ECE baselines.
- In Sec. G, we illustrate more visualization results generated by `TIger`.

A More Detailed Benchmark Construction Steps

A.1 COCO-EE

We built COCO-EE based on MSCOCO [7], which contains 123,287 images, and 5 ground-truth captions for each image. To ensure *criteria 1*, we selected all Ref-Caps and GT-Caps in COCO-EE from MSCOCO captions. Specifically, we constructed each editing instance following these steps:

1. **Image-Caption Similarity Filter.** To guarantee *criteria 2*, for each image labeled with 5 captions, we used a pre-trained CLIP [12] model to filter 300 captions from all the rest captions in its respective split set (training/validation/test) based on the CLIP score, where a higher score indicates higher similarity between the image and the caption. Meanwhile, to save the computation cost in the rest filtering steps, we then randomly selected 30 captions from them as Ref-Cap candidates, and we treated all the 5 ground-truth captions as the GT-Cap candidates.

*Zhen Wang and Long Chen are co-first authors with equal contributions.
†Jun Xiao is the corresponding author.

Text Premise:
Two motorcyclists racing neck and neck around a corner

Entailment: Motorcyclists are in a close race around a corner

Neutral: Motorcyclists dressed in different uniforms race around a corner in a stadium

Contradiction: Motorcyclists are stopped at a stop sign

Fig. 6. Instance from e-SNLI-VE.

2. **Caption Similarity Filter.** To guarantee *criteria 3*, we filtered each image's Ref-Cap candidates based on the BLEU [11] score between the Ref-Cap and GT-Cap candidates. We only kept the Ref-Cap candidates whose BLEU scores are greater than a certain threshold (BLEU-2 > 0.4 & BLEU-3 > 0.3).
3. **Caption Differences Filter.** To guarantee *criteria 4*, we filtered each image's Ref-Cap candidates based on the SPICE [1] score between the Ref-Cap and GT-Cap candidates. The SPICE scores reflect the similarity of scenes described by different captions, and we only kept the Ref-Cap candidate whose SPICE score is less than a certain threshold (*i.e.*, SPICE < 0.35).
4. **Edit Distance Filter.** Finally, for each filtered Ref-Cap candidate, we only selected the caption with the shortest edit distance from the corresponding GT-Cap candidates to form a Ref-GT caption pair.

Following the above steps, we constructed the COCO-EE, and divided it into training, validation, and test sets following the "Karpathy" split [5].

A.2 Flickr30K-EE

We built Flickr30K-EE based on dataset e-SNLI-VE [6]. e-SNLI-VE is a visual entailment dataset using the same image set as the image captioning dataset Flicrk30K [17]. Specifically, e-SNLI-VE was built based on the text entailment SNLI [3] dataset, SNLI used the captions from Flickr30K as text premises. For each text premise, there are three human-annotated sentence hypotheses, and each sentence hypothesis has a different relationship with the text premise. The e-SNLI-VE then replaced all the text premises with corresponding Flickr30K images and relabeled the sentence hypothesis to correct labeling errors.

Fig. 6 shows an instance of e-SNLI-VE. For each image in e-SNLI-VE, there are three sentence hypotheses, and each sentence hypothesis has a different relationship with the image premise:

- **Entailment**: if there is enough evidence in the image premise to conclude that hypothesis is true.

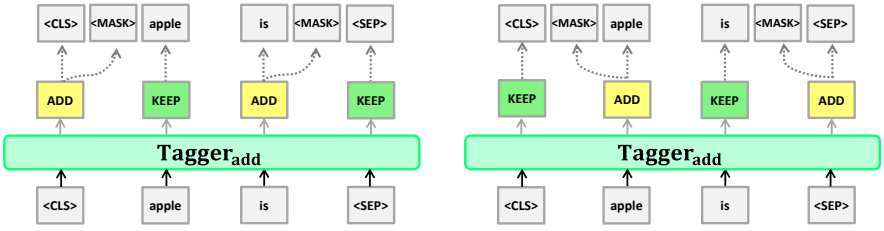


Fig. 7. Two ways of adding new words in $\text{Tagger}_{\text{add}}$.

- **Neutral**: if there is not enough evidence to conclude whether the hypothesis is true or false.
- **Contradiction**: if there is enough evidence in the image premise to conclude that hypothesis is false.

For each image and its textual hypotheses in the e-SNLI-VE, we only selected the contradiction and entailment hypothesis as a Ref-GT caption pair if they used to have the same text premise. We divided it into training, validation, and test sets based on e-SNLI-VE splits.

B Two Ways of Adding New Words in $\text{Tagger}_{\text{add}}$

As shown in Fig. 7, there are two ways to add new words in $\text{Tagger}_{\text{add}}$. Take the simple caption "`apple is`" as an example, to change it into "`the apple is red`", each newly added word (*i.e.*, [`MASK`] token) can be added either *after* the current token or *before* the current token.

When adding *after* the current token, [`CLS`] token should predict `ADD`, meanwhile we never need to add after the final [`SEP`] token, *i.e.*, the ground-truth edit operation for [`SEP`] token is constant (`KEEP`) and could be excluded from the loss computations. When adding *before* the current token, [`SEP`] token should predict `ADD`, and [`CLS`] is constant to predict `KEEP` because we never need to add words before it. There is no essential difference between the two ways in terms of model training. In our experiments, we use the way of adding *after* the current token for `Tiger`.

C Details of Calculating the Editing Steps

As mentioned in Sec. 5.1, the metric Editing Steps (**ES**) is the total number of meaningful editing steps, in this paper, we regard the sum of `DELETE` and `ADD` operations as **ES** since all baselines apply the same set of edit operations (*i.e.*, the three basic operations `KEEP`, `DELETE`, `ADD`, and the combination of them).

Specifically, if an edit operation is a combination of the above basic operations, we can decompose them into the three basic operations. For the V-Felix and V-LaserTagger baselines, we calculated ES as follows:

1. Felix [9] predicts the number of adding words together with the keep and delete operation, *e.g.*, the edit operation `(DELETE|N)` means delete the current token and add N new words after this token. This can save the number edit operation compared to predict each `DELETE` and `ADD` operation individually, but their contribution to the editing process are the same, and `(DELETE|N)` essentially achieves the editing by each `DELETE` and `ADD` operation separately. We thus count `(DELETE|N)` as one `DELETE` operation and N `ADD` operations (N+1 editing steps). Similarly, `(KEEP|N)` is counted as one `KEEP` operation and N `ADD` operations (N editing steps).
2. LaserTagger [10] predicts new words before the deleted or preserved tokens, *e.g.*, the edit operation `(this is|DELETE)` means delete the current token and add two new words "`this is`" before this token. Thus, we count it as one `DELETE` operation and two `ADD` operations for **ES** computing. Similarly, `(this is|KEEP)` is counted as one `KEEP` operation and two `ADD` operations.

Besides the basic edit operations and their combination operations, different (or future) ECE models may design other special or high-level edit operations. For edit operations that essentially changes the token in the sentence, we split them into basic `DELETE` and `ADD` operations for **ES** computing, *e.g.*, `REPLACE`, which replace the current token with a new one, we regard it as deleting the current token and adding a new one, so it will be counted as one `DELETE` operation and one `ADD` operation. For other edit operations that only change the order of the input tokens, we count each of them as one editing step.

D Implementation Details

For visual token features, we used the same bottom-up features from [2], which are extracted by a Faster R-CNN [13] pre-trained on VG [16]. For multimodal BERTs, we used the 12-layer base ViLBERT model [8] and used the checkpoint pre-trained on the Conceptual Captions [16] for initialization. All three modules are trained separately with a XE loss. The batch size was set to 64. We trained these modules with Adam optimizer for 20 epochs, and the initial learning rate was set to 2e-6. We used a linear decay learning rate schedule with warm up to train these modules. Besides, we expanded the editing instances based on the iterative editing process to train $\text{Tagger}_{\text{add}}$ and Inserter (*e.g.*, an editing instance which needs a three-round editing will be expanded into three training samples corresponding to three rounds respectively).

E Details of these Compared Baselines

In this section, we describe the detailed architectures of the compared state-of-the-art baselines.

Three implicit caption editing baselines are all built on top of the widely-used UpDn architecture [2]. Fig. 8 shows their architectures.

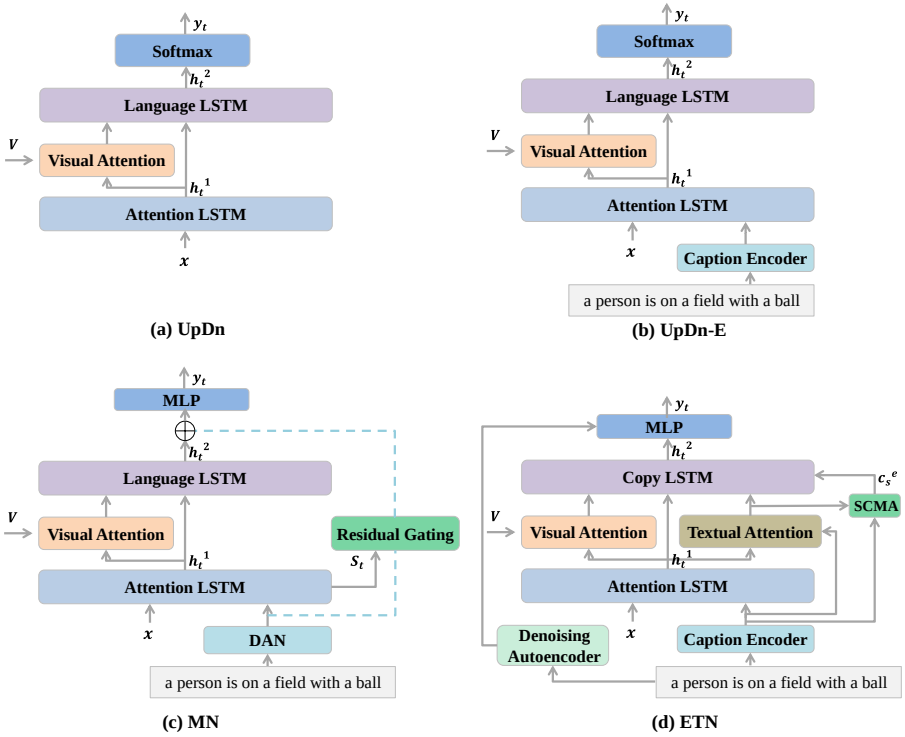
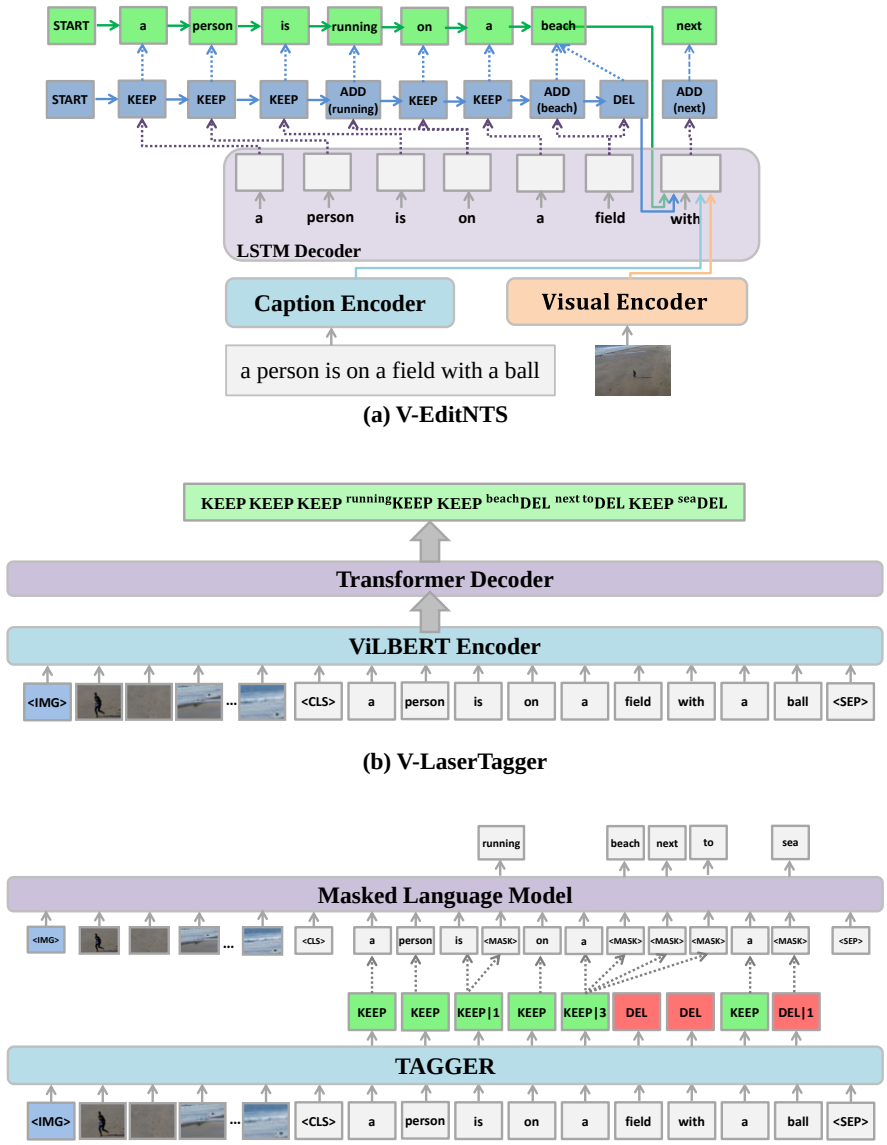


Fig. 8. Architectures of the implicit caption editing baselines

In UpDn, the input vector to the attention LSTM at each time step consists of the previous output of the language LSTM, concatenated with the mean-pooled image feature and the encoding of the previously generated word. The output of the language LSTM at each time step is then used to predict the output word.

1. **UpDn-E** [2]: It uses an extra caption encoder to encode the Ref-Cap, the caption encoder is a bi-directional LSTM same as the one in ETN [15], and the output of caption encoder is concatenated to the input vector;
2. **MN** [14]: It uses a pre-trained Deep Averaging Network (DAN) to encode the Ref-Cap, the output of the DAN is concatenated to the input vector. A residual LSTM gate is used to extract residual information from attention LSTM and DAN, which are then summed with the output of the language LSTM to predict the output word.
3. **ETN** [15]: It uses a Selective Copy Memory Attention (SCMA) to select and copy memory states corresponding to words in the Ref-Cap, and a Copy-LSTM to generate words. Meanwhile, it uses a bi-directional LSTM to encode the Ref-Cap. Besides, it further uses a denoising autoencoder to boost Copy-LSTM and gets the final prediction.



(a) V-EditNTS

(b) V-LaserTagger

(c) V-Felix

Fig. 9. Architecture of the three explicit baselines.

We also extended three text explicit editing models into ECE. Fig. 9 shows their architectures.

1. **V-EditNTS** [4]: It predicts edit operation sequence iteratively by an LSTM, including `KEEP`, `DELETE`, and `ADD`, it also predicts the specific word for `ADD` at the same time. We used an extra visual encoder that projects the mean-

Model	Type	COCO-EE		Flickr30K-EE		FLOPs(M)
		IT(ms)	C	IT(ms)	C	
V-EditNTS [9]	L	68.11	149.0	51.51	129.1	4.55
V-Felix [24]	T	93.80	139.5	76.40	127.4	15.02
V-LaserTagger [25]	T	325.45	127.1	313.08	104.0	10.42
TIger (round=1)	T	105.46	180.2	105.45	142.8	22.53
TIger (round=2)	T	172.83	189.8	170.00	148.1	37.55
TIger (round=3)	T	237.93	193.9	238.62	148.3	52.57
TIger (round=4)	T	304.36	194.8	302.21	148.3	67.59

Table 9. Results of average inference time (IT), CIDEr-D (C) and inference FLOPs of ECE models. Type "L" and "T" denote LSTM-based and Transformer-based models, respectively.

pooled image feature to match the dimension of the caption encoding. The input vector to the decoder LSTM at each time step consists of the caption encoding of Ref-Cap, concatenated with the visual encoding of the input image, the embedding of the current token, the LSTM encoding of the previous edit operation and previous output word.

2. **V-LaserTagger** [10]: It combines a ViLBERT [8] encoder with an autoregressive Transformer decoder. Specifically, it uses three edit operations: keeping a token, deleting a token, and adding new words before the token. The adding words are restricted to the phrase vocabulary that is derived from respective training data (COCO-EE and Flickr30K-EE)[3].
3. **V-Felix** [9]: It is composed of two ViLBERT [8] models including a tagging model that predicts operations to keep or delete the token, it will also predict the number of new words to add after the token. And an insertion model that predicts specific words for new adding. For fairness, we used the V-Felix without the reordering mechanism and the way of insertion was set to [`MASK`] prediction.

F Computational Efficiency

For more complete experiment results, we reported general computational efficiency evaluation metrics of our model and the ECE baselines. As shown in Table 9, due to model structure, Transformer-based models tend to have longer inference time and larger FLOPs than LSTM-based counterparts. Indeed, the computational cost of TIger increases continuously with more round editing. However, one-round TIger can still achieve much superior performance with nearly computational efficiency compared to other ECE baselines. In this paper, we hope TIger can serve as a strong baseline, and we mainly focus on **editing efficiency** of ECE models. In contrast, *the computational cost like FLOPs*

[3] We also tried different vocabulary sizes (*e.g.*, fixing to 500 as in [10]), the empirical results are similar without obvious differences.

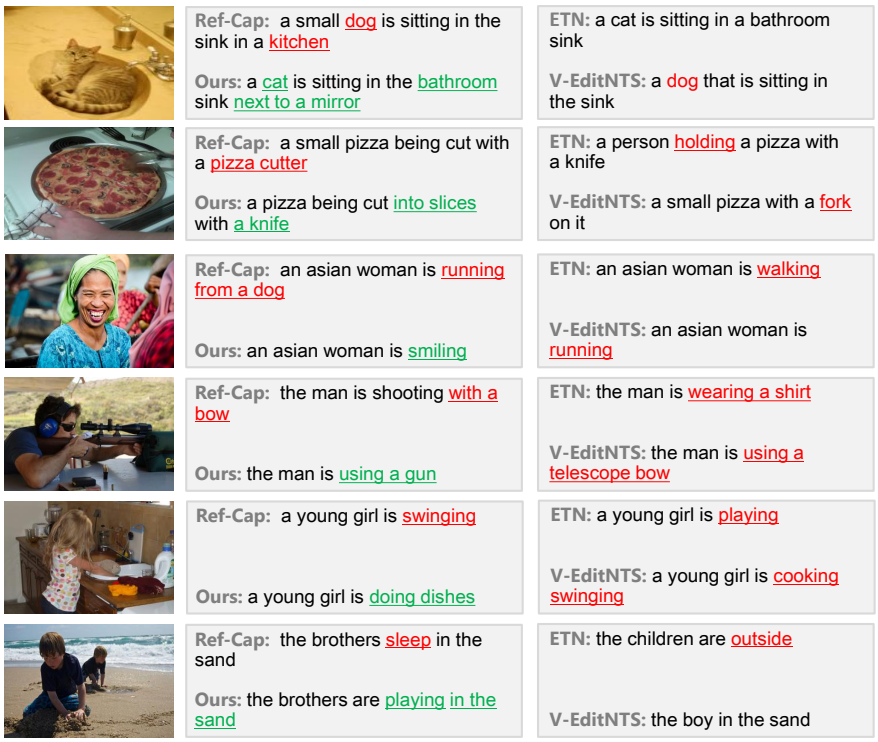


Fig. 10. Visualization results of our model compared to two baselines (ETN and V-EditNTS) in COCO-EE (top two samples) and Flickr30K-EE (bottom four samples).

was hardly reported as a key metric of models in existing caption generation or editing works.

G More Qualitative Results

Fig. 10 shows more results generated by `TIger` compared to baselines. The first two samples are from COCO-EE, we can observe that our model is not only capable of recognizing and correcting incorrect details (*i.e.*, "`dog`" to "`cat`", "`kitchen`" to "`bathroom`", and "`cutter`" to "`knife`") but also adding new details (*e.g.*, "`next to a mirror`"). The rest examples are from Flickr30K-EE and their Ref-Caps are relatively short, we can observe that our model can still correct the incorrect details (*e.g.*, change from "`running from a dog`" to "`smiling`", "`with a bow`" to "`using a gun`", "`swinging`" to "`doing dishes`, and "`sleep`" to "`playing`"") without breaking the structure of the caption (*e.g.*, `in the sand`).

References

1. Anderson, P., Fernando, B., Johnson, M., Gould, S.: Spice: Semantic propositional image caption evaluation. In: ECCV. pp. 382–398. Springer (2016) 2
2. Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., Zhang, L.: Bottom-up and top-down attention for image captioning and visual question answering. In: CVPR. pp. 6077–6086 (2018) 4, 5
3. Bowman, S.R., Angeli, G., Potts, C., Manning, C.D.: A large annotated corpus for learning natural language inference. arXiv (2015) 2
4. Dong, Y., Li, Z., Rezagholizadeh, M., Cheung, J.C.K.: Editnts: An neural programmer-interpreter model for sentence simplification through explicit editing. In: ACL (2019) 6
5. Karpathy, A., Fei-Fei, L.: Deep visual-semantic alignments for generating image descriptions. In: CVPR. pp. 3128–3137 (2015) 2
6. Kayser, M., Camburu, O.M., Salewski, L., Emde, C., Do, V., Akata, Z., Lukasiewicz, T.: e-vil: A dataset and benchmark for natural language explanations in vision-language tasks. arXiv (2021) 2
7. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV. pp. 740–755 (2014) 1
8. Lu, J., Batra, D., Parikh, D., Lee, S.: Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. NeurIPS (2019) 4, 7
9. Mallinson, J., Severyn, A., Malmi, E., Garrido, G.: Felix: Flexible text editing through tagging and insertion. arXiv (2020) 4, 7
10. Malmi, E., Krause, S., Rothe, S., Mirylenka, D., Severyn, A.: Encode, tag, realize: High-precision text editing. arXiv (2019) 4, 7
11. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: ACL. pp. 311–318 (2002) 2
12. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. arXiv (2021) 1
13. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. NeurIPS (2015) 4
14. Sammani, F., Elsayed, M.: Look and modify: Modification networks for image captioning. arXiv (2019) 5
15. Sammani, F., Melas-Kyriazi, L.: Show, edit and tell: A framework for editing image captions. In: CVPR. pp. 4808–4816 (2020) 5
16. Sharma, P., Ding, N., Goodman, S., Soricut, R.: Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In: ACL. pp. 2556–2565 (2018) 4
17. Young, P., Lai, A., Hodosh, M., Hockenmaier, J.: From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. TACL **2**, 67–78 (2014) 2