This CVPR 2020 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Show, Edit and Tell: A Framework for Editing Image Captions

Fawaz Sammani[1], Luke Melas-Kyriazi[2]
[1]Multimedia University, [2]Harvard University
fawaz.sammani@aol.com, lmelaskyriazi@college.harvard.edu

Abstract

Most image captioning frameworks generate captions directly from images, learning a mapping from visual features to natural language. However, editing existing captions can be easier than generating new ones from scratch. Intuitively, when editing captions, a model is not required to learn information that is already present in the caption (i.e. sentence structure), enabling it to focus on fixing details (e.g. replacing repetitive words). This paper proposes a novel approach to image captioning based on iterative adaptive refinement of an existing caption. Specifically, our caption-editing model consisting of two sub-modules: (1) EditNet, a language module with an adaptive copy mechanism (Copy-LSTM) and a Selective Copy Memory Attention mechanism (SCMA), and (2) DCNet, an LSTM-based denoising auto-encoder. These components enable our model to directly copy from and modify existing captions. Experiments demonstrate that our new approach achieves state-of-art performance on the MS COCO dataset both with and without sequence-level training. Code can be found at https://github.com/fawazsammani/show-edit-tell.

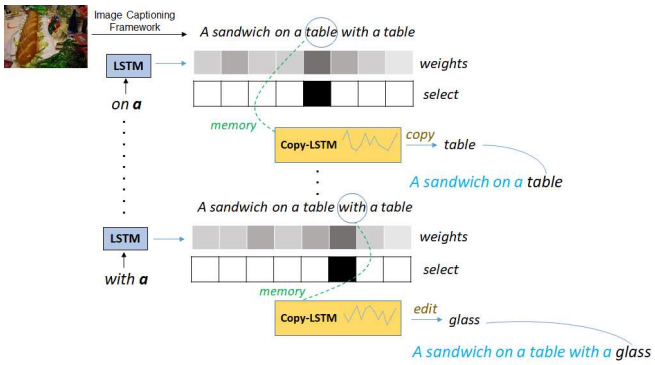

Figure 1. Our model learns how to edit existing image captions. At each decoding step, attention weights (grey) are generated; these correspond to the importance of each word in the existing caption for the word currently being generated in the new caption. Using a selective copy memory attention (SCMA) mechanism, we select the word with the highest probability and directly copy its corresponding LSTM memory state to our language LSTM (Copy-LSTM). That is, rather than learning to copy words directly from the input caption, we learn whether to copy the hidden states corresponding to these words. We then generate our new caption from this (possibly copied) hidden state. Best viewed in color.

1. Introduction

Image captioning is the task of producing a natural language description of a visual scene. As one of the prototypical examples of multimodal learning, image captioning combines techniques from computer vision (e.g. recognizing salient objects in an image), with those from natural language processing (e.g. generating coherent sentences describing these objects). Applications of image captioning include content-based image retrieval [18] and assisting the visually impaired by converting visual signals into text, which can then be transformed to speech using text-to-speech technologies [17].

Over the past five years, neural encoder-decoder systems have gained immense popularity in the field of image captioning due to their superior performance compared to previous image processing-based techniques. The current state-of-art image captioning models are composed of a CNN encoder, an LSTM (or Transformer) decoder, and one or more attention mechanisms. The input image is first encoded by a CNN into a set of feature vectors, each of which captures semantic information about an image region, and these feature vectors are decoded using an LSTM-based or Transformer-based network, which generates words sequentially. Attention mechanisms enable the decoding process to "focus" on particular image regions during generation, and the specific formulation of these mechanisms has been the center of much research [8, 2, 28, 16, 30].

High-quality captions consist of two elements: coherent natural language sentences (i.e. sentence/caption structure), and visually-grounded content (i.e. accurate details). Current image captioning models learn a ground-up map-

ping from image features to full captions, hoping to capture both elements simultaneously. Examining the outputs of prior image captioning models [28, 2, 29], we observe that models learn global sentence/caption structure exceptionally well, but they often produce incorrect, inconsistent, or repetetive content.

Motivated by this observation, in recent months, researchers have begun to consider the problem of editing inputs independent from the problem of generating inputs [7, 23]. Intuitively, editing should be easier than generating from scratch, because a caption-editing model can focus on visually-grounded details rather than on caption structure [23]. For example, consider Figure 1: A state-of-art image captioning framework [8] outputs "A sandwich on a table with a table." The network produces a sensible sentence structure for this particular image ("A __ on a __ with a __") but fails to properly fill in the nouns, repeating the main object in the image ("table"). A caption-editing model, applied to this caption, should be able to recognize this error (the noun repetition) and modify the caption to read "A sandwich on a table with a glass of wine" or perhaps simply "A sandwich on a table".

We propose a novel approach to image captioning based on iterative adaptive refinement of an existing caption rather than from-scratch caption generation. At each decoding step of the caption editing process, a word from the existing caption is selected and its corresponding memory state is copied into the internal structure of the LSTM (the *Copy-LSTM*). This Copy-LSTM includes a separate selective copy attention mechanism (SCMA), enabling it to further edit or copy the existing word into the final output caption. For example, in Figure 1, our model chooses to copy the first instance of the word "table" and edit the second instance to "glass". Ultimately, our model produces: "A sandwich on a table with a glass of wine".

In summary, our contributions are as follows:

- We propose EditNet, a framework for editing existing image captions that consists of a Copy-LSTM equipped with a Selective Copy Memory Attention (SCMA) mechanism. Alongside EditNet, we propose DCNet, a denoising auto-encoder that learns to denoise previous captions. We optimize DCNet with a novel objective function (MSE between hidden states), finding it to be a simple and effective way to improve the performance of our decoder.

- Our method achieves a new state-of-the-art performance on MS COCO dataset.

- We present an ablation analysis of the components of our model, demonstrating that each aspect contributes non-trivially to our model's final performance.

2. Related Work

2.1. Image Captioning

Image captioning has been widely studied in the computer vision and natural language processing communities for multiple decades. Traditional captioning systems, which were primarily used for video captioning, involved detecting keywords and using these keywords to fill in hand-made templates [20, 19]. These models had the advantage of always producing logical sentence structures, but their expressive power was severely limited due to the need for researchers to manually design templates.

In the past five years, neural network-based image captioning models have risen to prominence. Introduced by [27], these approaches fall into the broader category of encoder-decoder models, alongside those for machine translation, summarization, speech recognition, and a host of other tasks [24]. Specifically, [27] proposed a captioning model consisting of a CNN encoder and an LSTM decoder, in which the output of the CNN encoder (the final convolutional layer) was used as input to the LSTM. [28] dramatically improved upon the model introduced by [27] with the addition of an attention mechanism. These mechanisms have engendered large performance improvements across sequence learning tasks [28, 3, 6].

Of the attention mechanisms designed specifically for image captioning, bottom-up and top-down attention (Up-Down; [2]) and the recent attention-on-attention (AoA; [8]) have proven among the most successful. [2] adds a top-down attention LSTM before the language LSTM to selectively attend to spatial image features. [8], currently the state-of-the-art, adds an attention-on-attention module after both the language LSTM and the standard attention mechanism. This module is designed to measure the relevance between the attention result and the query; it transforms the output of the standard attention mechanism, multiplying it element-wise by an attention gate (a different transformation of the output followed by a sigmoid function).

Finally, parallel to improvements in attention mechanisms, [22] proposed a new optimization objective for image captioning. Traditionally, image captioning models are trained to minimize the cross-entropy between their word-level output and the ground truth caption. [22] instead optimizes a sequence-level objective, such as CIDEr [26] or METEOR [4], using methods adopted from reinforcement learning. It is now common in the literature to evaluate the performance of new models using both cross-entropy and self-critical training.

2.2. Sequence-to-Sequence Editing

In the past year, a new paradigm based around editing the output of sequence-to-sequence models has been shown to improve the performance of a large class of models. [7]

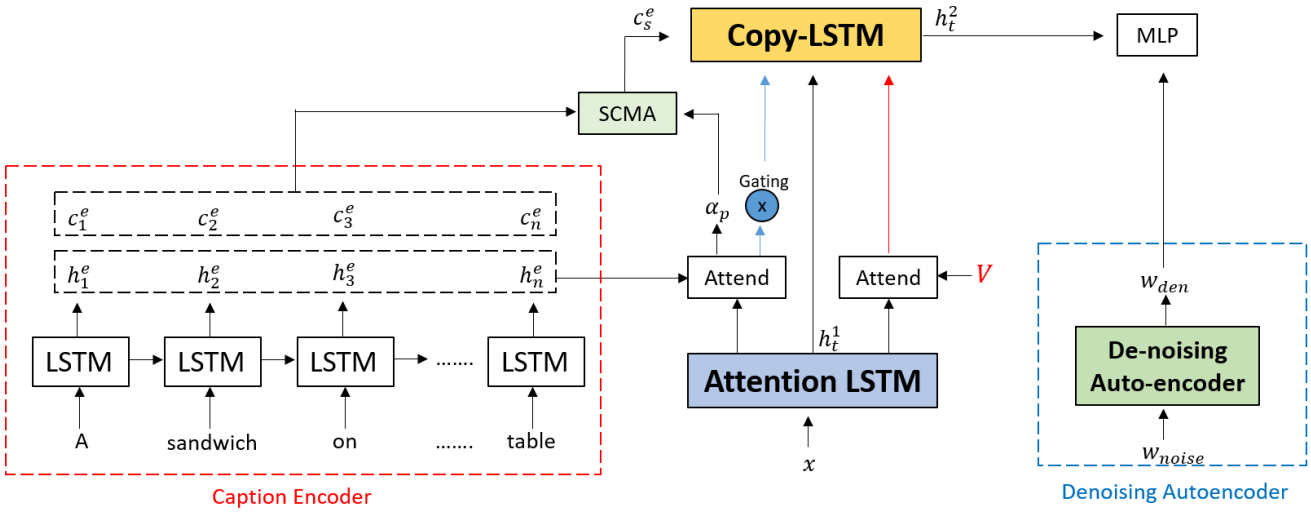

Figure 2. Our proposed model which includes two submodules: On the left, EditNet along with its decoder (middle) is shown. For EditNet, the existing caption is first encoded using a uni-directional LSTM, where each LSTM cell outputs a word representation (the hidden state h_t^e and a memory state c_t^e). The hidden states are used to calculate attention weights which are then passed to the SCMA mechanism along with the memory states. The SCMA selects a single memory state and directly copies it to the Copy-LSTM which includes an adaptive copy mechanism in its internal structure and can choose weather to "copy" or "edit" an existing word. The textual attended vector is gated to remove incorrect attended words before being passed as an input to the Copy-LSTM along with the visual attention vector. EditNet is equiped with an LSTM-based denoising auto-encoder (right) which takes as input the existing caption, encodes it into a compressed representation and then decodes the compressed representation to its expected output.

proposed a retrieve-and-edit framework for generating sequences such as source code from natural language inputs. The authors showed strong performance gains over standard sequence-to-sequence models on a code autocompletion task and the Hearthstone cards benchmark; these results suggests that editing existing outputs may be easier than generating new outputs from scratch.

Most recently, [23] proposed the task of editing image captions. [23] used a deep averaging network to encode an existing caption (outputted by a traditional sequence-to-sequence model) into a vector, and added a gated output of this vector along with the LSTM output, enabling the LSTM to model the "residual" information. This model improved upon the performance of some baseline methods, but failed to outperform the state-of-the-art in image captioning.

In this work, we introduce a new framework for the editing task proposed in [23], where we employ copy mechanisms to better take advantage of the information in the existing caption. Compared to both [23] and the state-of-the-art in image caption generation (discussed above) [8], we show significant performance improvements across image captioning metrics.

3. Proposed Methodology

Our model consists of two submodules: *EditNet* and *DCNet*. In the following sections, we describe each of these submodules in detail. A complete overview of our model is shown in Figure 2.

3.1. EditNet

EditNet is a model designed to learn whether to copy or edit each word in an input caption. It has an encoder-decoder structure with two components: (1) a Selective Memory Attention Mechanism (SCMA) and (2) a Copy-LSTM decoder. We describe these parts in detail in the following subsections.

3.1.1 EditNet Architecture

We base the general structure of EditNet on the widely-used bottom-up and top-down architecture from [2].

Given an image, our encoder extracts a set of 36 visual features using an R-CNN based network. We denote these features by $V = \{v_1, v_2, \ldots, v_k\}$ where $v_i \in \mathbb{R}^{2048}$ and k is the number of objects (in our case, $k = 36$).

Given output of our encoder and an input caption, our decoder produces an edited version of the input caption. Like [2], our decoder contains an attention LSTM and a language LSTM. Unlike previous work, we add an input caption LSTM and a novel SCMA module, and we replace the language LSTM with a Copy-LSTM.

First, we encode the input caption using a uni-directional one-layer LSTM (see the dashed red box in Figure 2). In the following sections, we denote the encoded input caption by $\overline{h}_s = [h_1^e \ldots h_n^e]$, where n is the number of words in the

input caption. We denote the memory states of the corresponding LSTM cells by $\bar{c}_s = [c_1^e \dots c_n^e]$.

Next, we feed the following inputs to the attention LSTM: the word embedding vector, the last hidden state of the caption encoder, the mean-pooled image features $\overline{\boldsymbol{v}} = \frac{1}{k}\sum_i \boldsymbol{v}_i$, and the previous hidden state of the language LSTM. That is, we input $x_t^1 = \left[w_t; h_n^e; \bar{v}; h_{t-1}^2\right]$ where ; indicates concatenation. Note that this attention LSTM is a standard LSTM, not a Copy-LSTM, because it does not take input from the SCMA module. The output of the attention LSTM h_t^1 is used to compute two attention vectors, one over the visual features and another over the textual features. These are fused with a gating mechanism and used as input to the Copy-LSTM.

The attention weights over textual features are also used as input to the SCMA module; this module may be thought of as learning to select and copy from the input caption LSTM, and its output is used as input to the Copy-LSTM. The Copy-LSTM takes as input the outputs of the attention LSTM along with the visual attended vector and the gated textual vector. It outputs a hidden state h_t^2, which is passed to a final linear layer to predict the softmax probability distribution over the vocabulary. Finally, this distribution is fused with the output of the denoising autoencoder (described in section 3.2) to produce the final output word.

3.1.2 Selective Copy Memory Attention (SCMA)

The SCMA (Figure 3) enables our model to select and copy memory states corresponding to words in the input caption.

We measure the similarity between the current initial decoder output h_t^1 and each word in the previous caption $\overline{h}_s$ using a shallow neural network followed by a softmax:

$$\alpha_p = \text{softmax}\left(w_a^T \tanh\left(W_s \bar{h}_s + W_h h_t^1\right)\right) \quad (1)$$

Different from the conventional attention mechanism, we do not utilize α_p directly. Instead, we utilize the corresponding memory state in the input caption encoder LSTM. To be precise, we copy the corresponding memory state c_t^e from the input caption encoder with the highest similarity (i.e. highest softmax output from α_p).

Notably, this indexing operation is non-differentiable. To get around this problem, we employ the re-parametrization trick [12]. We construct two masks, a binary mask and a shifting mask. The binary mask m_b includes a 1 in the index of the maximum probability value of the softmax output α_p, while the shifting mask m_s includes the residual values that shift the result of $\alpha_p m_b$ of the maximum word to 1 and 0 otherwise. Mathematically, this operation is:

$$c_S^e = \sum_{i=1}^{n} \left(\alpha_{p_i} m_{b_i} + m_{s_i}\right) c_i^e \quad (2)$$

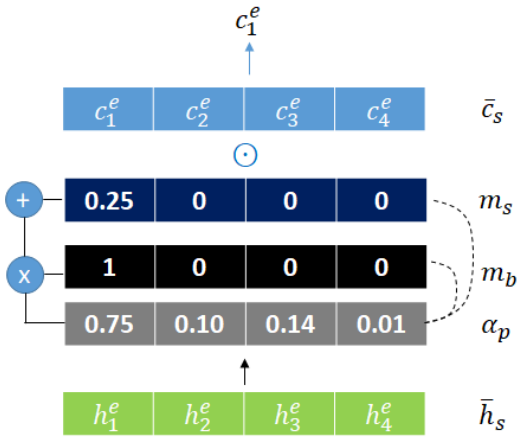

Figure 3. The operational flow of SCMA. Attention weights (grey) are computed from the encoded output of the input caption and highlight the importance of each word in accordance to the current generated word at the language model. The attention weights are then used to calculate two masks: a binary mask m_b which is multiplied with the corresponding attention weight of each word, and a shifting mask m_s which shifts the multiplication result to 1. Finally, each resulting element is multiplied with the corresponding memory state. Eventually, all memory states are eliminated except for the one with the maximum attention weight, which is the final copied output.

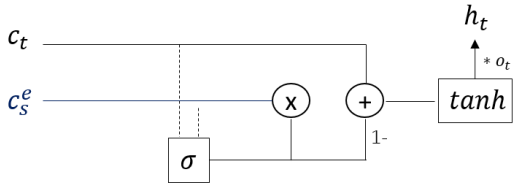

Figure 4. The structure of our Copy-LSTM (Equations 4-6).

For example, if the attention weight for the maximum word is 0.8, then $m_{b_i} = 1$ and $m_{s_i} = 0.2$. Therefore, the extracted memory state would be $c_i^e(0.8 \cdot 1 + 0.2) = c_i^e$. Similarly, if the attention weight for a non-maximum word is 0.3, then $m_{b_i} = 0$ and $m_{s_i} = 0$. In this case, $c_i^e(0.3 \cdot 0 + 0) = 0$, and c_i^e would be eliminated. Consequently, all words with a probability lower than the maximum value would be multiplied by 0, and the memory cell c_i^e with the maximum probability would remain.

We utilize the copied memory state, denoted C_S^e, in the Copy-LSTM described below.

3.1.3 Copy-LSTM

To incorporate the information from the input caption and SCMA module into the language decoder, we augment the LSTM cell with an adaptive copy mechanism. Our modified LSTM cell, which we denote Copy-LSTM, includes a "copy gate" that controls how much information is taken

from the SCMA module relative to the other input sources (the visual features and the hidden state).

We now give a mathematical formulation for the Copy-LSTM. As in a standard LSTM, the forget gate, input gate, output gate and memory state calculation are given as:

$$\begin{aligned} f_t &= \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \\ i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\ \tilde{C}_t &= \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \\ C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\ o_t &= \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \end{aligned} \tag{3}$$

In addition, we incorporate a copy gate, c_{g_t}, which may be thought of as calculating the similarity between the copied memory state and the word currently being generated:

$$c_{g_t} = \sigma\left(W_n \cdot [C_t, C_s^e]\right) \tag{4}$$

We then compute the amount to take from both memory states, and modify the LSTM memory state to:

$$C_{ap_t} = c_{g_t} * C_s^e + (1 - c_{g_t}) * C_t \tag{5}$$

The hidden state is then computed with a tanh activation function of the newly constructed memory state, multiplied by the output gate:

$$h_t = o_t * \tanh\left(C_{ap_t}\right) \tag{6}$$

With these modifications, the Copy-LSTM is able to incorporate the desired information into its output representation h_t. It passes this hidden state to the output layer, which predicts the next word in the caption. Note that if the gate c_{g_t} is 1, then the word from the input caption is fully copied, and if it is 0, then the previous caption is ignored and the word is generated anew. An overview of the modified internal structure of the LSTM (Copy-LSTM) is shown in Figure 4.

3.1.4 Context Gating

As mentioned earlier, our model attends over the textual features $\bar{h}_s$ of the existing caption. Intuitively, however, attending over the textual features may mislead the language LSTM when the existing caption contains incorrect information. Inspired by the recent advances in neural machine translation, we incorporate a "context gate" that learns how much to focus on the source context (the textual attended feature vector) and the target context (the word embedding vector and the current LSTM hidden state). That is:

$$c_m = z_t \odot \tanh\left(W_s c_t\right) + (1 - z_t) \odot \tanh\left(W_t \cdot [w_t; h_t]\right)) \tag{7}$$

where

$$z_t = \sigma\left(W_Z \cdot [w_t; h_t; c_t]\right) \tag{8}$$

and

$$c_t = \sum_{i=1}^{n} \alpha_{p_i} h_i^e \tag{9}$$

Note that equation 7 is different from [25], where we include the gate and its compliment before the activation function. We find that this operation performs better in completely removing the unwanted elements in the attention vector. Also note that α_p in equation 9 is same as that of equation 1. We find that sharing the parameters give better similarity scores and reduces the number of overall parameters.

3.2. Denoising Captioner (DCNet)

In parallel to EditNet, we use a denoising auto-encoder, denoted DCNet, to denoise our input caption. Denoising auto-encoders are traditionally used to re-construct noisy images. In our case, we may think of our input caption as a noisy version of a true caption.

DCNet is composed of a bi-directional LSTM encoder, which encodes the noisy caption into a compressed representation, and an LSTM decoder, which decodes the compressed representation. Note that DCNet operates only on textual features, without any visual features. Additionally, the parameters in DCNet are not shared with the parameters in EditNet. Further details on DCNet are included in the supplementary material.

3.3. Objectives

We first train our model by optimizing the cross entropy (XE) loss:

$$L_{XE}(\theta) = -\sum_{t=1}^{T} \log\left(p_\theta\left(y_t^* | y_{1:t-1}^*\right)\right) \tag{10}$$

After training with cross-entropy, we additionally optimize our language decoder using mean-squared error between the last decoder hidden state of the language model and the last hidden state of the ground truth caption. This ground truth caption hidden state is obtained by running the ground-truth caption through the encoder of the denoising auto-encoder. In sum, this loss is given by:

$$L_{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left(h_n^d - h_n^g\right) \tag{11}$$

where we linearly project the last hidden state of the language model h_n^2 without using any activation function:

$$h_n^d = W_d h_n^2 + b_d \tag{12}$$

We provide ablation studies on this scheme in section 4.4, where we demonstrate an increase in the CIDER score of DCNet from 1.171 to 1.183. This optimization scheme

is simple and not restricted to our model; it can be applied to general sequence-to-sequence or vector-to-sequence task. Our final loss (for non-sequence-level training) is:

$$L = L_{XE}(\theta) + L_{MSE} \tag{13}$$

Finally, for comparison with other works, we directly optimize for CiDEr-D using sequence-level training [22]. As in [22], the policy gradient is:

$$\nabla_\theta L_{RL}(\theta) \approx -\left(r\left(C^s\right) - b\right) \nabla_\theta \log p_\theta\left(C^s\right) \tag{14}$$

where $r\left(C_i^s\right)$ is the CIDEr score of the sampled caption and b is the CIDEr score of a greedily decoded caption [22].

3.4. Implementation Details

EditNet: For visual features, we use bottom-up features from [2]. For textual features, we use captions from [8].[1]

We set the embedding and hidden size of both the LSTM encoder and decoder network to 1024 and the attention dimension to 512. We train EditNet for 15 epochs using cross-entropy, as in equation 12. Note that for EditNet, we do not use MSE optimization after training with cross-entropy. However, we still provide ablation studies on training EditNet with MSE optimization.

We use the ADAM optimizer [11] with batch size 80, initial learning rate 5e-4, and decay the learning rate decay by a factor of 0.8 every 3 epochs. We increase the scheduled sampling probability by 0.05 every 5 epochs [5]. We optimize the CIDEr-D score with sequence-level training for another 25 epochs with an initial learning rate of 5e-5 and anneal it by 0.5 when the CIDER-D score shows no improvement for one epoch. We do not use label smoothing.

DCNet: DCNet fully operates on textual features, without using any visual features. For the encoder LSTM, we set the hidden size to 512 for each direction, ending up with a dimension of 1024 for both directions. For the decoder, we choose the top-down decoder [2] with a hidden size of 1024. The embedding dimension is set to 1024 and the attention dimension to 512. We train DCNet for 4 epochs using cross-entropy, as in equation 10 and optimize it with MSE for one additional epoch, as in equation 13. We set the batch size to 60 and use the same optimization settings (for both XE and CIDER-D optimization) as EditNet.

4. Experiments

4.1. Dataset

We evaluate our proposed method on the popular MS COCO dataset [14], which contains 123,287 images labeled with 5 captions for each by 5 different people. We use the standard "Karpathy" data split [10] for the offline performance comparisons, in which 5,000 images are used for validation, 5,000 are used for testing and 113,287 are used for training. We convert all sentences to lower case and remove words that occur fewer than 3 times from our vocabulary, ending up with a vocabulary of 13,368 words. For evaluation, we use 4 different metrics: BLEU (1- to 4-grams) [21], ROUGE-L [13], CIDEr-D [26] and SPICE [1]. All metrics are computed with the standard public evaluation code. [2]

4.2. Quantitative Analysis

Offline Evaluation: We report the performance of our model compared with the current state-other-art in Tables 1 and 2. These models include NIC [27], which uses a vanilla CNN-LSTM encoder-decoder framework; SCST [22], which optimizes a standard attention-based model using non-differentiable metrics; Adaptive [15] which uses a visual sentinel to eliminate visual attention over non-visual words; Up-Down [2] which uses an attention LSTM to attend over image features extracted from a Faster R-CNN model; RFNet [9] which uses multiple CNNs and LSTMs that are connected to each other; GCN-LSTM [29] which predicts an image scene graph and fuses it with the visual features to produces better feature vectors; and finally AoANet [8] which uses a Transformer-based language model and filters incorrect elements out of the attended visual vector.

For the cross-entropy loss training stage in Table 1, our single model achieves the highest score on all metrics with the exception of SPICE, where its score is marginally lower than [8]. For the sequence-level optimization stage, our model also achieves the highest scores across all metrics except CIDER-D, where is slightly below the published results from [8] and equal to the pretrained model released by [8]. Our model also dramatically outperforms the only other caption-editing model, Modification Networks (MN) [23], when using cross-entropy training (sequence-level results for MN are not reported).

Online Evaluation: The performance of our model on the official MS-COCO online testing server is provided in the supplementary material.

4.3. Qualitative Analysis

Figure 5 and 6 show some results generated by our editing framework. In Figure 5, we can see that the current state-of-art framework [8] generates a sentence where it recognizes the correct objects, but fails to arrange them in the correct format (standing *on a person*). Using these captions as input to our editing framework, our model is able to fix the sentence and arrange the objects in the correct format (standing *on top of a floor*). The right side of Figure 5 shows a visualization of the textual alignment between

[1] We use the pretrained model: https://github.com/husthuaan/AoANet

[2] https://github.com/tylin/coco-caption

Table 1. Performance of our **single model** and other state-of-the-art models on MS-COCO "Karpathy" test split under cross-entropy training. All values are reported as percentage (%). * indicates the results obtained from the publicly available pre-trained model. - indicates that the results are not provided. † indicates results from previous models trained to edit captions, rather than generate captions.

Model	Cross-Entropy Loss						
Metric	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	CIDEr-D	SPICE
NIC[27]	-	-	-	29.6	52.6	94.0	-
SCST [22]	-	-	-	30.0	53.4	99.4	-
Adaptive [15]	74.2	58.0	43.9	33.2	54.9	108.5	19.4
Up-Down [2]	77.2	-	-	36.2	56.4	113.5	20.3
MN (Up-Down)† [23]	76.9	61.2	47.3	36.1	56.4	112.3	20.3
RFNet [9]	76.4	60.4	46.6	35.8	56.8	112.5	20.5
GCN-LSTM [29]	77.3	-	-	36.8	57.0	116.3	20.9
AoANet [8]	77.4	-	-	37.2	57.5	119.8	**21.3**
AoANet* [8]	77.3	61.6	47.9	36.9	57.3	118.4	**21.6**
ETN (Ours)	**77.9**	**62.5**	**48.9**	**38.0**	**57.7**	**1.200**	21.2

Table 2. Performance of our **single model** and other state-of-the-art models on MS-COCO "Karpathy" test split under CIDER-D score optimization. All values are reported as percentage (%). * indicates the results obtained from the publicly available pre-trained model. - indicates that the results are not provided.

Model	Sequence-Level Optimization						
Metric	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	CIDEr-D	SPICE
NIC [27]	-	-	-	31.9	54.3	106.3	-
SCST [22]	-	-	-	34.2	55.7	114.0	-
Up-Down [2]	79.8	-	-	36.3	56.9	120.1	21.4
RFNet [9]	79.1	63.1	48.4	36.5	57.3	121.9	21.2
GCN-LSTM [29]	80.5	-		38.2	58.3	127.9	22.0
AoANet* [8]	80.5	65.2	50.1	39.1	58.9	**128.9**	22.7
AoANet [8]	80.2	-	-	38.9	58.8	**129.8**	22.4
ETN (Ours)	**80.6**	**65.3**	**51.1**	**39.2**	**58.9**	**128.9**	**22.6**

the detected words (y-axis) and the existing words (x-axis). From this, we can see which words the SCMA mechanism selected and copied to the Copy-LSTM.

Figure 6 demonstrates that our model is also capable of replacing repetitive words and adding details to captions. The first three examples show that AoANet often repeats words when it is unable to recognize the correct details in the image (e.g. *with a train station*, *and a stove*, *a bench*). Our editing model successfully fixes these issues by replacing the repetitive words. The last example in Figure 6 demonstrates that our model can add additional details to an existing caption, even when the visual features are minimal in the image (*with its landing gear down*).

4.4. Ablation Studies

In this section, we provide ablation studies on using mean-squared error (MSE) optimization after training with cross-entropy, and on using the Copy-LSTM alone along with the SCMA mechanism.

Table 3 gives results for EditNet and DCNet with and without using MSE optimization. The results without MSE are obtained after training each submodule with cross-entropy (XE) loss, while the results with MSE optimization are after optimizing EditNet with both XE (first alone) and MSE (together with XE). For DCNet, the addition of one epoch of MSE training boosts the BLEU-4 score from 36.9 to 37.2 and the CIDEr-D score from 117.1 to 118.3. We also examine the performance of our Copy-LSTM alone: we remove the visual features, the context gate and the DCNet sub-module, and we train the EditNet with cross-entropy (without MSE optimization). Our scores for BLEU-1/BLEU-2/BLEU-3/BLEU-4/ROUGE-L/CIDER-D are **77.3, 61.7, 48.0, 37.0, 57.2** and **117.3**, respectively. This translates to no improvement over

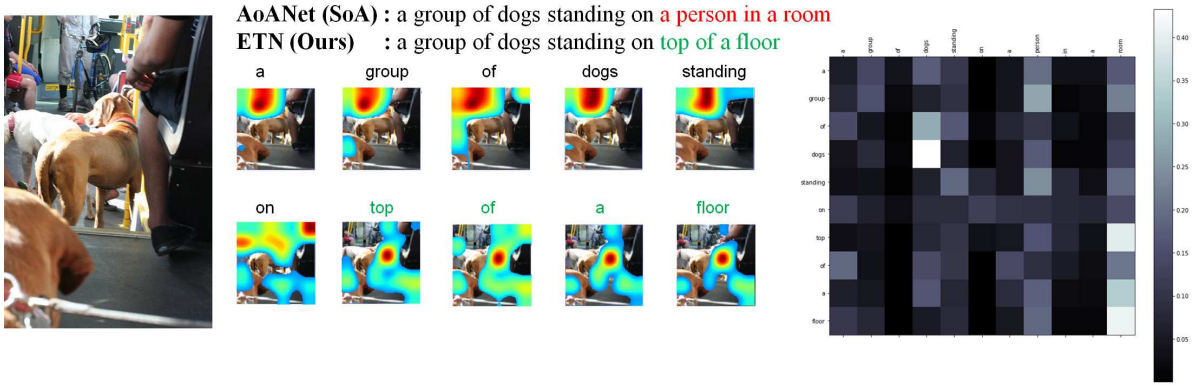

Figure 5. A caption generated by our editing framework when supplied with an input captions from AoANet [8], along with its visual attention maps (left) and textual alignment plots (right). The alignment plot visualizes the words selected by the SCMA mechanism and copied to the Copy-LSTM.

Figure 6. More results from our model compared to AoANet [8]

.

the pre-trained AoANet model for some metrics and a very small improvement for others. Moreover, we examine the performance of the context gate in our EditNet sub-module without any visual features, and find that the context gate improves the CIDER-D score from 117.3 to 117.5. By contrast, EditNet with visual features and textual context gating gives better scores across the board. Finally, we examine the performance of the non-differentiable indexing in the SCMA mechanism. We find that simply using soft-attention on the memory states achieves a CIDER-D score of 119.2, which is lower than the achieved score of 1.200 when using non-differentiable indexing.

Table 3. The effect of using MSE optimization after cross-entropy training. B-4 indicates BLEU-4 and C indicates CIDEr-D.

Model	**DCNet**		**EditNet**	
Metrics	**B-4**	**C**	**B-4**	**C**
w/o MSE Optimization	36.9	117.1	38.0	118.0
w/ MSE Optimization	37.2	118.3	38.0	118.5

5. Conclusion

In this paper, we propose a framework for editing image captions based on iterative adaptive refinement of an existing caption. This new perspective enables our framework to focus on fixing details of existing captions, rather than generating new captions from scratch. Specifically, our model consists of two novel sub-modules: (1) *EditNet*, a language module with an adaptive copy mechanism (*Copy-LSTM*) and a Selective Copy Memory Attention mechanism (*SCMA*), and (2) *DCNet*, an LSTM-based denoising auto-encoder. Experiments on the MS COCO dataset demonstrate that our approach achieves state-of-art performance across image captioning metrics. In the future, our framework may be extended to related tasks such as visual question answering and neural machine translation.

References

[1] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. In *ECCV*, 2016. 6

[2] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6077–6086, 2017. 1, 2, 3, 6, 7

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015. 2

[4] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *IEEvaluation@ACL*, 2005. 2

[5] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, 2015. 6

[6] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Advances in neural information processing systems*, pages 577–585, 2015. 2

[7] Tatsunori B. Hashimoto, Kelvin Guu, Yonatan Oren, and Percy Liang. A retrieve-and-edit framework for predicting structured outputs. In *NeurIPS*, 2018. 2

[8] Lun Huang, Wenmin Wang, Jie Chen, and Xiao-Yong Wei. Attention on attention for image captioning. *ArXiv*, abs/1908.06954, 2019. 1, 2, 3, 6, 7, 8

[9] Wenhao Jiang, Lin Ma, Yu-Gang Jiang, Wei Liu, and Tong Zhang. Recurrent fusion network for image captioning. In *ECCV*, 2018. 6, 7

[10] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015. 6

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 6

[12] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013. 4

[13] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *ACL 2004*, 2004. 6

[14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context. In *ECCV*, 2014. 6

[15] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3242–3250, 2016. 6, 7

[16] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 375–383, 2017. 1

[17] Haley MacLeod, Cynthia L Bennett, Meredith Ringel Morris, and Edward Cutrell. Understanding blind people's experiences with computer-generated captions of social media images. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 5988–5999. ACM, 2017. 1

[18] Vicente Ordonez, Xufeng Han, Polina Kuznetsova, Girish Kulkarni, Margaret Mitchell, Kota Yamaguchi, Karl Stratos, Amit Goyal, Jesse Dodge, Alyssa Mensch, et al. Large scale retrieval and generation of image descriptions. *International Journal of Computer Vision*, 119(1):46–59, 2016. 1

[19] Jia-Yu Pan, Hyung-Jeong Yang, Pinar Duygulu, and Christos Faloutsos. Automatic image captioning. In *2004 IEEE International Conference on Multimedia and Expo (ICME)(IEEE Cat. No. 04TH8763)*, volume 3, pages 1987–1990. IEEE, 2004. 2

[20] Jia-Yu Pan, Hyung-Jeong Yang, Christos Faloutsos, and Pinar Duygulu. Gcap: Graph-based automatic image captioning. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 146–146. IEEE, 2004. 2

[21] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2001. 6

[22] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1179–1195, 2016. 2, 6, 7

[23] Fawaz Sammani and Mahmoud Elsayed. Look and modify: Modification networks for image captioning. *British Machine Vision Conference (BMVC)*, abs/1909.03169, 2019. 2, 3, 6, 7

[24] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014. 2

[25] Zhaopeng Tu, Yang P. Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. Context gates for neural machine translation. *Transactions of the Association for Computational Linguistics*, 5:87–99, 2016. 5

[26] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4566–4575, 2014. 2, 6

[27] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015. 2, 6, 7

[28] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015. 1, 2

[29] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. Exploring visual relationship for image captioning. In *ECCV*, 2018. 2, 6, 7

[30] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659, 2016. 1