PicSOM Experiments in TRECVID 2020

Pre-workshop draft – Revision: 1.5

Jorma Laaksonen and Zixin Guo

Department of Computer Science
Aalto University School of Science
P.O.Box 15400, FI-00076 Aalto, Finland
firstname.lastname@aalto.fi

Abstract

This year, the PicSOM team participated only in the Video to Text Description (VTT), Description Generation subtask. In total, the PicSOM team submitted four runs. We had two goals in our submissions, first, to study the performance of our recent developments in the architectures of the captioning model, and second, to see the effect of using the VATEX dataset in model training. The submitted four runs are as follows:

- PicSOM.1.primary: Our latest and best stacked attention model, trained with three datasets.
- PicSOM.2: Model architecture similar to our best VTT 2019 submission, trained with three datasets.
- PicSOM.3: Another well-performing stacked attention model, trained with two datasets.
- PicSOM.4: Model architecture similar to our best VTT 2019 submission, trained with two datasets.

The runs aim at comparing different implementations of stacked attention on the visual features and the benefit from using the VATEX dataset. Based on our results we can conclude that the use of the VATEX dataset had more effect on the improvement of the results than the stacked attention, which also produced small but noticeable improvement. Based on the results of the runs, it seems that our latest attention model combined with self-critical reinforcement learning was the best approach.

I. Introduction

In this notebook paper, we describe the PicSOM team's experiments for the TRECVID 2020 evaluation [1]. We participated only in the Video to Text Description (VTT) subtask Description Generation. Our approaches are variations of the "Show and tell" model [2], augmented with a richer set of contextual features [3] and self-critical training [4]. The captioning models are described in more detail in Section II and their used training loss functions in Section III. Then, we describe the features in Section IV and the datasets used for training in Section V. In Section VI we intoduce our stacked attention model. Our experiments, submitted runs and results are discussed in Section VII and conclusions are drawn in Section VIII.

II. DeepCaption neural captioning model

The PicSOM team's LSTM [5] model has been implemented in PyTorch and is available as open source.[1] The features are translated to the hidden size of the LSTM by using a fully connected layer. We apply dropout and batch normalization [6] at this layer. As the loss function, we similarly use cross entropy, in addition to Reinforcement Learning with self-critical loss function [4] in order to fine-tune a well-performing model. The fine-tuning is implemented either by switching to the self-critical loss in training time or by specifying a pre-trained model to load and fine-tune.

III. Training loss functions

In order to train the architecture so that its output distribution approximates the target distribution at each decoding step t, several optimisation objectives are used. Recent progress on sequence training enables new optimisation paradigms, which are applied and compared in this work.

A. Cross-entropy

Traditionally, the teacher forcing algorithm [7] is the most common method to maximise the log-likelihood of a model output X to match the ground truth $y = \{y_1, y_2, \cdots, y_T\}$. It minimises the cross-entropy objective

$$\mathcal{L}_{CE} = -\sum_{t=1}^{T} \log p_\theta \left(y_t \mid y_{t-1}, \mathbf{h}_{t-1}, X \right), \tag{1}$$

where $\mathbf{h}_{t-1}$ is the hidden state of the RNN from the previous step and p_θ the probability of an output parametrized by θ. In the inference time, the output can be produced simply by greedy sampling of the sequence being generated.

B. Self-critical

Lately, Reinforcement Learning ideas have been used to optimise a captioning system based on recurrent neural network language models. Such a system can be seen as an agent taking actions according to a policy π_θ and outputting a word $\hat{y}_t$ as an action.

One proposed approach is the self-critical algorithm [4], where the output at inference time of the model $\hat{y}^g_{i,t}$ is used, normally applying greedy search. The sequences are scored using a reward function r. Thanks to the properties of this optimisation, NLP metrics can be used as reward to affect

[1] https://github.com/aalto-cbir/DeepCaption

the actual loss. In our case, CIDErD [8] is used. The final objective reads

$$\begin{aligned}\mathcal{L}_\theta = \tfrac{1}{N}\textstyle\sum_{i=1}^{N}\sum_t \log \pi_\theta \left(\hat{y}_{i,t} \mid \hat{y}_{i,t-1}, \mathbf{s}_{i,t}, \mathbf{h}_{i,t-1}\right) \\ \cdot \left(r(\hat{y}_{i,1}, \cdots, \hat{y}_{i,T}) - r(\hat{y}^g_{i,1}, \cdots, \hat{y}^g_{i,T})\right)\end{aligned} \quad . \qquad (2)$$

IV. Features

Table I summarizes the features used in our experiments and their dimensionalities.

TABLE I
Summary of the features used in our experiments.

abbr.	feature	dim.	modality
rn	ResNet-152	2048	image
i3d	I3D	2048	video
fake-i3d	Fake I3D	2048	image

A. ResNet-152

We are using pre-trained CNN features from ResNet-152 so that the 2048-dimensional features from the pool5 layer averaged across five crops from the original and horizontally flipped images. When applied to a video object, we have used the middlemost frame of the video.

B. I3D

To encode video features, we adopted Inflated 3D Convolutional Network (I3D) [9]. It builds upon already competent image recognition models (2D) and inflates the filters and kernels to 3D, thus creating an additional temporal dimension. Concretely, the base network used is ImageNet-pretrained Inception-V1 [10] using two streams [11]. The videos were first resampled to 25 frames per second as in the original I3D paper and 128 frames were taken from the center. For DeepCaption, the extractor is applied convolutionally over the whole video and the output is average-pooled in order to produce a 2048-dimensional feature vector.

C. Fake I3D

When we used still images of the COCO dataset for training a captioning model, we naturally were not able to extract and use I3D features for those images. Therefore we had calculated the average value of the I3D feature vectors in the TGIF dataset and used that vector as a “fake I3D” feature for all COCO images.

V. Training data

Table II gives a summary of the databases and the features we have extracted for them. In Tables II and III, we have shortened the dataset names with one letter abbreviations.

A. COCO

The *Microsoft Common Objects in COntext (MS COCO)* dataset [12] has 2,500,000 labeled instances in 328,000 images, consisting on 80 object categories. COCO is focused on non-iconic views (or non-canonical perspectives) of objects, contextual reasoning between objects, and precise 2D localization of objects.

TABLE II
Summary of the training datasets used in our experiments.

	dataset	items	captions	features
C	COCO	82,783 img	414,113	rn fake-i3d
T	TGIF	125,713 vid	125,713	rn i3d
V	VATEX	41,250 vid	825,000	rn i3d

B. TGIF

The *Tumblr GIF (TGIF)* dataset [13] contains 100,000 animated GIFs and 120,000 natural language sentences. This dataset aims to provide motion information involved between image sequences (or frames).

C. VATEX

As an addition to the training datasets we have used earlier, we have now started to use the new VATEX video captioning dataset [14]. VATEX contains over 41,250 videos and 825,000 captions in both English and Chinese.

VI. Stacked attention

Figure 1 shows the overall architecture of DeepCaption’s new caption generation model. The visual features and captions are treated as inputs to the encoding and decoding layers, and the last output of the decoding layers are processed by an LSTM. All the outputs from the decoding layers are collected and used to attend the representations generated by the recurrent language model to produce the output words.

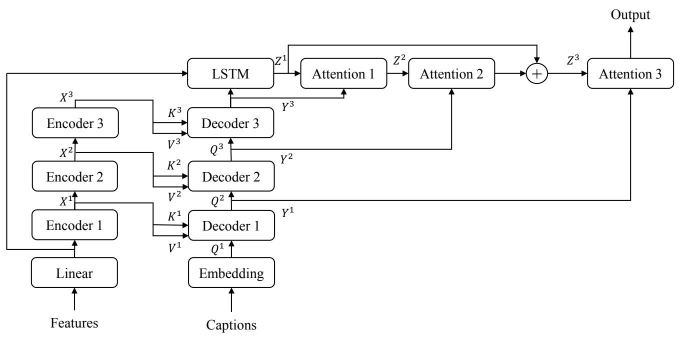

Fig. 1. The architecture of DeepCaption’s new stacked attention model.

The stacked attention model is based on the Transformer model [15], in which the intra- and cross-relations between the visual and the text features are calculated via scaled dot-product attention. The attention function receives three sequential sets with length s, and d_{model} dimensions, denoted as queries Q, keys K, and values V. The attention function is defined as

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_{model}}})V \ , \qquad (3)$$

where $Q \in \mathbb{R}^{s \times d_{model}}$ is a matrix of query vectors and K and $V \in \mathbb{R}^{s \times d_{model}}$ are matrices of key and value vectors. Given a set of features from videos, intra-modality attention is obtained in the encoder with self-attention on the different feature inputs. Cross-modality dependencies are modeled in the decoder via cross-modal attention operations between the visual and textual features. Multihead attention is employed for improving the feature representation and with k heads it is formulated as

$$Multihead(Q,K,V) = concat(h_1,\dots,h_k)W^O \tag{4}$$
$$h_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{5}$$
$$i = 1,\dots,k \ ,$$

with matrices $W_i^Q \in \mathbb{R}^{d_{model}\times d_{model}/k}$, $W_i^K \in \mathbb{R}^{d_{model}\times d_{model}/k}$ and $W_i^V \in \mathbb{R}^{d_{model}\times d_{model}/k}$ used in each of the k attention heads, and $W^O \in \mathbb{R}^{d_{model}\times d_{model}}$.

In our stacked attention model, we have used the depth of $N = 3$ layers, as seen in Figure 1. Both the encoding and decoding layers are stacked sequentially. The stack of N encoding layers generates multi-level outputs $X = (X^1,\dots,X^N)$ which are used as the key and value inputs, K and V, of the cross-modal attention in each corresponding decoder. The query inputs Q come there from the word embeddings of the caption. The multi-level cross-modal relations of visual and textual features provide refined inputs for the attention on the recurrent language model. The decoding layers depend on the visual features and the previously generated words. We collect them and exploit the outputs level by level.

The stacked attention mechanism always uses the decoder output Y^{N-j+1} to attend the attention-stacked LSTM output Z^j. First, with $j = 1$ and given the decoder output Y^N and the LSTM output Z^1, the stacked attention mechanism concatenates Y^N and Z^1 and transforms them linearly to the same dimension with Z. The stacked attention is then defined as element-wise or Hadamard product

$$StackedAttention(Y^{N-j+1}, Z^j) = \alpha(Y,Z) \odot Z \ , \tag{6}$$

where we have dropped the superscripts on the right for clarity and $\alpha(\cdot,\cdot)$ is a function that generates a element-wise multiplication matrix which has the same dimensions as Z. The function $\alpha(\cdot,\cdot)$ is defined as

$$\alpha(Y,Z) = \sigma(W\,[Y,Z] + b) \ , \tag{7}$$

where we have dropped the superscripts on the right for clarity and $\alpha(\cdot,\cdot)$ is a function that generates a element-wise multiplication matrix which has the same dimensions as Z. The function $\alpha(\cdot,\cdot)$ is defined as

$$\alpha(Y,Z) = \sigma(W\,[Y,Z] + b) \ , \tag{8}$$

where $[\cdot,\cdot]$ stands for concatenation, $\sigma(\cdot)$ is the sigmoid function, and W and b are the weight and the bias. The stacked attention for the full sequence of LSTM outputs is then formed by applying the attention (6) sequentially with $j = 1,\dots,N$. As can be seen in Figure 1, we have additionally utilized a skip connection from the LSTM output Z^1 to Z^3.

Word-level cross-entropy (XE) is used to pre-train the model, which is then fine-tuned via reinforcement learning. During the XE training, the model predictions are conditioned on the previous annotated words. Training with reinforcement learning employs the self-critical (SC) [16] training method. During the decoding, both greedy and stochastic samples of the output sequences are used at each time step. We employ the CIDEr-D [17] score as the reward of the SC reinforcement learning. The reward is baselined by a greedy sample rather than the mean of rewards. The gradient is then defined as

$$\nabla_\theta L(\theta) = -\frac{1}{M}\sum_{i=1}^{M}\Big((r(w^i) - r(\hat{w}))\nabla_\theta \log p(w^i)\Big) \ , \tag{9}$$

where w^i is the i-th stochastic sample in a batch, $\hat{w}$ is the greedy search sample and $r(\cdot)$ is the CIDEr-D reward function. When predicting, we perform greedy search and keep words with the highest predicted probabilities within the vocabulary.

VII. Experiments and Results

During the development stage, we mostly kept the selection of the trainig datasets and features fixed and concentrated on selecting the best model architecture and training strategy. We evaluated our results using the previously released ground truth of TRECVID VTT 2018 test set. The four runs submitted are identified as "s1" to "s4" in Table III.

Our four runs were mutually different fromeach other in the following:

- s1: Our latest and best stacked attention model, trained with all three dtasets: COCO, TGIF and VATEX.
- s2: Model architecture similar to our best VTT 2019 submission, trained with all three datasets.
- s3: Another well-performing stacked attention model, trained with two datasets: COCO and TGIF.
- s4: Model architecture similar to our best VTT 2019 submission, trained with two datasets: COCO and TGIF.

Based on evaluation on the TRECVID VTT 2018 and 2019 test sets, we ended up using a 2-layer LSTM for DeepCaption with an embedding vector size of 512, and 1024 for the hidden state dimensionality in all PicSOM team's runs. Both in the input translation layer and in the LSTM we applied a dropout of 0.5. We used Adam optimiser [18] for the self-critical stage with a learning rate of 5×10^{-5} and no weight decay. Additionally, gradient clipping is performed when a range $[-0.1, 0.1]$ is exceeded. The models were pretrained using centered RMSprop [19] with a learning rate of 0.001 and weight decay (L2 penalty) of 10^{-6}.

Our results compared to those of the other submitted runs are visualized with bar charts for each automatic performance measure in Figures 2–7.

VIII. Conclusions

There were two main research questions in the PicSOM team's set of four submissions. We wanted to see the benefit we could get from our novel stacked attention model in our DeepCaption captioning system. The results with the stacked attention model were quite systematically better than those without it, but the advantage was quite limited. We also studied, how much the intriduction of the VATEX dataset as an additional training data to COCO and TGIF improved the results. In this case the positive effect was clear and the

TABLE III
RESULTS OF OUR SUBMISSIONS S1,...,4.

	2020					
id	**METEOR**	**CIDEr**	**CIDErD**	**BLEU**	**SPICE**	**STS**
s1	**0.2617**	**0.319**	**0.200**	*0.0527*	**0.079**	0.4406
s2	*0.2556*	*0.312*	*0.191*	**0.0536**	*0.076*	*0.4293*
s3	0.2414	0.278	0.129	0.0485	0.069	**0.4581**
s4	0.2323	0.278	0.124	0.0201	0.067	0.4458

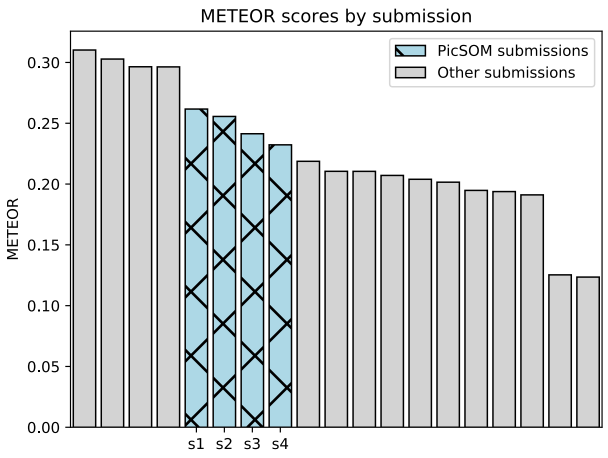

Fig. 2. METEOR results of the PicSOM team and others.

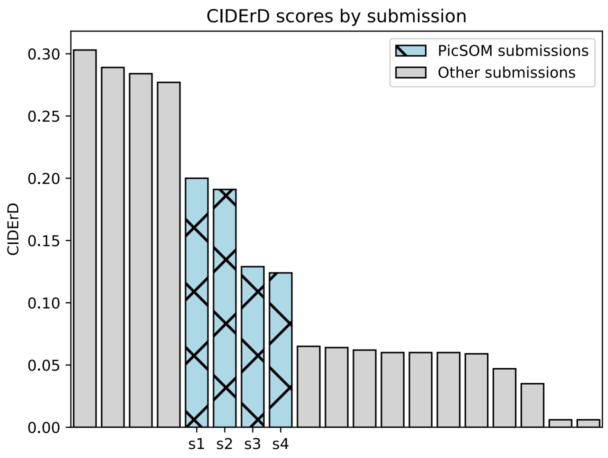

Fig. 4. CIDErD results of the PicSOM team and others.

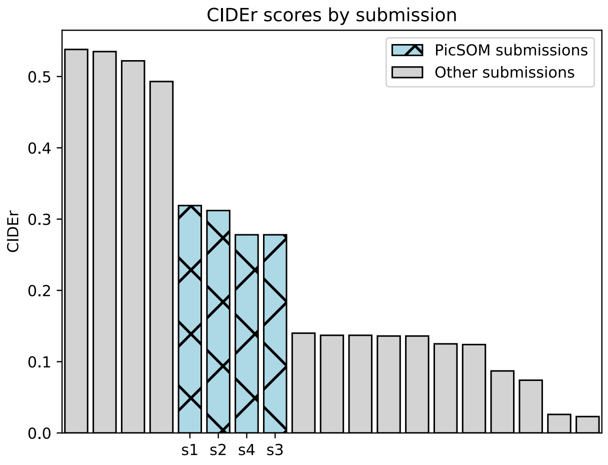

Fig. 3. CIDEr results of the PicSOM team and others.

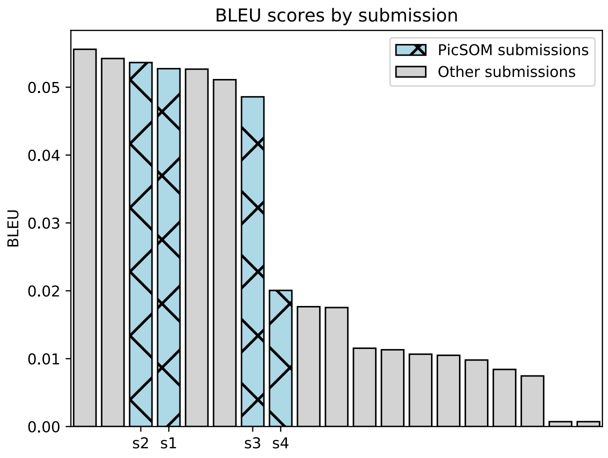

Fig. 5. BLEU results of the PicSOM team and others.

our results were clearly improved from the performance level where we were in the last year's submissions.

ACKNOWLEDGMENTS

This work has been funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 780069 *Methods for Managing Audiovisual Data: Combining Automatic Efficiency with Human Accuracy* (MeMAD). This work was supported by the Academy of Finland Flagship programme: Finnish Center for Artificial Intelligence, FCAI. The calculations were performed using computer resources provided by the Aalto University's *Aalto Science IT* project and CSC – IT Center for Science Ltd. We also acknowledge the support of NVIDIA Corporation with the donation of TITAN X and Quadro P6000 GPUs used for parts of this research.

REFERENCES

[1] George Awad, Asad A. Butt, Keith Curtis, Yooyoung Lee, Jonathan Fiscus, Afzal Godil, Andrew Delgado, Jesse Zhang, Eliot Godard, Lukas Diduch, Jeffrey Liu, Alan F. Smeaton, Yvette Graham, Gareth J. F. Jones, Wessel Kraaij, and Georges Quénot. Trecvid 2020: comprehensive campaign for evaluating video retrieval tasks across multiple application domains. In *Proceedings of TRECVID 2020*. NIST, USA, 2020.

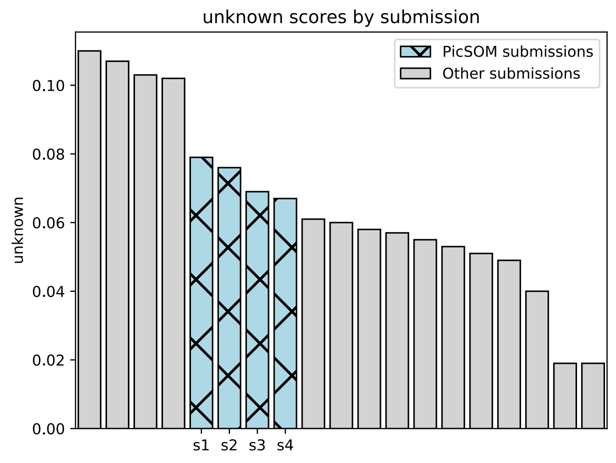

Fig. 6. SPICE results of the PicSOM team and others.

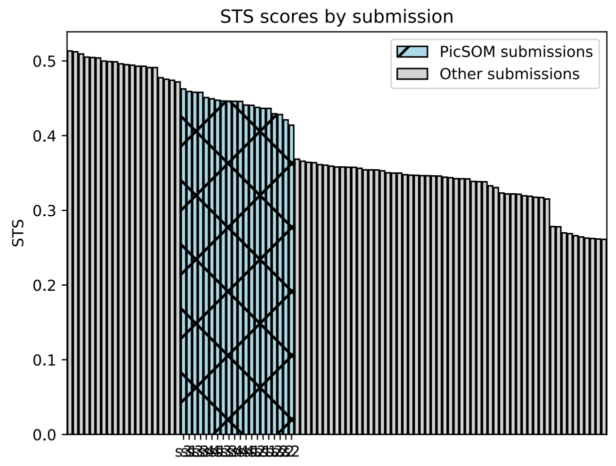

Fig. 7. STS results of the PicSOM team and others.

[2] 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 (CVPR)*, June 2015.

[3] Rakshith Shetty, Hamed R.-Tavakoli, and Jorma Laaksonen. Image and video captioning with augmented neural architectures. *IEEE MultiMedia*, 25(2):34–46, April 2018.

[4] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. *CoRR*, abs/1612.00563, 2016.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[7] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *CoRR*, abs/1506.03099, 2015.

[8] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[9] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. *CoRR*, abs/1705.07750, 2017.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[11] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. *CoRR*, abs/1604.06573, 2016.

[12] 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 *European Conference on Computer Vision (ECCV)*, 2014.

[13] Yuncheng Li, Yale Song, Liangliang Cao, Joel Tetreault, Larry Goldberg, Alejandro Jaimes, and Jiebo Luo. TGIF: A new dataset and benchmark on animated gif description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4641–4650, 2016.

[14] Xin Wang, Jiawei Wu, Junkun Chen, Lei Li, Yuan-Fang Wang, and William Yang Wang. Vatex: A large-scale, high-quality multilingual dataset for video-and-language research, 2019.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[16] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[17] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. CIDEr: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4566–4575, 2015.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[19] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.