Supplementary Material for Submission "End-to-End Dense Video Captioning with Masked Transformer"

Luowei Zhou*
University of Michigan
luozhou@umich.edu

Yingbo Zhou*
Salesforce Research
yingbo.zhou@salesforce.com

Jason J. Corso
University of Michigan
jjcorso@eecs.umich.edu

Richard Socher
Salesforce Research
richard@socher.org

Caiming Xiong†
Salesforce Research
cxiong@salesforce.com

A. Implementation Details

The sizes of the temporal convolution kernels in the proposal module are 1 2, 3, 4, 5, 7, 9, 11, 15, 21, 29, 41, 57, 71, 111, 161, 211 and 251. We set the hyper-parameters for End-to-end Masked Transformer as follows. The dropout ratio for Transformer is set to 0.2 and that for visual input embedding is set to 0.1. We set the loss coefficients $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ to $10, 1, 1, 0.25$. For training, we use stochastic gradient descent (SGD) with Nesterov momentum, the learning rate is set between 0.01 and 0.1 depending on the convergence, and the momentum is set at 0.95. We decay the learning rate by half on plateau. We also clip the gradient [2] to have global ℓ_2 norm of 1. For inference, we first pick event proposals with prediction score higher than a pre-defined threshold (0.7). We remove proposals that have high overlap (*i.e.* ≥ 0.9) with each other. For each video, we have at least 50, and at most 500 event proposals. The descriptions are then generated for each of the proposal, and we use greedy decoding for text generation with at most 20 words. We implement the model in PyTorch and train it using 8 Tesla K80 GPUs with synchronous SGD. The model typically takes a day to converge.

The implementation for proposal-only and captioning-only model is slightly different. We apply Adam for training rather than SGD and set the learning rate to 0.0001. When training the captioning-only model, we apply scheduled sampling [1]. We set the sampling ratio to 0.05 at the beginning of training, and increase it by 0.05 every 5 epoch until it reaches 0.25. Note that applying scheduled sampling to End-to-end Masked Transformer yield no improvements and hence disabled. In the proposal-only model, we report the results on a single-layer Transformer with the model size and hidden size to be 512 and 128. The temporal conv. stride factor s is set to 10.

*Equal contribution
†Corresponding author

Table A1. Additional ablation experiments on ActivityNet.

Method	B@3	B@4	M
SelfAttn + LSTM TempoAttn	2.91	1.35	7.88
BiLSTM + SelfAttn	4.06	1.92	9.05
Our Method (1-layer)	4.49	2.10	9.27

Table A2. Evaluating only short events from ActivityNet.

	GT Proposals		**Learned Proposals**	
Method	B@4	M	B@4	M
Bi-LSTM+TempoAttn	0.74	5.29	0.23	4.43
Our Method	0.87	5.82	0.68	5.06

B. Additional Results

To see the effectiveness of self-attention, we performed additional ablation studies, where we apply self-attention module at the encoder or decoder of the LSTM-based baseline. From the result it is clear that self-attention have significant impact on the performance of the model (see Tab. A1), especially as in the language decoder.

Note that the performance of captioning models over ground-truth segments vary little from number of layers. We choose to use 2-layer transformer for the rest of the experiments because 1) the 4 and 6 layer models are more computational expensive; 2) the learning is more complicated when the learned proposals are approximate, and a 2-layer model give us more flexibility for handling this case (see Tab. A1 for results on the 1-layer model).

Self-attention facilitates the learning of long-range dependencies, which should not hurt the performance on modeling relative short-range dependencies. To validate this we tested our model on shorter activities, where the activities

are at most 15 seconds long. The result is shown in Tab. A2.

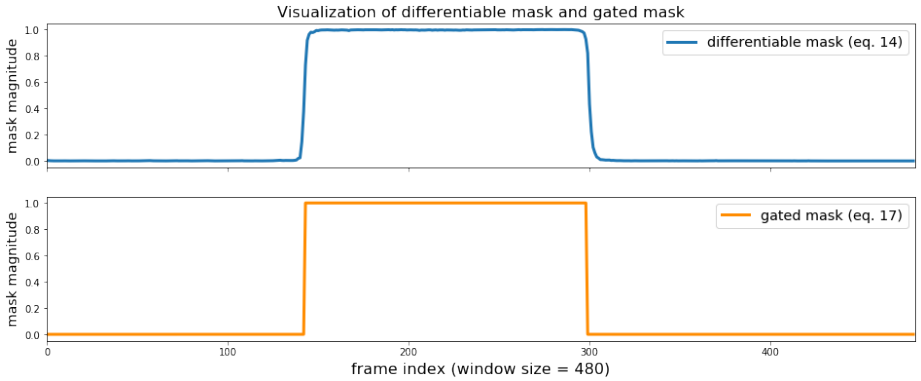
C. Additional Qualitative Results

We visualize the learned masks in Fig. A1. The first two correspond to the case where the proposal prediction is confident, i.e., proposal scores are high (> 0.9) and the last two correspond to the case where the prediction is less confident, i.e., proposal scores are low (< 0.7). We visualize the cross module attention in Fig. A3. For convenience, we randomly choose one of the attention matrices from the multi-head attention. Also, we notice that attention weights from higher-level self-attention layer is tend to be flatter than these from the lower-level layer. Qualitative results for YouCookII are shown in Fig. A2. The visual recognition is challenging result from the small and ambiguous objects (e.g., black pepper, lamb).

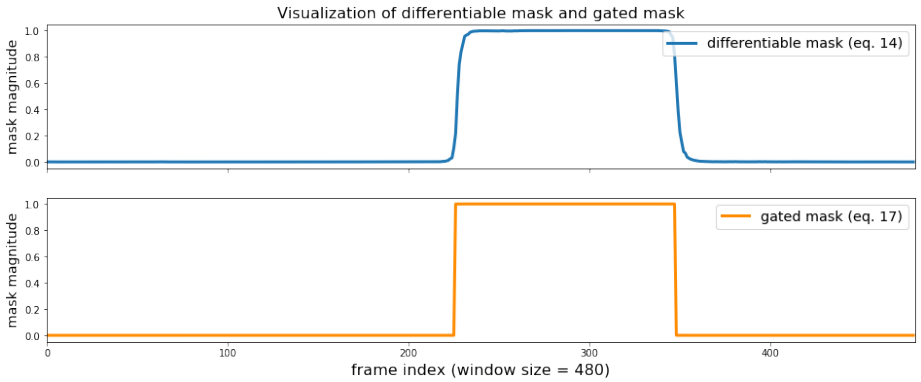
References

[1] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, pages 1171–1179, 2015. 1

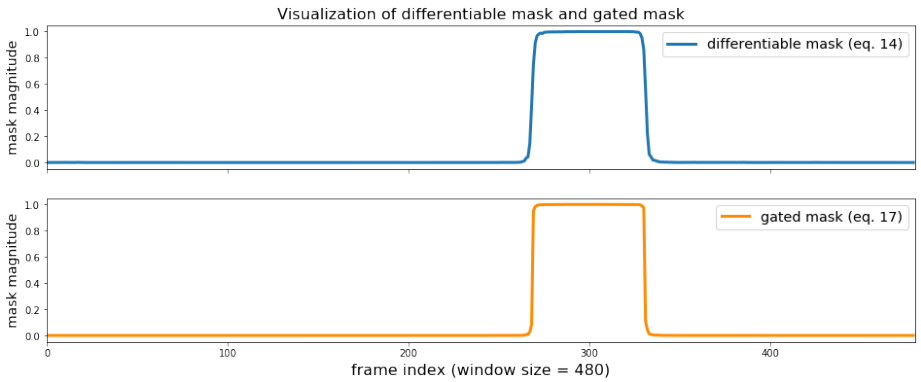
[2] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *ICML*, pages 1310–1318, 2013. 1



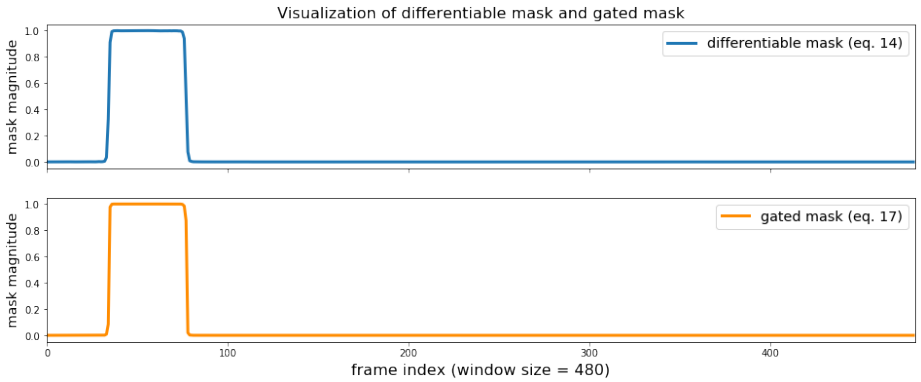
(a) High proposal score.



(b) High proposal score

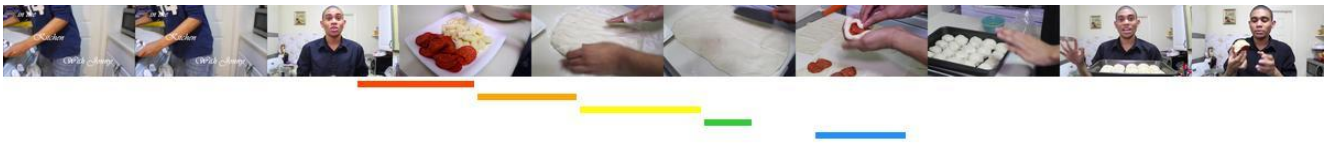


(b) Low proposal score



(b) Low proposal score

Figure A1. Visualization of differentiable masks and final masks under hight (a and b) and low proposal score (c and d). Videos from ActivityNet Captions validation set.



Ground-truth
Event 0: stretch the dough
Event 1: cut the dough into squares
Event 2: lay pepperoni and cheese on the dough and roll into a ball
Event 3: put the rolls in a pan
Event 4: brush each pizza bite with some melted butter and sprinkle some italian seasoning on top

Masked Trans. (ours)
Event 0: knead the dough
Event 1: cut the dough into thin slices roll
Event 2: cut the meat into thin slices roll
Event 3: dip the fish in the batter and place on the
Event 4: dip the fish in the batter and coat the batter the batter the batter

Bi-LSTM+TempoAttn
Event 0: cut the roll into pieces the edges and roll the dough
Event 1: cut the salmon into thin slices the sheet
Event 2: cut the salmon into thin slices the sheet
Event 3: place the filling on the bread the bread
Event 4: place the chicken on the pan the grill and serve

Ground-truth
Event 0: pour some oil into a hot pan
Event 1: add chopped onions and carrots to the pan
Event 2: add salt to the pan and mix
Event 3: add butter and garlic to the pan and mix
Event 4: add lamb to the pan and break it up
Event 5: add salt black pepper and italian seasoning to the meat
...

Masked Trans. (ours)
Event 0: heat a pan with oil a pan add the pork
Event 1: add the onions and garlic to the pan stir
Event 2: add the onions and carrots to the pan stir
Event 3: add the vegetables to the pan and stir
Event 4: add the pork to the pan stir
Event 5: add the vegetables to the pan and stir
...

Bi-LSTM+TempoAttn
Event 0: add oil to a pan heat <unk> <unk> <unk> <unk> <unk>
Event 1: add the chicken to the pan heat
Event 2: add the chicken to the pan heat
Event 3: add the chicken to the pan heat
Event 4: add the chicken to the pan heat
Event 5: add the chicken to the pan and stir heat
...

Figure A2. Qualitative results on YouCookII videos. We only showed result for the first 6 events in the second example.

Figure A3. Visualization of weights from the cross-module attention layer. X axis represents the generated words at each time step. Y axis indicates the sampled frames.