1 Supplementary

1.1 Proposition 1 for Eqn. 8 in the paper

Proposition 1: Suppose that we have n matrices, $I_1, I_2, ..., I_n$, and n vectors, $w_1, w_2, ..., w_n$. The space of I_l is $\mathbb{R}^{d\times t_l}$ and the space of w_l is $\mathbb{R}^{1\times t_l}$. Then

$$(\otimes\{I_1, ..., I_n\}) \odot (\otimes\{w_1, ..., w_n\}) = (\otimes\{I_1 \odot w_1, ..., I_n \odot w_n\}) \quad (1)$$

Proof: We use C_l to denote the left side of the equation and C_r to denote the right side of the equation. We utilize the element-wise comparison in two tensors. Following Eqns. 3 and 4 in the paper, the $(r_1, ..., r_n)$-th entry of C_l is expressed as

$$(C_l)_{r_1,...,r_n} = \mathbf{1}_d((I_1)_{r_1} \odot ... \odot (I_n)_{r_n}) \cdot (w_1)_{r_1} \cdot ... \cdot (w_n)_{r_n} \quad (2)$$

where $(I_l)_{r_l}$ is a vector which denotes r_l-th column of the I_l, $(w_l)_{r_l}$ is the r_l-th value of the vector. Since $(w_l)_{r_l}$ is a single element, we directly multiply it with the corresponding vector $(I_l)_{r_l}$.

$$(C_l)_{r_1,...,r_n} = \mathbf{1}_d(((I_1)_{r_1} \cdot (w_1)_{r_1}) \odot ... \odot ((I_n)_{r_n} \cdot (w_n)_{r_n})) \quad (3)$$

$$= (C_r)_{r_1,...,r_n} \quad (4)$$

The proposition is proven and is used to convert Eqn. 7 to Eqn. 8 in the paper.

1.2 Proposition 2 for Eqn. 9 in main paper

Proposition 2: Suppose that we have n matrices, $I_1, I_2, ..., I_n$. The space of I_l is $\mathbb{R}^{d\times t_l}$. Then

$$\sum(\otimes\{I_1, ..., I_n\}) = \mathbf{1}_d((I_1)\mathbf{1}_{t_1} \odot ... \odot (I_n)\mathbf{1}_{t_n}) \quad (5)$$

here we use vectors $\mathbf{1}_d$ and $\mathbf{1}_{t_l}$ which consist of 1 to represent the summation operation for matrix I_l in d dimension and t_l dimensions, respectively.

Proof: We use v_l to denote the left side of the equation and v_r to denote the right side of the equation. We can express v_l as

$$v_l = \sum_{r_1=1}^{t_1} ... \sum_{r_n=1}^{t_n} C_{r_1,r_2,...,r_n} \quad (6)$$

$$C = \otimes\{I_1, ..., I_n\} \quad (7)$$

Following Eqns. 3 and 4 in the paper, we can express $C_{r_1,r_2,...,r_n}$ as

$$C_{r_1,r_2,...,r_n} = \mathbf{1}_d((I_1)_{r_1} \odot ... \odot (I_n)_{r_n}) \quad (8)$$

We apply Eqn. 8 to Eqn. 6,

$$v_l = \sum_{r_1=1}^{t_1} ... \sum_{r_n=1}^{t_n} \mathbf{1}_d((I_1)_{r_1} \odot ... \odot (I_n)_{r_n}) \quad (9)$$

$$= \mathbf{1}_d(\sum_{r_1=1}^{t_1} ... \sum_{r_n=1}^{t_n} (I_1)_{r_1} \odot ... \odot (I_n)_{r_n}) \quad (10)$$

$$= \mathbf{1}_d((I_1)\mathbf{1}_{t_1} \odot ... \odot (I_n)\mathbf{1}_{t_n}) = v_r \quad (11)$$

The proposition is proven and is used to convert Eqn. 8 to Eqn. 9 in the paper.

1.3 The calculation process of the needed storage for different structures

As supposed in the paper, we take the calculation process of three modalities as an example.

HOCA The output space of each "tanh" module is $\mathbb{R}^{50\times 80\times 512}$. The space of the correlation tensor C_3 is $\mathbb{R}^{50\times 80\times 80\times 80}$, and the calculation process of C_3 generates other outputs which have much larger size. Since the tensor (matrix) multiplication operation in existing deep-learning framework (tensorflow) can be only used to integrate two tensors, for example, we can not directly integrate the three tensors with the same space $\mathbb{R}^{50\times 80\times 512}$ to the correlation tensor C_3. Thus, we implement it through a 80-step loop. In each step, we use one time step (with space $\mathbb{R}^{50\times 1\times 512}$) of the first tensor to multiply each column of the second tensor, the output space is $\mathbb{R}^{50\times 80\times 512}$. Then, we integrate the second and the third tensors, the output space is $\mathbb{R}^{50\times 80\times 80}$. After the loop, the total size of all the steps is $\mathbb{R}^{50\times 80\times 80\times (80+512)}$. The space of the weight W_2 for each modality is $\mathbb{R}^{80\times 80}$. The input space of each "softmax" module is $\mathbb{R}^{50\times 80}$. The final storage is mainly determined by the size of correlation tensor and the outputs generated by the calculation process, which is about 722 MB ($50 \times 80 \times 80 \times 592$) with $float32$ type.

Low-Rank HOCA The output space of each "tanh" is $\mathbb{R}^{50\times 80\times 512}$. The output space of each "linear(t)" module is $\mathbb{R}^{50\times 1\times 512}$. And the space of B_l is also $\mathbb{R}^{50\times 1\times 512}$. The output space of each "mul" module is $\mathbb{R}^{50\times 80\times 512}$. Since the number of modalities is 3, the needed storage is about 47 MB($50 \times 486 \times 512$).

1.4 Preprocessing

For MSVD and MSR-VTT, there is much redundancy between video frames. We sample video data for extracting image (2D CNN) features. Each video is sampled to 80 frames. For extracting 3D CNN features, we divide the raw video data into video chunks centered on 80 sampled frames at the first step. Each video chunk includes 64 frames. For extracting audio features, we obtain the audio file from the raw video data with FFmpeg. Note that MSVD dataset has no audio information, we need to download the original video data from YouTube. For both datasets, we use a pre-trained Inception-ResNet-v2 [Szegedy *et al.*, 2017] model to extract image features from the sampled frames and we keep the activations from the penultimate layer. In addition, we use a pre-trained inflated 3D network (I3D) [Carreira and Zisserman, 2017] to extract motion features from video chunks. We employ the activations from the last convolutional layer and implement a mean-pooling in the temporal dimension. We use the pre-trained VGGish [Hershey *et al.*, 2017] model to extract audio features, which is different from the traditional method.

For SumMe and TVSum, we extract image features every 15 frames similar to that of [Zhang *et al.*, 2016] with GoogleNet [Szegedy *et al.*, 2015]. We downsample the video data twice for extracting the motion features, we set the number of chunks to 100. In terms of audio features, we apply the same method as mentioned above.

1.5 The setting of rank

In this paper, we utilize three modalities in video data. The order of the decomposed tensor is relatively small. So we set the rank h to a small value. We try several values and find that rank 1 is enough to obtain competitive results with a high cost performance.

1.6 Experimental details

Video Captioning The hidden size is 512 for all LSTMs. The attention layer size for image, motion, audio attention is also 512. We add layer normalization in the LSTM decoder. The dropout rate for the input and output of LSTM decoder is 0.5. In the training stage, we use Adam algorithm to optimize the loss function; the learning rate is set to 0.0001. In the testing stage, we use beam-search method with beam-width 5. We use a pre-trained word2vec embedding to initialize the word vector matrix. Each word is represented as a 300-dimension vector. Those words which are not in the word2vec matrix are initialized randomly.

Video Question Answering The hidden size is 256 for all LSTMs. The attention layer size for image, motion, audio attention is also 256. The dropout rate for the input and output of LSTM decoder is 0.5. In the training stage, we use Adam algorithm to optimize the loss function; the learning rate is set to 0.001. We use a pre-trained word2vec embedding to initialize the word vector matrix. Each word is represented as a 300-dimension vector. Those words which are not in the word2vec matrix are initialized randomly.

Video Summarization The hidden size of Bi-LSTM is 512. We use the processed image features as query. The attention layer size for image, motion, audio attention is also 512. The dropout rate for the input and output of the Bi-LSTM is 0.5. In the training stage, we use Adam algorithm to optimize the loss function; the learning rate is set to 0.0001.

References

[Carreira and Zisserman, 2017] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, pages 4724–4733, 2017.

[Hershey *et al.*, 2017] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. Cnn architectures for large-scale audio classification. In *ICASSP*, pages 131–135, 2017.

[Szegedy *et al.*, 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. In *CVPR*, 2015.

[Szegedy *et al.*, 2017] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

[Zhang *et al.*, 2016] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. Video summarization with long short-term memory. In *ECCV*, pages 766–782, 2016.