AdaFuse: Adaptive Temporal Fusion Network for Efficient Action Recognition

Seungmin Lee (profile2697@gmail.com; 2020-20866),
Dept. of Electrical and Computer Engineering, Seoul National University

1. Introduction and Motivation

In video action recognition, both temporal modeling and employing temporal redundancy are crucial. For the purposes, the authors propose AdaFuse, named from an adaptive temporal fusion network, that dynamically selects channels of current and past feature maps and fuses them.

2. Method

2.1. 2D-CNN for Action Recognition

For action recognition, many methods first extract feature maps from each frame using 2D-CNN ($\mathcal{F}$) and colligate the feature maps using a consensus operation ($\mathcal{G}$) used for final prediction. So, the final prediction can be written as follows:

$$P(X_1, ..., X_T) = \mathcal{G}(\mathcal{F}(X_1), ..., \mathcal{F}(X_T))$$

where T is the number of sampled frames, and $\{X_1, ..., X_T\}$ are the sampled frames. The consensus operation can be an average operation or LSTM.

2.2. Adaptive Temporal Fusion

Assume that $x_t \in \mathbb{R}^{c \times h \times w}$ is a feature map, and $y_t = \text{conv}(x_t) \in \mathbb{R}^{c' \times h' \times w'}$ is a feature map of the next layer. Let $v_t = GAP(x_t) \in \mathbb{R}^c$ is a global average pooled feature. AdaFuse adopts an agent g that takes v_t and v_{t-1} and outputs $p_t \in \{0, 1, 2\}^c$ where p_t is used to fuse y_t and y_{t-1}:

$$\hat{y}_t^i = \mathbf{1}[p_t^i = 0] \cdot y_t^i + \mathbf{1}[p_t^i = 1] \cdot y_{t-1}^i$$

where $\mathbf{1}$ is the indicator function. The equation means if p_t^i is 0, then we use the i-th channel of y_t; else if p_t^i is 1, we use the i-th channel of y_{t-1}; finally, if p_t^i is 2, we mask the i-th channel as zeros.

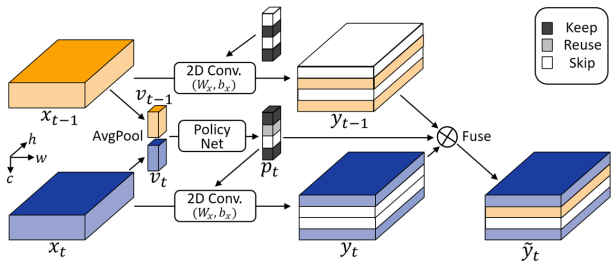

Figure 1. (a) The proposed encoder-decoder transformer architecture, (b) The encoder's attention maps: the encoder separate each instance even though those are occluded.

2.3. Loss functions

The loss function is the sum of the cross-entropy loss and the FLOPS of AdaFuse layers. The agent learns to minimize the FLOPS while maintaining the accuracy.

3. Results

The authors evaluate the effectiveness of the proposed method on various datasets such as Something-Something V1, Jester, and Mini-Kinetics. AdaFuse consistently improves performance while maintaining the FLOPS of the network, and in some experiments, the method shows better performance while reducing the FLOPS drastically.

4. Personal Note

Even though integrating with Reinforce learning seems a little bit complicated, the method seems simple, neat, and efficient.

References