DejaVid: Encoder-Agnostic Learned Temporal Matching for Video Classification

Supplementary Material

Additional Implementation Details. VideoMAE V2-g takes 16 frames of shape 224×224 as input and outputs a length-1408 representation and then a length-N_c logit vector, where N_c is the number of classes in the dataset. Thus the size of the encoder output N_f is $1408 + N_c$. For DejaVid, we apply a temporal sliding window across the input video, take the center crop, and resize to 224×224 to feed into the encoder. This gives us a TSE of shape $T \times N_f$ for some T. Then, for each action class, we randomly sample 50 TSEs, reshape each of them to $T_c \times N_f$ with linear interpolation, and then run 100 iterations of the DBA algorithm [23] to produce the centroid.

We now describe our choice of the temporal sliding window widths and strides. For Kinetics-400 and HMDB51, given a video, VideoMAE V2 temporally segments the video into 5 clips of the same length and takes 3 crops at the left, center, and right to produce $5 \times 3 = 15$ logit vectors, from which they then take the mean to produce the class prediction. Note that the temporal treatment is equivalent to a sliding window of width $\frac{|vid|}{5}$ and stride $\frac{|vid|}{5}$, where $|vid|$ is the video length. On the other hand, we only use the center crop, but deploy a sliding window of width $\frac{|vid|}{5}$ and stride $\frac{|vid|}{40}$, so we produce $(\frac{40}{5} \cdot (5-1)+1) \times 1 = 33$ logit vectors per video, with the resulting TSE having dimension $33 \times N_f$.

For Something-Something V2, unlike the other two datasets, VideoMAE V2 does not temporally segment but instead performs a strided slice on the frames with a step of 2. This means that the encoder is finetuned to an input window width of $|vid|$, which complicates our sliding window application. The vast majority of Kinetics-400 videos are of length ~ 300 frames, but videos in Something-Something V2 vary more in frame count, ranging from the teens to over a hundred, which means its encoder window width varies more too. In order to provide DejaVid with both constant-width and variable-width information, we apply four sliding windows with width $\{16, 32, 64, |vid|\}$ and stride 1 in parallel, and thus obtain for each video a TSE of dimension $|vid| \times (4 \cdot (1408 + N_c))$. The average video length in Something-Something V2 is ~ 40 frames, so on average, we produce $\frac{4 \cdot 40}{33} = 4.8$ times more embeddings per video than for Kinetics-400 and HMDB51.

Applying DejaVid to non-SOTA video encoders.

Our algorithm is model-agnostic and can be applied to other encoders. To demonstrate this, we apply DejaVid to other encoders that are not SOTA. The results, as presented, show a significant improvement. Here, we report as baselines numbers that the code on Hugging Face achieves on our machine rather than the numbers from the original papers, which are somewhat higher.

Model	Dataset (Clips$\times$ Crops)	Accuracy without DejaVid	Accuracy with DejaVid
facebook/timesformer-base-finetuned-ssv2 @ huggingface [2, 6]	SSv2[16] ($4\|vid\| \times 1$)	55.5%	57.6%
google/vivit-b-16x2-kinetics400 @ huggingface [3, 25]	K400[18] (33×1)	62.4%	66.6%

Formulas for loss gradients $\frac{\partial L_w}{\partial U}$ and $\frac{\partial L_w}{\partial C}$.

This section supplements Sec. 3.2 by proving the differentiability of the Algorithm 2 neural network of DejaVid, namely the detailed formulas for $\frac{\partial L_w}{\partial U}$ and $\frac{\partial L_w}{\partial C}$, which are omitted at the end of Sec. 3.2.

Recall from the end of Sec. 3.1 that we calculate the time-weighted distance from a training or validation TSE m to the centroid TSE C_i of each class i and then feed the class-wise distances to soft-min for class prediction. We first observe that before the soft-min, the distance calculations for each class are independent of each other; they do not share any elements of C or U, nor do they have any inter-class connections. So we can individually calculate $\frac{\partial L_w}{\partial U[c]}$ and $\frac{\partial L_w}{\partial C[c]}$ for each class c, then stack them together for the final $\frac{\partial L_w}{\partial U}$ and $\frac{\partial L_w}{\partial C}$.

Note that $\frac{\partial L_w}{\partial U[c]}$ and $\frac{\partial L_w}{\partial C[c]}$ are the combination of three components:

$$\frac{\partial L_w}{\partial U[c]} = \frac{\partial L_w}{\partial D_w[c]} \sum_l \frac{\partial D_w[c]}{\partial SC[c,l]} \frac{\partial SC[c,l]}{\partial U[c]}$$

$$\frac{\partial L_w}{\partial C[c]} = \frac{\partial L_w}{\partial D_w[c]} \sum_l \frac{\partial D_w[c]}{\partial SC[c,l]} \frac{\partial SC[c,l]}{\partial C[c]}$$

where l is the index of the diagonal, $D_w \in \mathbb{R}^{N_c}$ the distance from m to the centroid of each class, $SC[c,l]$ is the l-th skip-connection for class c as in Algorithm 2, and $C \in \mathbb{R}^{N_c \times T_c \times N_f}$, $U \in \mathbb{R}_{>0}^{N_c \times T_c \times N_f}$ are as defined in Sec. 3.1. The following tackles each of the three components respectively.

For $\frac{\partial L_w}{\partial D_w[c]}$, we use the standard cross-entropy and soft-min, so the derivative is well-known to be:

$$\frac{\partial L_w}{\partial D_w[c]} = \mathbf{y}[c] - p[c]$$

where $\mathbf{y}$ is the one-hot ground truth vector and $p[c]$ is the predicted probability of class c.

For $\frac{\partial D_w[c]}{\partial SC[c,l]}$, the standard trick for calculating loss gradients of a min-pooling layer is to define an indicator matrix. Note that the l-th min-pooling layer for class c has length $\|SC[c,l]\|$. Let $R[c,l]$ of shape $\|SC[c,l]\| \times \|SC[c,l-1]\|$ be the indicator matrix of the i-th min-pooling for class c. We have:

$$R[c,l,a,b] = \begin{cases} 1 & \text{if } b \in \{a-1, a\} \text{ and} \\ & \text{the } a\text{-th output of the min-pooling} \\ & = \text{ the } b\text{-th input of the min-pooling} \\ 0 & \text{otherwise} \end{cases}$$

And since the min-pooling layers are chained, we have:

$$\frac{\partial D_w[c]}{\partial SC[c,l]} = \prod_{i=n+m-2}^{l+1} R[c,i]$$

Notably, $\|SC[c,n+m-2]\| = 1$, so the matrix product results in a shape of $\|SC[c,n+m-2]\| \times \|SC[c,l]\| = 1 \times \|SC[c,l]\|$.

Finally, for $\frac{\partial SC[c,l]}{\partial U[c]}$, first notice that for any given i, $U[c,i]$ can only contribute to $SC[c,l]$ at the entry with $dist_w(U[c,i], C[c,i], m[l-i])$. Denoting $\frac{\partial SC[c,l]}{\partial U[c]}$ as $dU_l[c] \in \mathbb{R}^{\|SC[c,l]\| \times T_c \times N_f}$, we thus have:

$$dU_l[c,i,j,f] = \begin{cases} |C[c,i+\text{start},f] - m[j,f]| \\ \text{if } i + \text{start} + j = l \\ 0 \text{ otherwise} \end{cases}$$

where $\text{start} = \max(0, l - \dim_0(m) + 1)$ is the offset for the 0-th element of $SC[c,l]$, as in Line 6 of Algorithm 2.

Similarly for $\frac{\partial SC[c,l]}{\partial C[c]}$, first notice that for any given i, $C[c,i]$ can only contribute to $SC[c,l]$ at the entry with $dist_w(U[c,i], C[c,i], m[l-i])$. Denoting $\frac{\partial SC[c,l]}{\partial C[c]}$ as $dC_l[c] \in \mathbb{R}^{\|SC[c,l]\| \times T_c \times N_f}$, we thus have:

$$dC_l[c,i,j,f] = \begin{cases} U[c,i+\text{start},f] \cdot \text{sign}(C[c,i+\text{start},f] - m[j,f]) \\ \text{if } i + \text{start} + j = l \\ 0 \text{ otherwise} \end{cases}$$

which concludes the formulas for loss gradients $\frac{\partial L_w}{\partial U}$ and $\frac{\partial L_w}{\partial C}$. This demonstrates the differentiability of the Algorithm 2 neural network of DejaVid, which enables optimization via backpropagation.