A. Additional Experiments

A.1. Actionformer Downstream Head

Additional results using Actionformer [11], a recently introduced state-of-the-art TAL head, is presented in Table 3. Results using GTAD and BMN are also shown in parallel with the Actionformer results in Table 3 as exactly the same feature is utilized. Consistent performance improvement is observed with the adoption of SoLa strategy regardless of the downstream heads.

A.2. Additional Ablation Study

Inspired by Adapters [7, 8], the SoLa module has two parallel passes: a skip connection and a 1D CNN pass with a learnable scalar gating parameter initialized as 0. This structure makes the SoLa module the identity mapping at its first stage of training. As the training precedes, the gating parameter will deviate from 0 and the SoLa module will start to slowly enhance the input feature sequence. Table 1 demonstrates the ablation study of the design choice. We can see that while the SoLa module with a direct pass works reasonably well, the Adaptor style SoLa module brings additional performance gain thanks to its conservative enhancement of the feature sequence.

Additional ablation studies on the remaining design choices of the SoLa strategy are presented in Table 5 and 6. It is worth noting that, results on varying $K$ values suggest that appropriate $\lambda$ assignment is an essential part in terms of achieving temporally sensitive snippet feature sequences, which justifies our $\lambda$ function design. Moreover, result in Table 6 alludes that asymmetric projector significantly stabilizes the training procedure.

A.3. Error bars of the main results

To validate the robustness of our method, we report the main results’ error bars with standard deviations in Figure 1. 12 and 5 independent downstream head training with varying random seeds was done in Activitynet1.3 [1] and HACS [12] experiments respectively. The result shows that our method significantly outperforms the baselines with a strong statistical significance.

B. Connection to the Contrastive Learning

In this section, we provide a connection between the well-known contrastive learning loss function and our Similarity Matching loss.

Due to the unlabeled target dataset assumption, the training of the SoLa module must be done in a self-supervised manner. Since recent studies [4, 5] have shown remarkable success of contrastive learning in the self-supervised representation learning domain, it is natural to start with the standard NT_XENT loss [2]. For the positive sample representation pair $(z_i, z_j)$ and the similarity measure $\text{sim}(\cdot, \cdot)$, $\text{alignment}$ is defined as follows:

$$L^\text{NT_XENT}_{i,j} = -\frac{\text{sim}(z_i, z_j)}{L^\text{alignment}} + \log \left( \exp\left(\text{sim}(z_i, z_j)\right) + \sum_{k \in I \setminus \{i,j\}} \exp\left(\text{sim}(z_i, z_k)\right) \right),$$

where $I$ is the index set containing all the sample index. However, unlike the common contrastive learning setting, we cannot exploit data augmentation to generate positive samples from the given data since we do not have any means to manipulate the feature-level sample without hurting its essence. This makes the standard positive/negative pair concept inapplicable and thereby requires devising a different approach for training the SoLa module.

In this regard, we start with the concept of the temporal structure that videos naturally convey: “adjacent frames should be similar, while remote frames should remain distinct”. In fact, the instantiation of the above temporal structure can substitute the standard positive/negative pair concept. But unlike the discrete positive/negative distinction in Equation (1), the temporal structure offers a softened version of the distinction in that “the similarity decreases as the distance between the two features increases”. Therefore, we introduce a softened indicator function $\lambda(\cdot, \cdot) \rightarrow [0, 1]$, whose output represents the input pair’s $\text{positiveness}$ by a continuous real value. For instance, a high $\lambda(z_i, z_j)$ value...
indicates \((z_i, z_j)\) to be treated more like a semantically close pair, whereas a low \(\lambda(z_i, z_j)\) value represents larger semantic distance.

To incorporate \(\lambda(\cdot, \cdot)\) into Equation (1), we pay careful attention to the following two points: (i) \(L_{alignment}\) and \(L_{distribution}\) should be more influential with high \(\lambda(z_i, z_j)\) and low \(\lambda(z_i, z_k)\) respectively to match our soft \(\lambda\) positiveness concept, and (ii) Equation (1) must be recovered from Equation (2) and (3) by setting \(\lambda(\cdot, \cdot)\) as a discrete indicator function that only returns 1 if the given pair is a positive pair and 0 otherwise. From these points, we placed the coefficients \(\lambda(z_i, z_j)\) and \(1 - \lambda(z_i, z_k)\) in front of the positive/negative pairs in Equation (1). Two possible softened NT_XENT losses can be derived from the coefficients’ positions:

\[
\mathcal{L}_{in}^{\text{soften}}(z_i, z_j) = -\lambda(z_i, z_j) \text{sim}(z_i, z_j) \\
+ \log \left( \exp(\lambda(z_i, z_j) \text{sim}(z_i, z_j)) \right) \\
+ \sum_{k \in \{i, j\} \setminus \{z_i, z_j\}} \exp((1 - \lambda(z_i, z_k)) \text{sim}(z_i, z_k)) \\
L_{distribution} \tag{2}
\]

\[
\mathcal{L}_{out}^{\text{soften}}(z_i, z_j) = -\lambda(z_i, z_j) \text{sim}(z_i, z_j) \\
+ \log \left( \lambda(z_i, z_j) \exp(\text{sim}(z_i, z_j)) \right) \\
+ \sum_{k \in \{i, j\} \setminus \{z_i, z_j\}} (1 - \lambda(z_i, z_k)) \exp(\text{sim}(z_i, z_k)) \\
L_{distribution} \tag{3}
\]

We observed that \(\mathcal{L}_{out}^{\text{soften}}\) performs well, whereas \(\mathcal{L}_{in}^{\text{soften}}\) degrades the downstream task performance significantly (Table 2). Note that the gradient with respect to \(\text{sim}(z_i, z_k)\) in \(\mathcal{L}_{in}^{\text{soften}}\)'s \(L_{distribution}\) term is scaled by \(1 - \lambda\). As \(1 - \lambda \in [0, 1]\), we attribute the deterioration of the performance to the broken gradient balance between \(L_{alignment}\) and \(L_{distribution}\) terms in \(\mathcal{L}_{in}^{\text{soften}}\).

While we can directly work with the Equation (3), another interesting observation in Equation (3) is that it results in a non-zero term even if only one pair \((z_i, z_j)\) is given for its computation, while a trivial cancellation of \(L_{alignment}\) and \(L_{distribution}\) occurs in Equation (1) and (2). As \(I\{i, j\} = \emptyset\) in the one pair case, the single pair \(\mathcal{L}_{out}^{\text{soften}}\) can be represented as follows:

\[
\mathcal{L}_{out}^{\text{soften}}(z_i, z_j) \\
= -\lambda(z_i, z_j) \text{sim}(z_i, z_j) + \log \left( \lambda(z_i, z_j) \exp(\text{sim}(z_i, z_j)) \right) \\
= -\lambda(z_i, z_j) \text{sim}(z_i, z_j) + \log \lambda(z_i, z_j) + \text{sim}(z_i, z_j) \\
= (1 - \lambda(z_i, z_j)) \text{sim}(z_i, z_j) + \text{const} \\
= -(1 - \lambda(z_i, z_j)) \log(1 - p) + \text{const}, \tag{4}
\]

where \(p = 1 - \exp(-\text{sim}(z_i, z_j))\). Note that \(p\) monotonically increases as \(\text{sim}(z_i, z_j)\) increases for all \(\text{sim}(z_i, z_j)\), allowing the interpretation of \(p\) as \(\text{sim}(z_i, z_j)\) without the loss of generality. However, the single pair \(\mathcal{L}_{out}^{\text{soften}}\) goes to 0 when the \(\lambda\) goes to 1, regardless of the \(p\) value. To resolve this issue, we added symmetric term \(-\lambda(z_i, z_j) \log p\) to the single pair \(\mathcal{L}_{out}^{\text{soften}}\). Here, if we assume the strictly positive similarity measure (e.g., \(\text{sim}(z_1, z_2) = \frac{1}{\|z_1 - z_2\|^2}, z_1 \neq z_2\)), \(p\) is bounded to \((0, 1)\). Denoting the bounded \(p\) as \(\hat{p}\), similarity matching loss \(\mathcal{L}_{SM}^{\text{soften}}\) is formulated as

\[
\mathcal{L}_{SM}^{\text{soften}}(z_i, z_j) = -\lambda(z_i, z_j) \log \hat{p} - (1 - \lambda(z_i, z_j)) \log(1 - \hat{p}), \tag{5}
\]

where \(\hat{p} \in (0, 1)\) is the network’s prediction of the given pair’s similarity. Intuitively, our loss term simply minimizes the Binary Cross Entropy (BCE) between the network prediction \(\hat{p}\) and the given label \(\lambda(z_i, z_j)\), as its name “similarity matching” suggests. We empirically found out that our sim-
Method & Temporal Action Localization (GTAD) & mAP@0.5 & @0.75 & @0.95 & Avg & gain \\ Baseline & 49.78 & 34.46 & 7.96 & 33.84 & - &  \\ SoLa(ours) & 51.17 & 35.70 & 8.31 & 34.99 & +1.15 &  \\ & Temporal Action Localization (ActionFormer) & mAP@0.5 & @0.75 & @0.95 & Avg & gain \\ Baseline & 50.22 & 33.81 & 7.75 & 33.19 & - &  \\ SoLa(ours) & 51.64 & 34.81 & 8.02 & 34.21 & +1.02 &  \\ & Temporal Action Proposal (BMN) & AR@1 & @10 & @100 & AUG & gain \\ Baseline & 33.59 & 56.79 & 75.05 & 67.16 & - &  \\ SoLa(ours) & 34.25 & 57.75 & 75.86 & 68.07 & +0.91 &  

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
<th>Setting</th>
<th>Temporal Action Localization (GTAD [10])</th>
<th>mAP@0.5</th>
<th>@0.75</th>
<th>@0.95</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epoch</td>
<td>500</td>
<td>Baseline</td>
<td>49.78</td>
<td>34.46</td>
<td>7.96</td>
<td>33.84</td>
<td></td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.0001</td>
<td>4 8</td>
<td>50.16</td>
<td>34.92</td>
<td>8.17</td>
<td>34.35</td>
<td></td>
</tr>
<tr>
<td>Hidden Units</td>
<td>1024</td>
<td>8 8</td>
<td>50.59</td>
<td>35.29</td>
<td>7.94</td>
<td>34.53</td>
<td></td>
</tr>
<tr>
<td>Conv layers</td>
<td>3</td>
<td>16</td>
<td>50.80</td>
<td>35.33</td>
<td>8.56</td>
<td>34.80</td>
<td></td>
</tr>
<tr>
<td>Kernel sizes</td>
<td>[5, 1, 1]</td>
<td>16</td>
<td>50.63</td>
<td>35.39</td>
<td>8.23</td>
<td>34.69</td>
<td></td>
</tr>
<tr>
<td>Optimizer</td>
<td>AdamW [6]</td>
<td>Batch size</td>
<td>256</td>
<td>50.96</td>
<td>35.61</td>
<td>7.65</td>
<td>34.83</td>
</tr>
<tr>
<td>s</td>
<td>8</td>
<td>16</td>
<td>50.88</td>
<td>35.37</td>
<td>7.58</td>
<td>34.64</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>8</td>
<td>16</td>
<td>50.88</td>
<td>35.37</td>
<td>7.58</td>
<td>34.64</td>
<td></td>
</tr>
<tr>
<td>TSM size</td>
<td>8×8</td>
<td>16 8</td>
<td>51.17</td>
<td>35.70</td>
<td>8.31</td>
<td>34.99</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. TAL performance in Activitynet1.3 [1] dataset with various downstream heads.

Table 4. SoLa module hyperparameters. $K$ is a constant for the $\lambda$ assignment (Equation 1 in the main paper) and $s$ is the step size described in the caption of the main paper’s Figure 2.

C. Justification of the similarity assumption

Assignment of soft pseudo-label (Eq.1) is from the local similarity assumption:

“Only adjacent features should be similar while distant features remain distinct.”,

which is based on the empirical observation on general videos. One might suspect that the above assumption over-simplifies the complex characteristics of untrimmed videos. However, the fact that only “sampled subsequence of the given video” is utilized in the SoLa training procedure should not be neglected. It indicates that for the SoLa training, the assumption does not have to hold in the whole video, but only in the sampled video clip which is relatively shorter than the whole video. Thus, the exact local similarity assumption utilized in the SoLa training procedure should be noted as

“Only adjacent features should be similar while distant features remain distinct, if both features are from the same sampled subsequence.”,

which is a far more relaxed one compared to that of its whole-video version. It’s worth noting that adjusting the length of the sampled subsequence can accommodate videos with repetitions because if the subsequence length is much shorter than the repetition duration, the local similarity assumption still holds.

Moreover, there is no need for all the samples to strictly satisfy the local similarity assumption. Although there might be some counterexamples, we have found that training with the above assumption is reasonable as long as the number of them does not exceed the samples that obey the local similarity assumption. To support our claim, empirical analysis on this issue is presented in our main paper (Figure 3) which shows that general and widely used untrimmed video datasets mostly follow the local similarity assumption.

D. Implementation Details

We adopted a simple feedforward neural network for the SoLa module architecture. It consists of three layers: 1DConv-ReLU-1DConv-ReLU-1DConv, with additional residual connection from the very first layer (before the first 1DConv) and the last layer (after the last 1DConv). Here, the 1DConv pass is scaled with $\alpha$, a learnable parameter which is initialized as 0. Other detailed hyperparameters are presented in Table 4.

We do not think our SoLa module’s architecture is globally optimal. Rather, we show that despite its overly simplified architecture, the SoLa strategy still works well. Future
research will include devising better SoLa module architecture.

References


