Supplementary Material for ACCV 2020 Paper BaSSL: Boundary-aware Self-Supervised Learning for Video Scene Segmentation

Jonghwan Mun[1,⋆] Minchul Shin[1,⋆] Gunsoo Han[1]
Sangho Lee[2] Seongsu Ha[2] Joonseok Lee[2,†] Eun-Sol Kim[3,†]

[1]Kakao Brain
[2]Graduate School of Data Science, Seoul National University
[3]Department of Computer Science, Hanyang University
[1]{jason.mun,craig.starr,coco.han}@kakaobrain.com
[2]{sangho.lee,sha17,joonseok}@snu.ac.kr [3]eunsolkim@hanyang.ac.kr

A Additional Implementation Details

Additional details of the shot encoder (*i.e.*, ResNet-50) and the contextual relation network (*i.e.*, Transformer) are as follows. For the shot encoder, each shot is given by three key-frames (*i.e.*, $N_k = 3$) and a shot encoding $\mathbf{e}$ is given by the averaged feature after inferring individual three key-frames using ResNet-50; note that, to speed up the pre-training, we use randomly sampled one key-frame out of three. For Transformer, the hyperparameters are set to $(L = 2, H = 768, A = 8)$ where L, H and A mean the number of stacked transformer blocks, the dimension of hidden activation and the number of attention heads, respectively. We apply the Dropout technique [8] on hidden states and attention weights with a probability of 10% and use GELU [3] as an activation function.

For data augmentation of key-frames in a shot, we adopt PyTorch's torchvision package. Given a sequence of shots, we apply random crop (with resize), random flip, random color jitter and random Gaussian blur. In detail, firstly, the cropping is performed with a random size (*i.e.*, scales between [0.14, 1.0] of the original size) and a random aspect ratio (between 3/4 to 4/3), and then the cropped one is resized to (224,224). Secondly, we apply a random horizontal flip with a probability of 50%. Thirdly, as a color augmentation, we perform a random color jitter (with a probability of 80%) and a random color dropping to gray scale (with a probability of 20%). The color strength parameters for jittering are set to $\{\text{brightness}, \text{contrast}, \text{saturation}, \text{hue}\} = (0.2, 0.2, 0.2, 0.05)$. Finally, Gaussian blur is applied with a probability of 50% where a standard-deviation of spatial kernel is set to [0.1, 2.0]. Note that the same augmentations are applied to all key-frames in the input sequence $\mathbf{S}_n$ of shots while different color jittering is applied on individual shots. Also, for $\mathbf{S}_n^{\text{slow}}$, we perform a different augmentation compared to that applied on $\mathbf{S}_n$.

During the pre-training stage, the model parameters are randomly initialized and then trained using the proposed pretext tasks. We use LARS [9] to learn

⋆ Equal contribution † Corresponding authors

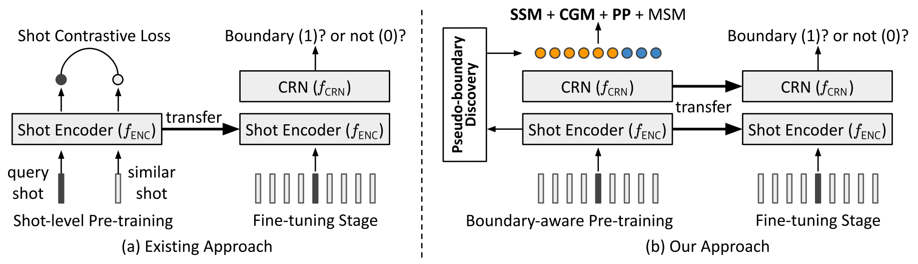

Fig. 1. Comparison between existing approaches and ours for video scene segmentation. The existing approach focuses only on learning shot-level representation given by shot encoder (f_{ENC}). In contrast, our boundary-aware pre-training method focuses on learning contextual representation by taking neighbor shots into account. Thus, our method can learn both the shot encoder (f_{ENC}) and the contextual relation network (CRN; f_{CRN}) and transfer their parameters during the fine-tuning stage.

the model (except for parameters of bias and Batch-Normalization) with a mini-batch of 256 shot sequences, a base learning rate of 0.3, momentum of 0.9, weight decay of 10^{-6} and trust coefficient of 0.001. We pre-train the model for 10 epochs with a linear warm-up strategy for 1 epoch (*i.e.*, 10% of whole training epochs) followed by learning rate decaying with a cosine schedule. The temperature τ in Eq. (5) in the main paper is set to 0.1. Using 16 V100 GPUs with mixed precision training, it takes less than 2 days for pre-training.

In the fine-tuning stage, we initialize the parameters of the shot encoder and the contextual relation network by that of the pre-trained ones. However, we freeze the parameters of the shot encoder following [2]. We fine-tune the contextual relation network and the scene boundary detection head for 20 epochs using Adam [5] with a learning rate of 10^{-5} and a mini-batch of 1024 training examples. The learning rate is decayed with a cosine schedule without warm-up.

B Comparison with Shot-level Self-supervised Learning

As mentioned in the main paper, our approach is distinguishable from the shot-level pre-training approach [2] in that the objectives used in our approach (BaSSL) is to learn contextual representations by taking neighbor shots into account. Fig. 1 provides a clear summary of comparison between shot-level pre-training and our boundary-aware pre-training, BaSSL. Firstly, shot-level pre-training takes a pair of two shots as an input while BaSSL takes a sequence of shots. Secondly, shot-level pre-training aims to train shot encoder (f_{ENC}) only, while BaSSL trains both the shot encoder and the contextual relation network (f_{ENC} and f_{CRN}). In contrast to the shot-level pre-training that requires to train f_{CRN} from scratch during the fine-tuning stage, BaSSL benefits from weight transfer by pre-training the parameters of f_{CRN} with large-scale in-domain data in advance. Note that the results (M6-7) in Table 2 in the main paper show

Table 1. Comparison between the shot-level and the boundary-aware pre-training.

Check List	Shot-level Pre-training	Boundary-aware Pre-training
Network architecture	f_{ENC}	$f_{\text{ENC}} + f_{\text{CRN}}$
Training input	a pair of shots (#shots: 2)	a sequence of shots (#shots: 2K+1)
Weights transferable for f_{ENC}?	yes	yes
Weights transferable for f_{CRN}?	no	yes
Positive pair in contrastive learning	shot-shot	shot-scene

Table 2. Comparison of existing video scene segmentation datasets. Note that we brought the table from [6] with an update on the MovieNet-SSeg dataset.

Dataset	#Video	#Scene	#Shot	Time (h)	Source
BBC [1]	11	670	4.9K	9	Documentary
OVSD [7]	21	300	10K	10	MiniFilm
MovieNet-SSeg [4]	318	42K	500K	-	Movies
MovieNet [4]	1,100	-	1.6M	-	Movies

Table 3. Comparison between our method and shot-level pre-training baselines on BBC and OVSD datasets in an unsupervised setting. The numbers mean AP and the best model is highlighted in bold.

Model	SimCLR (instance)	SimCLR (temporal)	SimCLR (NN)	BaSSL
BBC	32.34	34.18	32.92	**39.98**
OVSD	25.45	24.92	25.02	**28.68**

that the weight transfer of f_{CRN} is important to improve the video scene segmentation performance. Finally, the contrastive learning objective in shot-level pre-training drives the representations of two shots (query and positive) to be close to each other, whereas Shot-Scene Matching objective in our approach performs the same task but with a shot (query) and its associated scene (positive; a sequence of shots). The Table 1 summarizes the aforementioned comparisons.

C Results on Additional Datasets

Table 2 shows the data statistics of different video scene segmentation datasets. We found the limited number of datasets that provide the scene boundary annotations and, as far as we know, the MovieNet-SSeg [4] is the largest-scale video scene segmentation dataset. We further compare BaSSL with shot-level pre-training baselines on two additional datasets—BBC [1] and OVSD [7]. Note that the training and test splits are not available and the dataset size is extremely limited (11 and 21 videos in BBC and OVSD, respectively); in addition, 2 out of 21 videos in OVSD is not available. Thus, we infer predictions using models trained on MovieNet-SSeg without fine-tuning on BBC and OVSD (*i.e.*, unsupervised setting). The results are summarized in Table 3. The result shows the superiority of our method compared to shot-level pre-training baselines.

Algorithm 1 DTW-based pseudo-boundary discovery

```
1: Input: Shot encoder $f_{\text{enc}}$, contextual relation network $f_{\text{CRN}}$, and an input shot
   sequence $\mathbf{S}_n = \{\mathbf{s}_{n-K}, ..., \mathbf{s}_n, ..., \mathbf{s}_{n+K}\}$ centered at $n^{\text{th}}$ shot $\mathbf{s}_n$ with neighbor size
   $K$, two image augmentation functions $\lambda^1_{\text{aug}}, \lambda^2_{\text{aug}}$.
2: $(\mathbf{E}_n, \mathbf{E}^{\text{slow}}_n) \leftarrow ([], [])$
3: for $i = n - K$ to $n + K$ do
4:     $\mathbf{e}_i \leftarrow f_{\text{ENC}}(\lambda^1_{\text{aug}}(\mathbf{s}_i))$   // extract shot-level representations for all shots
5:     $\mathbf{E}_n \leftarrow \{\mathbf{E}_n; \mathbf{e}_i\}$   // append
6: end for
7: for $i$ in $\{n - K, n + K\}$ do
8:     $\mathbf{e}_i \leftarrow f_{\text{ENC}}(\lambda^2_{\text{aug}}(\mathbf{s}_i))$   // extract shot-level representations for slow sequence
9:     $\mathbf{E}^{\text{slow}}_n \leftarrow \{\mathbf{E}^{\text{slow}}_n; \mathbf{e}_i\}$   // append
10: end for
11: $\mathbf{S}^{\text{left}}_n$, $\mathbf{S}^{\text{right}}_n$, $b^* \leftarrow \text{DTW}(\mathbf{E}_n, \mathbf{E}^{\text{slow}}_n)$   // apply dynamic time warping
12: Output: Two continuous non-overlapping sub-sequences $\mathbf{S}^{\text{left}}_n$ and $\mathbf{S}^{\text{right}}_n$ and a
    pseudo boundary shot $\mathbf{s}_{n+b^*}$.
```

D Algorithm for Pseudo-boundary Discovery

In this section, we describe the details of pseudo-boundary discovery method applying DTW on $\mathbf{S}_n$ and $\mathbf{S}^{\text{slow}}_n$. In practice, $\mathbf{S}_n$ is given as a mini-batch resulting in a tensor with a shape of (B, S, N_k, C, H, W) where individuals mean the batch size, the number of shots in $\mathbf{S}_n$ (*i.e.*, $2K + 1$), the number of key-frames in a shot, channels, frame height and frame width, respectively. Then, we obtain $\mathbf{S}^{\text{slow}}_n \in \mathbb{R}^{B\times2\times N_k\times C\times H\times W}$ that is composed of the first and last shots in $\mathbf{S}_n$. We apply two different augmentation functions into key-frames in $\mathbf{S}_n$ and $\mathbf{S}^{\text{slow}}_n$, respectively. Next, we compute encoded representation of shots from $\mathbf{S}_n$ and $\mathbf{S}^{\text{slow}}_n$ using f_{ENC}. Note that during the pre-training stage, we randomly sample one key-frame among N_k candidates in a shot and then reshape the input tensor as (B*S, C, H, W) or (B*2, C, H, W) to be forwarded by the shot encoder f_{ENC}; thus the tensor shape of the encoded shot representation is given by (B, S, D_e) or (B, 2, D_e) after apply reshaping, where D_e means the dimension of encoded feature. Finally, given two sequences of encoded representation for $\mathbf{S}_n$ and $\mathbf{S}^{\text{slow}}_n$, DTW provides two sub-sequences $\mathbf{S}^{\text{left}}_n$ and $\mathbf{S}^{\text{right}}_n$ and a pseudo boundary shot $\mathbf{s}_{n+b^*}$. The algorithm 1 illustrates the details. In addition, to demonstrate the simplicity of the alignment computation using DTW, we include the PyTorch code in Listing 1.1. The implementation of DTW can be done in 6 lines of python code using *tslearn* package.

E More Ablation Study using NMI

Using NMI, we additionally perform ablation study of our algorithm by adding pretext tasks one by one, and measure the corresponding NMI scores. The result in Table 4 shows that better NMI score is achieved as more pretext tasks are combined together. This tendency is also observed in our ablation in Table 3 in

```python
from tslearn import metrics
import numpy as np
def compute_dtw_path(seq_1, seq_2):
  """
  Input:
    seq_1: sparse shots embedding, shape = torch.size([2, dim])
    seq_2: dense shots embedding, shape = torch.size([N, dim]), N > 2
  Output:
    dtw_path: output of DTW algorithm, shape = torch.size([N, dim])
  """

  cost = (1-torch.bmm(seq_1, seq_2.transpose(1, 2))).numpy()
  dtw_path = []
  for bsz in range(cost.shape[0]):
    _path, _ = metrics.dtw_path_from_metric(
        cost[bsz], metric="precomputed")
    dtw_path.append(np.asarray(_path)) # torch.Size([N, dim])
  return dtw_path
```

Listing 1.1. PyTorch code for alignment computation using DTW given two sequences. The *tslearn* package is used for DTW path calculation.

Table 4. Ablation study on the combination of boundary-aware pretext tasks measured by NMI.

Pretext Tasks	NMI	Gain (Δ%)
SSM	85.48	0.00%
SSM+MSM	85.64	+0.19%
SSM+MSM+CGM	85.93	+0.33%
SSM+MSM+CGM+PP	86.71	+0.91%

the main paper, which indicates the NMI score of pre-trained models is highly correlated with the final performance after the fine-tuning.

F More Qualitative Analysis

Pseudo-boundaries We compare the quality of discovered pseudo-boundaries with the ground truth scene boundaries in Fig. 2. In most cases, we observe the pseudo-boundaries identified by the DTW algorithm are successfully located in close distance with the ground truth ones. This result validates our idea considering the problem of discovering pseudo-boundary as a temporal alignment problem between two sequences with different frequencies ($\mathbf{S}_n$ and $\mathbf{S}_n^{\text{slow}}$). At the same time, we illustrate the failure cases. Although discovered pseudo-boundary does not match the ground truth in this case, we figure the determined boundary is not always arbitrary. For example, the mismatch is often caused by the noise existing in the ground truth (see the first row in the failure cases). On the other hand, in case all shots are visually similar (see the third row in the failure cases), the DTW solely relying on the visual modality fails to find the correct boundary.

Predicted scene boundaries The Fig. 3 illustrates the scene boundary predictions of different models. Comparing with the baselines, we observe that our

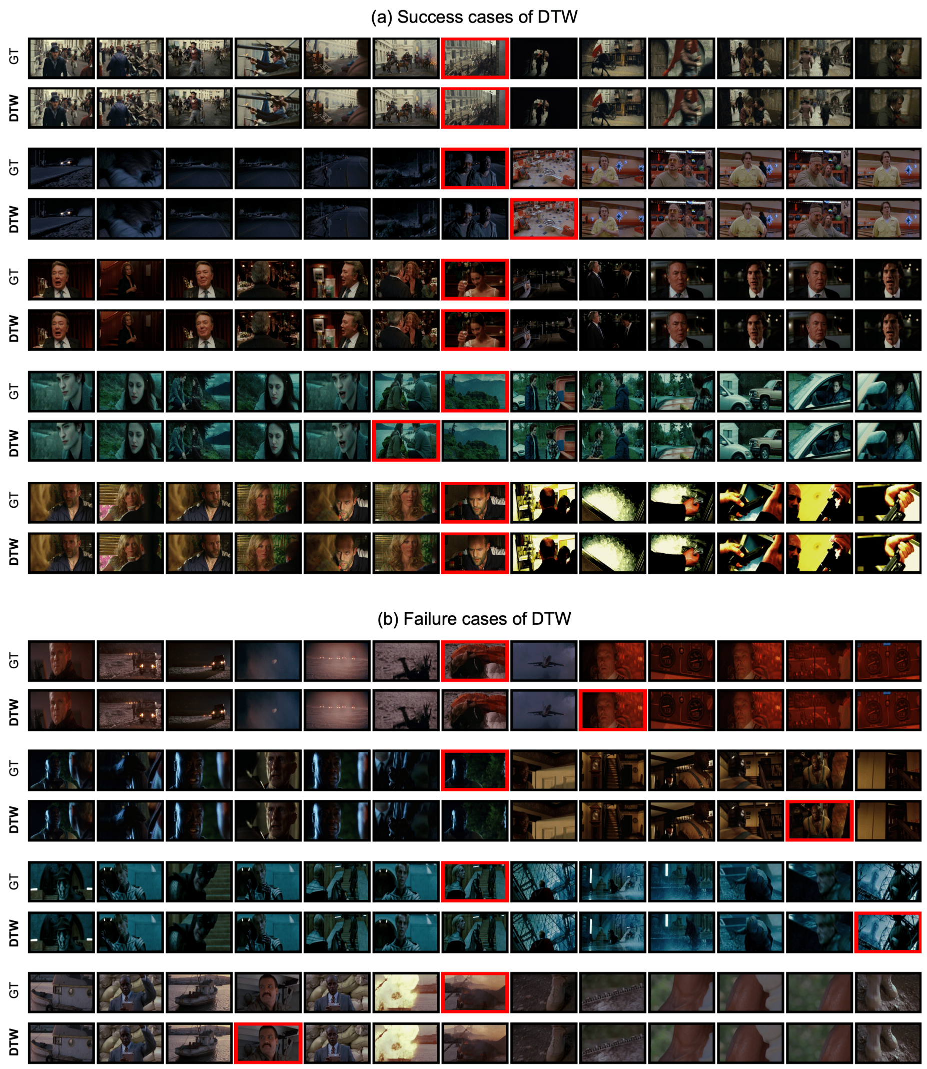

Fig. 2. Comparison between the ground truth scene boundaries and the discovered pseudo-boundaries based on the DTW algorithm. The examples are sampled from the MovieNet-SSeg dataset. All boundary shots are highlighted in red.

approach, BaSSL, shows qualitatively better result for video scene segmentation. On the other hand, we observe the over-segmentation issue in many cases using any competing methods (including ours). Our finding implies that achieving the highest recall only does not guarantee the highest performance in practice. We reckon that further studies on this over-segmentation problem would be a highly important topic when it comes to real-world application.

References

1. Baraldi, L., Grana, C., Cucchiara, R.: A Deep Siamese Network for Scene Detection in Broadcast Videos. In: ACM MM (2015)
2. Chen, S., Nie, X., Fan, D., Zhang, D., Bhat, V., Hamid, R.: Shot Contrastive Self-Supervised Learning for Scene Boundary Detection. In: CVPR (2021)
3. Hendrycks, D., Gimpel, K.: Gaussian Error Linear Units (GELUs). arXiv:1606.08415 (2016)
4. Huang, Q., Xiong, Y., Rao, A., Wang, J., Lin, D.: MovieNet: A Holistic Dataset for Movie Understanding. In: ECCV (2020)
5. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. In: ICLR (2015)
6. Rao, A., Xu, L., Xiong, Y., Xu, G., Huang, Q., Zhou, B., Lin, D.: A Local-to-Global Approach to Multi-modal Movie Scene Segmentation. In: CVPR (2020)
7. Rotman, D., Porat, D., Ashour, G.: Robust and efficient video scene detection using optimal sequential grouping. In: IEEE international symposium on multimedia (ISM) (2016)
8. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of machine learning research **15**(1), 1929–1958 (2014)
9. You, Y., Gitman, I., Ginsburg, B.: Large Batch Training of Convolutional Networks. arXiv:1708.03888 (2017)

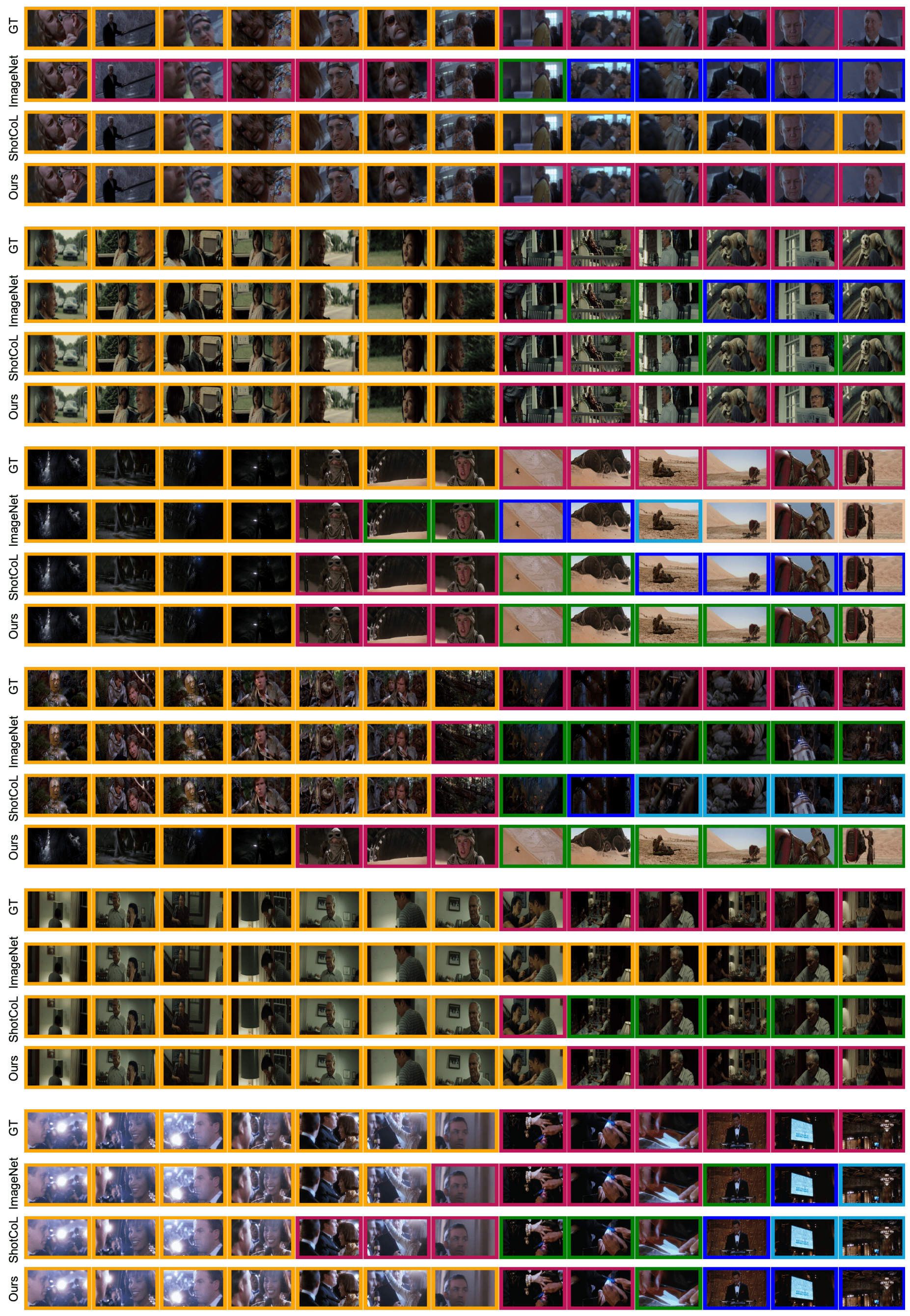

Fig. 3. Comparison of boundary detection results from three pre-training approaches: ImageNet pre-trained ResNet, ShotCoL, and BaSSL. The first row shows the reference that is composed of two adjacent scenes divided by the ground truth boundary. We visualize the shots that are assigned to the same scene segments with the same colored border.