Supplementary Materials for From Detection to Anticipation: Online Understanding of Struggles across Various Tasks

Shijia Feng, Michael Wray, Walterio Mayol-Cuevas
University of Bristol
{shijia.feng.2019, michael.wray, walterio.mayol-cuevas}@bristol.ac.uk

1. Implementation Details

Input Features. Since both LSTR [8] and CMeRT [5] are feature-based temporal models, we use pre-extracted video features from the EvoStruggle dataset [3]. These features are obtained using the SlowFast Network [1, 2], pre-trained on the Kinetics dataset [4].

The SlowFast model processes each clip of 32 frames sampled from the original 50 FPS video with a stride of 16 frames, producing one feature vector per 16-frame segment. This results in a temporal resolution of one feature every $16/50 = 0.32$ seconds, corresponding to a feature-level frame rate of approximately 3.125 FPS.

Each feature vector is 2304-dimensional and consists of two parts: a 2048-dimensional component from the Slow pathway, capturing spatial visual information, and a 256-dimensional component from the Fast pathway, encoding motion information. When loading the features, we explicitly separate these two components and feed them into the feature fusion head of the respective model. The fusion head concatenates the visual and motion features and projects them via an MLP before passing them to the temporal backbone.

For the anticipation task, the model predicts future struggle states over a target horizon (e.g., 2 seconds), corresponding to approximately 6 feature frames under a feature sampling interval of 0.32 seconds. When computing anticipation length in terms of frames, the feature frame rate is rounded to 3 FPS.

To align the ground-truth annotations with the feature frames, we convert the temporal start and end times into frame-level labels by multiplying them by the feature frame rate (3.125 FPS) and rounding to the nearest integers.

LSTR [8] for Online Detection and Anticipation. We train our models from scratch using the Adam optimizer with a base learning rate of 1×10^{-6} and a weight decay of 5×10^{-3}. A cosine learning rate scheduler is applied for 20 epochs, with a linear warm-up over the first five epochs starting from 1×10^{-7}. Unless otherwise stated, the training batch size is set to 16. For within-activity evaluations, such as the *Origami* activity, we adopt the LSTR model [8] with the following settings:

- **Architecture:** The encoder module consists of 3 transformer decoder layers for compressing long-term memory, each with a feed-forward dimension of 1024 and 16 attention heads. The decoder module comprises two transformer decoder layers that attend to the compressed long-term memory using the short-term memory as a query.
- **Memory Settings:** The long-term memory includes 1600 frames (equivalent to 512 seconds of video), sampled at a rate of 4. The short-term memory consists of 25 frames (8 seconds), sampled at a rate of 1.
- **Anticipation Settings:** For struggle anticipation, the anticipation length is set to 2 seconds, corresponding to 6 frames of input features.
- **Regularization:** A dropout rate of 0.2 is applied to alleviate overfitting.

The same training and architectural settings are applied to the *Tangram* and *Shuffle Cards* activities. For the *Tying Knots* activity, the dropout rate is increased to 0.4 to alleviate overfitting.

In the experiments where the model is trained jointly on all four activities in the EvoStruggle dataset [3], we extend training to 50 epochs and reduce the base learning rate to 1×10^{-7}.

CMeRT [5] for Online Detection and Anticipation. For within-activity evaluations, such as the *Tying Knots* activity, we train the model using the Adam optimizer with a base learning rate of 5×10^{-6} and a weight decay of 5×10^{-3}. A cosine learning rate scheduler is applied for 15 epochs, with a linear warm-up over the first five epochs starting from 5×10^{-7}. The batch size is set to 32.

- **Architecture:** The CMeRT [5] model builds on the LSTR architecture [8], with the encoder module consisting of 3 transformer decoder layers for compressing long-term memory. In addition, one transformer decoder

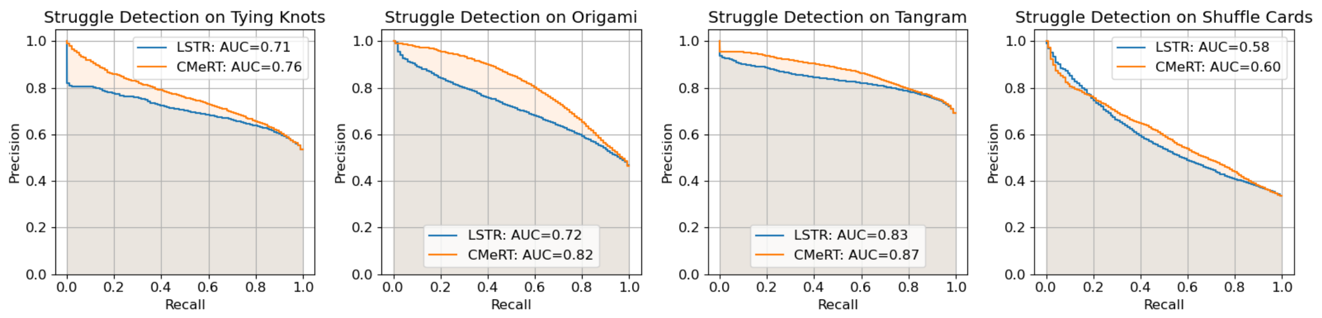

Figure 1. Precision-Recall Curve of models trained on individual activities, evaluated by the cAP on the *online struggle detection* task.

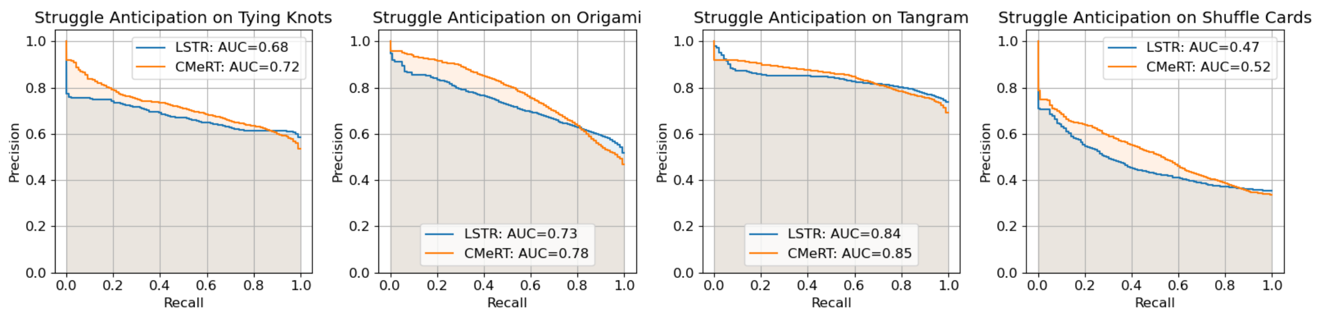

Figure 2. Precision-Recall Curve of models trained on individual activities, evaluated by the cAP on the *struggle anticipation* task.

layer is used for generating future features, and two transformer decoder layers act as the decoder module to produce the final features before classification. Each transformer layer uses eight attention heads, and the feed-forward dimension is set to 1024.

- **Memory Settings:** The long-term memory includes 1600 frames (512 seconds) sampled at a rate of 4. The short-term memory consists of 25 frames (8 seconds), sampled at a rate of 1.
- **Anticipation Settings:** The anticipation length is set to 2 seconds (6 frames), and the internal future feature generation length is set to 48 frames (16 seconds).
- **Regularization:** A dropout rate of 0.3 is used to alleviate overfitting.

The same training and architectural settings are applied to the *Origami*, *Tangram*, and *Shuffle Cards* activities.

For experiments where the model is trained jointly on all four activities in the EvoStruggle dataset [3], we also train for 15 epochs using the same base learning rate of 5×10^{-6}. In additional experiments, such as those examining activity-level and task-level generalization, the same hyperparameters are used.

Sampling Strategy for Training and Inference. During training, feature frames are sampled using a non-overlapping sliding window based on the short-term working memory length (25 frames, equivalent to 8 seconds). For each sampled video sequence, a random start index is selected within the video.

At inference time, we apply a batch inference strategy: the short-term memory window starts at the beginning of the video and advances one frame at a time. If there are insufficient frames preceding the short-term memory window to fill the long-term memory, we pad it by repeating the first frame index and apply masking up to the rightmost valid index. This ensures that all inputs maintain consistent dimensions while preserving causal structure.

2. Precision-Recall Evaluation

To fairly compare LSTR [8] and CMeRT [5], precision-recall (PR) curves are plotted across all activities for both online struggle detection and anticipation, offering a more comprehensive evaluation, especially suited to the imbalanced nature of struggle data. Figure 1 presents the PR curves for online detection, while Figure 2 shows the PR curves for anticipation. The Area Under the Curve (AUC) is also computed for each PR curve to quantify performance. As shown in the two figures, the PR curves for CMeRT [5] generally exhibit larger AUCs than those of LSTR [8], with particularly notable gains in the Origami activity. The performance gap is smaller in the Tangram and Shuffle Cards activities. These results further demonstrate that CMeRT [5] is more effective at online struggle detection and anticipation.

Struggle Anticipation					
Activities	**Anticipation Avg. (cAP%)**	**ECE (frame-level)**	**Event-Level F1@$\{0.1, 0.3, 0.5\}$**	**ECE (event-level)**	**Lead Time (s)**
Tying Knots	70.16	0.0800	0.5745	0.3763	1.29
Origami	80.82	0.1240	0.3726	0.3726	2.22
Tangram	72.45	0.1375	0.4107	0.3890	2.02
Shuffle Cards	69.12	0.0774	0.4218	0.3293	1.59
Online Struggle Detection					
Activities	**Detection (cAP%)**	**ECE (frame-level)**	**Event-Level F1@$\{0.1, 0.3, 0.5\}$**	**ECE (event-level)**	**Mean Detection Delay (s)**
Tying Knots	74.09	0.0909	0.5846	0.3612	0.68
Origami	83.29	0.1376	0.3998	0.3505	0.01
Tangram	75.94	0.1424	0.4574	0.3692	0.95
Shuffle Cards	73.85	0.0935	0.5269	0.3206	0.69

Table 1. Main experimental results with additional event-level metrics, including averaged F1 scores over multiple IoU thresholds, expected calibration error (ECE), and latency measures (time-to-anticipation for struggle anticipation and mean detection delay for online struggle detection) using the CMeRT model with SlowFast features.

3. Additional Evaluation Metrics

Additional evaluation metrics are reported for the main experimental results in Table 1. These include: (1) event-level averaged F1 scores at IoU thresholds of 0.1, 0.3, and 0.5, (2) expected calibration error (ECE) computed at both the frame level and the event level, and (3) latency metrics, namely time-to-anticipation (lead time) and mean detection delay. The extension experiments are conducted using the CMeRT model with SlowFast-extracted video features. The model CMeRT generates predictions at the frame level, while event-level predictions are obtained by grouping consecutive positive frames into segments defined by their start and end points. IoU is then used to determine matches between predicted struggle events and ground-truth annotations. The ECE is computed separately for frame-level and event-level outputs according to the definitions in [7]. Anticipation lead time reflects how far in advance the model can predict the onset of a struggle, whereas mean detection delay measures the average lag in detecting a struggle in the online setting.

According to the results in Table 1, the frame-level expected calibration error (ECE) remains low (approximately 0.1), suggesting that the frame-level predictions are well calibrated. The event-level F1 scores range from 0.3726 to 0.5846, with only minor differences between the anticipation and online detection tasks. In contrast with the frame-level ECE, the event-level ECE is higher, between 0.3 and 0.4, likely due to the conversion of frame-level predictions into continuous struggle segments. For anticipation, the model achieves an average lead time of 1–2 seconds before the ground-truth struggle onset, demonstrating its ability to predict struggle in advance. Meanwhile, the mean detection delay remains within 1 second, indicating that the model responds promptly to struggle moments in the online streaming setting.

4. Additional Details for the Runtime Analysis

More detailed tables and discussions are presented in this section. The full runtime results are presented in Table 2.

Overall, the S3D provides a slightly faster backbone than SlowFast, while LSTR is significantly lighter than CMeRT. When combined, the fastest setting (S3D + LSTR) achieves 23.7 FPS, and the slowest (SlowFast + CMeRT) runs at 18.2 FPS, both of which remain feasible for real-time applications.

Model	**GFLOPs**	**Params (M)**	**Runtime / FPS**
Feature Extraction Backbones			
SlowFast-R50	66.42	33.64	27.80 ms / 24.6 FPS
S3D	94.85	15.82	28.00 ms / 28.4 FPS
Temporal Models (feature input)			
LSTR	3.20	25.75	6.99 ms / 143 FPS
CMeRT	5.87	42.58	14.27 ms / 70 FPS
Combined Runtime (Feature + Model)			
SlowFast + LSTR	69.62	59.39	47.69 ms / 21.0 FPS
SlowFast + CMeRT	72.29	76.22	54.97 ms / 18.2 FPS
S3D + LSTR	98.05	41.57	42.23 ms / 23.7 FPS
S3D + CMeRT	100.72	58.40	49.51 ms / 20.2 FPS

Table 2. Runtime analysis of feature extraction backbones and temporal models on a single NVIDIA 1080Ti GPU (batch size = 1). FPS is reported with preprocessing included.

Setting	**Feat. FPS**	**Ant. cAP**	**Det. cAP**
SlowFast + CMeRT, stride 16	3.125	76.28	78.51
SlowFast + CMeRT, stride 32	1.562	73.04	74.80
S3D + CMeRT, stride 16	3.125	74.28	75.71

Table 3. Impact of backbone and stride (window size = 32) on feature throughput and cAP (%).

We further examine the effect of backbone choice and feature stride on performance using CMeRT as the temporal model, shown in Table 3. With SlowFast at stride

16, the system processes 3.125 feature FPS and achieves 76.28% cAP for anticipation and 78.51% for online detection. Increasing the stride to 32 (non-overlapping windows) reduces the throughput to 1.562 feature FPS and lowers performance to 73.04% and 74.80%, respectively. Using S3D with a stride of 16 yields a similar 3.125 feature FPS but slightly lower accuracy (74.28% for anticipation and 75.71% for detection). All of the above show the following trend: A larger stride improves computational efficiency but comes with a modest drop in accuracy. S3D provides higher efficiency than SlowFast, with slightly lower accuracy overall, yet still outperforms SlowFast when a larger stride is used. These findings underscore the trade-off between backbone choice, stride selection, and model accuracy.

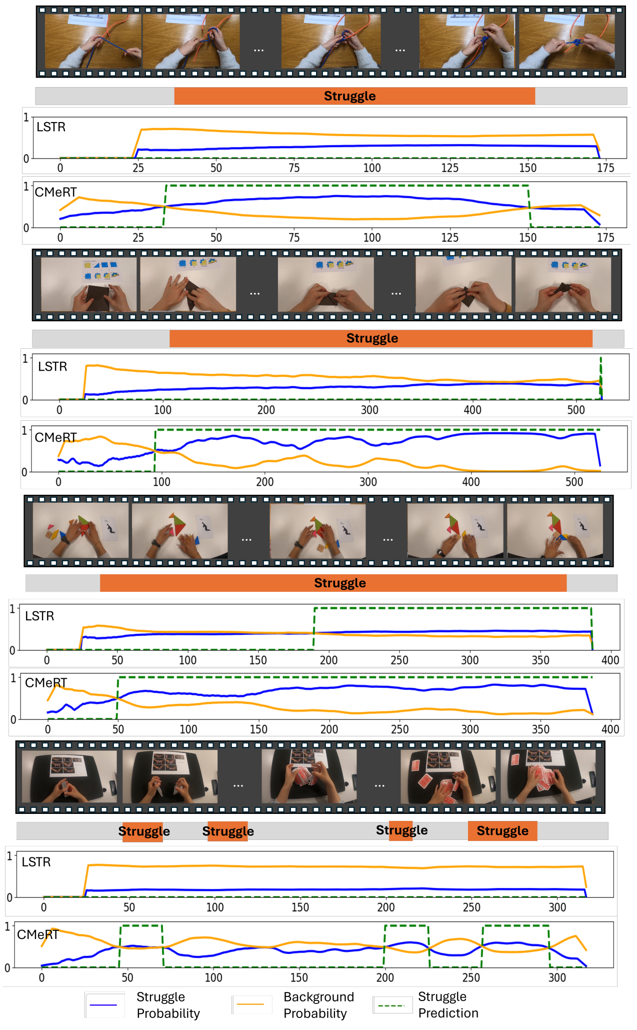

Figure 3. Qualitative Results on **Struggle Anticipation** Comparing Model LSTR [8] and Model CMeRT [5] Across Activities (from top to bottom) Tying Knots, Origami, Tangram, and Shuffle Cards.

5. Additional Visualization Results

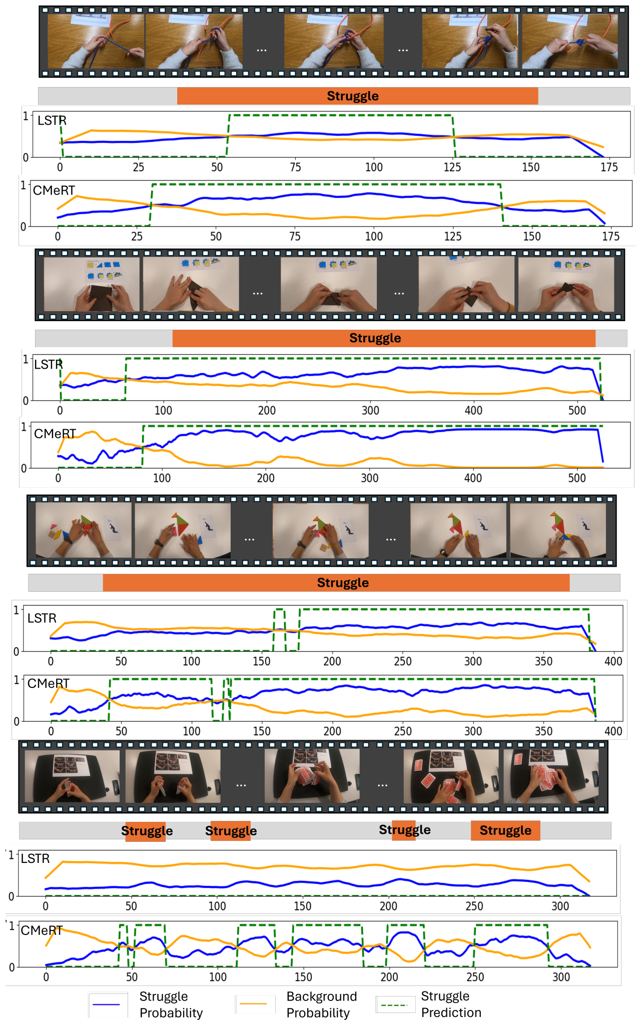

Figure 4. Qualitative Results on **Online Struggle Detection** Comparing Model LSTR [8] and Model CMeRT [5] Across Activities (from top to bottom) Tying Knots, Origami, Tangram, and Shuffle Cards.

Figure 3 and Figure 4 present additional prediction results for struggle anticipation and online struggle detection, respectively. These visualizations include comparisons between the two models: LSTR [8] and CMeRT [5].

The LSTR model [8] is generally less accurate than CMeRT [5] in both detecting and anticipating struggle, partly due to the latter's use of future-feature prediction during training. While LSTR can detect some struggle moments in the online detection task, its temporal boundaries are often imprecise. In the more challenging anticipation task, LSTR frequently fails to detect any struggle moments, as illustrated in Figure 3.

In contrast, CMeRT [5] demonstrates significantly better performance in both tasks, with more accurate boundaries that align closely with the ground truth. However, it still struggles with repeated or short-duration struggle moments—leading to false negatives in anticipation (e.g., Shuffle Cards activity in Figure 3) and occasional false positives in detection (Figure 4), highlighting the inherent difficulty of both tasks.

Finally, Figure 5 presents a qualitative comparison between the reformulated online struggle detection and the original offline approach in [3]. Both methods are able to detect struggle moments in the provided examples; however, the online model identifies them more promptly. In the first three examples, the online method captures the onset of struggle more accurately, and in some cases even slightly earlier than the annotated ground truth, whereas the offline method typically detects the onset with noticeable delay. On the other hand, the online approach can occasionally overfit, leading to false positives—as illustrated in the fourth example, where a spurious struggle detection appears mid-video.

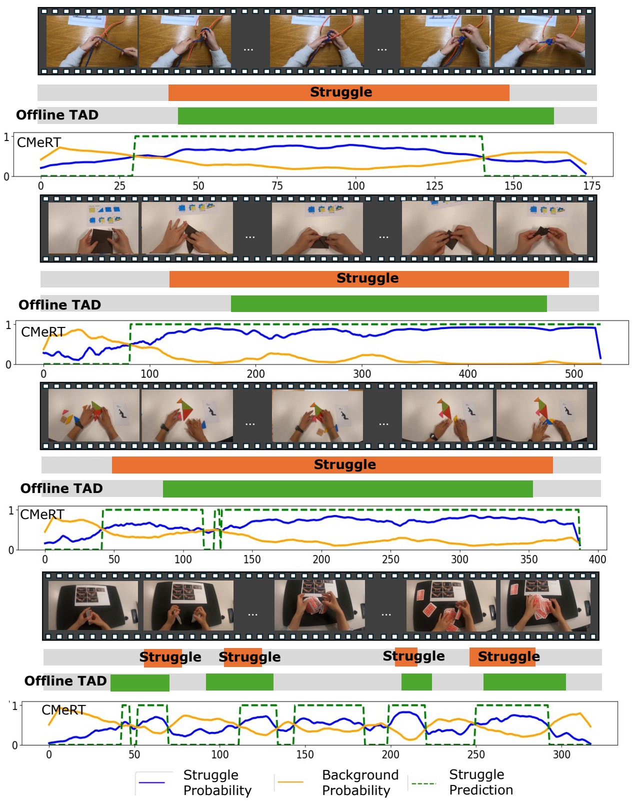

Figure 5. Qualitative comparisons between the **Online Struggle Detection** with Model CMeRT [5] and the **offline Struggle Detection** in [3] using Model TriDet [6] (shown using green bars), across Activities (from top to bottom) Tying Knots, Origami, Tangram, and Shuffle Cards.

References

[1] Haoqi Fan, Yanghao Li, Bo Xiong, Wan-Yen Lo, and Christoph Feichtenhofer. Pyslowfast. https://github.com/facebookresearch/slowfast, 2020. 1

[2] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6202–6211, 2019. 1

[3] Shijia Feng, Michael Wray, and Walterio Mayol-Cuevas. Evostruggle: A dataset capturing the evolution of struggle across activities and skill levels, 2025. 1, 2, 5

[4] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset, 2017. 1

[5] Zhanzhong Pang, Fadime Sener, and Angela Yao. Context-enhanced memory-refined transformer for online action detection. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 8700–8710, 2025. 1, 2, 4, 5

[6] Dingfeng Shi, Yujie Zhong, Qiong Cao, Lin Ma, Jia Lit, and Dacheng Tao. Tridet: Temporal action detection with relative boundary modeling. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18857–18866, 2023. 5

[7] Cheng Wang. Calibration in deep learning: A survey of the state-of-the-art, 2025. 3

[8] Mingze Xu, Yuanjun Xiong, Hao Chen, Xinyu Li, Wei Xia, Zhuowen Tu, and Stefano Soatto. Long Short-Term Transformer for Online Action Detection. In *Advances in Neural Information Processing Systems*, pages 1086–1099. Curran Associates, Inc., 2021. 1, 2, 4