MiniROAD: Minimal RNN Framework for Online Action Detection

Joungbin An[1] Hyolim Kang[1] Su Ho Han[1] Ming-Hsuan Yang[1,2,3] Seon Joo Kim[1]
[1]Yonsei University [2]UC Merced [3]Google Research

Abstract

Online Action Detection (OAD) is the task of identifying actions in streaming videos without access to future frames. Much effort has been devoted to effectively capturing long-range dependencies, with transformers receiving the spotlight for their ability to capture long-range temporal structures. In contrast, RNNs have received less attention lately, due to their lower performance compared to recent methods that utilize transformers. In this paper, we investigate the underlying reasons for the inferior performance of RNNs compared to transformer-based algorithms. Our findings indicate that the discrepancy between training and inference is the primary hindrance to the effective training of RNNs. To address this, we propose applying non-uniform weights to the loss computed at each time step, which allows the RNN model to learn from the predictions made in an environment that better resembles the inference stage. Extensive experiments on three benchmark datasets, THUMOS, TVSeries, and FineAction demonstrate that a minimal RNN-based model trained with the proposed methodology performs equally or better than the existing best methods with a significant increase in efficiency. The code is available at https://github.com/jbistanbul/MiniROAD.

1. Introduction

Online Action Detection (OAD) is a challenging task that involves identifying actions taking place in a streaming video without access to future frames. Since its introduction [7], OAD has gained great attention in research circles, as it has numerous real-world applications, such as autonomous driving [20], smart surveillance systems [26, 27], and anomaly detection [31]. These applications require the ability to process streaming videos in real-time, making OAD an essential component of many advanced systems. As a result, there has been a growing interest in developing more accurate and efficient methods for performing OAD.

To address the challenges of OAD, recent efforts have focused on developing novel models for analyzing streaming videos in real-time. The conventional process of OAD consists of two stages. In the first stage, frames are converted into a sequence of features by a feature extractor. In the second stage, the extracted features are processed to capture the temporal information required for performing OAD. Much attention on OAD research has been paid to this second stage where the two mainstream models are RNNs [11, 30, 18, 5] and Transformers [9, 10, 33].

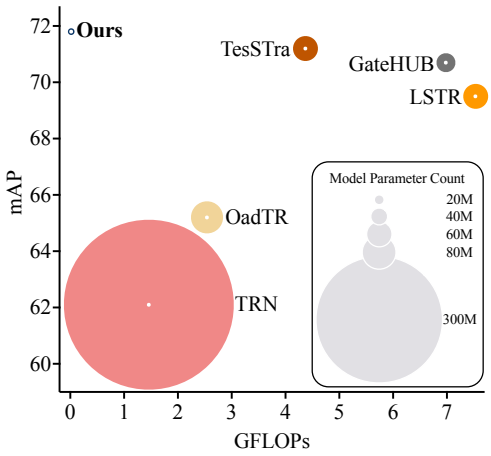

Figure 1: Performance of recent OAD methods with respect to GFLOPs and the model's size.

Each model has its strengths and weaknesses. RNNs can capture temporal transitions effectively by modeling sequences recurrently but are known for their training difficulties [23, 30]. Transformers can capture long-term dependencies through self-attention, however, attention computation is computationally expensive. Although recent research has shifted towards using Transformer models, our paper revisits RNN, which has been overlooked in recent studies.

In this work, we show that RNN's intrinsic inductive bias—prioritizing the current input while preserving meaningful temporal information from the histories—makes them well-suited for OAD. Unlike in natural language processing, where a distant word can significantly affect the next sequence prediction, we observe that in OAD, the current and its neighboring frames have a dominant influence on the current prediction. In addition, RNN is efficient, as it requires a small computational overhead compared to

the transformer models, making them well-suited for online tasks that require running in real-time.

Despite its architecture being well-suited for the OAD task, RNN has been recently overlooked due to the promising results shown by transformer-based models. The prevailing reasoning behind the lower performance of RNN based method was the difficulty of learning long-range dependencies [37, 23]. In this paper, we find that it is the *training-inference discrepancy* that hinders the effective training of RNNs.

During inference, the video is processed frame-by-frame in a streaming fashion. During training, however, the video is divided into multiple clips, which are then fed to the model, as illustrated in Figure 2. This creates a significant discrepancy between the two phases since videos are usually much longer than the training clips. Consequently, the clips provide limited temporal information, leading to a restricted view of the video during training. This limitation is not as problematic for recent OAD models using transformers as it is for models using RNNs. This is because recent transformer models utilize the temporal information between clips using explicit long-term memory even during training. At the same time, RNNs do not since they receive a non-informative hidden state initialized as zero at the beginning of the clip. As a result, RNNs make most predictions during training with the hidden state that lacks temporal information, whereas it makes most predictions during inference with the hidden state accumulated with sufficient temporal information.

We demonstrate that addressing the training and inference discrepancy in RNNs alone can significantly improve their performance without requiring any changes to their architecture. Specifically, we find that introducing non-uniform weights to losses at different time steps can alleviate this discrepancy, which is a primary factor contributing to RNN's suboptimal performance. This approach differs from the conventional assumption of averaging the loss over the joint prediction, as we demonstrate that non-uniform weights to the loss are essential for effectively training the model. We demonstrate the effectiveness of our methodology using **Mini**mal **R**ecurrent neural network for **O**nline **A**ction **D**etection (MiniROAD), which is a lightweight GRU based model. Figure 1 shows the comparison of our model with other recent models in terms of the model's performance, GFLOPs, and size. It shows that our method with only $\sim 0.4\%$ computational power needed for the previous state-of-the-art, is able to achieve better performance.

Our model is evaluated on conventional OAD benchmark datasets, including THUMOS'14 and TVSeries, as well as on the recently introduced FineAction dataset. As conventional datasets used for benchmarking OAD tasks are relatively small, the much larger FineAction dataset offers a more challenging testbed that better depicts real-life scenarios. Larger datasets like FineAction open up more possibilities for future work on OAD and hence our results can serve as a strong baseline. Our experiments show that our proposed methodologies achieve on-par or better results on the benchmarked dataset with significant improvements in efficiency.

The main contributions of this work are:

- We revisit the potential of RNNs and identify the training-inference discrepancy as a primary cause of their lower performance. To address this issue, we propose a simple yet effective method that makes RNN both effective and efficient in capturing the essential temporal information for online action detection.
- Extensive experimental results show that the proposed method performs favorably against the state-of-the-art with high efficiency; our model with only $\sim 0.4\%$ computational overhead compared to the existing methods, achieves on par or better performance on a challenging benchmark dataset, with $\sim 17\%$ performance improvement over the state-of-the-art in a large-scale dataset.
- We expand our methodology and carry out performance evaluation on Action Anticipation and show the applicability of our method on other online tasks.

2. Related Work

2.1. Online Action Detection

The Online Action Detection task [7] involves detecting actions in the current frame online, using information from the past up to the present. Earlier approaches to Online Action Detection relied on recurrent models. Geest et al. propose a two-stream feedback network [8], utilizing LSTM [18] where one stream focus on feature representation and the other on temporal patterns. RED [14] takes advantage of reinforcement learning to make action predictions as early as possible. Recent works using RNNs focus on designing more effective RNN cells. TRN [39] models a greater temporal context by having the LSTM cell [18] anticipate the future. IDN [12] uses the GRU cell [4] to explicitly differentiate relevant information for ongoing actions. In [21], FATSnet employs GRU to perform future anticipation and temporal smoothing for more effective current prediction. On the other hand, WOAD [15] introduces a method for weakly supervised online action detection. Existing RNN-based models mainly concentrate on developing more powerful RNN architectures to handle the task.

Transformers have also been used for the OAD task. OadTR [36] introduces a transformer encoder-decoder architecture for capturing long-term relations through self-attention. In [40], LSTR achieves significant performance improvements by employing a two-stage memory compres-

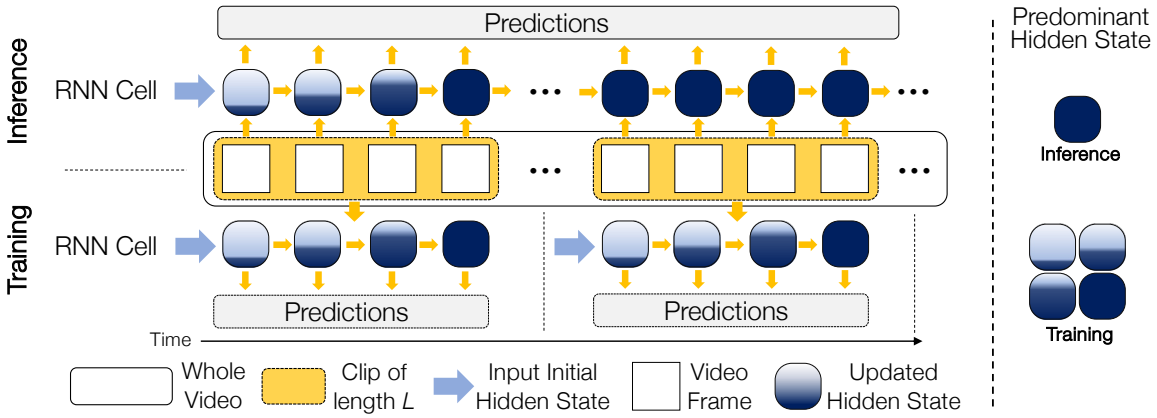

Figure 2: Illustration of Training and Inference Discrepancy. During the inference stage (top), frames are fed to the model sequentially in a streaming fashion. The hidden state evolves across the entire video, progressively incorporating accumulated temporal context (depicted as darkening over time). In the training stage (bottom), the video is divided into shorter clips of length L, serving as model inputs. Here, the model solely relies on temporal information within each clip, lacking inter-clip associations. This yields dominant predictions constrained by clip-specific information.

sion design with transformers, allowing for capturing long-term temporal and short-term context through self-attention. This two-stage memory design has been further advanced in subsequent works. GateHUB [3] improves the two-stage architecture with a Gated History Unit to address the limitations of prior cross-attention methods. This avoids incorrect current frame predictions by appropriately weighting informative and uninformative frames. Recently, TeSTra [43] extends the LSTR architecture by introducing two temporal smoothing kernels to reduce inference computation cost.

Although these transformer models have achieved state-of-the-art results, the attention computation involved in encoding long-term and short-term memory is computationally expensive. In this work, we show that RNNs, without modifying the internals of the architecture, can be just as effective in processing streaming videos with the proposed methodology alone.

2.2. Sequence Modeling

As many real-world applications involve processing time-series data, sequential modeling has been a topic of ongoing research. Recurrent Neural Networks (RNNs) are one of the most commonly used models for sequence modeling, with popular variants including [18] and [5]. These variants introduce mechanisms within the RNN cell to improve its memory capacity while attending to important parts of the sequential data. 1D CNNs [22] incorporate convolution operations to capture the temporal structure of time-series data. Recent advances have been made using transformers [33] that incorporate attention mechanisms for capturing short and long-term dependencies. Ever since its success in the language domain, its use has been widespread in various fields [10, 9, 1, 38].

There have been ongoing efforts on exploring non-transformer architectures. Gu *et al*. [16] introduces the State Space Model (SSM) in tackling sequence data, proving its efficacy in effectively handling long sequences. A recent work [25] demonstrates that RNNs can match the performance and training speed of SSMs by applying techniques such as linearizing and diagonalizing the recurrence, along with better parameterizations and initializations. Recently, there has been a notable surge in interest surrounding a novel variant of RNN called RWKV [28]. This architecture is the first non-transformer model to be successfully scaled to tens of billions of parameters, bringing together the efficient parallelizable training of transformers with the inference efficiency of RNNs. Our research aligns with these works in resurrecting RNNs in the transformer era.

3. Methodology

Provided a live streaming video, our objective is to recognize actions in the current frame based on past observations up to the current time. Since v_1^t, a streaming video from the start until the current time t only provides observations $\{v_1, ..., v_t\}$ for the past t frames, it is not possible to access any information beyond v_t. In this setting, the task is to use v_1^t to predict $\hat{y}_t \in \{0, 1, ..., K\}$ where K is the number of action classes and index 0 is the background class. During training, the video is divided into clips, each of length L, and hence clips of v_{t-L}^t are given as input. Following the convention, we suppose that a pre-trained feature extractor is available which can convert each frame v_t into a feature vector $f_t \in \mathbb{R}^C$. Our method employs the feature vector f_t as the input at current time t to predict $\hat{y}_t$.

3.1. Overview

Our method is developed based on the observation that a discrepancy exists between the training and the inference stages, which hinders the effective training of sequence-to-sequence models. Figure 2 illustrates this training-inference discrepancy in more detail. While during inference the model receives temporal information from the beginning of the video to the current time step t, in training, the model is provided with temporal information from $t-L$ to t. Consequently, for each video, clips make predictions using the hidden state that lacks temporal information, which critically affects the losses associated with the early predictions. However, we observe that the conventional approach unknowingly assigns the same uniform weight to all losses for each time step. Our proposed method applies different weights to losses at different time steps, resulting in better training convergence and improved model performance.

3.2. Alleviating Training-Inference Discrepancy

The conventional loss calculation for OAD follows the following cross-entropy calculation:

$$-\sum_{t=1}^{L}\sum_{k=0}^{K}\alpha_t \hat{y}_{t,k}\log(\hat{s}_{t,k}), \tag{1}$$

where $\hat{y}_{t,k}$ and $\hat{s}_{t,k}$ represent the ground truth and predicted logit at time t, $\alpha_t \in \{\alpha_1, \alpha_2, ..., \alpha_L\}$ is the weight applied to the loss at each time step, and K and L are the number of classes and the length of the clip, respectively. We pay attention to the fact that the conventional approach has been unknowingly setting the weight α_t uniformly for every time-step in the clip, resulting in $\alpha_t = 1/L$. However, we show that setting weights uniformly like this, equivalent to using the joint prediction of all time steps, hinders the RNN from being effectively trained as it does not resemble the inference stage.

Our goal is to alleviate the problem caused by the discrepancy between the training and inference stages. To do so, we first pay careful attention to what is happening in the inference stage where RNN's temporal capacity plays an important role. During the inference stage, temporal information can accumulate as much as the length of the whole video. Although sequence-to-sequence modeling, in theory, can have an unlimited continuous context length, it is known to have a limited temporal capacity [30], the maximum amount of information in which the RNN can retain. For the task of handling streaming videos, as the videos are long, most predictions are made utilizing the information within its limited capacity. This means that RNN uses the temporal information within RNN's temporal capacity for most predictions during inference.

Selected Weights. From the above reasons, we choose to apply the weights that enable the RNN to utilize its maximum temporal capacity. The weight α_t is set as:

$$\alpha_t = \begin{cases} 1 & \text{if } t = L \\ 0 & \text{Otherwise} \end{cases}, \tag{2}$$

which is a step function that equals putting masks on all the loss except for the loss at the very last time step in the clip. Applying weights like this during training allows us to 1) disregard the dominant predictions made by using an uninformative hidden state and 2) train the RNN using predictions made by the RNN's sufficient temporal capacity, both of which better reflect the inference stage. As long as the length of the clip L is reasonable enough to exploit RNN's full potential, the prediction made at the end of the clip is more representative of what is actually happening in the inference stage. The choice of L, the length of the clip which directly affects the receptive field of RNN during training, is further experimented with in Section 5.2.

While the concept is very simple, its impact on the performance is substantial. By addressing the training-inference discrepancy, the model is able to learn more effectively. Section 5.3 provides further experiments on various weight choices that support the choice of our weights.

4. Experimental Setup

4.1. Dataset

Similar to prior works on online action detection, we evaluate MiniROAD on two publicly-available datasets: THUMOS'14 and TVSeries. In addition, we use a large FineAction dataset for performance evaluation.

THUMOS'14 [19] comprises over 20 hours of sports video, e.g. baseball pitch, each annotated with 20 actions. Following the prior works [3, 43], we train on the validation set (200 untrimmed videos) and evaluate on the test set (213 untrimmed videos).

TVSeries [7] consists of 27 episodes of six popular TV shows, spanning 16 hours. It is annotated with 30 real-life everyday actions like opening doors, running, and drinking. It proposes unique challenges as it has a diverse viewpoint, heavy occlusion, and a large portion of background frames.

FineAction [24] is a large dataset introduced for Temporal Action Detection with more detailed activity descriptions with varying difficulties. It consists of 16,732 untrimmed videos spanning 705 video hours with 103,324 temporal instances of 106 action categories.

4.2. Setting

Feature Extraction. As the selection of a feature extractor is an important factor that directly affects the performance, we follow the experimental settings of the state-of-the-art methods [43, 3, 40]. For THUMOS and TVSeries, we first sample the videos into 24 FPS and feed non-overlapping snippets to the feature extractor. The snippet

size is set to 6 and TSN [34] is adopted as the feature extractor. We employ implementation and checkpoints from MMACTION [6] and experiment with feature extractors pretrained on Kinetics-400 [2] and Activitynet1.3 [17]. For FineAction Dataset, we use the officially available features extracted using a Two-Stream architecture [32]. This architecture feeds non-overlapping snippets of 16 RGB or optical flow frames to two separate I3D [2] models that were pretrained with Kinetics. Since the officially available feature is too condensed (16 frames being converted into one feature) and causes overlooking of the fine-grained action instances, we linearly interpolate the temporal resolution of the feature sequence by a factor of four. For all experiments, the visual and motion features are concatenated along the channel dimension before feeding into the model.

Implementation Details. To substantiate our claim that mitigating the training-inference discrepancy leads to more effective training of RNN models, we adopt a minimalist approach in designing our model. Our model comprises an embedding layer, a GRU, and a classification layer. Despite its simplicity, our model achieves comparable or even better performance to the state-of-the-art methods thanks to maximizing RNN's potential with our proposed methodology, while its lightweight property makes it highly efficient.

Implementation details can be found in the officially available code. Experiments are conducted using Nvidia RTX3090 Graphics cards.

4.3. Evaluation Metrics

To evaluate the performance of our methodology, we follow prior works and report per-frame mean Average Precision (mAP) on the THUMOS'14 dataset and per-frame mean calibrated Average Precision (mcAP) on the TVSeries dataset. Per-frame mAP is adopted for the FineAction Dataset as well. For measuring mAP, we collect all the classification scores for each frame and calculate the precision and recall based on the sorted results. We then calculate the Average Precision for each class using the precision and recall values and average the results over all classes to obtain the mAP. The mcAP, introduced by [7] with the TVSeries dataset, was proposed to address the imbalance between positive and negative samples. Calibrated average precision (cAP) is calculated using the formula

$$cAP = \frac{\sum_i cPrec(i) \cdot I(i)}{\sum_i I(i)}, \tag{3}$$

where $I(i)$ is 1 if the frame i is a true positive and *cPrec* is calculated by

$$cPrec = \frac{w \cdot TP}{w \cdot TP + FP}. \tag{4}$$

where w is the ratio between negative and positive samples.

Method	Backbone	Parameters	GFLOPs	mAP (%)
TRN[39]		402.9M	1.46	62.1
OadTR[36]		75.8M	2.54	65.2
LSTR[40]	TSN	58.0M	7.53	69.5
GateHUB[3]	(Kinetics)	45.2M	6.98	70.7
TeSTra[43]		58.9M	4.37	71.2
MiniROAD (Ours)		**15.8M**	**0.0158**	**71.8**

Table 1: Comparison to other recent OAD methods in terms of parameter count, GFLOPs, and mAP on THUMOS'14 dataset using the Kinetics pretrained feature extractor.

5. Results and Analysis

5.1. Comparisons with State of the Art

THUMOS'14. Table 1 presents a comparison of MiniROAD with recent methods on THUMOS'14, using features extracted by the Kinetics-pretrained TSN backbone. We measured the parameter count and GFLOPs of TeSTra and ours by ourselves, and the other numbers are from [3]. We can see that MiniROAD outperforms state-of-the-art methods on THUMOS'14 by 0.6 % in mAP while requiring only 0.4 % GFLOPs and 27% parameters. There is a significant gain in efficiency thanks to the RNN architecture. RNN's advantage in efficiency can be also seen by TRN. Despite its large number of parameters, TRN requires the least number of computations compared to the recent works as recent works employ transformer architecture that needs a lot more computation overhead for calculating attention. These large gains in efficiency suggest that RNNs have a significant advantage in online settings, specifically for those that require real-time processing. To validate the generalization of our method, we also evaluate the same dataset using the feature extractor pretrained with Activitynet. The results are shown in Table 2a and our method achieves the best results in this setting as well.

TVSeries. Table 2b shows the comparison of our method with the other methods using the features extracted by Kinetics and Activitynet pretrained features. Our method is able to achieve comparable or better performance to the state-of-the-art in this dataset as well.

FineAction. Table 3 compares the performance of our approach and recent methods on the FineAction Dataset. As we are the first to evaluate this dataset for the OAD task, we reproduced all the results in Table 3 using the publicly available code. Our model outperforms the recent methods, achieving a 17% performance gain over the next best-performing method. This dataset is much larger and contains activities of varying difficulties, in contrast to previously tested datasets that had limited action instances of similar categories. Thus, it provides a more realistic setting and can serve as a valuable benchmark for future OAD research.

Method	mAP (%)	
	Anet	Kinetics
RED [14]	45.3	-
IDN [12]	50.0	60.3
FATS [21]	51.6	59.0
TRN [39]	47.2	62.1
LAP [29]	53.3	-
TFN [13]	55.7	-
OadTR [36]	58.3	65.2
LSTR [40]	65.3	69.5
GateHUB [3]	69.1	70.7
TeSTra [43]	68.2	71.2
Ours	**69.3**	**71.8**

(a)

Method	mcAP (%)	
	Anet	Kinetics
RED[14]	79.2	-
FATS[21]	81.7	84.6
IDN[12]	84.7	86.1
TRN[39]	83.7	86.2
TFN[13]	85.0	-
LAP[29]	85.3	-
PKD[42]	-	86.4
OadTR[36]	85.4	87.2
LSTR[40]	88.1	89.1
GateHUB[3]	88.4	**89.6**
Ours	**88.5**	**89.6**

(b)

Table 2: Comparison to other OAD methods on (a) THUMOS'14 [19] dataset and (b) TVSeries dataset [7]. Results using the backbone pre-trained using Activitynet [17] and Kinetics [2] are shown.

Method	mAP (%)
OadTR[36]	31.8
LSTR[40]	31.6
TeSTra[43]	31.2
Ours	**37.1**

Table 3: Comparison to other OAD methods on FineAction dataset [24].

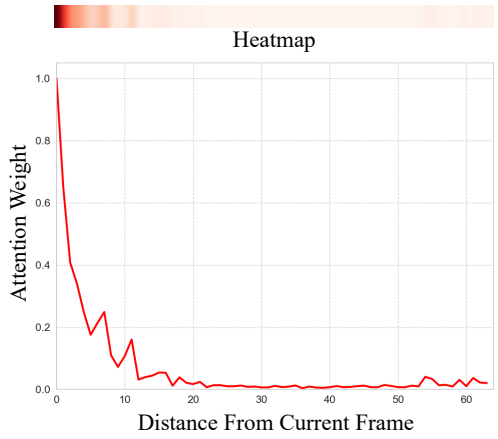

Figure 3: Attention heatmap (top) and its specific values (bottom) of the transformer encoders in OadTR [36]. For the heatmap calculation, all the attention values in the encoder (of every layer and head) are collected for the entire test set input which are then averaged out over the heads. The above shows the collected values after normalization.

5.2. Why is RNN adequate for the task?

The prevailing reasoning behind the lower performance of RNNs is that they struggle with learning long-range dependencies. This is often attributed to training difficulties such as gradient vanishing that can arise when processing prolonged sequences. Therefore, recent works have been using transformers for OAD as they are particularly proficient at capturing long-term dependencies. However, it is worth asking how much temporal information is actually needed for OAD.

What do Transformers Actually See? Inspired by the recent success of transformer models in OAD, we analyzed the attention weights within the transformer multi-head attention to gain insights into the model's decision-making process. The visualization results are presented in Figure 3. As evident from the heat map and the graph, the transformer model predominantly focuses on the recent and neighboring frames, indicating the recent input's dominant influence on the prediction. Surprisingly, despite the transformer's intended capability of capturing long-term dependencies, the model mainly relies on the recent frames for making the current frame prediction. Our analysis reinforces the argument that RNN's inductive bias, which prioritizes the current input while retaining meaningful temporal information from the past, is well suited for the task.

Optimal Length of Temporal Information. Figure 4 shows the performance of our model with varying clip lengths using the proposed methodology. As our methodology trains the model with the loss computed at the end of the clip, the graph shows the trade-off between providing more temporal information and ignoring the supervision of the earlier steps. The figure shows that a saturation point is reached at 32 seconds and providing more information than 32 seconds does not improve the performance. To the commonly held belief that the RNN does not benefit from prolonged sequence length, there exists a model's maximum capacity. It is important to note, however, that RNN's capacity is already enough to perform well on the OAD task, validated by the promising performances shown in Section 5.1. A slight drop in performance after 32 seconds is a result of RNN not being able to use temporal information beyond a certain point and much supervision lost during training.

5.3. What is the optimal weight?

Our work shows that the uniform weights put on computed loss at each time-step hinders the effective training of the RNNs. To make the most use of the finding, we experimented with varying weight choices α_t in (1) at each time step. Different choices of weights including *uniform*, *linearly increasing*, *exponentially increasing*, *learnable*, and *step function*, are experimented. The results are shown in Table 4a. The results indicate that assigning *uniform*

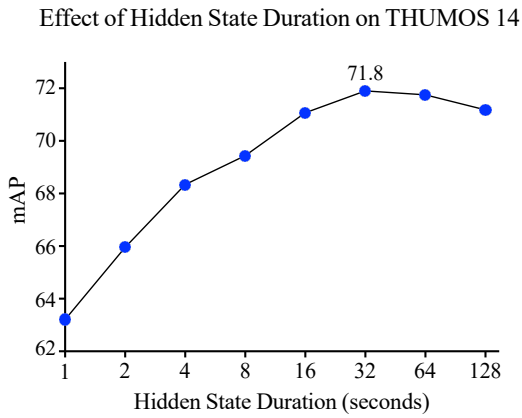

Figure 4: Effect of varying the length of the clip L during training.

weights to the loss computed at each time-step results in the worst performance, which is consistent with the findings discussed in the paper that this conventional approach impedes the effective training of the model. Applying *linearly increasing* weights performs better than assigning uniform weights because it places less emphasis on early-stage predictions. The next best performing choice is *exponentially increasing* weights, which heavily emphasizes the loss computed at later stages. The next best performing weight choice is *learnable* weights. For this experimental setup, learnable embeddings of size L (length of the clip) were trained. When computing the loss, we apply softmax activation to the learnable weights, which are then multiplied by their corresponding loss at their time steps. This approach shows even better performance than exponentially increasing weights. The best-performing choice of weight is *step-function* which uses only the loss computed at the end. This results in the best performance as the loss associated with predictions using the informative hidden state best resemble the dominant predictions made at the inference stage.

5.4. Effectiveness of the method

Table 4b shows the result of our model with and without the proposed non-uniform weights. We can observe that there is a 5 % mAP increase by using our proposed methodology. In addition, we also experiment with our proposed methodology using the transformer architecture. For the experiment, we adopt the transformer encoder from OadTR and set the encoding layers to 3 and the number of heads to 8 which is the default setting in the publicly available code. We observe that the performance increases from 63.0 to 64.9 % mAP with a 1.9 % mAP increase using our proposed methodology.

Weight Type	mAP (%)
Uniform	66.4
Linear	68.6
Exponential	69.8
Learnable	70.8
Step-Function	**71.8**

(a)

	Method	
	Transformer	MiniROAD
Uniform Weight	63.0	66.4
Proposed	**64.9**	**71.8**

(b)

Table 4: (a) Result of applying different weights α_t in (1). Otherwise stated, Step-Function is the default choice in our paper which only uses the loss computed at the end of the clip. (b) Result of applying the proposed weights on the transformer-based model and on our model.

Method	OF. Comp.	RGB Feat.	OF. Feat.	Model FPS	Overall FPS	mAP (%)
TRN				143	20.5	62.1
OadTR				145	20.5	65.2
LSTR	28.8	383	219	187	21.2	69.5
TeSTra				169	20.9	71.2
Ours				**37300**	**23.8**	**71.8**

Table 5: Runtime Comparison of OAD methods on THUMOS'14 dataset with flow features extracted using TV-L1 [41] algorithm. *OF. Comp.*, *RGB Feat.*, and *OF. Feat.* are the speed of extracting optical flows, rgb features, and optical flow features respectively, measured in FPS.

5.5. Runtime

Efficiency is a critical aspect of the online nature of the OAD task. To assess efficiency, we compare MiniROAD's runtime with recent methods in frames per second (FPS) using videos from the THUMOS'14 dataset on a system with a single RTX3090 GPU. The results in Table 5 demonstrate that our model's lightweight RNN architecture is very efficient, running about 200 times faster than previous methods. However, it is worth noting that the model's speed is not the final speed, as the input of the OAD model requires the extraction of visual and motion features. As illustrated in Table 5, computing optical flow accounts for the majority of the computation time. The end-to-end speed, which encompasses the speed after computing optical flows, extracting visual and motion features, and passing through the model, is reported as *Overall FPS* in the table. Although extracting optical flow acts as a bottleneck, the efficiency of our model yields 2.9 gains in FPS overall. These gains in overall speed are further improved when a faster optical flow computing algorithm is used. To demonstrate the efficacy of this approach, we also test using the NVIDIA Optical Flow SDK[1] (NVOFA) for faster optical flow computa-

[1]https://developer.nvidia.com/opticalflow-sdk

Method	Pre-Train	mAP@τ_o 0.25	0.5	0.75	1	1.25	1.5	1.75	2	Average
RED [14]		45.3	42.1	39.6	37.5	35.8	34.4	33.2	32.1	37.5
TRN [39]		45.1	42.4	40.7	39.1	37.7	36.4	35.3	34.3	38.9
TTM [35]		45.9	43.7	42.4	41.0	39.9	39.4	37.9	37.3	40.9
LAP-Net [29]		49.0	47.4	45.3	43.2	41.3	39.7	38.3	37.0	42.6
OadTR [36]	ANet1.3	50.2	49.3	48.1	46.8	45.3	43.9	42.4	41.1	45.9
LSTR [40]		-	-	-	-	-	-	-	-	50.1
GateHUB [3]		-	-	-	-	-	-	-	-	54.2
TeSTra [43]		64.7	61.8	58.7	55.7	53.2	51.1	49.2	**47.8**	55.3
Ours		**65.4**	**63.3**	**60.5**	**57.4**	**54.8**	**51.9**	**49.7**	47.7	**56.3**
TTM [35]		46.8	45.5	44.6	43.6	41.9	41.1	40.4	38.7	42.8
LSTR [36]		60.4	58.6	56.0	53.3	50.9	48.9	47.1	45.7	52.6
OadTR [36]	K400	59.8	58.5	56.6	54.6	52.6	50.5	48.6	46.8	53.5
TeSTra [43]		66.2	63.5	60.5	57.4	54.8	52.6	50.5	48.9	56.8
Ours		**68.5**	**66.3**	**63.5**	**60.7**	**57.9**	**55.6**	**53.5**	**51.4**	**59.7**

Table 6: Action Anticipation Results on THUMOS'14 dataset with feature extractor pretrained with Activitynet (top) and Kinetics (bottom).

Method	OF. Comp.	RGB Feat.	OF. Feat.	Model FPS	Overall FPS	mAP (%)
TRN†				143	66.0	61.6
OadTR†				145	66.3	63.3
LSTR†	1K	383	219	187	73.8	67.1
TeSTra				169	70.9	67.3
Ours				**37300**	**122**	**68.4**

Table 7: Comparison to other OAD methods on THUMOS'14 dataset with flow features extracted using NVIDIA OPTICAL FLOW. † was reproduced by us. All the values indicate the speed measured in FPS.

tion, which runs at around 1K FPS. By utilizing the faster flow algorithm, our model runs at 122 FPS which is 1.72 times faster than the state-of-the-art model. Although there is a degradation in the final performance compared to the traditionally used flow algorithm, this highlights the availability of an alternative approach that can deliver faster end-to-end speed while balancing performance and speed trade-offs. The significant increase in end-to-end speed when using the faster optical flow algorithm indicates that additional increases can be gained with RGB-only inputs, leaving open opportunities for future work.

5.6. Action Anticipation

As our methodology is also applicable to other online tasks, we extend our method to Action Anticipation, which is a task of anticipating upcoming actions using the histories. Instead of predicting the current time frame, the model is trained to predict the next upcoming actions every 0.25 seconds up to 2 seconds. Table 6 shows the results of predictions at each of that time-step. It shows that our method performs favorably to the other methods with 1 % mAP gain using activitynet pretrained backbone and 2.9% mAP gain using kinetics pretrained backbone over the previous state-of-the-art. While our methodology was initially designed for OAD, its applicability extends beyond OAD and can be used for other similar tasks.

6. Conclusion

We revisit RNN for online action detection and show its potential in dealing with the task. We identify the training-inference discrepancy as a primary cause of the lower performance of RNNs and address the issue by introducing non-uniform weights to the loss computed at each time-steps. We demonstrate the effectiveness of our proposed method using MiniROAD, a minimal RNN-based OAD model. MiniROAD achieves equal or better accuracy than the existing best method on challenging datasets, with a significant increase in efficiency. With the faster NVOFA, MiniROAD runs at an impressive 122 FPS.

We note that recent OAD methods rely heavily on optical flows, which impede end-to-end speed. While faster optical flow algorithms like NVOFA can mitigate this issue, flow-free OAD remains an open yet challenging problem. By highlighting the suitability of RNNs for processing streaming videos, we hope to pave the way for further advancements and shed light on developing more efficient models.

Acknowledgements

This work was supported by Samsung Electronics Co., Ltd. (Mobile eXperience Business), Yonsei University Artificial Intelligence Graduate School Program under Grant 2020-0-01361, and by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00124, Development of Artificial Intelligence Technology for Self-Improving Competency-Aware Learning Capabilities). The work of Joungbin was in part supported by the Hyundai Motor Chung Mong-Koo Foundation.

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *IEEE International Conference on Computer Vision (ICCV)*, pages 6836–6846, 2021. 3

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6299–6308, 2017. 5, 6

[3] Junwen Chen, Gaurav Mittal, Ye Yu, Yu Kong, and Mei Chen. Gatehub: Gated history unit with background suppression for online action detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3, 4, 5, 6, 8

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 2

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 1, 3

[6] MMAction2 Contributors. Openmmlab's next generation video understanding toolbox and benchmark. https://github.com/open-mmlab/mmaction2, 2020. 5

[7] Roeland De Geest, Efstratios Gavves, Amir Ghodrati, Zhenyang Li, Cees Snoek, and Tinne Tuytelaars. Online action detection. In *European Conference on Computer Vision (ECCV)*, pages 269–284. Springer, 2016. 1, 2, 4, 5, 6

[8] Roeland De Geest and Tinne Tuytelaars. Modeling temporal structure with lstm for online action detection. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1549–1557. IEEE, 2018. 2

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1, 3

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 3

[11] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990. 1

[12] Hyunjun Eun, Jinyoung Moon, Jongyoul Park, Chanho Jung, and Changick Kim. Learning to discriminate information for online action detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 6

[13] Hyunjun Eun, Jinyoung Moon, Jongyoul Park, Chanho Jung, and Changick Kim. Temporal filtering networks for online action detection. *Pattern Recognition*, 111:107695, 2021. 6

[14] Jiyang Gao, Zhenheng Yang, and Ram Nevatia. Red: Reinforced encoder-decoder networks for action anticipation. *arXiv preprint arXiv:1707.04818*, 2017. 2, 6, 8

[15] Mingfei Gao, Yingbo Zhou, Ran Xu, Richard Socher, and Caiming Xiong. Woad: Weakly supervised online action detection in untrimmed videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1915–1923, 2021. 2

[16] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021. 3

[17] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–970. IEEE, 2015. 5, 6

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 1, 2, 3

[19] Haroon Idrees, Amir R Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The thumos challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding(CVIU)*, 155:1–23, 2017. 4, 6

[20] Jinkyu Kim, Teruhisa Misu, Yi-Ting Chen, Ashish Tawari, and John Canny. Grounding human-to-vehicle advice for self-driving vehicles. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10591–10599, 2019. 1

[21] Young Hwi Kim, Seonghyeon Nam, and Seon Joo Kim. Temporally smooth online action detection using cycle-consistent future anticipation. *Pattern Recognition*, 116:107954, 2021. 2, 6

[22] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398, 2021. 3

[23] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015. 1, 2

[24] Yi Liu, Limin Wang, Yali Wang, Xiao Ma, and Yu Qiao. Fineaction: A fine-grained video dataset for temporal action localization. *IEEE Transactions on Image Processing*, 2022. 4, 6

[25] Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. *arXiv preprint arXiv:2303.06349*, 2023. 3

[26] Guansong Pang, Cheng Yan, Chunhua Shen, Anton van den Hengel, and Xiao Bai. Self-trained deep ordinal regression for end-to-end video anomaly detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12173–12182, 2020. 1

[27] Hyunjong Park, Jongyoun Noh, and Bumsub Ham. Learning memory-guided normality for anomaly detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14372–14381, 2020. 1

[28] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, et al. Rwkv:

Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023. 3

[29] Sanqing Qu, Guang Chen, Dan Xu, Jinhu Dong, Fan Lu, and Alois Knoll. Lap-net: Adaptive features sampling via learning action progression for online action detection. *arXiv preprint arXiv:2011.07915*, 2020. 6, 8

[30] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997. 1, 4

[31] Tianmin Shu, Dan Xie, Brandon Rothrock, Sinisa Todorovic, and Song Chun Zhu. Joint inference of groups, events and human roles in aerial videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4576–4584, 2015. 1

[32] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *Conference on Neural Information Processing Systems (NeurIPS)*, 27, 2014. 5

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Conference on Neural Information Processing Systems (NeurIPS)*, 30, 2017. 1, 3

[34] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision (ECCV)*, pages 20–36. Springer, 2016. 5

[35] Wen Wang, Xiaojiang Peng, Yanzhou Su, Yu Qiao, and Jian Cheng. Ttpp: Temporal transformer with progressive prediction for efficient action anticipation. *Neurocomputing*, 438:270–279, 2021. 8

[36] Xiang Wang, Shiwei Zhang, Zhiwu Qing, Yuanjie Shao, Zhengrong Zuo, Changxin Gao, and Nong Sang. Oadtr: Online action detection with transformers. *IEEE International Conference on Computer Vision (ICCV)*, 2021. 2, 5, 6, 8

[37] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. 2

[38] Chao-Yuan Wu, Yanghao Li, Karttikeya Mangalam, Haoqi Fan, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Memvit: Memory-augmented multiscale vision transformer for efficient long-term video recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13587–13597, 2022. 3

[39] Mingze Xu, Mingfei Gao, Yi-Ting Chen, Larry S. Davis, and David J. Crandall. Temporal recurrent networks for online action detection. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 2, 5, 6, 8

[40] Mingze Xu, Yuanjun Xiong, Hao Chen, Xinyu Li, Wei Xia, Zhuowen Tu, and Stefano Soatto. Long short-term transformer for online action detection. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 2, 4, 5, 6, 8

[41] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Pattern Recognition*, pages 214–223. Springer, 2007. 7

[42] Peisen Zhao, Lingxi Xie, Ya Zhang, Yanfeng Wang, and Qi Tian. Privileged knowledge distillation for online action detection. *arXiv preprint arXiv:2011.09158*, 2020. 6

[43] Yue Zhao and Philipp Krähenbühl. Real-time online video detection with temporal smoothing transformers. In *European Conference on Computer Vision (ECCV)*, 2022. 3, 4, 5, 6, 8