Re^{2}TAL: Rewiring Pretrained Video Backbones for Reversible Temporal Action Localization

Chen Zhao[1] Shuming Liu[1] Karttikeya Mangalam[2] Bernard Ghanem[1]
[1]King Abdullah University of Science and Technology (KAUST), Saudi Arabia [2]UC Berkeley, US
{chen.zhao, shuming.liu, bernard.ghanem}@kaust.edu.sa mangalam@berkeley.edu

Abstract

Temporal action localization (TAL) requires long-form reasoning to predict actions of various durations and complex content. Given limited GPU memory, training TAL end to end (i.e., from videos to predictions) on long videos is a significant challenge. Most methods can only train on pre-extracted features without optimizing them for the localization problem, consequently limiting localization performance. In this work, to extend the potential in TAL networks, we propose a novel end-to-end method ***Re^{2}TAL****, which rewires pretrained video backbones for reversible TAL.* ***Re^{2}TAL*** *builds a backbone with reversible modules, where the input can be recovered from the output such that the bulky intermediate activations can be cleared from memory during training. Instead of designing one single type of reversible module, we propose a network rewiring mechanism, to* ***transform any module with a residual connection to a reversible module*** *without changing any parameters. This provides two benefits: (1) a large variety of reversible networks are easily obtained from existing and even future model designs, and (2) the reversible models require much less training effort as they reuse the pre-trained parameters of their original non-reversible versions. Re^{2}TAL, only using the RGB modality, reaches 37.01% average mAP on ActivityNet-v1.3, a new state-of-the-art record, and mAP 64.9% at tIoU=0.5 on THUMOS-14, outperforming all other RGB-only methods. Code is available at https://github.com/coolbay/Re2TAL.*

1. Introduction

Temporal Action Localization (TAL) [36,53,73] is a fundamental problem of practical importance in video understanding. It aims to bound semantic actions within start and end timestamps. Localizing such video segments is very useful for a variety of tasks such as video-language grounding [23,56], moment retrieval [9,21], video captioning [30,50]. Since video actions have a large variety of temporal durations and content, to produce high-fidelity localization, TAL approaches need to learn from a long temporal scope of the video, which contains a large number of frames. To accommodate all these frames along with their network activations in GPU memory is extremely challenging, given the current GPU memory size (*e.g.* the commodity GPU GTX1080Ti only has 11GB). Often, it is impossible to train one video sequence on a GPU without substantially downgrading the video spatial/temporal resolutions.

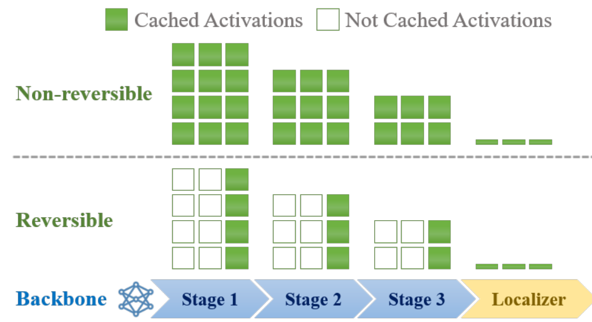

Figure 1. **Illustration of TAL network activations in training.** Top: non-reversible network stores activations of all layers in memory. Bottom: reversible network only needs to store the activations of inter-stage downsampling layers. Backbone activations dominate memory occupation, compared to Localizer.

To circumvent the GPU-memory bottleneck, most TAL methods deal with videos in two isolated steps (*e.g.* [4,67,69,71–73]). First is a snippet-level feature extraction step, which simply extracts snippet representations using a pre-trained video network (backbone) in inference mode. The backbone is usually a large neural network trained for an auxiliary task on a large dataset of trimmed video clips (*e.g.*, action recognition on Kinetics-400 [28]). The second step trains a localizer on the pre-extracted features. In this way, only the activations of the TAL head need to be stored in memory, which is tiny compared to those of the backbone (see the illustration of the activation contrast between backbone and localizer in Fig. 1). However, this two-step strategy comes at a steep price. The pre-extracted features can suffer from domain shift from the auxiliary pre-training task/data to TAL, and do not necessarily align with the representation needs of TAL. This is because they cannot be finetuned and must be used as-is in their misaligned state for TAL. A better alternative is to jointly train the backbone and localizer end to end. But as mentioned earlier, the enormous memory footprint of video activations in the backbone makes it extremely challenging. Is there a way for end-to-end training without compromising data dimensionality?

Reversible networks [20, 25, 31, 48] provide an elegant solution to drastically reduce the feature activation memory during training. Their input can be recovered from the output via a reverse computation. Therefore, the intermediate activation maps, which are used for back propagation, do not need to be cached during the forward pass (as illustrated in Fig. 1). This offers a promising approach to enable memory-efficient end-to-end TAL training, and various reversible architectures have been proposed, such as RevNet [20], and RevViT [48]. However, these works design a specific reversible architecture and train for a particular dataset. Due to their new architecture, they also need to train the networks from scratch, requiring a significant amount of compute resources.

Conversely, it would be beneficial to be able to convert existing non-reversible video backbones to reversible ones, which would **(1)** avail a large variety of architectures and **(2)** allow us to reuse the large compute resources that had already been invested in training the non-reversible video backbones. Since pre-trained video backbones are a crucial part of TAL, the ability to convert off-the-shelf backbones to reversible ones is a key to unleash their power in this task.

In this work, for end-to-end TAL, we propose a principled approach to Rewire the architectural connections of a pre-trained non-reversible backbone to make it Reversible, dubbed Re^{2}TAL. Network modules with a residual connection (res-module for short), such as a Resnet block [22] or a Transformer MLP/attention layer [14], are the most popular design recently. **Given any network composed of residual modules, we can apply our rewiring technique to convert it to a corresponding reversible network** without introducing or removing any trainable parameters. Instead of training from scratch, our reversible network can reuse the non-reversible network's parameters and only needs a small number of epochs for finetuning to reach similar performance. We summarize our contributions as follows.

(1) We propose a novel approach to construct and train reversible video backbones parsimoniously by architectural rewiring from an off-the-shelf pre-trained video backbone. This not only provides a large collection of reversible candidates, but also allows reusing the large compute resources invested in pre-training these models. We apply our rewiring technique to various kinds of representative video backbones, including transformer-based Video Swin and ConvNet-based Slowfast, and demonstrate that our reversible networks can reach the same performance of their non-reversible counterparts with only minimum finetuning effort (as low as 10 epochs compared to 300 epochs for training from scratch).

(2) We propose a novel approach for end-to-end TAL training using reversible video networks. Without sacrificing spatial/temporal resolutions or network capability, our proposed approach dramatically reduces GPU memory usage, thus enabling end-to-end training on one 11GB GPU. We demonstrate on different localizers and different backbone architectures that we significantly boost TAL performance with our end-to-end training compared to traditional feature-based approaches.

(3) With our proposed Re^{2}TAL, we use recent localizers in the literature to achieve a new state-of-the-art performance, 37.01% average mAP on ActivityNet-v1.3. We also reach the highest mAP among all methods that only use the RGB modality on THUMOS-14, 64.9% at tIoU$=0.5$, outperforming concurrent work TALLFormer [10].

2. Related Work

Reversible Networks are a family of neural models based on the reversible real-valued non-volume preserving (real NVP) transformation introduced in [12, 13]. The transformation has been originally applied widely for image generation using generative flows [24, 29], and later used for various applications, such as compression [38], denoising [43], and steganography recovery [49]. Furthermore, it has also been repurposed for memory-efficient neural network training for a variety of architectures such as ConvNets [20, 25], Masked ConvNet [57], RNNs [47], Graph Networks [31] and more recently, Vision Transformers [48]. However, each of these methods only focuses on one or several specific architectures, and trains their newly proposed reversible architectures from scratch, thereby requiring large compute resources. In this work, we propose a rewiring scheme to adapt off-the-shelf models to reversible architectures which only utilize a small amount of compute for finetuning. This allows democratizing the reach of reversible architecture to arbitrary models and dataset settings by leveraging the architecture design effort and computational cost already spent in vanilla off-the-shelf models.

Video Recognition Backbones. ConvNets have had a long and illustrious history of improving video understanding performance [7, 16–19, 26, 33, 52, 55, 61, 64, 65, 74]. Among those, Slowfast [17], which uses a slow pathway and a fast pathway to capture spatial semantics and temporal motion respectively, has been widely adopted for various tasks such as action detection [17], untrimmed video classification [44] and moment/natural language retrieval [21] for its high efficiency and efficacy. Recently, video transformers have ushered in a new life in the field of video representation learning. Leveraging long-term temporal connection via the attention mechanism, transformers have quickly gain popularity as the backbone of choice for several video recognition workloads [3, 5, 15, 32, 45, 70]. Among those, Video Swin Transformer [45] introduces an inductive bias of locality to video transformers, leading to an outstanding speed-accuracy trade-off. However, most of the models are vanilla non-reversible architectures. In this work, we unleash the potential of reversible architectures for these models such that the good properties of reversible models, *e.g.* memory efficiency, can be infused into them. In particular, we show that the pretrained Video Swin Transformer [45] and SlowFast [17] models can be rewired to be reversible, and finetuned cheaply to improve temporal action localization performance with end-to-end training.

Temporal Action Localization (TAL). Due to the conflict between large video data and the GPU memory limit, most TAL methods using deep networks are two-step methods [4, 8, 34, 36, 37, 41, 46, 51, 54, 69, 71–73]. For example, using pre-extracted features, G-TAD [69] and VSGN [73] utilize graph convolutions to model temporal relations between snippets, and ActionFormer [71] leverages transformers to capture long-range context. To mitigate the performance gap between the two-step mechanism and real end-to-end training, some methods explore post-pretraining to enhance the video feature representations for TAL [2, 66, 68]. In the meanwhile, researchers also attempt to perform real end-to-end training by reducing network/data complexity [10, 35, 39, 40, 42, 63, 66]. R-C3D [66] is the end-to-end pioneer, but it uses a shallow network C3D [60], thus restricting the performance. PBRNet [39] and ASFD [35] downscale the frame resolution to 96×96. DaoTAD [63] makes RGB-only enough by using end-to-end training with various data augmentations. TALLFormer [10] proposes to use a feature bank strategy and only updates a portion of features during end-to-end training. Compared to these approaches, our method doesn't sacrifice any data dimensionality or data samples in training, and it supports very deep networks. We significantly reduce memory consumption while preserving the full data fidelity. Moreover, our work is complementary to these methods, and can also be used jointly to further reduce memory cost.

3. Method: Re^2 Temporal Action Localization

3.1. TAL Formulation and Architecture

Temporal action localization (TAL) predicts the timestamps of actions from a video sequence, which is formulated as follows. Given a video sequence V of T frames $\{I_t \in \mathbb{R}^{3\times H\times W}\}_{t=1}^{T}$, TAL predicts a set m of actions $\Phi = \{\phi_m = (t_{m,s}, t_{m,e}, c_m, s_m)\}_{m=1}^{M}$, where $t_{m,s}$ and $t_{m,e}$ are action start and end time respectively, c_m is action label, and s_m is prediction confidence. To achieve this, the following two steps are required.

In the first step, the videos V are encoded as N features vectors $\{\mathbf{x}_n \in \mathbb{R}^C\}_{n=1}^{N}$ via a backbone. This backbone aggregates spatial information within video frames, as well as temporal information across frames. It is usually designed for an auxiliary task such as action recognition. Popular backbones can be categorized into Resnet-based architectures, such as I3D [7], R2+1D [62], SlowFast [17], and Transformer based architectures, such as ViViT [3], Video Swin Transformers [45].

In the second step, a localizer, uses a 'neck' to further aggregate the video features $\{\mathbf{x}_n\}_{n=1}^{N}$ in the temporal domain, and a 'head', to make predictions of action boundaries, *i.e.*, start and end timestamps $(t_{m,s}, t_{m,e})$ and categories. The neck contains layers of networks, *e.g.*, 1D convolutional networks in BMN [36], Graph networks in G-TAD [69] and VSGN [73], and Transformers in RTD-Net [59] and ActionFormer [71].

As mentioned in Sec. 1, an optimal way to train the entire TAL network is to jointly train the backbone and the localizer end to end. However, it is substantially challenging to fit all activation maps of the long video sequence into limited GPU memory. In the following sections, we provide end-to-end TAL solution Re^{2}TAL.

3.2. Rewire for Reversibility, and Reuse

Let's first analyze what in the GPU memory precludes end-to-end training of TAL. As mentioned in Sec. 1, the activation maps in the backbone are the major occupant in the memory. Concretely, given a batch of a video sequence of T frames, each with resolution $S\times S$, to train them on a backbone of L layers each with C channels, the training memory complexity is $\mathcal{O}(LCTS^2)$. Downgrading any data dimensionality or the backbone capacity can reduce the memory complexity, but at the same time, may harm the prediction accuracy. We aim to reduce memory consumption without sacrificing the data dimensionality or the model capacity.

Actually, the intermediate activations are stored in memory during training for the purpose of gradient computation in back propagation. If we can reconstruct them during back propagation, then there is no need for their memory occupation. Reversible networks [20, 25, 48, 49] is a superb solution for reconstructing the input from output, and various reversible architectures have been proposed recently (e.g., Revnet [20], RevViT [48]). However, the performance of TAL heavily depends on the backbone that is pre-trained on one or even multiple large-scale datasets. For example, the popular I3D [7] backbone is trained on Kinetics-400 [28] with its parameters initialized from Resnet [22] trained on ImageNet [11]. If we make up a new reversible network for TAL as is the case with previous reversible networks (e.g., [20, 25]), we need to go through all the expensive and time-consuming training stages to reach an equivalently

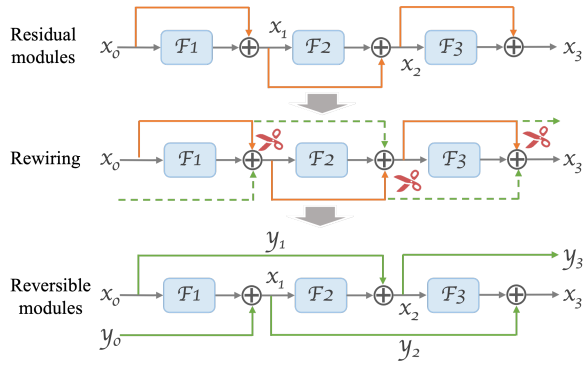

Figure 2. **Illustration of the proposed rewiring process.** In this example, 3 consecutive residual modules in the same stage are rewired into 3 reversible modules. After rewiring, each residual connection skips one more $\mathcal{F}$ block ahead. Note that beyond this example, our rewiring can process any number of such modules.

good initialization for TAL.

Instead of proposing a new reversible architecture, we propose a principled approach to rewire a pre-trained non-reversible backbone to make it reversible. Given *any* module with a residual connection, we can convert it into a reversible module while preserving the same set of parameters. This technique has two advantages. First, we obtain a large and growing collection of various reversible architectures out of the box, by rewiring the existing networks and even future networks of higher performance. Second, we can reuse their pre-trained weights to initialize our reversible networks, eliminating the tremendous effort to train from scratch. In the following, we will describe our rewiring for reversibility (**Re^2**) technique in detail.

Rewiring. Residual modules, *i.e.*, modules with a residual connection, are the most commonly adopted design in concurrent neural network architectures, such as Resnet [22] and Transformers [14]. Given any residual modules, we can rewire them into reversible modules. Fig. 2 illustrates this process with an example of 3 consecutive residual modules that are in the same stage, *i.e.*, there are no downsampling operations in between. Each of the original residual modules contains a block $\mathcal{F}_i$ (blue boxes) where $i = 1, 2, 3, \ldots$ and a residual connection (orange arrow). The blocks $\mathcal{F}_i$ have the same input and output dimensions, and can contain any computations. For example, in Resnet, the building block consisting of convolution, batch normalization, and ReLU can be one $\mathcal{F}_i$ block (*e.g*. Fig. 3 (a)). In Transformers, if $\mathcal{F}_i$ represents an attention layer, then $\mathcal{F}_{i+1}$ is an MLP layer (*e.g*. Fig. 3 (b) and (c)). The residual connection in each module only skips one $\mathcal{F}_i$ block in the same module.

To rewire, we simply let the residual connection skip the next block $\mathcal{F}_{i+1}$ as well as the current one $\mathcal{F}_i$. To be more specific, we keep the starting point of each residual connection, but make it end after two blocks, as shown in the second row of Fig. 2. As a result, two consecutive residual

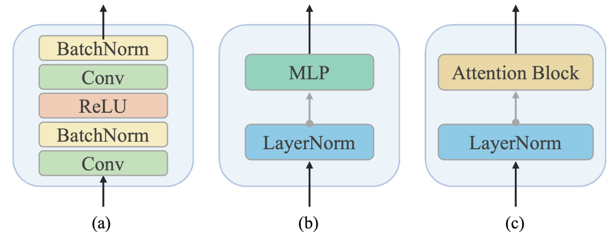

Figure 3. **Examples of $\mathcal{F}$ blocks.** (a) A resnet basic block. (b) A transformer MLP layer. (c) A transformer attention layer.

connections become overlapped, and there are always two pathways of activations (gray and green in Fig. 2) throughout the modules, as shown in the third row of Fig. 2.

To prepare a second pathway of input to the first module, we duplicate the input x_0 to obtain y_0. To combine the results x_3 and y_3 of the two pathways, we simply average them at the end of the modules. This design guarantees that the dimensions of the input and output of every block $\mathcal{F}_i$ are identical before and after the rewiring. Therefore, the structure of $\mathcal{F}_i$ stays exactly the same.

Reversibility. In the following, we mathematically formulate the rewired modules to show their reversibility. To be concise, we use the first two modules as an example. Given the input activations x_0 and y_0, and the blocks $\mathcal{F}_1$ and $\mathcal{F}_2$, the output of the two blocks are x_1, y_1 and x_2, y_2 respectively, computed as follows

$$\begin{cases} y_1 = x_0 \\ x_1 = \mathcal{F}_1(x_0) + y_0, \end{cases} \Rightarrow \begin{cases} y_2 = x_1 \\ x_2 = \mathcal{F}_2(x_1) + y_1. \end{cases} \tag{1}$$

The above equations are reversible, which means that we can recover the input x_1, y_1 from x_2, y_2, and then x_0, y_0 from x_1, y_1. The reverse computation is formulated in the following equation

$$\begin{cases} x_0 = y_1 \\ y_0 = x_1 - \mathcal{F}_1(x_0), \end{cases} \Leftarrow \begin{cases} x_1 = y_2 \\ y_1 = x_2 - \mathcal{F}_2(x_1). \end{cases} \tag{2}$$

A network with more modules follows the same strategy as in Eq. 1 and Eq. 2. When we stack multiple consecutive reversible modules as one network (as the example in Fig. 2), the entire network is reversible. In this case, we can start from the last module and sequentially reconstruct the input of each module with Eq. 2.

Reuse. As the structure and parameter dimensions of the $\mathcal{F}_i$ block in our reversible modules after rewiring stay exactly the same as the $\mathcal{F}_i$ block in the corresponding residual module, we can directly reuse the pre-trained parameters in the residual modules to initialize our reversible modules.

Our rewiring mechanism provides a large collection of reversible candidates. We can easily convert a network into a reversible one without designing a new architecture from scratch. Not only can we use the existing architectures, but we can also benefit from future even better models to obtain better reversible models. The reuse strategy

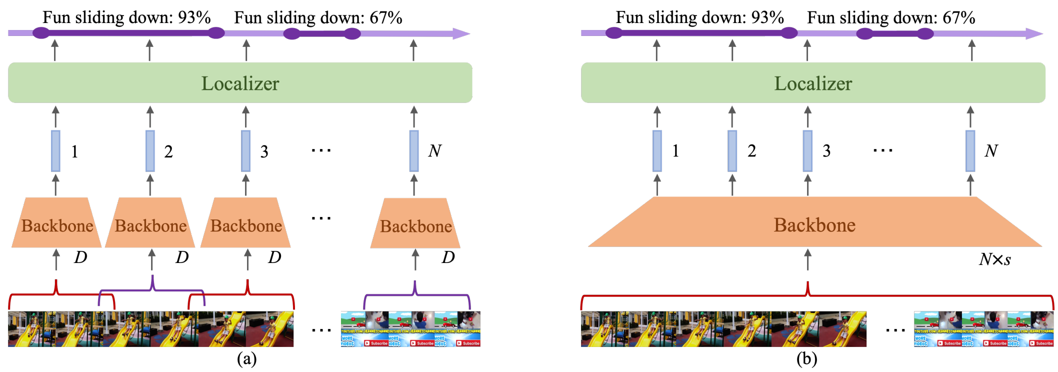

Figure 4. **Comparison of different input arrangements for a TAL network. (a) Snippets input:** the backbone processes many video snippets (a short clip of frames), often overlapped. **(b) Frames input:** the backbone takes all the frames in the video as one single input.

allows taking advantage of the large computing resources invested in those pre-trained models. We don't need to retrain the reversible network from scratch. Instead, we can train the reversible backbone with minimum effort (as low as 10 epochs compared to 300 epochs from scratch) while reaching the same performance as its residual counterpart.

3.3. Reversible Temporal Action Localization

Considering that the backbone is the heaviest part in the TAL network in terms of memory usage, as illustrated in Fig. 1, we target the backbone and apply our Re^2 technique described above to rewire it to obtain a reversible network.

A backbone network, *e.g.*, Video Swin Transformer [45], Slowfast [17], is usually comprised of several stages. Within each stage, the activation sizes stay the same. We convert all the modules into reversible ones following the rewiring method proposed in Sec. 3.2. During training, we can clear all the input and intermediate activations from GPU memory, and only store the final output of the stage during the forward pass (as shown in Fig. 1). In the back propagation, we re-compute all the activations based on the reverse process as in Eq. 2. As [48] pointed out, this re-computation doesn't incur too much more computational time since we can parallelize the process to make use of the spare computation of GPUs.

Across stages, there are downsampling layers, which reduce the activation sizes. We leave the downsampling layers as they are and cache the activations inside to enable back propagation. Considering that there are only several downsampling layers, the memory occupation of these activations is acceptable.

3.4. End-to-End TAL Training

For end-to-end training with our Re^{2}TAL, we just need to do the following: find a well-performing video backbone, rewire it into a reversible one, load the parameters from the original backbone to the reversible one and finetune for several epochs on the pretraining task, and train with a localizer end to end. This reversible TAL network is significantly more efficient in memory usage, enabling end-to-end training a GPU of limited memory (*e.g.*, as small as a 11GB commodity GPU). But is it a wise choice to directly adopt the training strategies from the feature-based methods?

Input frame arrangement. Since most TAL methods are designed and experimented with the pre-extracted, they have predisposed to particular design choices, such as extracting features with overlapped snippets to arrange the input frames. But we find it not ideal for end-to-end training.

Fig. 4 (a) illustrates the framework commonly used in two-step (or feature-based) methods. To extract N feature vectors as the localizer input, N snippets need to be processed by the backbone independently. Each snippet contains a sequence of D frames. The snippets are usually overlapped with one another to maintain temporal consistency, which causes duplicate computation and extra memory occupation. Consequently, at least ND frames need to go through the backbone.

This framework works fine with the two-step method, since feature extraction is one-off effort and all the snippets can be processed sequentially to circumvent the memory issue. However, end-to-end training cannot bear the extra memory cost. Therefore, we treat the entire video sequence as one single input instead, and the backbone aggregates all frames at one time, as shown in Fig. 4 (b). This mechanism not only reduces the cost incurred by the duplicate computation, but also has the advantage of aggregating long-term temporal information, in contrast to being restricted within the snippet as with the snippet mechanism. In this way, to obtain N feature vectors, we just need to process Ns frames, where s is the overall strides in the backbone (*e.g.* s=2 in Video Swin Transformers). Considering that s is usually much smaller than D, the frame-input mechanism is more efficient in memory and computation cost.

4. Experiments

Datasets and evaluation metrics. We present our experimental results on two representative datasets ActivityNet-v1.3 (ActivityNet for short) [6] and THUMOS-

Table 1. **Advantage of end-to-end training.** End-to-end training leads to significant mAP (%) boost for TAL with either backbone.

	Re^{2}Vswin-tiny				Re^{2}Slowfast-101			
Method	0.5	0.75	0.95	Avg.	0.5	0.75	0.95	Avg.
Features	51.18	35.09	9.72	34.47	51.98	36.00	9.47	35.24
End2End	**53.24**	**37.23**	**10.49**	**36.38**	**53.63**	**37.53**	**10.67**	**36.82**

14 (THUMOS for short) [27]. **ActivityNet** is a challenging large-scale dataset, with 19994 temporally annotated untrimmed videos in 200 action categories, which are split into training, validation and testing sets by the ratio of 2:1:1. **THUMOS** contains 413 temporally annotated untrimmed videos with 20 action categories, in which 200 videos are for training and 213 videos for validation[1]. For both datasets, we use mean Average Precision (mAP) at different tIoU thresholds as the evaluation metric. On ActivityNet, we choose 10 values in the range $[0.5, 0.95]$ with a step size 0.05 as tIoU thresholds; on THUMOS, we use tIoU thresholds $\{0.3, 0.4, 0.5, 0.6, 0.7\}$; following the official evaluation practice.

Backbones and localizers of Re^{2}TAL. For backbones, we choose two representative models from the Resnet and Transformer families respectively for experimental demonstration: Video Swin Transformers (Vswin for short) [45] and Slowfast [17], both well known for their powerful video representation and memory efficiency. For Vswin, we use three variants: tiny, small, and base; for Slowfast, we use 50, 101, and 152. For localizers, we experiment with the recent temporal action localization (TAL) methods VSGN [73] and ActionFormer [71].

Implementation Details of Re^{2}TAL. We initialize all our reversible backbones with their non-reversbile counterparts, and finetune for up to 30 epochs on Kinetics-400 with Cosine Annealing learning rate policy and Adamw (for Vswin) and SGD (for Slowfast) optimizers. Actually, in our experiments, we find 10 epochs of finetuning already reaches similarly good performance. With the pre-trained reversible backbone, we train TAL on a **single** GPU, A100 with batchsize $= 1$ for Vswin and V100 with batchsize $= 2$ for Slowfast. This is only possible due to the massive GPU memory savings from the reversible backbones. For the hyper-parameters, we follow the original training recipe of VSGN [73] and ActionFormer [71] for the learning rates and epochs in the localizers. We set the learning rates of the Vswin backbones 1 magnitude lower than the localizer, and those of the Slowfast backbones 2 magnitude lower. The spatial resolution is 224×224. For ActivityNet, the number of frame input is $T = 512$, the number of feature vectors is $N = 256$; and for THUMOS, the number of frame input is $T = 1024$, the number of features is $N = 512$.

[1] The training and validation sets of THUMOS are temporally annotated videos from the validation and testing sets of UCF101 [58], respectively.

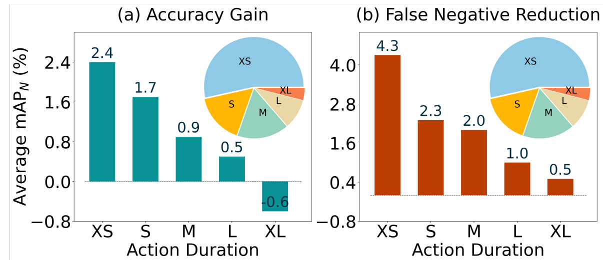

Figure 5. **Performance improvement of end-to-end TAL upon feature-based in terms of action temporal duration**.

4.1. Ablation and Analysis

In this section, we provide ablation study on ActivityNet and performance analysis to answer the following questions. (1) Why do we need end-to-end training? (2) How effective and efficient is our Re^{2}TAL? (3) What is the benefit of the proposed rewiring technique? (4) Which reversible backbone and localizer provides the best performance?

4.1.1 Why end-to-end training?

With end-to-end training, we are able to optimize the features to adapt to the TAL task and dataset such that we can achieve higher performance than the feature-based method. To verify this, we compare end-to-end training to the feature-based training on our reversible TAL. In Tab. 1, we demonstrate the comparison on two types of backbones: Re^2 Vswin-tiny and Re^2 Slowfast-101 with VSGN [73] on ActivityNet. It shows that for both backbones, using end-to-end training leads to significant performance boost, gaining almost 2% mAP.

To further diagnose what kinds of actions end-to-end training improves the most, in Fig. 5 (a), we plot the accuracy gains brought by end-to-end training for five different groups of actions based on their temporal durations (in seconds): XS: (0s, 30s], S: (30s, 60s], M: (60s, 120s], L: (120s, 180s], and XL: $>$ 180s (as suggested in [1]). We can see that the accuracy gains for different action durations are closely related to the numbers of samples in each duration category (the pie chart): short actions (XS and X) with many samples are obviously improved, though long actions (XL) with fewer samples is slightly sacrificed. Actually, short actions are a fundamental challenge in TAL [73]. End-to-end training provides an effective solution by enabling the backbone to learn feature representations to the benefit of short actions — the categories with more samples, thus preventing them fusing into the background. Short actions occupy the majority in the dataset, and their improvement leads to an overall performance boost. In Fig. 5 (b), we show that the false negative predictions are reduced by end-to-end training for all groups, more significantly for shorter ones.

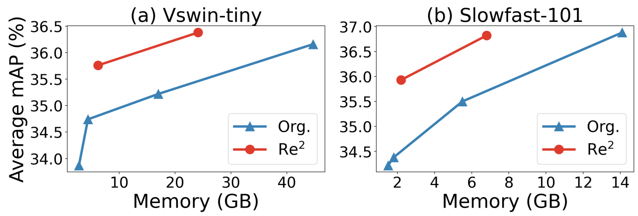

Figure 6. **Re^{2}TAL models compared with original models at different spatial and temporal downscaling ratios.** Spatial and temporal resolutions from left to right are as follows. Re^{2}TAL: (112, 512), (224, 512); Original: (56, 512), (112, 192), (224, 192), (224, 512). The localizer VSGN [73] is used.

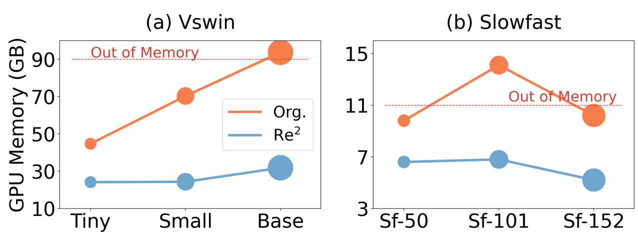

Figure 7. **GPU memory consumption v.s. network sizes.** The dot sizes represent the model sizes, and the y-axis is the GPU memory used for processing one video (batchsize=1). Our Re^{2}TAL almost keeps the memory constant when only the network depths increases (*e.g.* from Vswin-tiny to small, from Slowfast-50 to 101), whereas the original models easily go out of memory when the network becomes larger.

4.1.2 How effective and efficient is Re^{2}TAL?

An alternative way to enable end-to-end training is to reduce data resolutions [39, 63]. In Fig. 6, we downscale the input videos in the spatial or temporal dimensions to adjust the memory requirements, and compare our Re^{2}TAL models to their corresponding original models. We can see that under even smaller memory requirements, our Re^{2}TAL models achieve higher performance than the original models that relies on sacrificing video resolutions to reduce memory.

To further visualize the memory efficiency of our Re^{2}TAL compared to their non-reversible counterparts when trained end to end, we demonstrate the GPU memory consumption of the three Vswin backbones of different depths and widths and the three Slowfast backbones of different depths in Fig. 7. For Vswin backbones, we assume a GPU memory budget of 90GB, which is the size of A100. For Slowfast backbones, we assume our GPU budget is 11GB, which is the size of the commodity GPU such as GTX1080Ti. We can see that for either case, the non-reversible network will go out of memory when the model size reaches a certain level. In contrast, our Re^{2}TAL almost keeps constant memory usage when only the network depth increases (from Vswin-tiny to small, from Slowfast-50 to 101)[2]. With the low memory cost, our Re^{2}TAL enables end-to-end training with a deep Slowfast backbone on **one single** 11GB GPU. Moreover, Re^{2}TAL doesn't incur much extra training time compared to its non-reversible counterpart. The time to train one epoch is the following, Vswin-tiny (114 mins) *vs.* Re^{2}Vswin-tiny (135 mins); Slowfast-50 (147 mins) *vs.* Re^{2}Slowfast-50 (158 mins).

Table 2. **Comparison of feature representations between the Re^{2}TAL and original models**, in terms of average mAP (%) on the dataset ActivityNet. Vw: Vswin; Slowf: Slowfast.

Model	Tiny	Small	Base	Model	50	101	152
Vswin	34.36	33.86	**34.47**	Slowfast	34.60	35.04	**34.61**
Re^{2}Vw	**34.47**	**34.04**	34.09	Re^{2}Slowf	**34.93**	**35.24**	34.54

Table 3. **Effectiveness of rewiring and reusing pre-trained video models**. Reusing pre-trained models leads to significantly better performance than training from scratch (compare Row 1 to the rest). Reusing a better pre-trained model gives even higher performance (compare Row 2 to Row 3). Pret.: pretraining.

Pret. Model	Pret. Dataset	0.5	0.75	0.95	Avg.
Vswin-base	None	44.58	28.41	7.40	28.68
	Kinetics-400	52.72	36.73	8.88	35.73
	Kinetics-600	**52.46**	**37.37**	**10.39**	**36.28**

Table 4. **Comparison of localization performance with different backbones and localizers**, in terms of average mAP (%) on the dataset ActivityNet.

Backbone	Localizer	0.5	0.75	0.95	Avg.
Re^{2}Vswin-tiny	VSGN	53.24	37.23	10.49	36.38
	ActionFormer	54.75	37.81	9.03	**36.80**
Re^{2}Slowfast-101	VSGN	53.63	37.53	10.67	36.82
	ActionFormer	55.25	37.86	9.05	**37.01**

4.1.3 Why rewiring and reuse?

We rewire existing network architectures to make them reversible, and reuse their parameters for initialization. This way we dramatically reduce the effort of training the reversible networks, and still reach the representation capability of the original non-reversible ones. To compare the video representation capabilities of both types of networks, we use them to extract video features and train a localizer (VSGN [73] in this case) with the features. We demonstrate their performance in Tab. 2, showing that our reversible models are comparable to the original models.

If there are two versions of the same non-reversible model trained in different ways and with different performance, will our Re^{2}TAL models benefit more from the higher-performing version? Vswin-base happens to have such two versions: one trained on Kinetics-400 and the other trained on Kinetics-600, the latter with better performance. We initalize our Re^{2}Vswin-base with either ver-

[2]Memory increases from Vswin-small to base is due to the channel increase; memory reduction from Slowfast-101 to 152 is because the former one uses input configuration (8, 8) while the latter uses (4, 16).

Table 5. **Compared to the state-of-the-art for temporal action localization performance on ActivityNet-v1.3 and THUMOS-14**, measured by mAPs (%) at different tIoU thresholds according to their respective official metrics. E2E: end-to-end; Mem: memory (GB).

Method	Backbone	E2E	Flow	Mem	ActivityNet-v1.3				THUMOS-14				
					0.5	0.75	0.95	Avg.	0.3	0.4	0.5	0.6	0.7
TAL-Net [8]	I3D	✗	✓	-	38.23	18.30	1.30	20.22	53.2	48.5	42.8	33.8	20.8
BMN [36]	TSN	✗	✓	-	50.07	34.78	8.29	33.85	56.0	47.4	38.8	29.7	20.5
G-TAD [69]	TSN	✗	✓	-	50.36	34.60	9.02	34.09	54.5	47.6	40.2	30.8	23.4
TSI [41]	TSN	✗	✓	-	51.18	35.02	6.59	34.15	61.0	52.1	42.6	33.2	22.4
BC-GNN [4]	TSN	✗	✓	-	50.56	34.75	9.37	34.26	57.1	49.1	40.4	31.2	23.1
VSGN [73]	TSN	✗	✓	-	52.38	36.01	8.37	35.07	66.7	60.4	52.4	41.0	30.4
ActionFormer [71]	I3D	✗	✓	-	53.50	36.20	8.20	35.60	82.1	77.8	71.0	59.4	43.9
PBRNet [39]	I3D	✓	✓	-	53.96	34.97	8.98	35.01	58.5	54.6	51.3	41.8	29.5
AFSD [35]	I3D	✓	✓	12	52.40	35.30	6.50	34.40	67.3	62.4	55.5	43.7	31.1
R-C3D [66]	C3D	✓	✗	-	26.80	-	-	-	44.8	35.6	28.9	-	-
DaoTAD [63]	I3D	✓	✗	11	-	-	-	-	62.8	-	53.8	-	30.1
TALLFormer [10]	VSwin-Base	✓	✗	29	54.10	36.20	7.90	35.60	76.0	-	<u>63.2</u>	-	34.5
ActionFormer [71]	VSwin-Tiny	✗	✗	-	53.83	35.82	7.27	35.17	70.8	64.7	55.7	42.2	27.0
ActionFormer + Re^{2}TAL	Re^{2}VSwin-Tiny	✓	✗	24	<u>54.75</u>	<u>37.81</u>	<u>9.03</u>	<u>36.80</u>	<u>77.0</u>	<u>71.5</u>	62.4	<u>49.7</u>	<u>36.3</u>
ActionFormer [71]	Slowfast-101	✗	✗	-	53.98	37.00	8.87	36.09	72.7	66.9	58.6	46.4	33.1
ActionFormer + Re^{2}TAL	Re^{2}Slowfast-101	✓	✗	6.8	**55.25**	**37.86**	**9.05**	**37.01**	**77.4**	**72.6**	**64.9**	**53.7**	**39.0**

sion, and compare their performance on TAL in Tab. 3. By using the better version, the TAL performance obviously increases. In addition, using either version as initialization is better than training from scratch. This indicates that our rewiring-reuse strategy is very important for TAL performance, and reusing a better pre-trained model will benefit TAL even further. That means if a better training strategy for an existing model comes out, we can correspondingly obtain a more accurate reversible model.

4.1.4 Choice of Backbones and Localizers

In Tab. 4, we demonstrate the results using the two different backbones Re^{2}Vswin-tiny and Re^{2}Slowfast-101, and two different localizers VSGN [73] and ActionFormer [71] on ActivityNet. Since the ActionFormer localizer yields better performance for both Re^{2}Vswin-tiny and Re^{2}Slowfast-101, we compare the ActionFormer results to other methods in the literature, and also apply them to the THUMOS dataset, as shown in Sec. 4.2.

4.2. State-of-the-Art Comparisons

We compare the performance of our Re^{2}TAL to recent state-of-the-art (SOTA) methods in the literature in Tab. 5 on ActivityNet and THUMOS. On ActivityNet, our Re^{2}TAL reaches a new SOTA performance: average mAP 37.01%, outperforming all other methods by significant margins. On THUMOS, ours surpasses the concurrent work TaLLFormer [10] with mAP 64.9% at tIoU=0.5, and the highest among all methods that only use the RGB modality.

Furthermore, for an apple-to-apple comparison with the feature-based method ActionFormer, we re-ran ActionFormer using their official code with the RGB features extracted with the Vswin-tiny and Slowfast-101 backbones, corresponding to our Re^{2}Vswin-tiny and Re^{2}Slowfast-101, respectively. We see that our Re^{2}TAL always outperforms vanilla ActionFormer under the same backbone categories (*e.g.* Re^{2}Vswin-tiny and Vswin-tiny are in the same backbone category; Re^{2}Slowfast-101 and Slowfast-101 are in the same backbone category).

5. Conclusions

In this work, we propose a novel rewiring-to-reversibility (Re^2) scheme to convert off-the-shelf models into reversible models while preserving the number of trainable parameters. The procedure allows reusing the compute invested in training large models and adds only a tiny sliver of fine-tuning compute. We apply the procedure to video backbones such as Video Swin (Vswin) and SlowFast to obtain Re^{2}Vwin and Re^{2}Slowfast backbones respectively. Further, we utilize the Re^2 backbones for memory-efficient end-to-end temporal action localization, reaching mAP 64.9% at tIoU$=$ 0.5 on THUMOS-14, and average mAP 37.01% on ActivityNet-v1.3, establishing a new state-of-the-art. We hope that future work in this direction can explore extending the Re^2 method to other memory-bottlenecked tasks such as dense video captioning, movie summarization *etc*.

Acknowledgments. This work was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research through the Visual Computing Center (VCC) funding and SDAIA-KAUST Center of Excellence in Data Science and Artificial Intelligence.

References

[1] Humam Alwassel, Fabian Caba Heilbron, Victor Escorcia, and Bernard Ghanem. Diagnosing error in temporal action detectors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 6

[2] Humam Alwassel, Silvio Giancola, and Bernard Ghanem. TSP: Temporally-sensitive pretraining of video encoders for localization tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3

[3] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. ViViT: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3

[4] Yueran Bai, Yingying Wang, Yunhai Tong, Yang Yang, Qiyue Liu, and Junhui Liu. Boundary content graph neural network for temporal action proposal generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 1, 3, 8

[5] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021. 3

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

[7] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 3

[8] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A. Ross, Jia Deng, and Rahul Sukthankar. Rethinking the Faster R-CNN architecture for temporal action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3, 8

[9] Guo Chen, Sen Xing, Zhe Chen, Yi Wang, Kunchang Li, Yizhuo Li, Yi Liu, Jiahao Wang, Yin-Dong Zheng, Bingkun Huang, et al. InternVideo-Ego4D: A pack of champion solutions to ego4d challenges. *arXiv preprint arXiv:2211.09529*, 2022. 1

[10] Feng Cheng and Gedas Bertasius. TALLFormer: Temporal action localization with long-memory transformer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 2, 3, 8

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2009. 3

[12] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. In *International Conference on Learning Representations (ICLR) Workshop*, 2015. 2

[13] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations (ICLR)*, 2017. 2

[14] 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. In *International Conference on Learning Representations (ICLR)*, 2021. 2, 4

[15] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3

[16] Christoph Feichtenhofer. X3D: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2020. 2

[17] 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 (ICCV)*, 2019. 2, 3, 5, 6

[18] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[19] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2019. 2

[20] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 2, 3

[21] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4D: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 3, 4

[23] Lisa Anne Hendricks, O. Wang, E. Shechtman, Josef Sivic, Trevor Darrell, and Bryan C. Russell. Localizing moments in video with natural language. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 1

[24] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning (ICML)*, 2019. 2

[25] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-RevNet: Deep invertible networks. In *International Conference on Learning Representations (ICLR)*, 2018. 2, 3

[26] Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. STM: Spatiotemporal and motion encoding for action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 2

[27] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. http://crcv.ucf.edu/THUMOS14/, 2014. 6
[28] W. Kay, J. Carreira, K. Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, F. Viola, T. Green, T. Back, A. Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *ArXiv*, abs/1705.06950, 2017. 1, 3
[29] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 2
[30] R. Krishna, Kenji Hata, F. Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 1
[31] Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. Training graph neural networks with 1000 layers. In *International Conference on Machine Learning (ICML)*, 2021. 2
[32] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Improved multiscale vision transformers for classification and detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3
[33] Zhenyang Li, Kirill Gavrilyuk, Efstratios Gavves, Mihir Jain, and Cees GM Snoek. VideoLSTM convolves, attends and flows for action recognition. *Computer Vision and Image Understanding*, 2018. 2
[34] Chuming Lin, Jian Li, Yabiao Wang, Ying Tai, Donghao Luo, Zhipeng Cui, Chengjie Wang, Jilin Li, Feiyue Huang, and Rongrong Ji. Fast learning of temporal action proposal via dense boundary generator. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 3
[35] Chuming Lin, Chengming Xu, Donghao Luo, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Yanwei Fu. Learning salient boundary feature for anchor-free temporal action localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3, 8
[36] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. BMN: Boundary-matching network for temporal action proposal generation. In *Proceedings of The IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 1, 3, 8
[37] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. BSN: Boundary sensitive network for temporal action proposal generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 3
[38] Kang Liu, Dong Liu, Li Li, Ning Yan, and Houqiang Li. Semantics-to-signal scalable image compression with learned revertible representations. *International Journal of Computer Vision (IJCV)*, 2021. 2
[39] Qinying Liu and Zilei Wang. Progressive boundary refinement network for temporal action detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 3, 7, 8
[40] Shuming Liu, Mengmeng Xu, Chen Zhao, Xu Zhao, and Bernard Ghanem. ETAD: A unified framework for efficient temporal action detection. *arXiv preprint arXiv:2205.07134*, 2022. 3
[41] Shuming Liu, Xu Zhao, Haisheng Su, and Zhilan Hu. Tsi: Temporal scale invariant network for action proposal generation. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2020. 3, 8
[42] Xiaolong Liu, Song Bai, and Xiang Bai. An empirical study of end-to-end temporal action detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3
[43] Yang Liu, Zhenyue Qin, Saeed Anwar, Pan Ji, Dongwoo Kim, Sabrina Caldwell, and Tom Gedeon. Invertible denoising network: A light solution for real noise removal. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2
[44] 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 (TIP)*, 31, 2022. 2
[45] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3, 5, 6
[46] Fuchen Long, Ting Yao, Zhaofan Qiu, X. Tian, Jiebo Luo, and T. Mei. Gaussian temporal awareness networks for action localization. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3
[47] Matthew MacKay, Paul Vicol, Jimmy Ba, and Roger B Grosse. Reversible recurrent neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018. 2
[48] Karttikeya Mangalam, Haoqi Fan, Yanghao Li, Chao-Yuan Wu, Bo Xiong, Christoph Feichtenhofer, and Jitendra Malik. Reversible vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3, 5
[49] Chong Mou, Youmin Xu, Jiechong Song, Chen Zhao, Bernard Ghanem, and Jian Zhang. Large-capacity and flexible video steganography via invertible neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 3
[50] Jonghwan Mun, L. Yang, Zhou Ren, N. Xu, and B. Han. Streamlined dense video captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1
[51] Zhiwu Qing, Haisheng Su, Weihao Gan, Dongliang Wang, Wei Wu, Xiang Wang, Yu Qiao, Junjie Yan, Changxin Gao, and Nong Sang. Temporal context aggregation network for temporal action proposal refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3
[52] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[53] Merey Ramazanova, Victor Escorcia, Fabian Caba Heilbron, Chen Zhao, and Bernard Ghanem. OWL (Observe, Watch, Listen): localizing actions in egocentric video via audio-visual temporal context. *arXiv preprint arXiv:2202.04947*, 2022. 1

[54] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage CNNs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[55] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. 2

[56] Mattia Soldan, Alejandro Pardo, Juan León Alcázar, Fabian Caba, Chen Zhao, Silvio Giancola, and Bernard Ghanem. MAD: A scalable dataset for language grounding in videos from movie audio descriptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[57] Yang Song, Chenlin Meng, and Stefano Ermon. MintNet: Building invertible neural networks with masked convolutions. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019. 2

[58] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. In *CRCV-TR-12-01*, 2012. 6

[59] Jing Tan, Jiaqi Tang, Limin Wang, and Gangshan Wu. Relaxed transformer decoders for direct action proposal generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3

[60] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3D convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015. 3

[61] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 2

[62] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[63] Chenhao Wang, Hongxiang Cai, Yuxin Zou, and Yichao Xiong. RGB stream is enough for temporal action detection. *arXiv preprint arXiv:2107.04362*, 2021. 3, 7, 8

[64] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krähenbühl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[65] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2017. 2

[66] Huijuan Xu, Abir Das, and Kate Saenko. R-C3D: Region convolutional 3D network for temporal activity detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 3, 8

[67] Mengmeng Xu, Juan-Manuel Pérez-Rúa, Victor Escorcia, Brais Martinez, Xiatian Zhu, Li Zhang, Bernard Ghanem, and Tao Xiang. Boundary-sensitive pre-training for temporal localization in videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1

[68] Mengmeng Xu, Juan Manuel Perez Rua, Xiatian Zhu, Bernard Ghanem, and Brais Martinez. Low-fidelity video encoder optimization for temporal action localization. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021. 3

[69] Mengmeng Xu, Chen Zhao, David S Rojas, Ali Thabet, and Bernard Ghanem. G-tad: Sub-graph localization for temporal action detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 3, 8

[70] Shen Yan, Xuehan Xiong, Anurag Arnab, Zhichao Lu, Mi Zhang, Chen Sun, and Cordelia Schmid. Multiview transformers for video recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[71] Chenlin Zhang, Jianxin Wu, and Yin Li. ActionFormer: Localizing moments of actions with transformers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 1, 3, 6, 8

[72] Chen Zhao, Merey Ramazanova, Mengmeng Xu, and Bernard Ghanem. SegTAD: Precise temporal action detection via semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2023. 1, 3

[73] Chen Zhao, Ali K Thabet, and Bernard Ghanem. Video self-stitching graph network for temporal action localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1, 3, 6, 7, 8

[74] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 2