Dr^{2}Net: Dynamic Reversible Dual-Residual Networks for Memory-Efficient Finetuning
[Supplementary Material]

Chen Zhao[1] Shuming Liu[1] Karttikeya Mangalam[2] Guocheng Qian[1]
Fatimah Zohra[1] Abdulmohsen Alghannam[1] Jitendra Malik[2] Bernard Ghanem[1]
[1]King Abdullah University of Science and Technology, Saudi Arabia [2]UC Berkeley, US

In the paper, we have described the core techniques of Dr^{2}Net, and provided the key experiments that support our contributions. In this supplementary material, we provide additional details of the method and the experiment implementation, as well as extra experimental results.

A. Additional Details of the Method

A.1. Proof of invertibility of Dr^{2}Net

Our proposed Dr^{2}Net, as illustrated in Fig. 2 and Eq. 1 in the paper, is a reversible network, and mathematically, an invertible function. In this section, we mathematically prove its invertibility. Let's rewrite the computation of the i^{th} module (Eq. 1 in the paper) in the following equation for clarity.

$$\begin{cases} y_i = \beta \times x_{i-1} \\ x_i = \mathcal{G}_i(x_{i-1}) + y_{i-1}. \end{cases} \tag{1}$$

Let's make $I = (x_{i-1}, y_{i-1})$, which represents the input activations to the i^{th} module, and make $O = (y_i, x_i)$, which represents the output activations from the i^{th} module. The Jacobian matrix of Eq. 1 is computed as follows

$$J = \frac{\partial O}{\partial I} = \begin{bmatrix} \frac{\partial y_i}{\partial x_{i-1}} & \frac{\partial y_i}{\partial y_{i-1}} \\ \frac{\partial x_i}{\partial x_{i-1}} & \frac{\partial x_i}{\partial y_{i-1}} \end{bmatrix} = \begin{bmatrix} \beta \times I_d & 0 \\ \frac{\partial G_i}{\partial x_{i-1}} & I_d \end{bmatrix}. \tag{2}$$

In Eq. 2, I_d is the identity matrix of size d, where d is the dimension of the activations $x_i, y_i, x_{i-1}, y_{i-1}$. Its determinant is computed as

$$\det(J) = \det(\beta \times I_d) \cdot \det(I_d) = \beta^d. \tag{3}$$

As described in the paper, $\beta \neq 0$, and hence, the Jacobian determinant $\det(J)$ is not zero. Therefore, the function in Eq. 1 representing the i^{th} module in Dr^{2}Net is invertible.

If we stack multiple such reversible modules, represented by the above invertible functions, without inserting any downsampling operations, we will form *a stage* in Dr^{2}Net. One stage is mathematically composition of such invertible functions, and therefore, the entire stage of Dr^{2}Net is also invertible. Between stages where there are downsampling operations, we cache the activations after each stage following [18, 28].

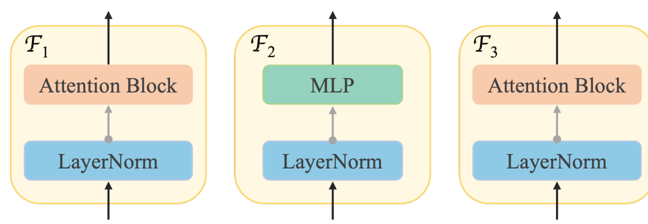

Figure 1. $\mathcal{F}_i$ **blocks in a transformer network.** If the pretrained model is a transformer network, e.g., Swin [15] or ViT [7], the $\mathcal{F}_i$ blocks in our Dr^{2}Net are attention layers or MLP layers. The two types of layers are interleaved, namely, if $\mathcal{F}_1$ is an attention layer, then $\mathcal{F}_2$ is an MLP layer, and $\mathcal{F}_3$ is an attention layer, and so on.

A.2. Illustration of the reverse computation

In Fig. 2 (c) in the paper, we have illustrated the architecture of our Dr^{2}Net with the $\mathcal{F}$ blocks and the two types of residual connections. In Fig. 2 (a), we re-illustrate this forward process by moving the $\mathcal{F}$ blocks along with their α-weighted residual connection inside the G blocks for conciseness and to be consistent with Eq. 1 in the paper. In Fig. 2 (b), we illustrate its corresponding reverse process.

For detailed mathematical formulation of the forward and reverse processes, we expand Eq. 1 in the paper as Eq. 4, and Eq. 2 in the paper as Eq. 5 to illustrate the computation in three modules. In the equations, $\mathcal{G}_i(x_{i-1}) = \mathcal{F}_i(x_{i-1}) + \alpha \times x_{i-1}$.

We can see from Fig. 2 (b) and Eq. 5 that during the reverse computation, given x_i and y_i where $i = 3$, we will compute all the intermediate activations x_i, y_i where $i = 0, 1, 2$ module by module. In the i^{th} module, x_{i-1} is computed first using $x_{i-1} = y_i/\beta$. Then x_{i-1} is used to

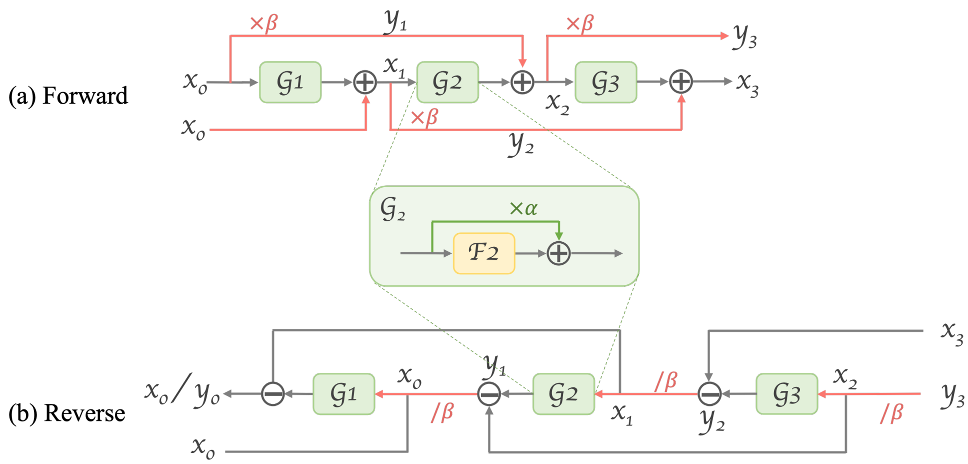

Figure 2. **Forward and reverse computation in Dr^{2}Net.** Gray arrows denote the pathway for x_i, and pink arrows denote the pathway for y_i. Compared to Fig. 2 in the paper, we place the $\mathcal{F}_i$ blocks along with their α-weighted residual connections inside the module $\mathcal{G}_i$.

$$\text{Forward:} \quad \left\{ \begin{array}{l} y_1 = \beta \times x_0 \\ x_1 = \mathcal{G}_1(x_0) + y_0, \end{array} \right. \Rightarrow \left\{ \begin{array}{l} y_2 = \beta \times x_1 \\ x_2 = \mathcal{G}_2(x_1) + y_1, \end{array} \right. \Rightarrow \left\{ \begin{array}{l} y_3 = \beta \times x_2 \\ x_3 = \mathcal{G}_3(x_2) + y_2. \end{array} \right. \tag{4}$$

$$\text{Reverse:} \quad \left\{ \begin{array}{l} x_0 = y_1/\beta \\ y_0 = x_1 - \mathcal{G}_1(x_0), \end{array} \right. \Leftarrow \left\{ \begin{array}{l} x_1 = y_2/\beta \\ y_1 = x_2 - \mathcal{G}_2(x_1), \end{array} \right. \Leftarrow \left\{ \begin{array}{l} x_2 = y_3/\beta \\ y_2 = x_3 - \mathcal{G}_3(x_2). \end{array} \right. \tag{5}$$

compute $\mathcal{G}_i(x_{i-1})$ to finally compute y_{i-1}.

A.3. Illustration of different types of F blocks

The basic blocks $\mathcal{F}_i$ in Dr^{2}Net, as illustrated in Fig. 2, can be any network block that doesn't change the feature dimensions. We use $\mathcal{F}_i$ and $\mathcal{F}$ interchangeably in the following text. The $\mathcal{F}_i$ blocks can be instantiated as different types of blocks when the pretrained networks have different architectures. In Fig. 1, we illustrate the $\mathcal{F}_i$ blocks of the popular transformer architectures, Swin [15] and ViT [7]. In this case, the $\mathcal{F}_i$ blocks in our Dr^{2}Net are attention layers or MLP layers. The two types of layers are interleaved, namely, if $\mathcal{F}_1$ is an attention layer, then $\mathcal{F}_2$ is an MLP layer, and $\mathcal{F}_3$ is an attention layer, and so on.

A.4. Gradient errors of different networks

In the paper, we have illustrated the gradient error levels of video Swin-tiny [16] in Fig. 3. In this subsection, we plot the error levels for another popular type of network Video ViT [24], and provide more detailed explanations about the error maps.

In Fig. 3, we plot the error levels of the two types of networks Video ViT-small (used in VideoMAE [24]) and Video Swin-tiny, both with 12 layers. As we described in the paper, customized back-propagation which computes gradients with recomputed intermediate activations through the reverse process (Eq. 2 in the paper), is used to save memory for the reversible networks. This may introduce numerical errors that are accumulated due to floating point computation with limited precision. The idea of the gradient error levels is to assess the precision of the customized back-propagation compared to using the default back-propagation that computes gradients with the activations cached in GPU memory. Concretely, the values in the gradient-error-level maps in Fig. 3 are obtained as follows. Given one point $\alpha = \alpha_0$ and $\beta = \beta_0$, we obtain one Dr^{2}Net architecture that is adapted from Video ViT-small or Video Swin-tiny. For this Dr^{2}Net architecture, we have two ways of implementations: (1) Dr^{2}Net-A with customized back-propagation, and (2) Dr^{2}Net-B with default back-propagation. We generate a random tensor, and feed it into Dr^{2}Net-A and Dr^{2}Net-B separately, and compute two versions of gradients respectively: G_A and G_B. We compare G_A and G_B using *torch.allclose(G_A, G_B, rtol=1e-05, atol=atol)*, and record the lowest *atol* value that gives *torch.allclose() == True* as the value at $(\alpha = \alpha_0, \beta = \beta_0)$ in the gradient-error-level maps.

As we see from Fig. 3, though Swin has slightly lower error levels than ViT, the error levels of the two types of networks are quite close, with the lowest in the bottom-left

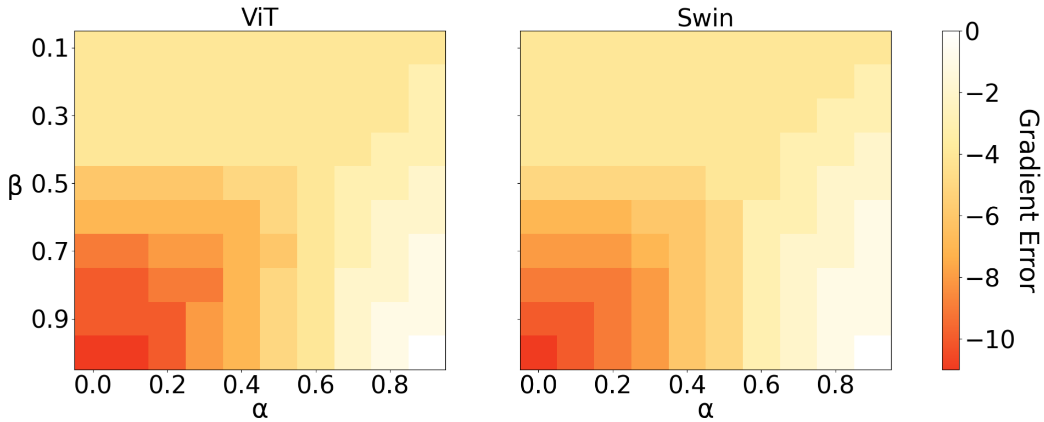

Figure 3. **Gradient error levels with different α and β values for Video ViT-small and Video Swin-tiny.** The error levels of the two types of networks are similar, with the lowest in the bottom-left corners, and the highest in the bottom-right corners. Swin has slight lower error levels.

corners, and the highest in the bottom-right corners. When we initialize Dr^{2}Net from the pretrained ViT or Swin, we set $\alpha = 1, \beta = 0.1$, meaning the finetuning starts from the top-right corners of the map, as we described in Sec. 3.3.2 in the paper. Considering that the errors at the top-right corner are too high to effectively train the networks, i.e., 10^{-4} and 10^{-5} for ViT and Swin respectively, we need the dynamic finetuning strategy to adjust the values of α and β to reach a point with sufficient precision, which is the bottom-left region. It can be observed from the maps that the shortest path to reach the bottom-left region with monotonically non-increased error levels is along the diagonal, meaning updating α and β simultaneously.

In addition, to make Dr^{2}Net with new values of α and β benefit from Dr^{2}Net with previous values of α and β, we need to update the values of α and β in small steps. We use η to determine the updating frequency of both coefficients, as described in Sec. 3.3.2 in the paper. Given the total number of epochs for which α and β are updated, a smaller η value indicates the changes of α and β are more frequent but more incremental each time. We have shown in Tab. 10 in the paper that a smaller η value results in higher performance for the task of action recognition with the VideoMAE [24] pretrained model.

B. Implementation Details

In this section, we provide the implementation details of the downstream tasks we have experimented in the paper.

B.1. Temporal action detection

Temporal action detection (TAD) [12, 25, 28] is a typical long-form video understanding task, that needs to process a long sequence of video frames to identify all the action instances. Given a long video, the task of TAD outputs the category as well as the start and end timestamps of each action. A representative dataset for this task is the largescale dataset ActivityNet-v1.3 [4], that uses mean Average Precision (mAP) at 10 tIoU thresholds in the range [0.5, 0.95] as well as average mAP as the evaluation metric.

In our experiment, we use a recent TAD method VSGN [27] as the detector, and Video Swin-tiny pretrained with Kinetics-400 classification as the backbone. For all the experiments of this task in Tab. 2 in the paper, we use the same setup as follows. As network input, we use 512 input frames, evenly sampled from the entire video regardless of the original video duration. The frame resolution is 224×224. We use the augmentation following [28]. The backbone learning rate is $1e-5$, the detector learning rate is $1e-4$, and the batch size is 2. The total number of epochs is 20. For Dr^{2}Net, the coefficient updating frequency is 3 epochs, and the updating ends at the 10^{th} epoch.

B.2. Video object segmentation

Video object segmentation aims to separate the foreground objects from the background region of a video at the pixel level [2, 6]. Recently, referring video object segmentation (RVOS) has drawn more attention [14, 19, 20]. Given a sequence of video frames and a text query, RVOS aims to segment all objects in the video referred by the input text prior to determining the referred instance [9]. In this paper, we evaluated our method on the dataset A2D-Sentence [9], which contains 3,754 videos with 8 action classes.

In the experiments, we utilize the method MTTR [3] as the segmentation head and the Kinetics-400 [5] pretrained Video Swin-tiny as the backbone. In MTTR, the window size is set to 10, and the total batch size is set to 6. The video frames are resized such that the short side is at least 320 pixels and the long side at most 576 pixels. The model is trained for 70 epochs. For Dr^{2}Net, the coefficient updating frequency is set to 2 iterations, and the updating ends at the

10^{th} epoch.

B.3. Action recognition

Action recognition (AR) [8, 10, 16, 24, 29] is a fundamental task in video understanding, which aims to classify a video clip into an action category. Though it doesn't require as long input sequences as TAD, its input is still 3D video data and it uses spatio-temporal attention with Transformers, which consumes a large amount of GPU memory. Therefore, memory-efficient finetuning is important. If we can save memory consumption during training, then we will be able to feed more input frames, use larger batch sizes, and train larger networks, which will lead to higher performance.

For the experiments, we adopt the widely used large-scale video dataset Something-Something V2 [10], which contains around 169k videos for training and 20k videos for validation, with 174 motion-centric action classes. We report the top-1 and top-5 accuracies as the evaluation metrics. We have two sets of experiments on the task of action recognition, Set-A with the Video ViT backbones pretrained with VideoMAE [24] (Sec. 4.1 in the paper), and Set-B with Image ViT backbones pretrained with DINOv2 [21] (Sec. C.1). Both sets of experiments use the dataset Something-Something V2 [10] and the finetuning recipe of VideoMAE [24] for the downstream finetuning. For both sets, the input video resolution is $224 \times 224 \times 16$, the batch size is 384, the learning rate $1e-3$, and the total number of epochs is 40. For Dr^{2}Net, the coefficient updating frequency is 2 iterations, and the updating ends at the 5^{th} epoch.

B.4. Object detection

Object detection (OD) involves identifying and locating potential objects within an image. A notable example of state-of-the-art object detection approaches is DINO [26], which enhances the performance of the DETR-based framework by denoisng its anchor boxes. For the downstream task of object detection in our work, we use DINO as the detection head and employ Swin Transformer [15] as the image backbone. We evaluate the model's performance using the mean Average Precision (mAP) metric on the COCO val2017 dataset [13].

In our experiments, we follow the training receipt of the original DINO. The Swin Transformer is pretrained on the ImageNet-22k dataset with the image classification task. We utilize 4 scales of feature maps to conduct the experiments. The short side of an input image is randomly resized between 480 and 800 pixels, and the long side is resized to at most 1333. The total batch size is 16, and the number of training epochs is 12. For Dr^{2}Net, the updating frequency of the two coefficients is 2 iterations, and the updating ends at the 5^{th} epoch.

B.5. 3D point cloud segmentation

3D point cloud segmentation (PCS) is the process of classifying point clouds into multiple meaningful regions, where the points in the same region have the same label. We conduct extensive experiments in *S3DIS* [1], which is the mostly-used benchmark for large-scale point cloud segmentation. S3DIS consists of 6 areas with 271 rooms, where area-5 is used in testing and the others are used in training. Each area is a large point cloud of a building. We used the same preprocessing as Pix4Point [23] to extract the point cloud per room, and leveraged sphere sampling to sample $16,384$ points as a batch in training and testing. Following the standard practice [22], our model is optimized using the cross-entropy loss with label smoothing of 0.1, the AdamW optimizer [17] with a learning rate 1e-4, a cosine learning rate scheduler, 10 warmup epochs, weight decay 1e-5, the batch size 8, and 600 total training epochs. We use data augmentation including rotation, scaling, color auto-contrast, and color dropping. For Dr^{2}Net, the coefficient updating frequency is 10 iterations, and the updating ends at the 50^{th} epoch.

C. Supplementary Experiments

C.1. More pretraining methods

In the paper, we have shown the effectiveness of our Dr^{2}Net on models with different pretraining methods, including fully-supervised classification, self-supervised learning with MAE [11] and VideoMAE [24]. In this subsection, we demonstrate our results with one more pretraining method DINOv2 [21].

DINOv2 is a self-supervised learning method that pretrain an image model on a largescale image dataset. We use it for the downstream task action recognition on the dataset Something-Something v2 [10]. Since the architecture of the DINOv2 model is ViT [7], which is agnostic of input data dimensions, we can directly apply the same ViT architecture to the video data and compute spatio-temporal attention. Considering that the patch embedding layer was pretrained for images which are 2D data, we inflate those convolutional kernels to 3D during initialization to perform tube embedding instead of patch embedding. In addition, we interpolate the position embedding to match the video dimension. Our implementation of finetuning the DINOv2 model on Something-Something v2 follows VideoMAE [24] for the setup of the spatio-temporal attention, tube embedding, and the training recipe.

We demonstrate the memory consumption and the recognition accuracy in Tab. 1. Compared to conventional end-to-end finetuning (Row 2), our Dr^{2}Net (Row 5) only uses less than 1/4 memory, and its accuracy surprisingly surpasses conventional finetuning by a large margin. Considering that the accuracies in the table are taken from the results of the

Table 1. **Memory and accuracy comparison on action recognition using DINOv2 [21] pretrained models.** The backbone ViT-small is used. Conventional: conventional non-reversible backbone; Reversible: previous reversible backbone [28]; Hard: directly initializing the reversible network using pretrained parameters.

Downstream training		Top-1 acc	Top-5 acc	Mem (GB)
Conventional	Frozen*	33.10%	/	/
	End-to-end	55.18%	82.79%	34.2
Reversible [28]	Scratch	14.31%	33.96%	8.0
	Hard	37.29%	66.22%	8.0
Dr^{2}Net	**End-to-end**	**64.98%**	**88.90%**	8.0

* Frozen: linear probing results from the DINOv2 [21] paper.

40^{th} epoch following VideoMAE [24], the training might not have fully converged. Still, that shows our Dr^{2}Net at least converges faster. This might be due to the domain gap between the image pretraining and the video downstream task, and is worth further exploration.

C.2. Using larger networks

Our Dr^{2}Net can significantly reduce the GPU memory consumption during finetuning. Using the saved GPU memory, we can support a larger backbone network to reach higher accuracy. We experiment with larger backbones for the tasks of action recognition with DINOv2 [21] pretrained models, action recognition with VideoMAE [24] pretrained models, and object detection with DINO [26]. We demonstrate the accuracy and the corresponding GPU memory consumption in Tab. 2, Tab. 3 and Tab. 4, respectively.

For the first two tasks (Tab. 2 and Tab. 3), which use ViT [7] as the backbone, we apply Dr^{2}Net to ViT-base in addition to ViT-small. Using the larger backbone ViT-base (Row 3), the accuracy is obviously increased for both tasks. Compared to both conventional finetuning (Row 1), Dr^{2}Net uses still less than half of the memory (16.6 GB *vs*. 34.2 GB, 13.0 GB *vs*. 29.3 GB), but reaches much higher performance.

For the task of object detection [26], we apply Dr^{2}Net to Video Swin-small and Video Swin-base in addition to Video Swin-tiny. Using the larger backbone Swin-small (Row 3), the accuracy is obviously increased, while the memory is almost the same (30.1 GB). Using a even larger backbone Swin-small, the accuracy is dramatically higher than conventional finetuning (54.7% *vs*. 51.3%), while memory cost is only 60% of it (32.4 GB *vs*. 54.0 GB).

Table 2. **Accuracy versus memory for action recognition [10] with DINOv2 [21] pretrained models.** Our Dr^{2}Net can utilize the saved memory to train a larger backbone (Row 3), leading to higher performance while still using less memory. Conventional: conventional non-reversible finetuning.

Finetuning	Backbone	Top-1 acc	Top-5 acc	Mem (GB)
Conventional	ViT-small	55.2%	82.8%	34.2
Dr^{2}Net	ViT-small	65.0%	88.9%	8.0
Dr^{2}Net	ViT-base	**68.2%**	**90.8%**	16.6

Table 3. **Accuracy versus memory for action recognition [10] with VideoMAE [24] pretrained models.** Our Dr^{2}Net can utilize the saved memory to train a larger backbone (Row 3), leading to higher performance while still using less memory. Conventional: conventional non-reversible finetuning.

Finetuning	Backbone	Top-1 acc	Top-5 acc	Mem (GB)
Conventional	ViT-small	66.5%	90.3%	29.3
Dr^{2}Net	ViT-small	64.6%	89.0%	6.0
Dr^{2}Net	ViT-base	**68.6%**	**92.0%**	13.0

Table 4. **Accuracy versus memory for object detection [26].** Our Dr^{2}Net can utilize the saved memory to train a larger backbone (Row 3&4), leading to higher performance while still using less memory. Conventional: conventional non-reversible finetuning. Conventional: conventional non-reversible finetuning.

Finetuning	Backbone	AP (%)	Mem (GB)
Conventional	Vswin-tiny	51.3	54.0
Dr^{2}Net	Vswin-tiny	51.3	30.0
Dr^{2}Net	Vswin-small	52.8	30.1
Dr^{2}Net	Vswin-base	**54.7**	32.4

References

[1] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3D semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4

[2] Goutam Bhat, Felix Järemo Lawin, Martin Danelljan, Andreas Robinson, Michael Felsberg, Luc Van Gool, and Radu Timofte. Learning what to learn for video object segmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 777–794. Springer, 2020. 3

[3] Adam Botach, Evgenii Zheltonozhskii, and Chaim Baskin. End-to-end referring video object segmentation with multimodal transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[4] 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. 3

[5] 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. 3

[6] Ho Kei Cheng and Alexander G Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *European Conference on Computer Vision*, pages 640–658. Springer, 2022. 3

[7] 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 16×16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. 1, 2, 4, 5

[8] 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. 4

[9] Kirill Gavrilyuk, Amir Ghodrati, Zhenyang Li, and Cees GM Snoek. Actor and action video segmentation from a sentence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5958–5966, 2018. 3

[10] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The “something something” video database for learning and evaluating visual common sense. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 4, 5

[11] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2022. 4

[12] 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. 3

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proceedings of the European conference on computer vision (ECCV)*, 2014. 4

[14] Si Liu, Tianrui Hui, Shaofei Huang, Yunchao Wei, Bo Li, and Guanbin Li. Cross-modal progressive comprehension for referring segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):4761–4775, 2021. 3

[15] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2022. 1, 2, 4

[16] 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. 2, 4

[17] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. 2019. 4

[18] 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. 1

[19] Bruce McIntosh, Kevin Duarte, Yogesh S Rawat, and Mubarak Shah. Visual-textual capsule routing for text-based video segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9942–9951, 2020. 3

[20] Ke Ning, Lingxi Xie, Fei Wu, and Qi Tian. Polar relative positional encoding for video-language segmentation. In *IJCAI*, page 10, 2020. 3

[21] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 4, 5

[22] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. In *NeurIPS*, 2022. 4

[23] Guocheng Qian, Abdullah Hamdi, Xingdi Zhang, and Bernard Ghanem. Image pretrained standard transformers for 3d point cloud understanding. In *International Conference on 3D Vision (3DV)*, 2024. 4

[24] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022. 2, 3, 4, 5

[25] 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. 3

[26] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Heung-Yeung Shum. DINO: DETR with improved denoising anchor boxes for end-to-end object detection. In *The Eleventh International Conference on Learning Representations*, 2023. 4, 5

[27] 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. 3

[28] Chen Zhao, Shuming Liu, Karttikeya Mangalam, and Bernard Ghanem. Re^{2}TAL: Rewiring pretrained video backbones for reversible temporal action localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 3, 5

[29] Andrew Zisserman, Joao Carreira, Karen Simonyan, Will Kay, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, et al. The Kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 4