# [Supplementary] 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

In this supplementary material, we discuss about data augmentation with our Re$^2$TAL (Section 1) and our strategy for a special case of Resnet modules ( Section 2).

## 1. Data Augmentation

Data augmentation, especially spatial augmentation is a common strategy to overcome overfitting. By randomly transforming the images when training a deep neural network, it is helpful to enhance the validation accuracy. However, it is rarely used for temporal action localization (TAL) due to the fact that most methods are based on 1D features, which have already lost the spatial dimensions.

Due to the end-to-end training framework, we can take advantage of the various augmentations in the image domain as well as the temporal domain. In Table 1, we compare the performance with different augmentation strategies: random spatial cropping, random flipping, temporal jittering, random rotation, and color jittering. We can see that the augmentations in the image domain are very important for improving performance. Using the combination of all the above augmentations leads to the highest performance.

Table 1. **Ablation study on different augmentations.** Performance is shown in mAP (%) on Re$^2$ Vswin-tiny.

| Rand. Crop. | Rand. Flip. | Temp. Jitter. | Rand. Rotat. | Color Jitter. | tIoU 0.5 | tIoU 0.75 | tIoU 0.95 | Avg. mAP |
|---|---|---|---|---|---|---|---|---|
| | | | | | 52.27 | 35.96 | 9.61 | 35.25 |
| ✓ | ✓ | | | | 52.59 | 36.74 | 10.03 | 35.87 |
| ✓ | ✓ | ✓ | | | 53.05 | <u>36.99</u> | 9.08 | 35.98 |
| ✓ | ✓ | | ✓ | | <u>53.16</u> | 36.80 | **10.55** | <u>36.20</u> |
| ✓ | ✓ | | | ✓ | 52.93 | 36.94 | 10.17 | 36.02 |
| ✓ | ✓ | ✓ | ✓ | ✓ | **53.24** | **37.23** | <u>10.49</u> | **36.38** |

## 2. Additional Explanations of A Special Case

In the original version of the Resnet module [1], there is a ReLU layer after each residual connection, between two $\mathcal{F}_i$ blocks, as illustrated in Fig. 1. Considering that the ReLU layer is not invertible, directly rewiring this type of modules would discontinue the reversibility across modules without caching the activations.

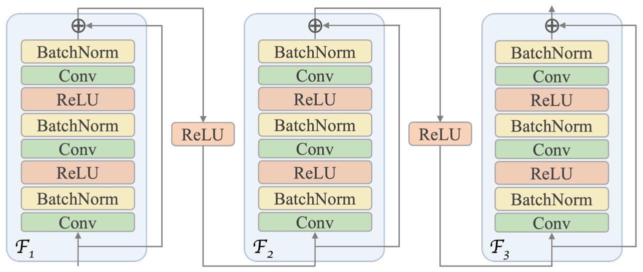

Figure 1. **Illustration of $\mathcal{F}_i$ blocks (with bottleneck) with ReLU layers in between.**

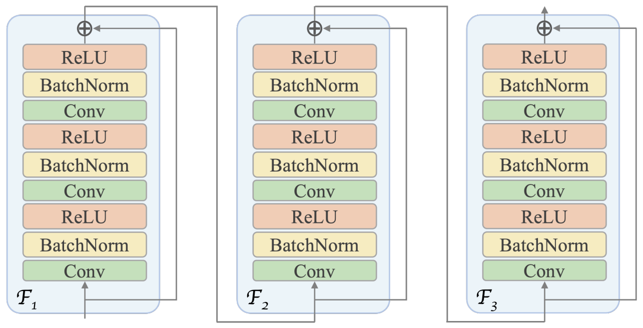

Figure 2. **Illustration of $\mathcal{F}_i$ blocks (with bottleneck) after ReLU layers are moved inside the blocks.**

To deal with this special case, we provide a simple ReLU-in strategy that enables reversibility without affecting module parameters. As shown in Figure 2, we simply move this ReLU layer inside the $\mathcal{F}_i$ block before rewiring. Because $\mathcal{F}_i$ block can be any function, it can include one more ReLU layer, which doesn't have any parameters. After this, we can follow the rewiring process in Section 3.2 in the paper to convert the modules to reversible modules.

Actually, the Slowfast networks we used in the experiments contain such Resnet modules. We followed the above ReLU-in strategy to build the Re$^2$ Slowfast backbones. We showed that the obtained Re$^2$ Slowfast can reach the performance of the original Slowfast networks in Table 2 in the paper.

## References

[1] 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 (ICCV)*, 2016. 1