ERA: Expert Retrieval and Assembly for Early Action Prediction (Supplementary Material)

Lin Geng Foo[1⋆], Tianjiao Li[1]1, Hossein Rahmani[2], Qiuhong Ke[3], and Jun Liu[1⋆⋆]

[1] ISTD Pillar, Singapore University of Technology and Design
{lingeng_foo,tianjiao_li}@mymail.sutd.edu.sg, jun_liu@sutd.edu.sg
[2] School of Computing and Communications, Lancaster University
h.rahmani@lancaster.ac.uk
[3] Department of Data Science & AI, Monash University
qiuhong.ke@monash.edu

1 Visualization of selection of the experts

We visualize the selection of experts on classes of NTU60 in Fig. 1. Samples from similar classes retrieve similar experts, while dissimilar classes retrieve different experts. This is qualitative evidence that the expert retrieval mechanism assigns each expert similar samples, which forces them to specialize in subtle cues to distinguish them. We show more quantitative analysis of this in the next section.

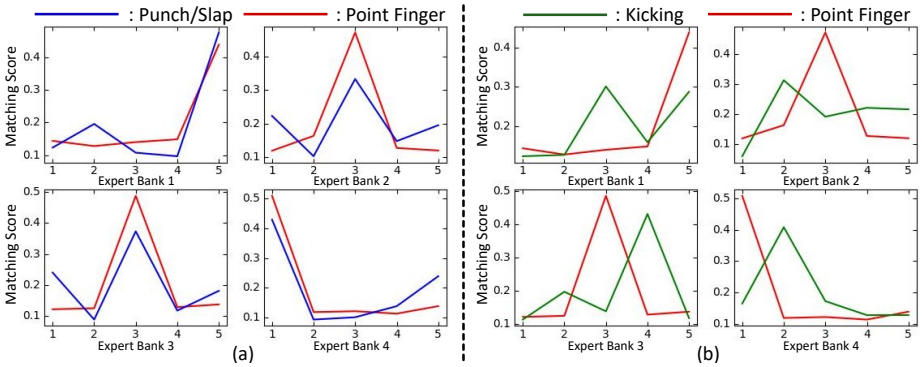

Fig. 1. Visualization of experts selection at 20% observation of the actions. Each subplot contains information from an Expert Bank, where the horizontal axis represents the M experts (where $M = 5$) and the vertical axis denotes their normalized matching scores. (Left) At 20% observation ratio, actions "Punch/Slap" and "Point finger" are similar, and their expert matching scores are also similar. (Right) At 20% observation ratio, actions "Point finger" and "Kicking" are different, and their expert matching scores are also different.

⋆ equal contribution
⋆⋆ corresponding author

2 Quantitative analysis of expert selection

In the section above, we have shown how samples from similar classes tend to retrieve similar experts, while dissimilar classes retrieve different experts. In order to show that this trend holds for more samples in general, here we perform further quantitative analysis.

We first average the expert matching scores (for each of the $d \times M$ experts within an ERA module) over all samples from C_1, and represent it as a vector $\mathbf{s}^{C_1} \in \mathbb{R}^{d \times M}$, which can be interpreted as the average expert matching scores for a class C_1. The same is done for the samples from C_2 to obtain $\mathbf{s}^{C_2}$. Then, we calculate $r_{C_1,C_2} \in [-1, 1]$ as the cosine similarity between the vectors $\mathbf{s}^{C_1}$ and $\mathbf{s}^{C_2}$. The higher the r_{C_1,C_2} values between the two classes, the higher the tendency of samples from those classes to activate the same experts.

We analyse two action classes that look visually similar ("Punch/Slap" and "Point finger"), and one action class that looks very different ("Kicking"), and calculate r_{C_1,C_2} between those classes in Table 1. The similar actions "Punch/Slap" and "Point finger" have a high cosine similarity score of 0.83, compared to the dissimilar actions "Punch/Slap" and "Kicking" with a score of -0.03, or "Point finger" and "Kicking" with a score of -0.01, showing that similar classes tend to share the same experts.

Table 1. Computed cosine similarity scores r_{C_1,C_2} between action classes from NTU60 dataset. r_{C_1,C_2} values tend to be higher for action classes that are more similar (e.g. Punch/Slap vs Point finger), and tend to be lower for action classes that are less similar (e.g. Point finger vs Kicking).

	C_2 = Punch/Slap	C_2 = Point finger	C_2 = Kicking
C_1 = Punch/Slap	-	0.83	-0.03
C_1 = Point finger	0.83	-	-0.01
C_1 = Kicking	-0.03	-0.01	-

3 Latency during Testing

We display the latency during testing in Table 2, as measured on a Nvidia Geforce RTX 3090 GPU. Replacing basic convolutional modules with ERA modules only slightly increases the latency during testing. Note that performance in early action prediction is improved significantly with our method, with improvement of around 7 AUC points on NTU60 and NTU120, more than 5 AUC points on SYSU and more than 4 AUC points for UCF101.

4 NTU60 early action prediction experiment details

We follow the experimental settings of [2] for the NTU60 dataset. We describe it in more detail below.

Table 2. Latency (ms) of testing a batch of samples. The testing latency for Baseline and ERA-Net refer to latency incurred for one forward pass only. The Baseline model uses the backbone without ERA modules, while ERA-Net replaces 25% of convolutional layers with ERA modules.

Method	Latency (ms) of testing a batch of 64 samples
Baseline	621
ERA-Net	701

For each of the 10 settings (from 10% to 100%), we simply select the corresponding number of consecutive frames from the full action sequence as the partial sequence, starting from the first frame. During training, the observation ratio of each sample in each batch is sampled randomly from a uniform distribution (to select evenly from all the 10 settings). If a partial sequence has less than 300 frames, the sequence is repeated until 300 frames is reached. During evaluation, the different observation ratios are systematically tested according to each setting (so we fix the observation ratios of input sequences). Input partial sequences are also repeated in a similar manner, until there are 300 frames in each input sequence.

For training and testing, we follow [2] where 20 subjects are employed for training and the remaining 20 subjects are left for testing. We train a single model to predict action labels at all the different observation ratios, and also evaluate that model on all observation ratios.

5 UCF101 early action prediction experiment details

As stated in the paper, we follow the experimental settings of [4] for the UCF101 dataset. Below, we explain the procedures in more detail.

Firstly, from each full video, we obtain 10 clips of different lengths, ranging from only using the first 10% of frames, to using the full 100% of frames. During training, when the video is selected to be used as input data, one of the 10 clips is randomly selected instead. From the clip, 16 frames are uniformly sampled, and re-sized into a $3 \times 16 \times 112 \times 112$ tensor as input to the network. Notably, a single model is trained to predict action labels at different observation ratios. That single model is also evaluated on all observation ratios systematically.

For evaluation, we followed [4] and used the first 15 groups of videos in UCF101 for training, while 3.6k action videos were used for testing.

6 Algorithm of Expert Learning Rate Optimization

In Algorithm 1, we show a summary of our Expert Learning Rate Optimization method.

Algorithm 1: Expert Learning Rate Optimization

Input: Training data $\mathcal{D}$, batch size B, model learning rate α;
Output: Trained network θ;
Initialize network θ with non-expert parameters w and experts $\mathcal{E}$ randomly.
Initialize set of expert learning rate parameters β, with each element set to α;
while *not converge* **do**
  (1) Virtual Training:
  $\{x_j^{tra}, y_j^{tra}\}_{j=1}^{B} = \text{SampleMiniBatch}(\mathcal{D})$;
  Calculate loss $\mathcal{L}_{tra} = \frac{1}{B}\sum_{j=1}^{B} \mathcal{L}(w, \mathcal{E}; x_j^{tra}, y_j^{tra})$;
  Update non-experts $\hat{w} = w - \alpha \nabla_w \mathcal{L}_{tra}$;
  With β frozen, update experts $\hat{\mathcal{E}} = \mathcal{E} - \beta \nabla_{\mathcal{E}} \mathcal{L}_{tra}$;
  (2) Meta-Expert Optimization:
  $\{x_j^{val}, y_j^{val}\}_{j=1}^{B} = \text{SampleMiniBatch}(\mathcal{D})$;
  Calculate loss $\mathcal{L}_{val} = \frac{1}{B}\sum_{j=1}^{B} \mathcal{L}(\hat{w}, \hat{\mathcal{E}}; x_j^{val}, y_j^{val})$;
  With w and $\mathcal{E}$ frozen, update $\beta' = \beta - \alpha \nabla_{\beta} \mathcal{L}_{val}$;
  (3) Model Training:
  Update non-experts $w' = w - \alpha \nabla_w \mathcal{L}_{tra}$;
  Update experts $\mathcal{E}' = \mathcal{E} - \beta' \nabla_{\mathcal{E}} \mathcal{L}_{tra}$;
  Set $w = w', \mathcal{E} = \mathcal{E}', \beta = \beta'$
end

7 Replacement of Layers in Backbones

We select convolutional layers in the backbone models to replace with our ERA modules. In this section, we explain more in detail which convolutional layers were selected for replacement. For both of our backbones, we select convolutional layers uniformly across the backbone, in order to learn subtle cues at different levels (i.e. to learn low, mid, and high level subtle cues).

The 2s-AGCN [3] backbone model consists of two AGCN streams that utilize two types of information (joint and bone). Both streams of 2s-AGCN consist of stacked TCN-GCN blocks. We thus pick one block from every 4 TCN-GCN blocks, and replace the convolutional layers within that block with our ERA module.

The 3D ResNeXt-101 [1] backbone model uses 3D ResNeXt blocks as a basic unit of its architecture. Thus, we select one block every 4 ResNext blocks, and the convolutional layers within that block are replaced with our ERA module.

References

1. Hara, K., Kataoka, H., Satoh, Y.: Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 6546–6555 (2018) 4
2. Li, T., Liu, J., Zhang, W., Duan, L.: Hard-net: Hardness-aware discrimination network for 3d early activity prediction. In: European Conference on Computer Vision. pp. 420–436. Springer (2020) 2, 3
3. Shi, L., Zhang, Y., Cheng, J., Lu, H.: Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12026–12035 (2019) 4
4. Wang, X., Hu, J.F., Lai, J.H., Zhang, J., Zheng, W.S.: Progressive teacher-student learning for early action prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3556–3565 (2019) 3