System-status-aware Adaptive Network for Online Streaming Video Understanding

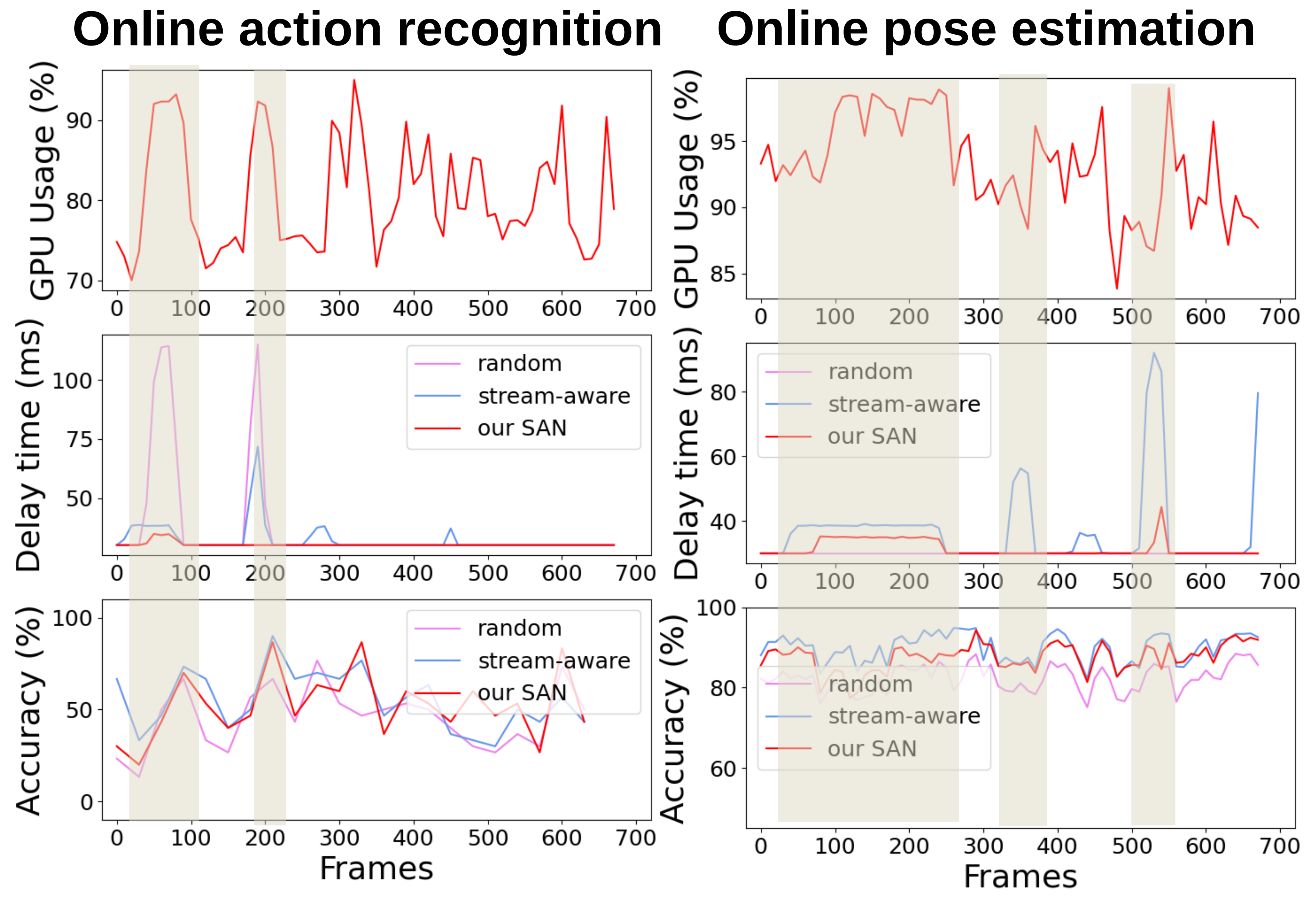
Lin Geng Foo[1]*, Jia Gong[1]*, Zhipeng Fan[2], and Jun Liu[1]
1 Singapore University of Technology and Design; 2 New York University; * Equal contribution

JUNE 18-22, 2023 CVPR VANCOUVER, CANADA

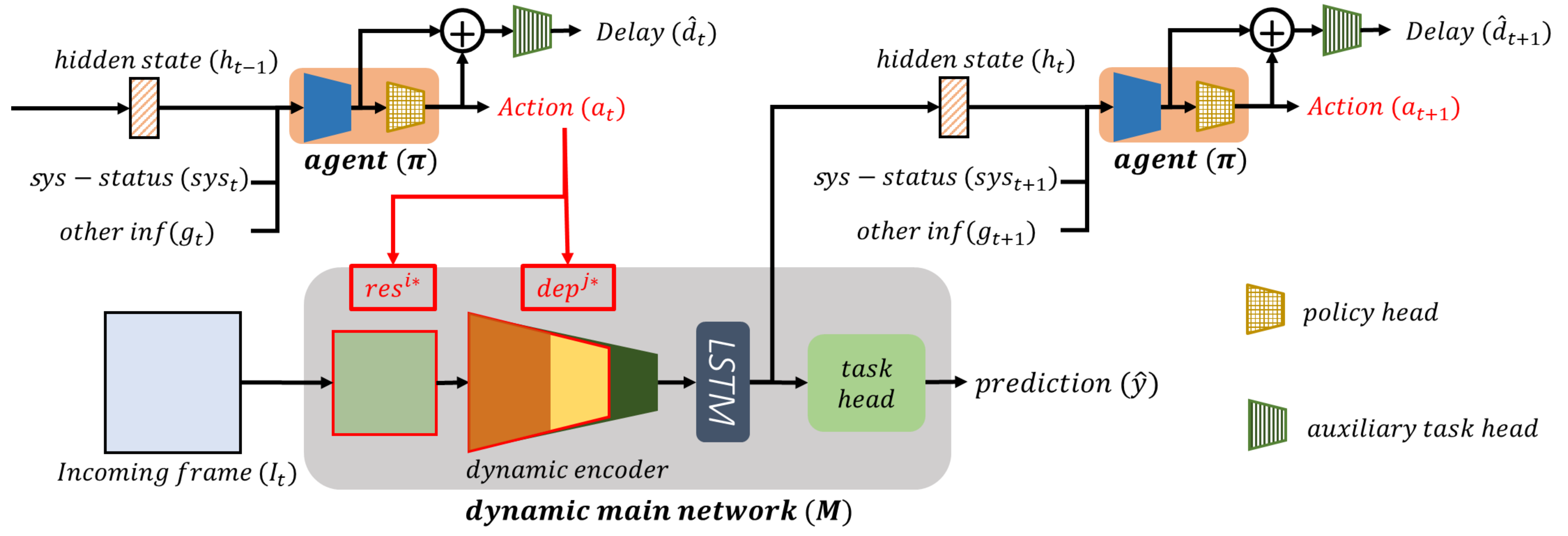
Motivation and Overview

- **Motivation:** To provide reliable yet timely responses for online video understanding tasks in a real-world setting, it is necessary to investigate the model's performance **under dynamic computational resources**. However, most works do not explicitly consider this problem.
- **Our method:** We propose (1) a **System-status-aware Adaptive Network** that considers **the device's real-time state**, improving efficiency and robustness to fluctuations of the system status; (2) a **Meta Self-supervised Adaptation (MSA)** method that adapts the policy to new hardware configurations at test-time. allowing for easy deployment of the model onto other **unseen** hardware platforms.

Impact of Dynamic System Status



Our Framework



System-status-aware Adaptive Network (SAN):

- **Dynamic main network:** We design a dynamic encoder that has both dynamic depth and dynamic resolution. Our dynamic depth mechanism gives the encoder the option of producing output features at shallower intermediate layers without having to wait for all layers to be executed. The dynamic resolution mechanism allows our encoder to selectively take in input images at a lower resolution.
- **RL-based agent:** The agent controls the dynamic main network by generating a frame-level processing policy (res_t, dep_t) at each time t, which aims to maintain the task-related accuracy at a high level with low delay

Meta Self-supervised Adaptation (MSA):

We first propose an **auxiliary task (Delay Prediction)** that can fine-tune the model on target platforms, where supervision signals can be obtained on-the-fly. Then, to better bind the quality of the developed policy to the delay prediction performance, we **meta optimize** agent π such that the update based on the delay prediction error leads to better adaptation of the policy. Specifically, we denote ***the auxiliary task as the meta-train task and the agent's policy generation as the meta-test task***. Our meta-optimization algorithm aims to learn a good initialization of agent's parameters, such that when it is updated via the meta-train task, it can achieve good performance on the meta-test task.

Experimental Results

Online action recognition (on Salad 50):

Method	Accuracy	Max Delay	Mean Delay	R\T Frames
RA [30]	42.6 %	76.6 ms	33.9 ms	79.1 %
DDLSTM [31]	41.1 %	407.4 ms	110.3 ms	5.9 %
LiteEval [40]	40.3 %	110.7 ms	68.9 ms	30.4 %
AR-Net [25]	40.9 %	91.1 ms	56.2 ms	62.1%
main network				
+ random policy	46.2%	80.5 ms	34.1 ms	92.1%
+ stream-aware	**55.4** %	80.8 ms	40.3 ms	82.0%
+ SAN	53.8%	**49.4** ms	**29.9** ms	**95.4**%

Online pose estimation (on Sub-JHMDB):

Method	Accuracy	Max Delay	Mean Delay	R\T Frames
DKD [29]	94.0	133.8 ms	47.9 ms	59.9 %
SimpleBaseline [41]	94.4	129.6 ms	45.5 ms	63.8 %
Skip-Convolution [12]	95.1	115.7 ms	51.2 ms	53.7 %
main network				
+ random policy	86.9	130.9 ms	36.1 ms	72.9%
+ stream-aware	**95.3**	126.1 ms	44.7 ms	63.7%
main network + SAN	93.4	**41.2** ms	**33.1** ms	**81.4**%

Across platforms (for pose estimation):

Method	a+b → c		
	Accuracy	Mean Delay	R\T Frames
Fully-supervised	93.0 %	28.3 ms	87.4 %
Direct transfer	87.3 %	28.4 ms	82.4 %
Feature alignment	90.8 %	29.0 ms	80.4 %
Our self-supervised	91.1 %	29.8 ms	79.8 %
Our MSA	92.8 %	29.7 ms	86.1 %

1) **Device a**: a laptop with an AMD Ryzen Threadripper 2950X and an NVIDIA Geforce GTX 1080 GPU (11 GB).2) **Device b**: a desktop with an Intel(R) Core(TM) i9-10900 and an NVIDIA Geforce RTX 2080 GPU (12 GB).3) **Device c**: a workstation with an AMD Ryzen Threadripper 3960X and an NVIDIA Geforce RTX 3090 GPU (24 GB).