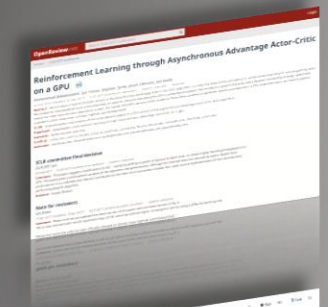
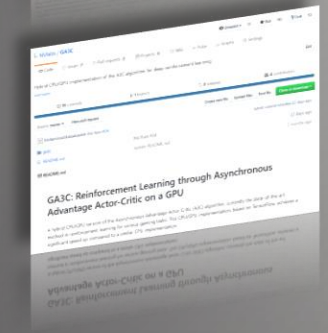
GPU-BASED A3C FOR DEEP REINFORCEMENT LEARNING

M. Babaeizadeh[†,‡]  , I.Frosio[‡]  , S.Tyree[‡]  , J. Clemons[‡]  , J.Kautz[‡] 

[†]University of Illinois at Urbana-Champaign, USA [‡] NVIDIA, USA



An ICLR 2017 paper



A github project

NVIDIA

2007

*Image from xkcd.com

NVIDIA

2017

*Image from xkcd.com

NVIDIA

GPU-BASED A3C FOR DEEP REINFORCEMENT LEARNING

GPU-BASED A3C FOR DEEP REINFORCEMENT LEARNING

Learning to accomplish a task

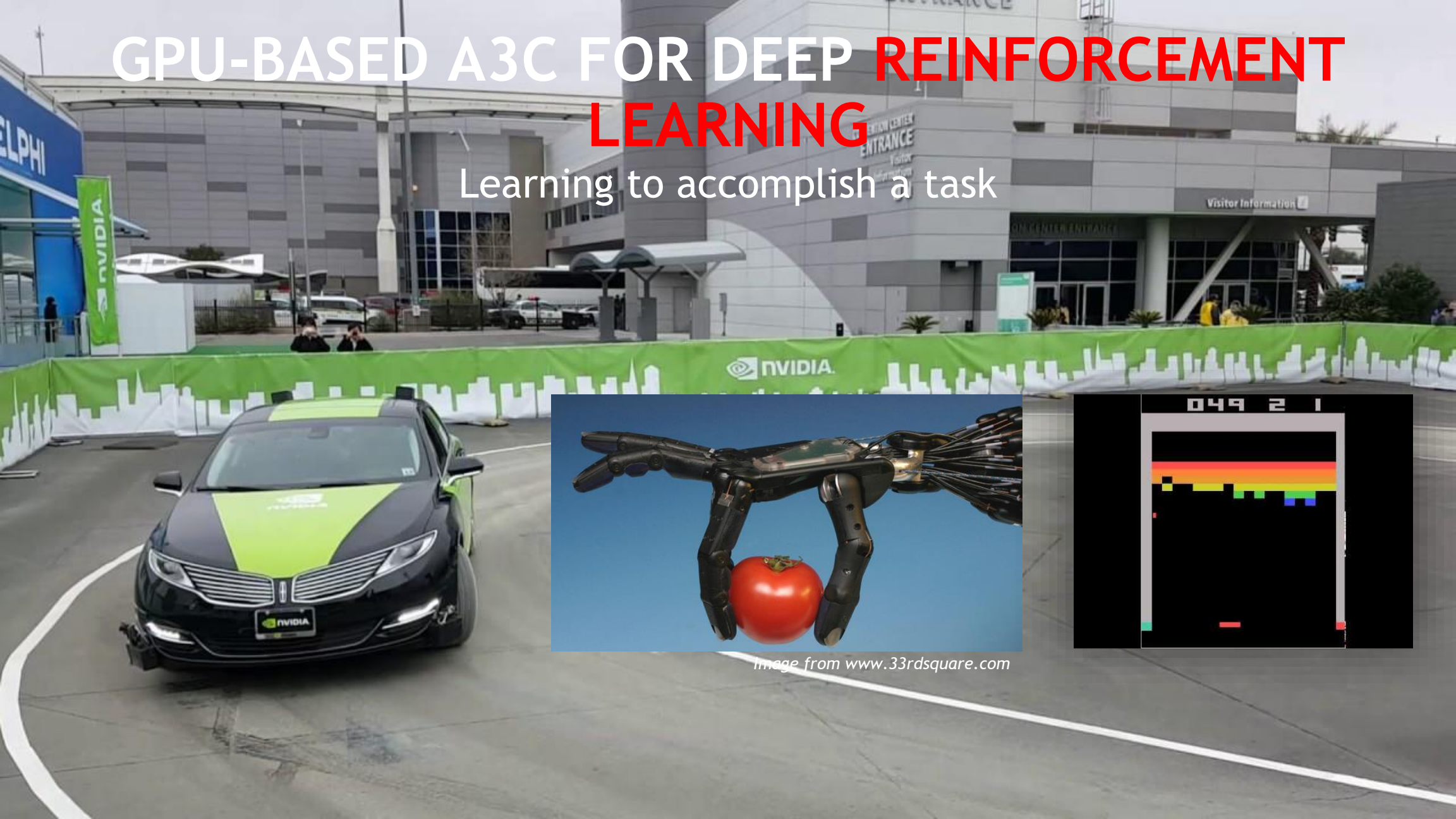


Image from www.33rdsquare.com

GPU-BASED A3C FOR DEEP REINFORCEMENT LEARNING

Definitions

S_t, R_t

RL agent

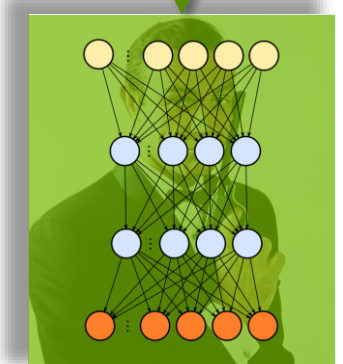
$a_t = \pi(S_t)$

- ✓ Environment
- ✓ Agent
- ✓ Observable status S_t
- ✓ Reward R_t
- ✓ Action a_t
- ✓ Policy $a_t = \pi(S_t)$

GPU-BASED A3C FOR DEEP REINFORCEMENT LEARNING

Definitions

S_t, R_t

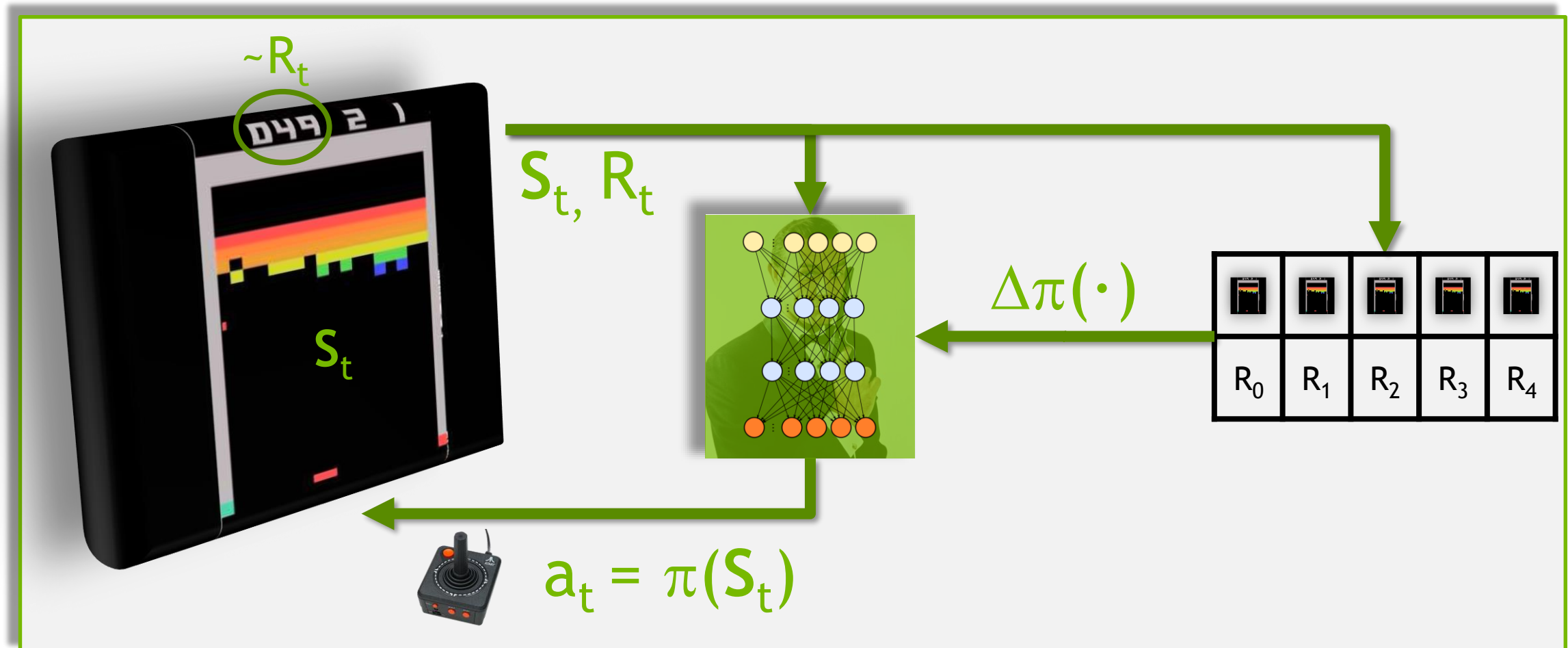


Deep RL agent

$a_t = \pi(S_t)$

GPU-BASED A3C FOR DEEP REINFORCEMENT LEARNING

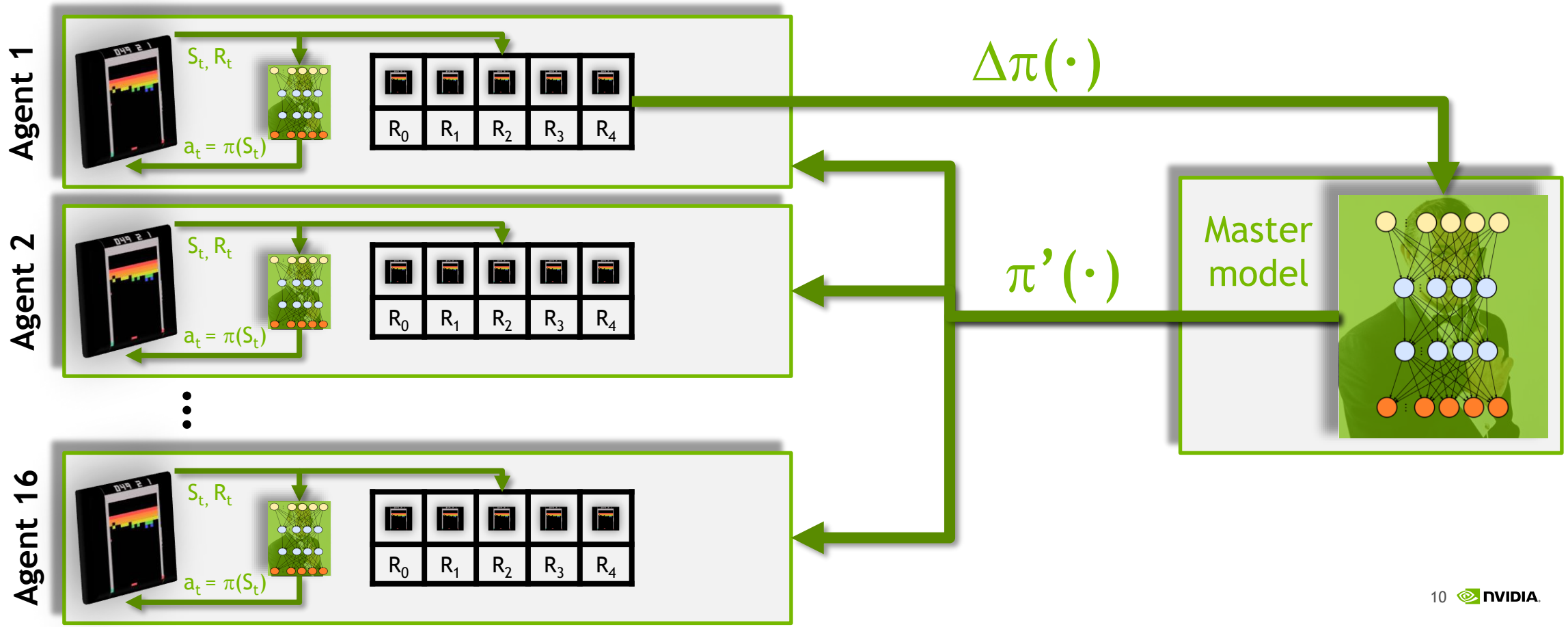
Definitions



GPU-BASED A3C FOR DEEP REINFORCEMENT LEARNING

GPU-BASED A3C FOR DEEP REINFORCEMENT LEARNING

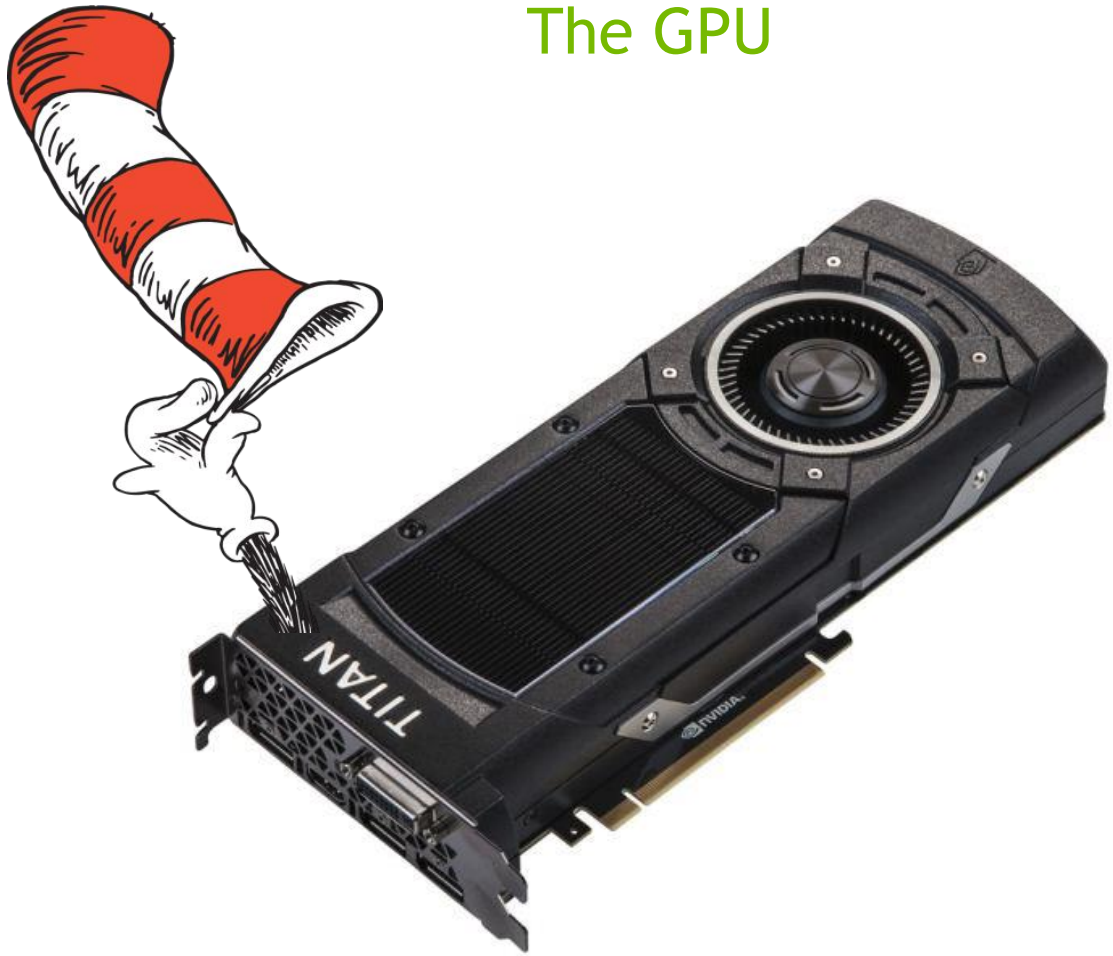
Asynchronous Advantage Actor-Critic (Mnih et al., arXiv:1602.01783v2, 2015)



GPU-BASED A3C FOR DEEP REINFORCEMENT LEARNING

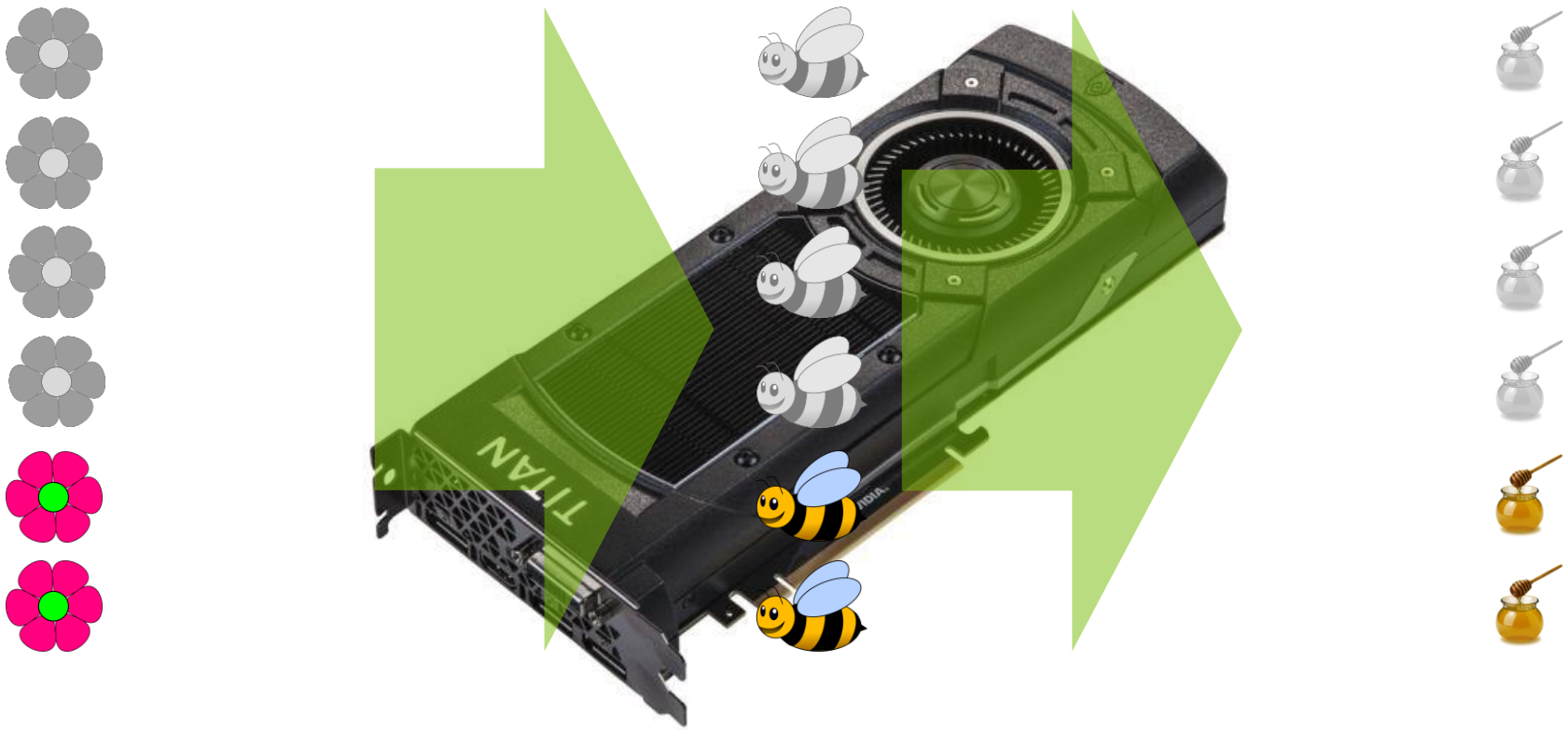
GPU-BASED A3C FOR DEEP REINFORCEMENT LEARNING

The GPU



GPU-BASED A3C FOR DEEP REINFORCEMENT LEARNING

LOW OCCUPANCY (33%)

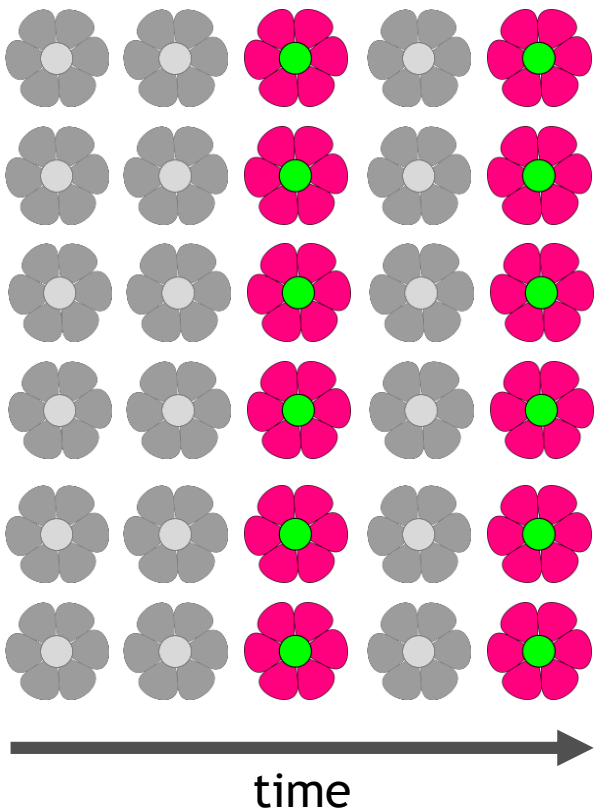


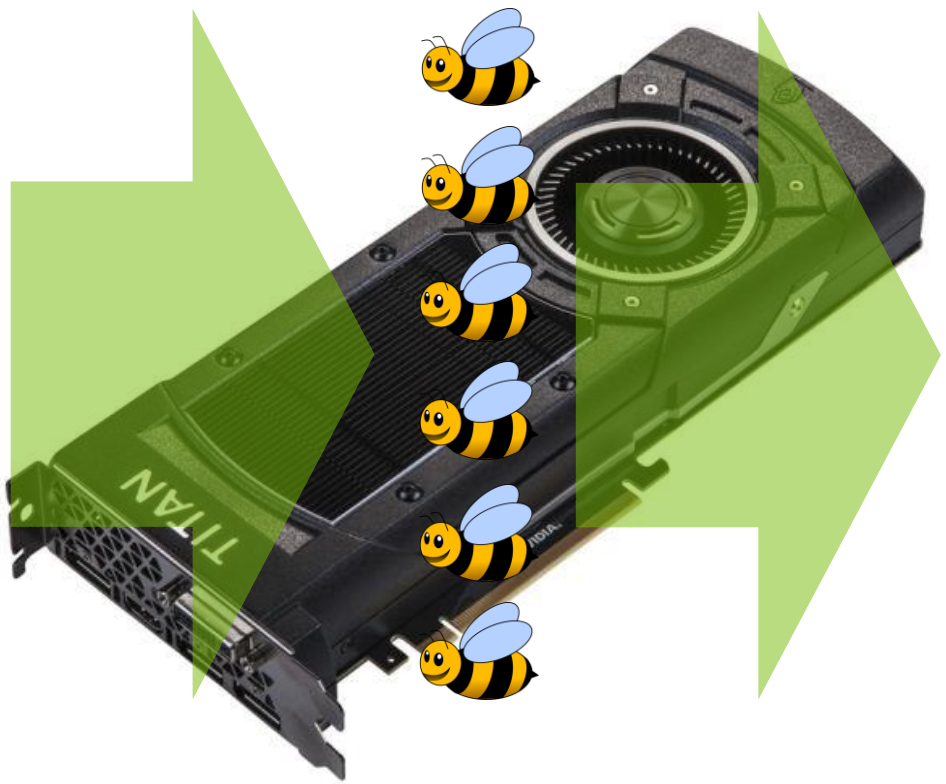
GPU-BASED A3C FOR DEEP REINFORCEMENT LEARNING

HIGH OCCUPANCY (100%)

GPU-BASED A3C FOR DEEP REINFORCEMENT LEARNING

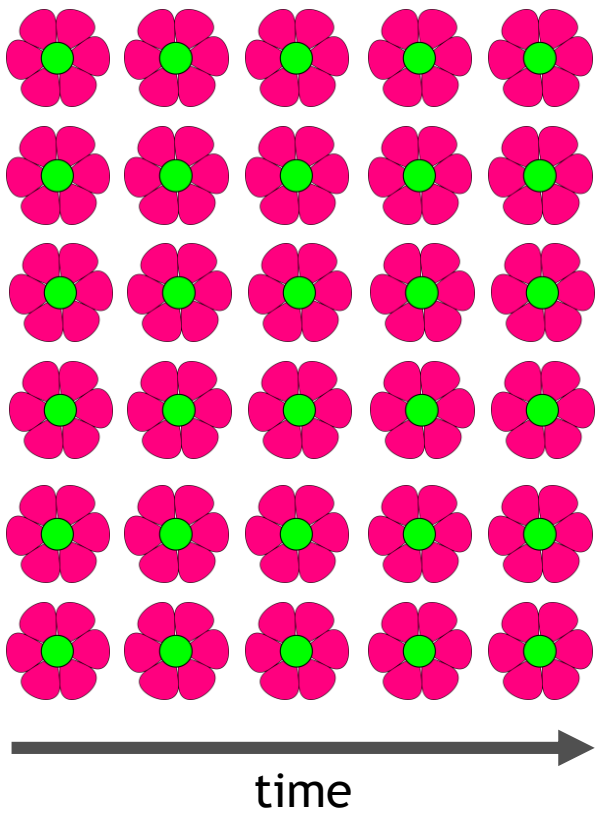
HIGH OCCUPANCY (100%), LOW UTILIZATION (40%)

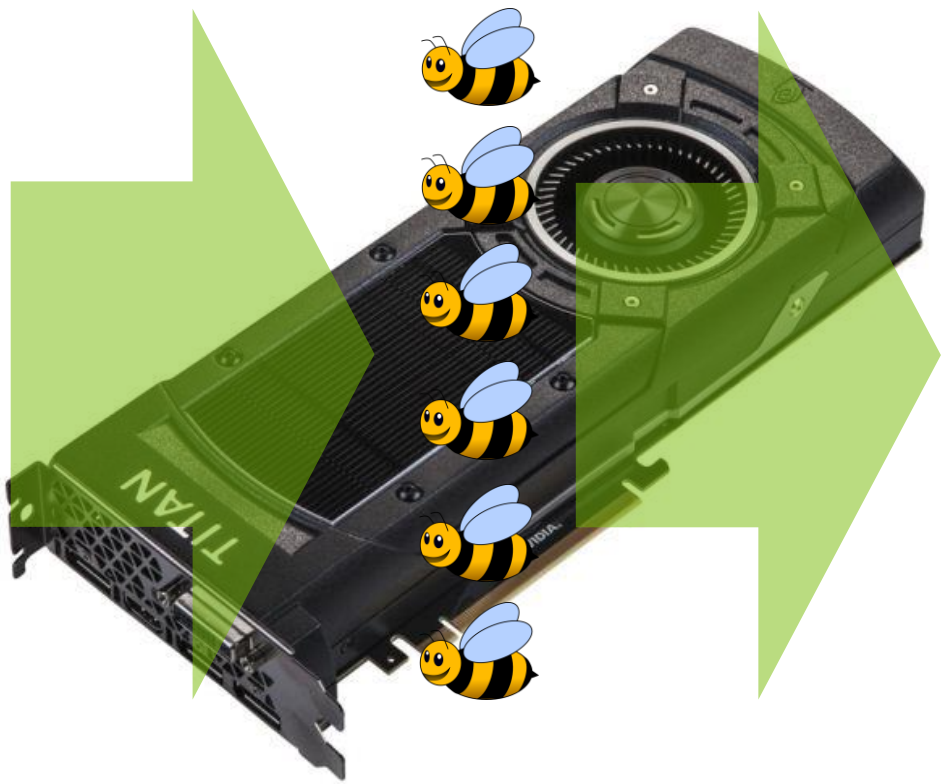




GPU-BASED A3C FOR DEEP REINFORCEMENT LEARNING

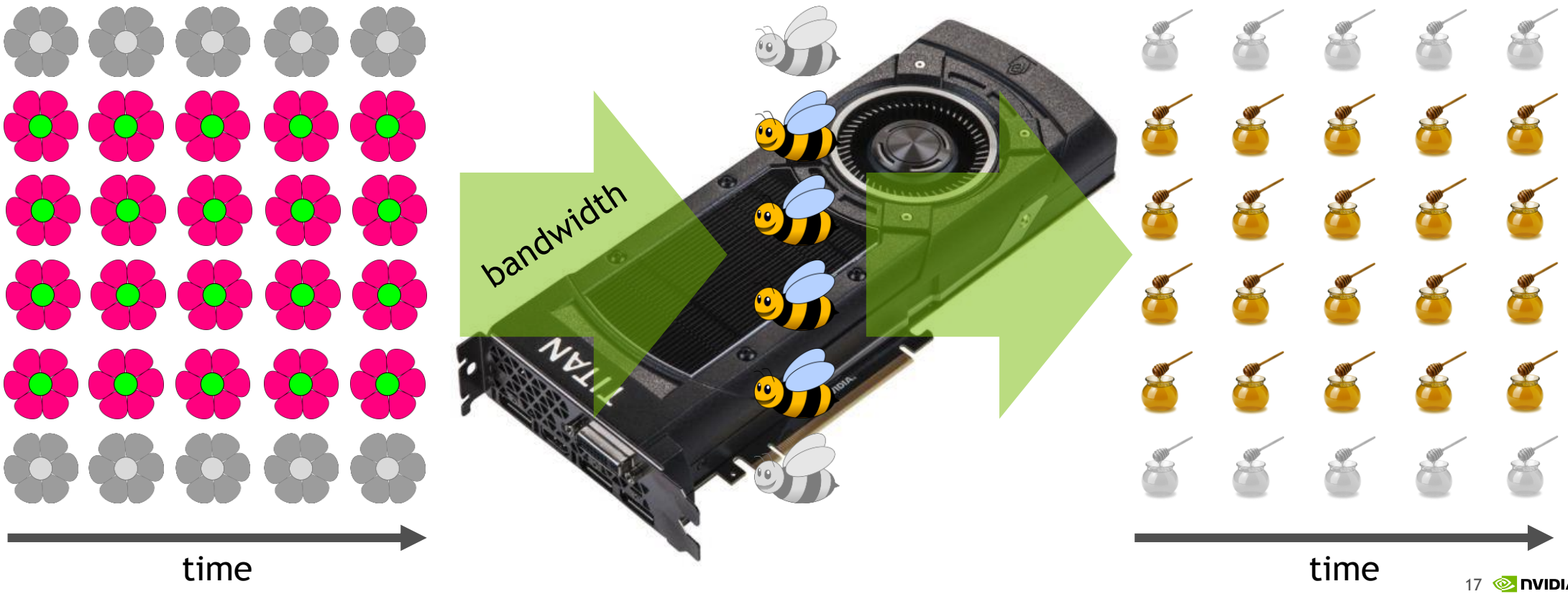
HIGH OCCUPANCY (100%), HIGH UTILIZATION (100%)

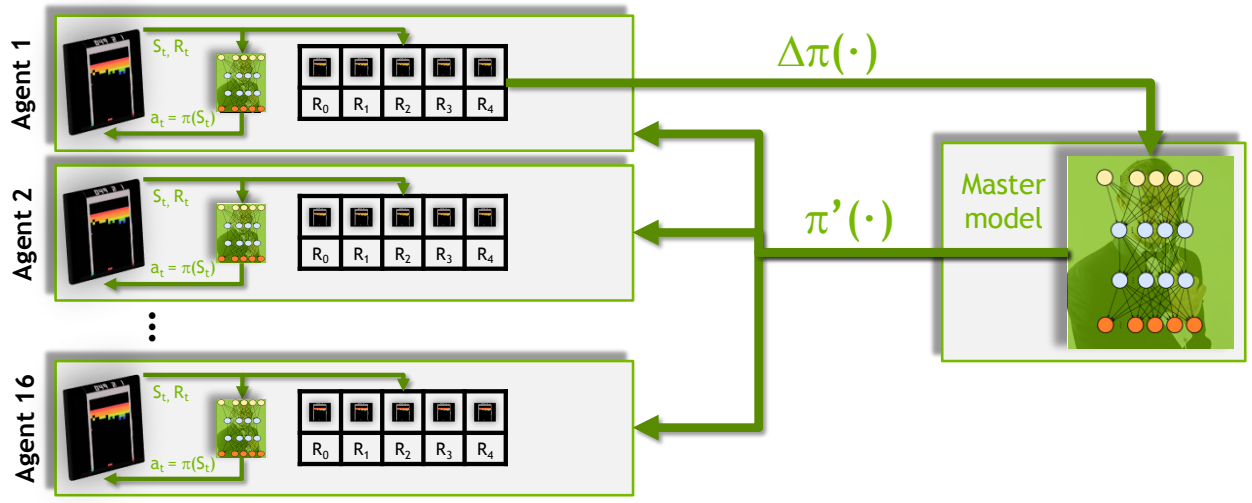




GPU-BASED A3C FOR DEEP REINFORCEMENT LEARNING

BANDWIDTH LIMITED




Agent 1
Agent 2
Agent 16
S_t, R_t
$a_t = \pi(S_t)$
R_0 R_1 R_2 R_3 R_4
$\Delta\pi(\cdot)$
$\pi'(\cdot)$
Master model


TITAN

MAPPING DEEP PROBLEMS TO A GPU

REGRESSION, CLASSIFICATION, …

data

100% utilization / occupancy

Pear, pear, pear, pear, …
Empty, empty, …
Fig, fig, fig, fig, fig, fig,
Strawberry, Strawberry,
Strawberry, …
…

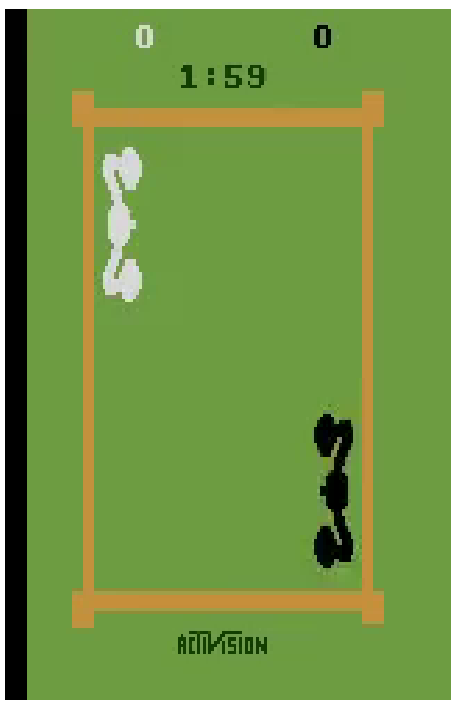
labels

REINFORCEMENT LEARNING

status, reward

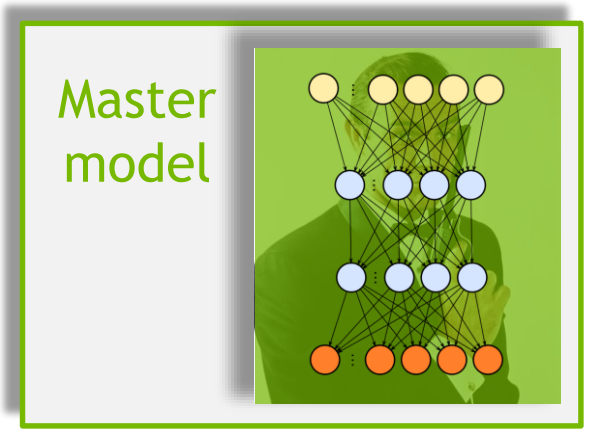
?

action



A3C

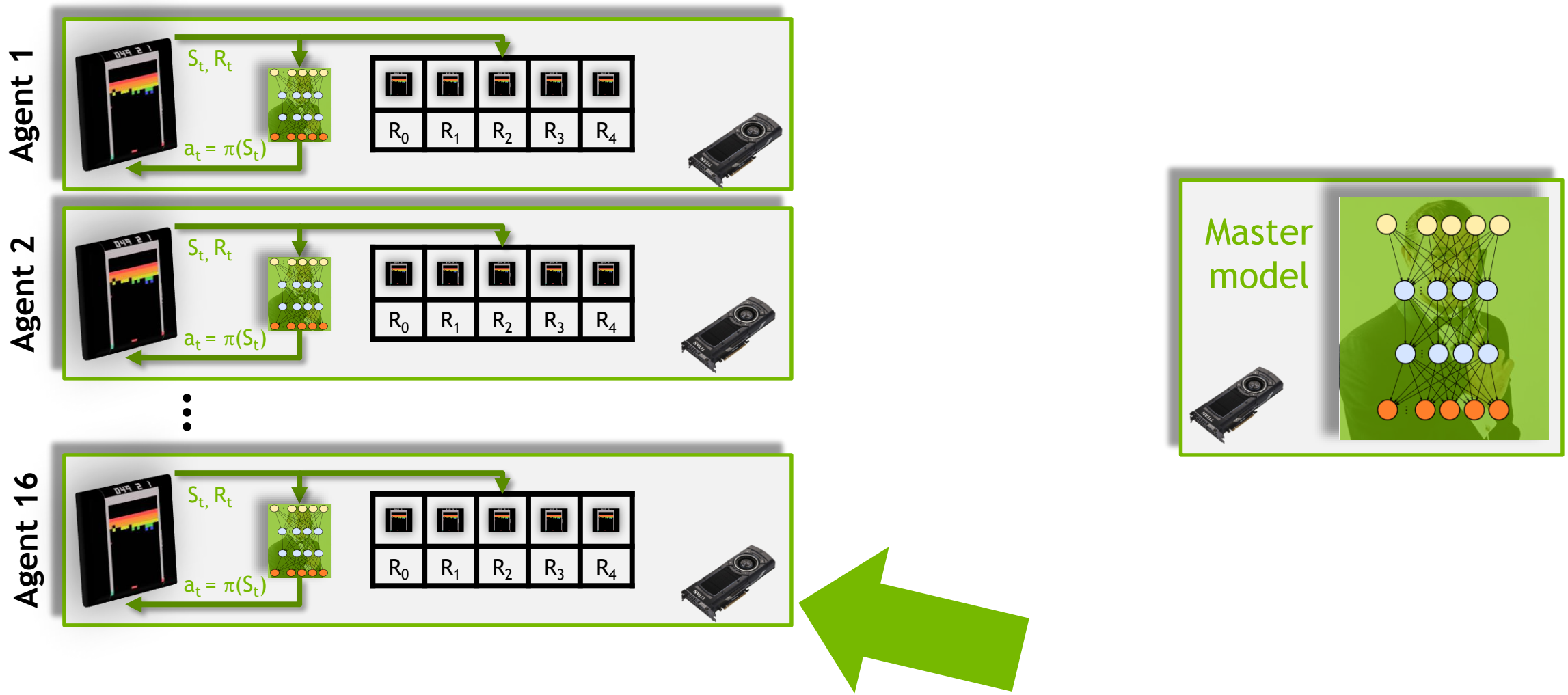

Agent 1
S_t, R_t
$a_t = \pi(S_t)$
R_0 R_1 R_2 R_3 R_4
Agent 2
S_t, R_t
$a_t = \pi(S_t)$
R_0 R_1 R_2 R_3 R_4
Agent 16
S_t, R_t
$a_t = \pi(S_t)$
R_0 R_1 R_2 R_3 R_4


Master model

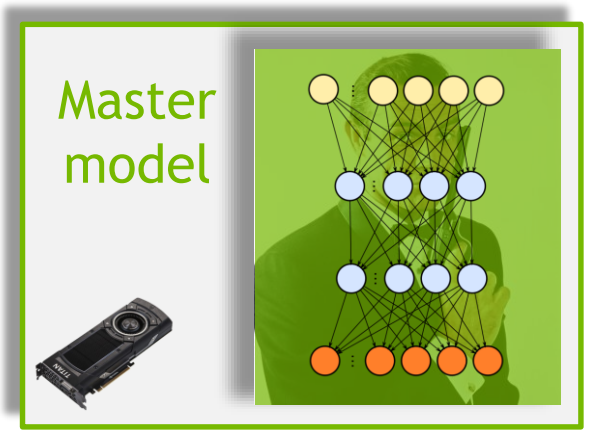
A3C


Agent 1
Agent 2
Agent 16
S_t, R_t
$a_t = \pi(S_t)$
R_0 R_1 R_2 R_3 R_4
CPU
Master model

A3C

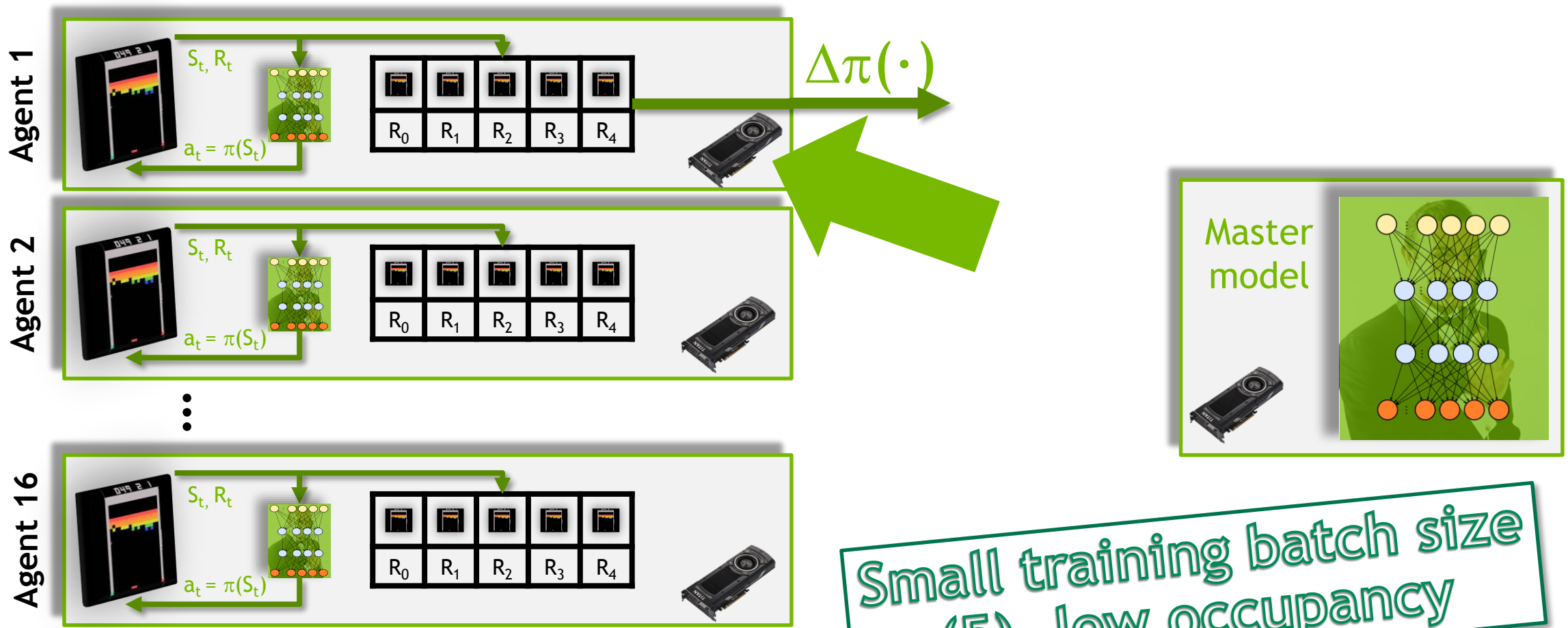

Agent 1
Agent 2
Agent 16
S_t, R_t
$a_t = \pi(S_t)$
R_0 R_1 R_2 R_3 R_4
Master model

A3C



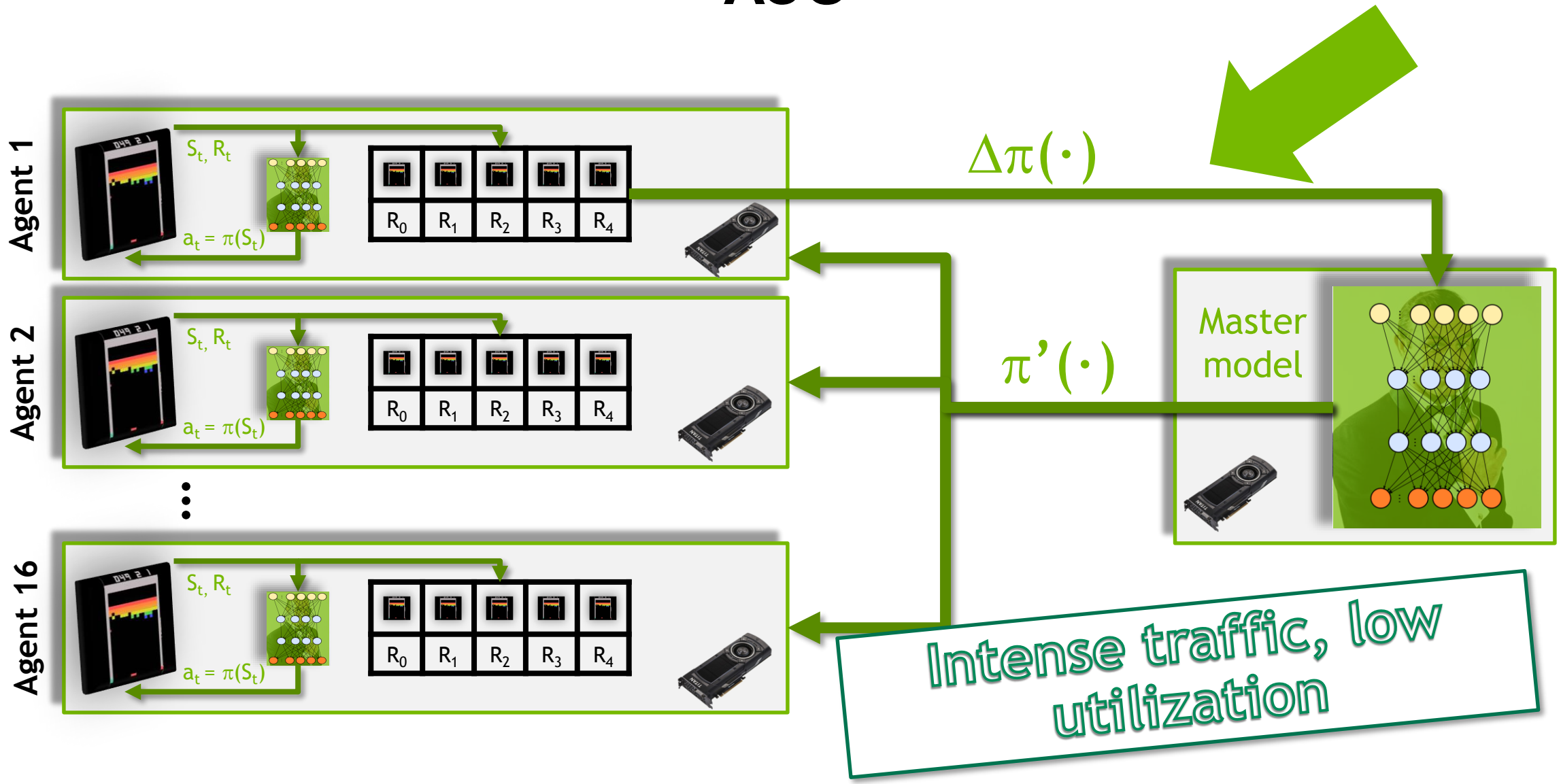
Small inference batch size (1), low occupancy

A3C



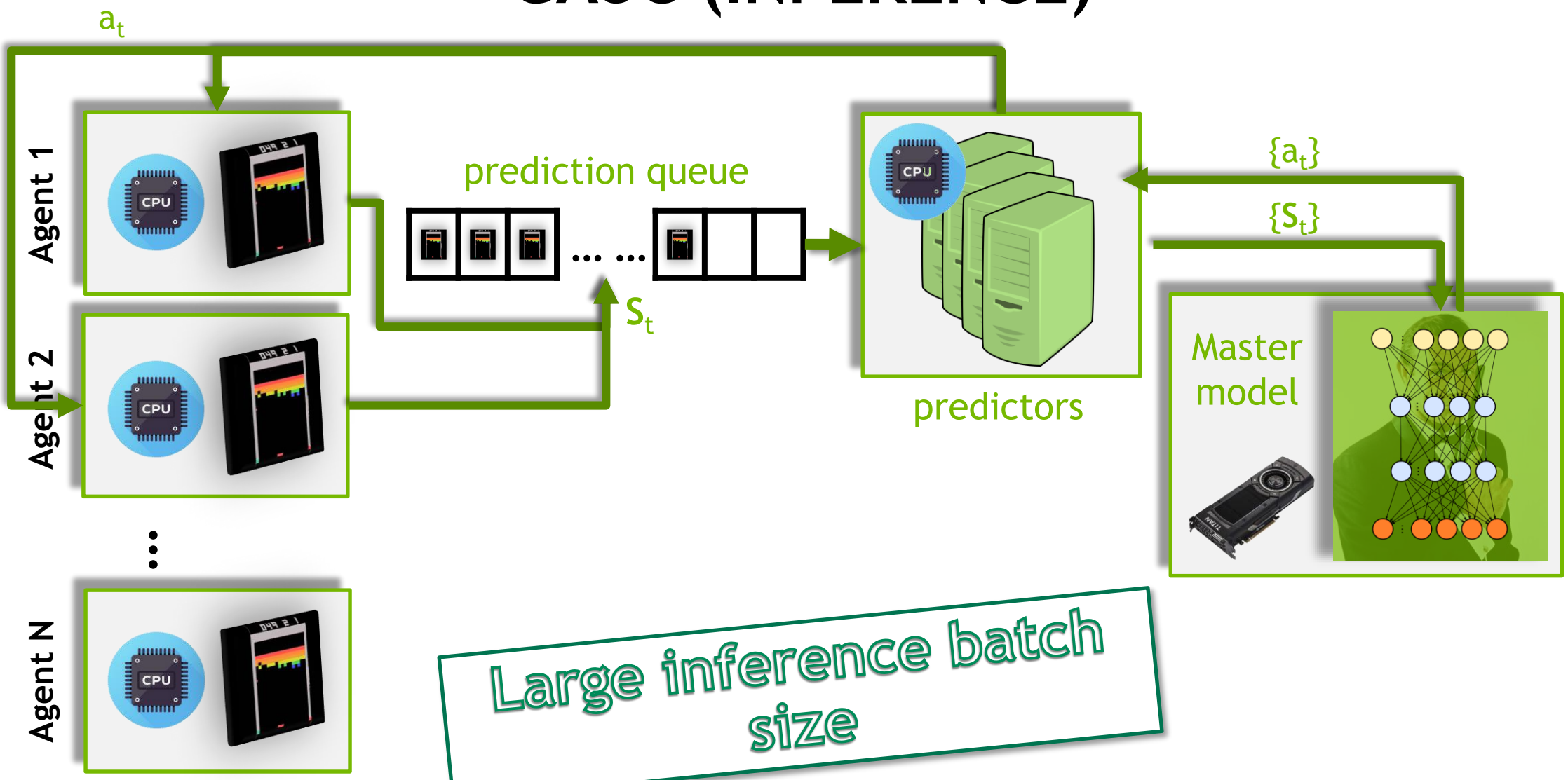
Small training batch size (5), low occupancy

A3C


Agent 1
Agent 2
Agent 16
S_t, R_t
$a_t = \pi(S_t)$
R_0 R_1 R_2 R_3 R_4
$\Delta\pi(\cdot)$
$\pi'(\cdot)$
Master model
Intense traffic, low utilization

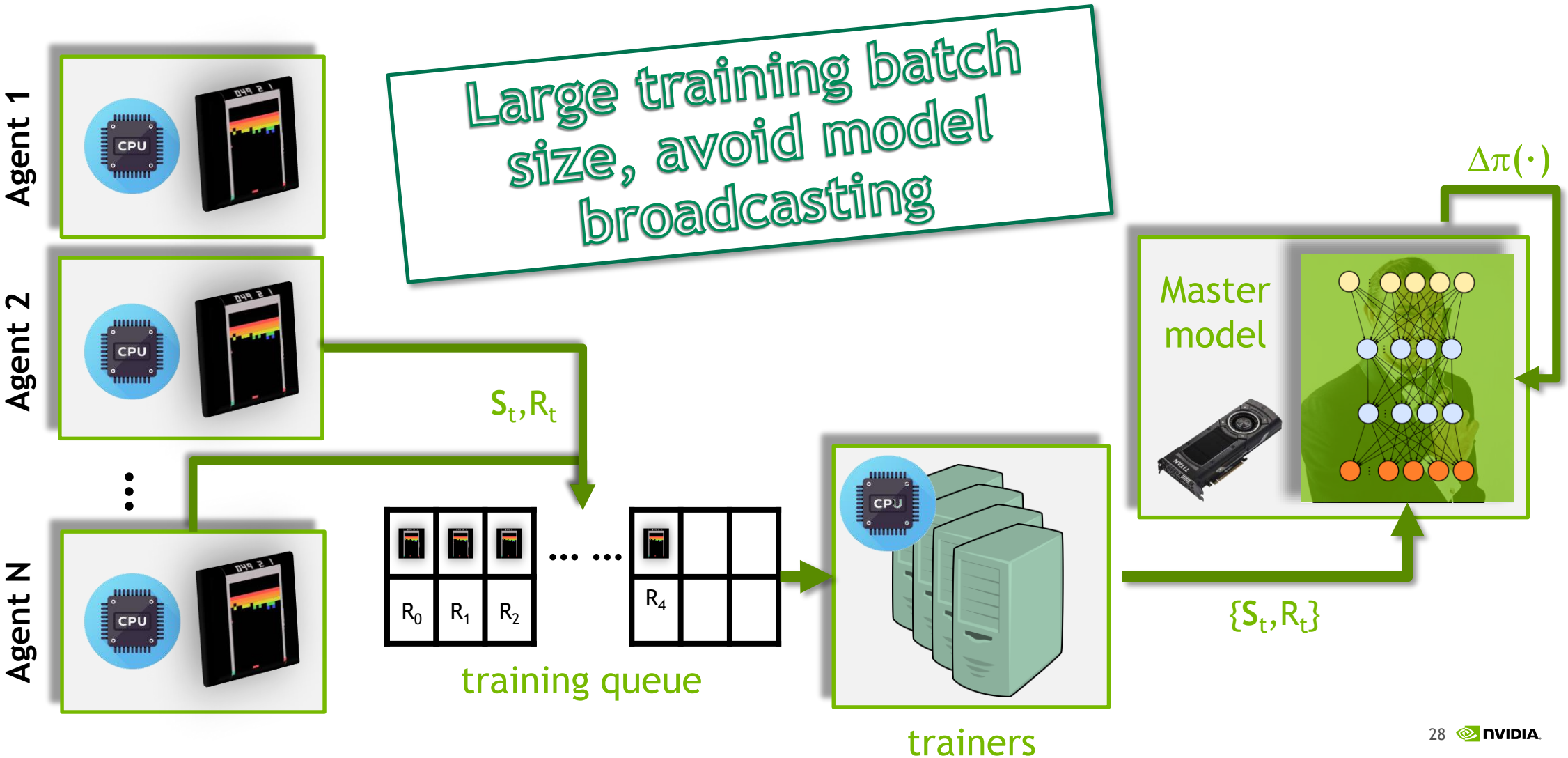
El Capitan big wall, Yosemite Valley

GA3C (INFERENCE)

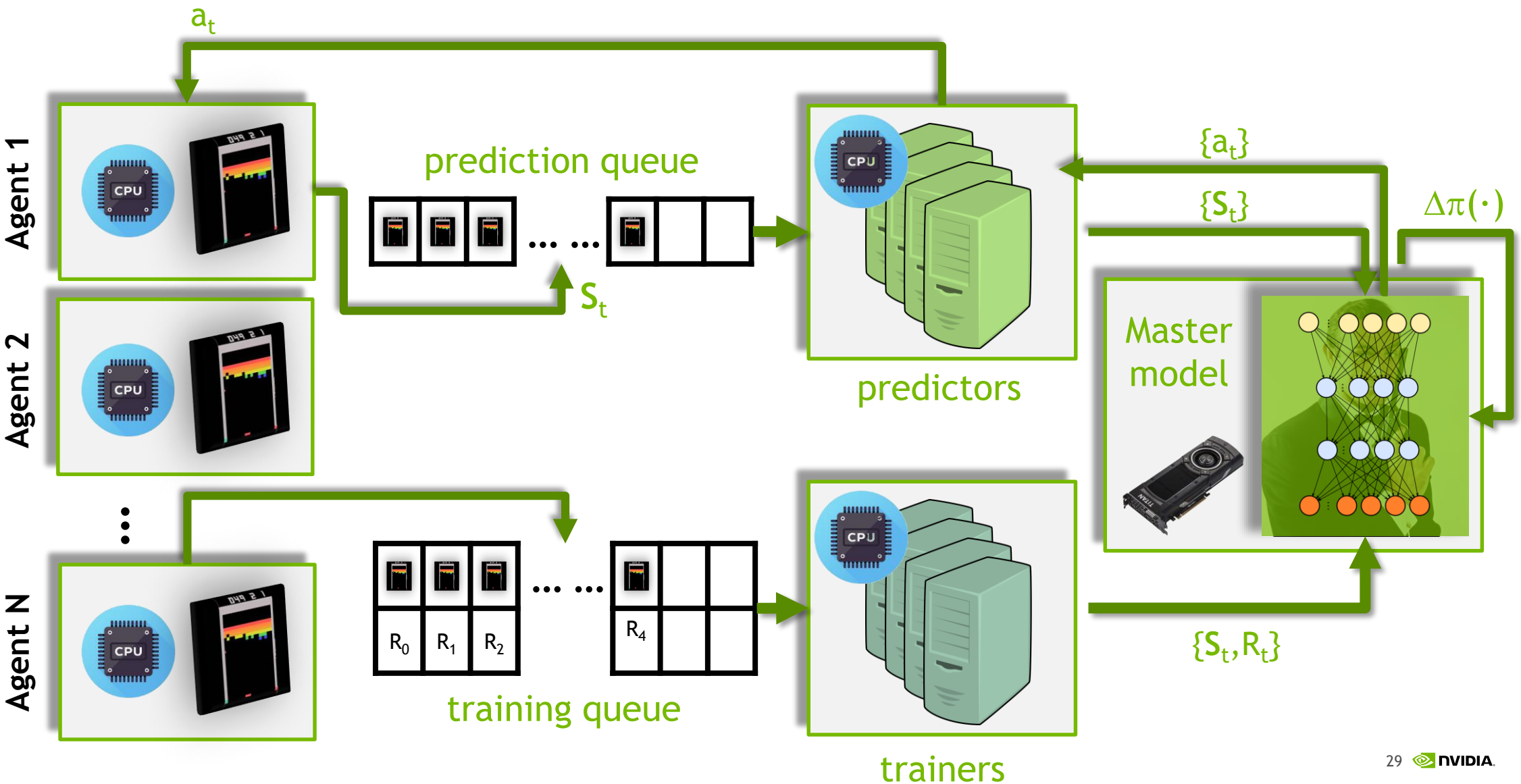


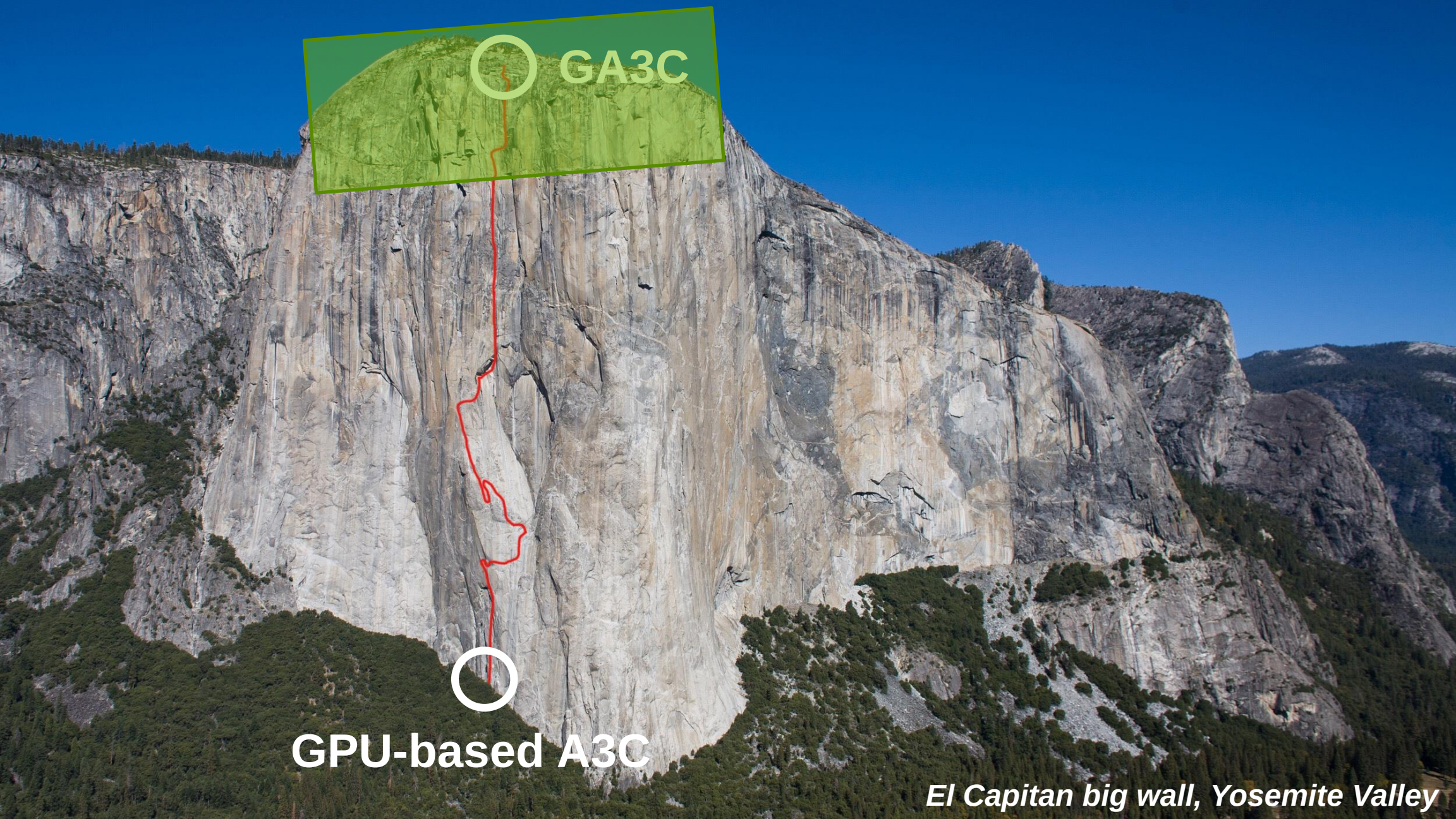
Large inference batch size

GA3C (TRAINING)

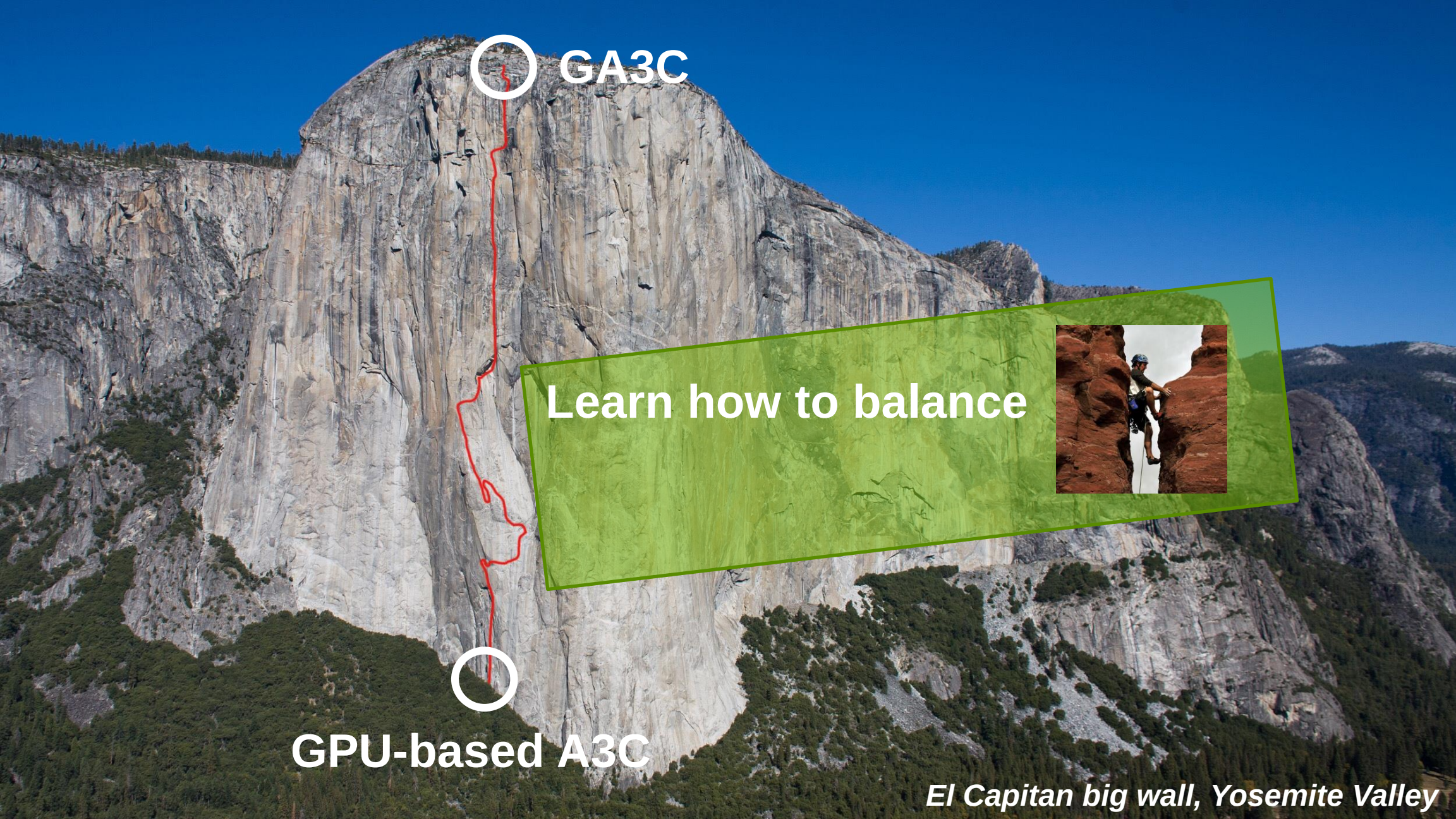


GA3C

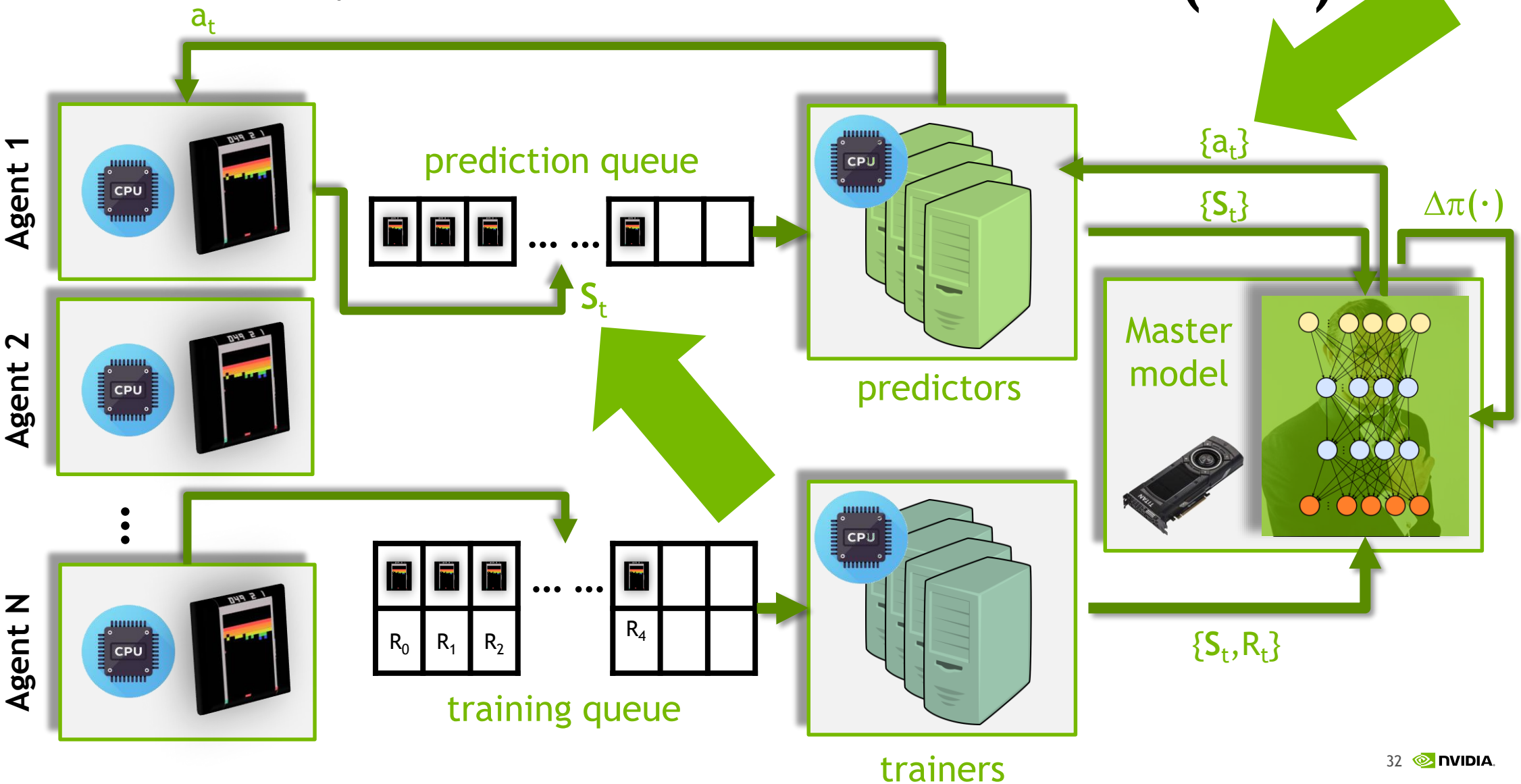

a_t
Agent 1
Agent 2
Agent N
CPU
prediction queue
s_t
predictors
$\{a_t\}$
$\{s_t\}$
$\Delta\pi(\cdot)$
Master model
R_0
R_1
R_2
R_4
training queue
trainers
$\{s_t, R_t\}$



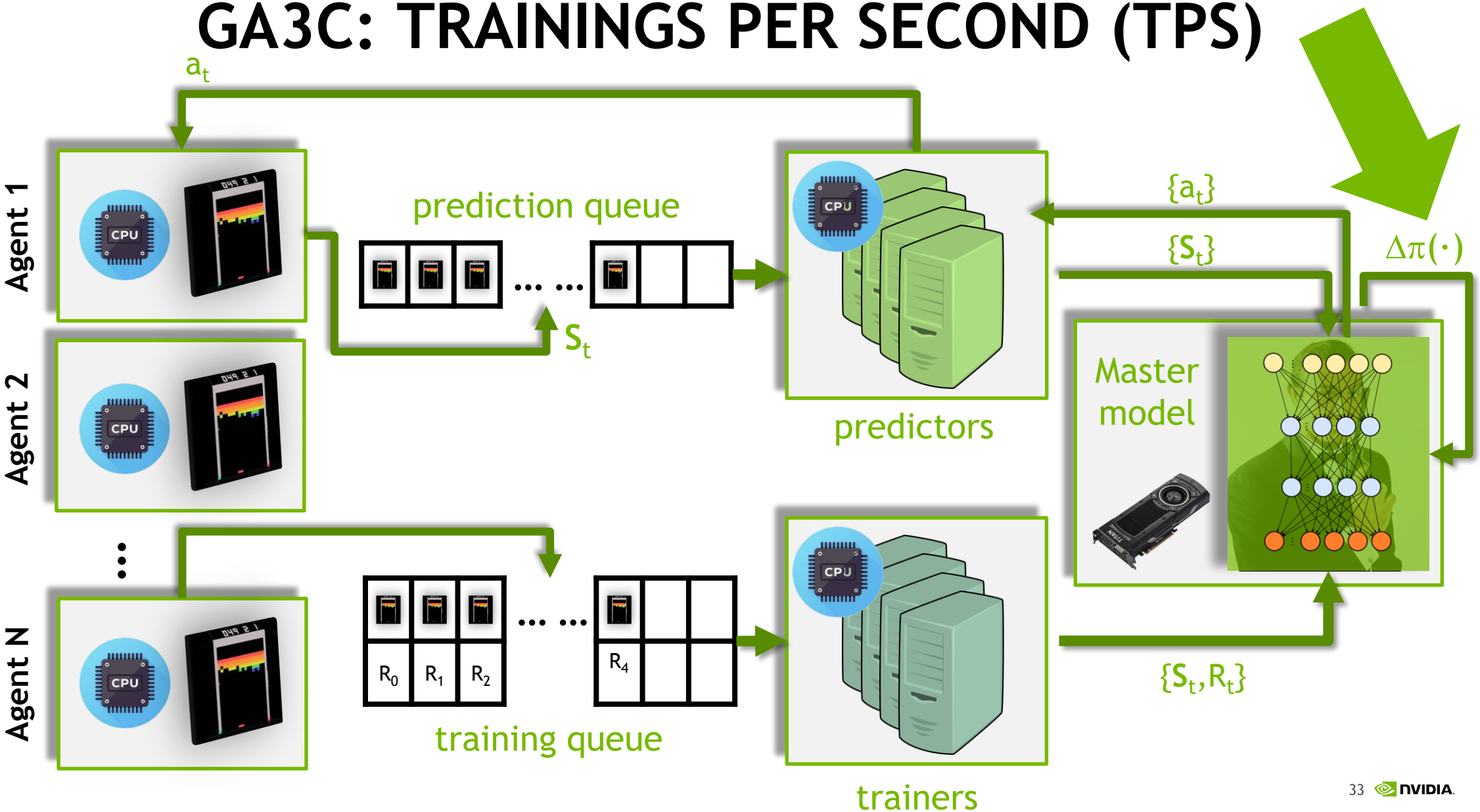
El Capitan big wall, Yosemite Valley


GA3C
Learn how to balance
GPU-based A3C
El Capitan big wall, Yosemite Valley

GA3C: PREDICTIONS PER SECOND (PPS)



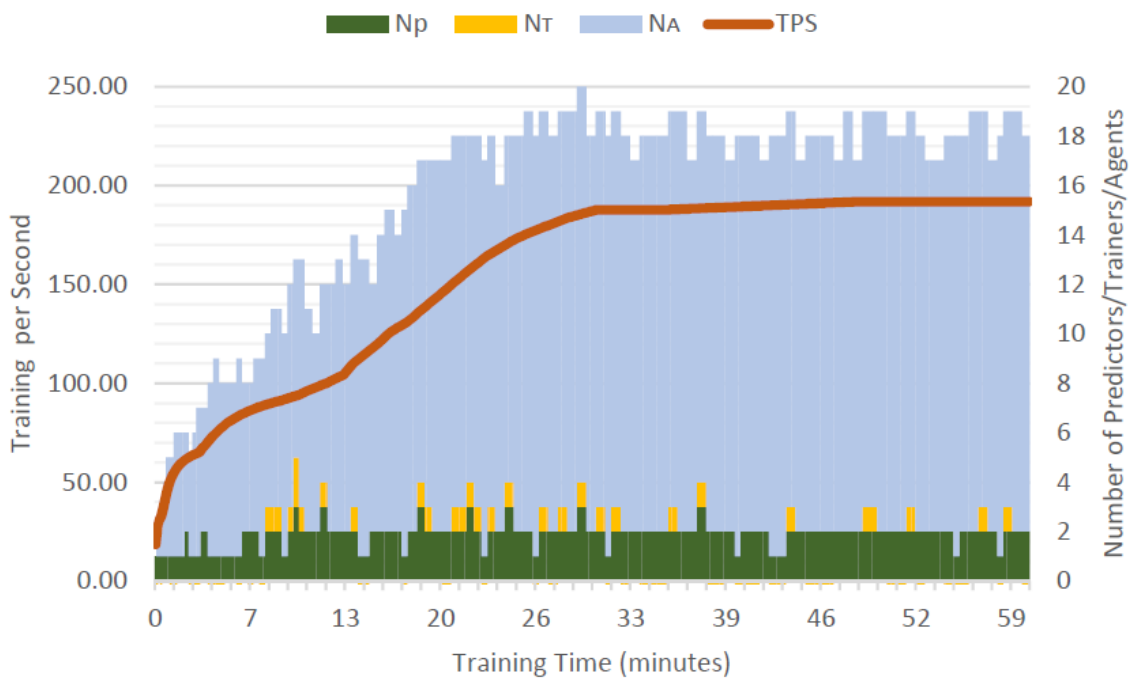
GA3C: TRAININGS PER SECOND (TPS)



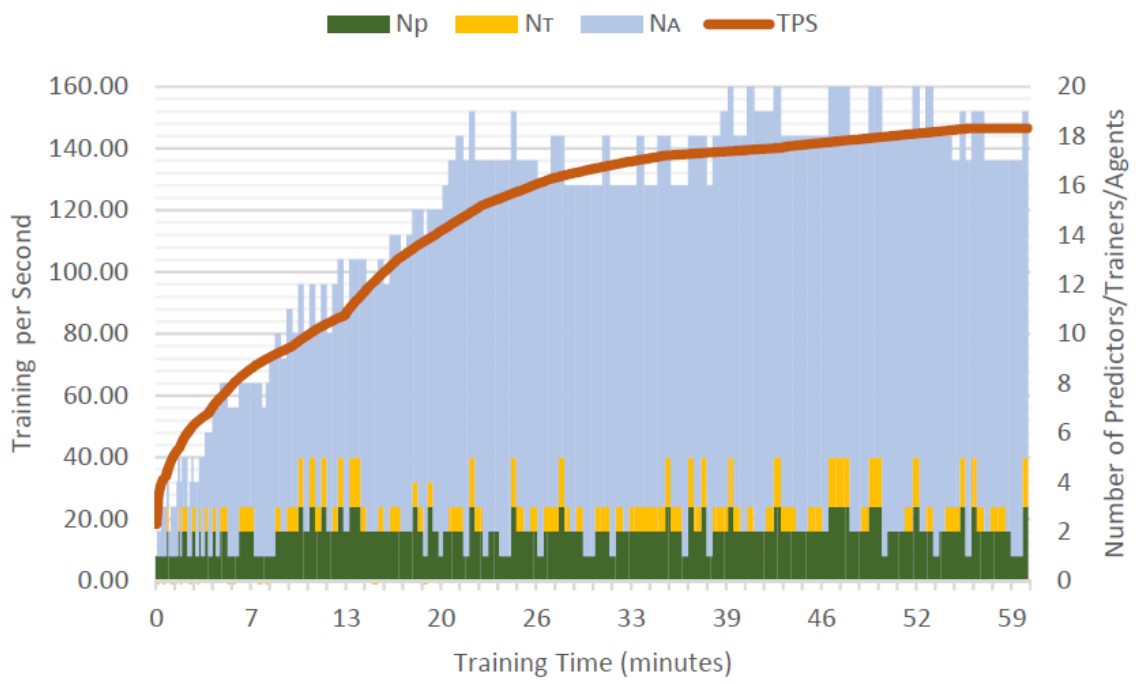
AUTOMATIC SCHEDULING

Balancing the system at run time

ATARI Boxing



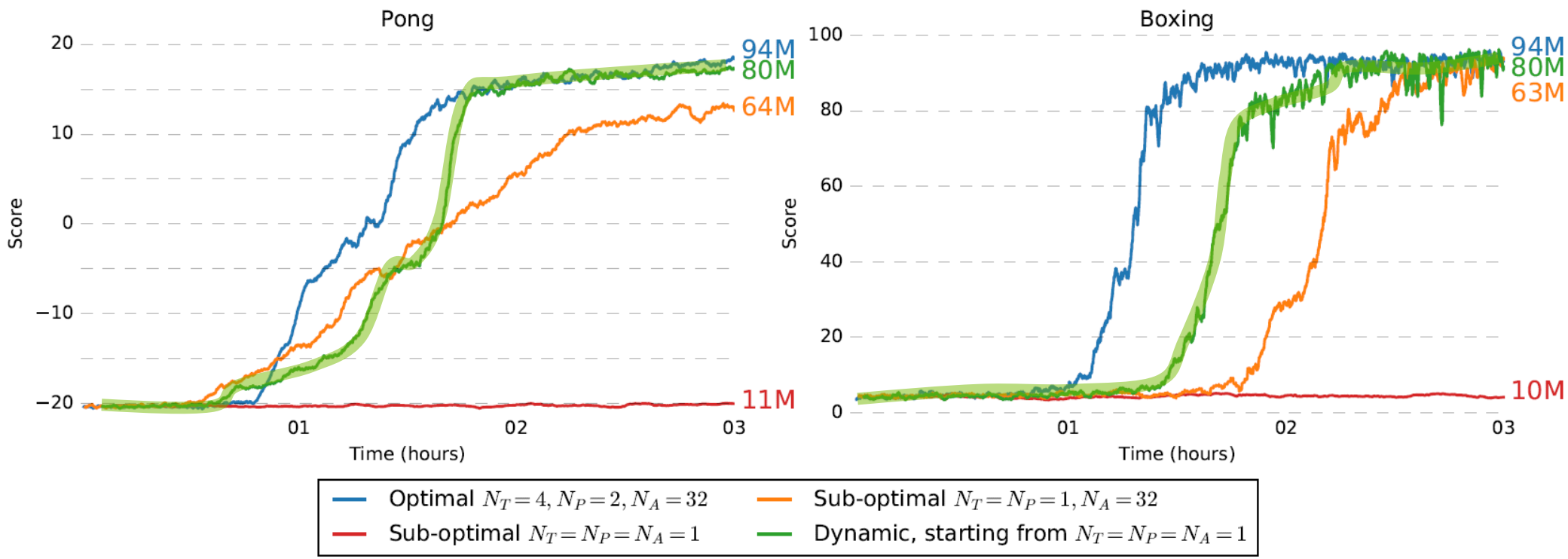
ATARI Pong



N_P = # predictors, N_T = # trainers, N_A = # agents, TPS = training per seconds

THE ADVANTAGE OF SPEED

More frames = faster convergence



LARGER DNNS

For real world applications (e.g. robotics, automotive)

A3C (ATARI)

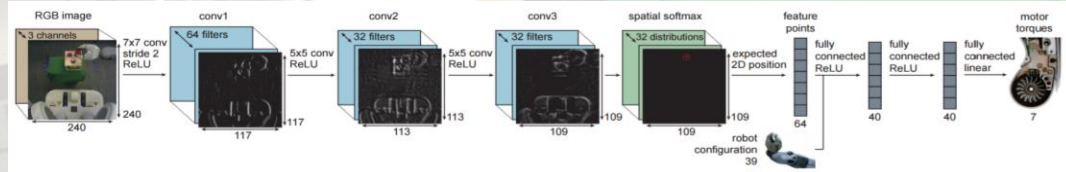
Conv 16 8x8 filters, stride 4 → Conv 32 4x4 filters, stride 2 → FC 256

Conv 32 8x8 filters, stride 1, 2, 3, 4 → Conv 32 4x4 filters, stride 2 → Conv 64 4x4 filters, stride 2 → FC 256 ?

Others (robotics)

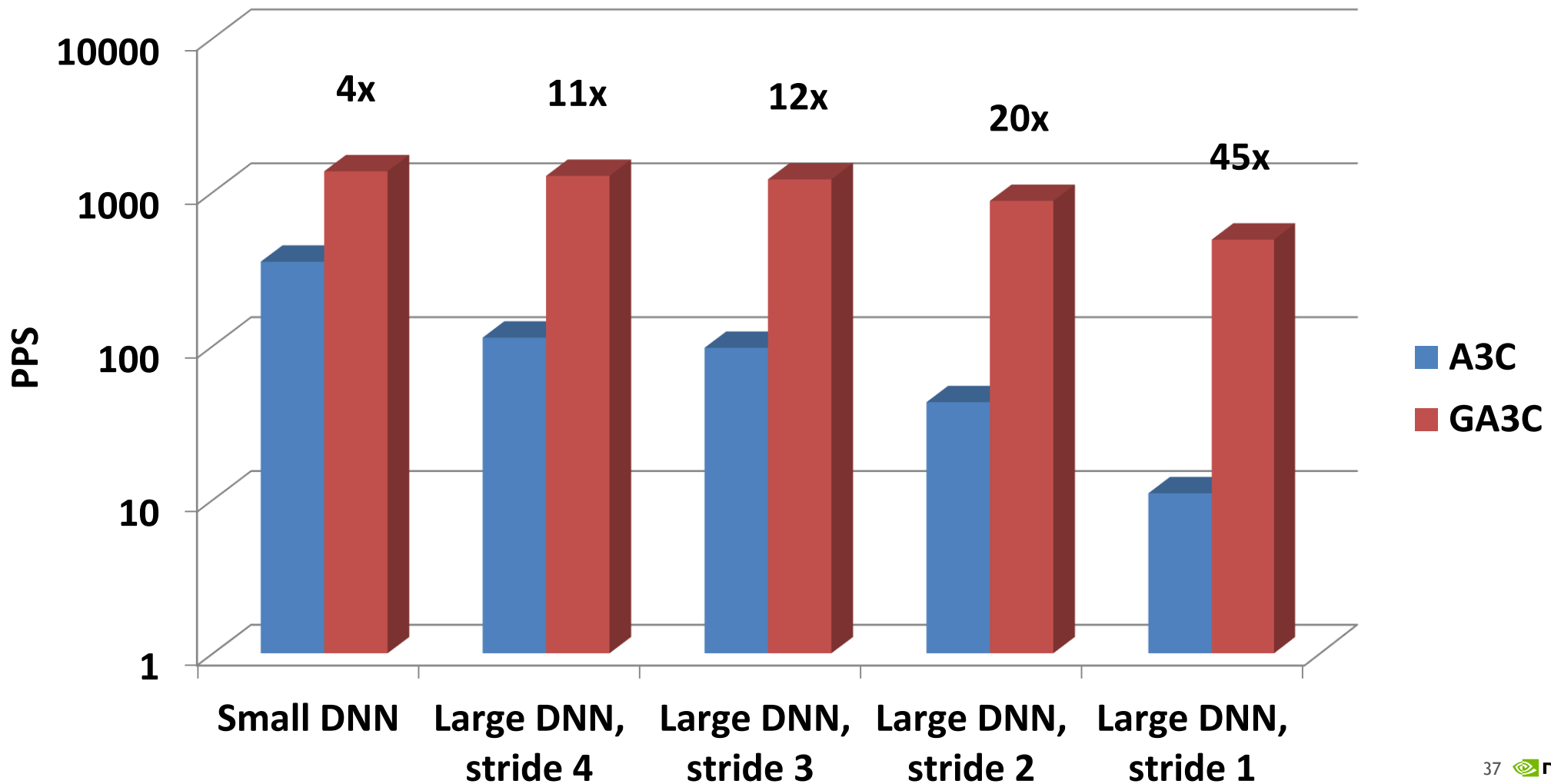
Timothy P. Lillicrap et al., Continuous control with deep reinforcement learning, International Conference on Learning Representations, 2016.

S. Levine et al., End-to-end training of deep visuomotor policies, Journal of Machine Learning Research, 17:1-40, 2016.



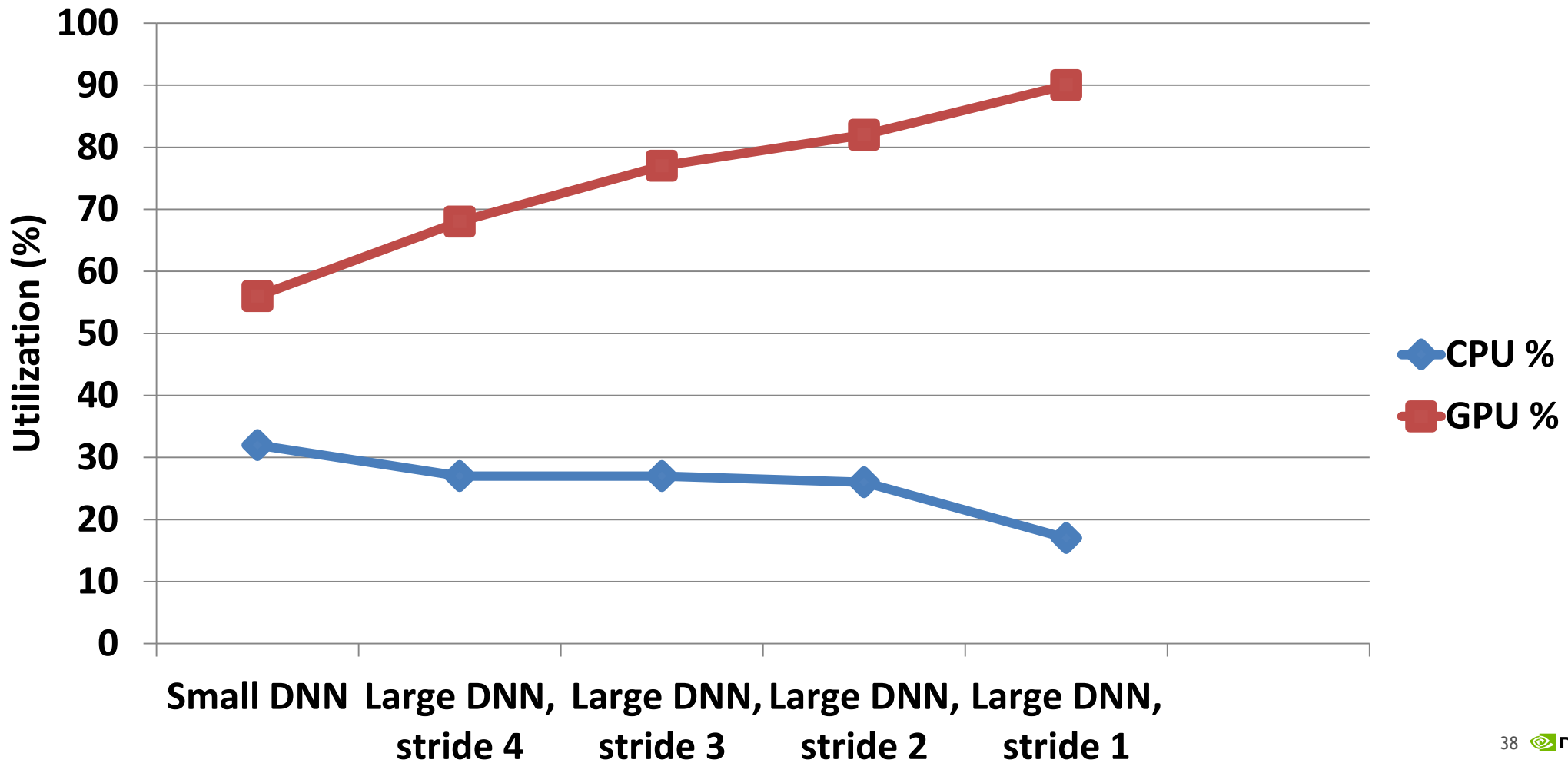
GA3C VS. A3C*: PREDICTIONS PER SECONDS

* Our Tensor Flow implementation on a CPU



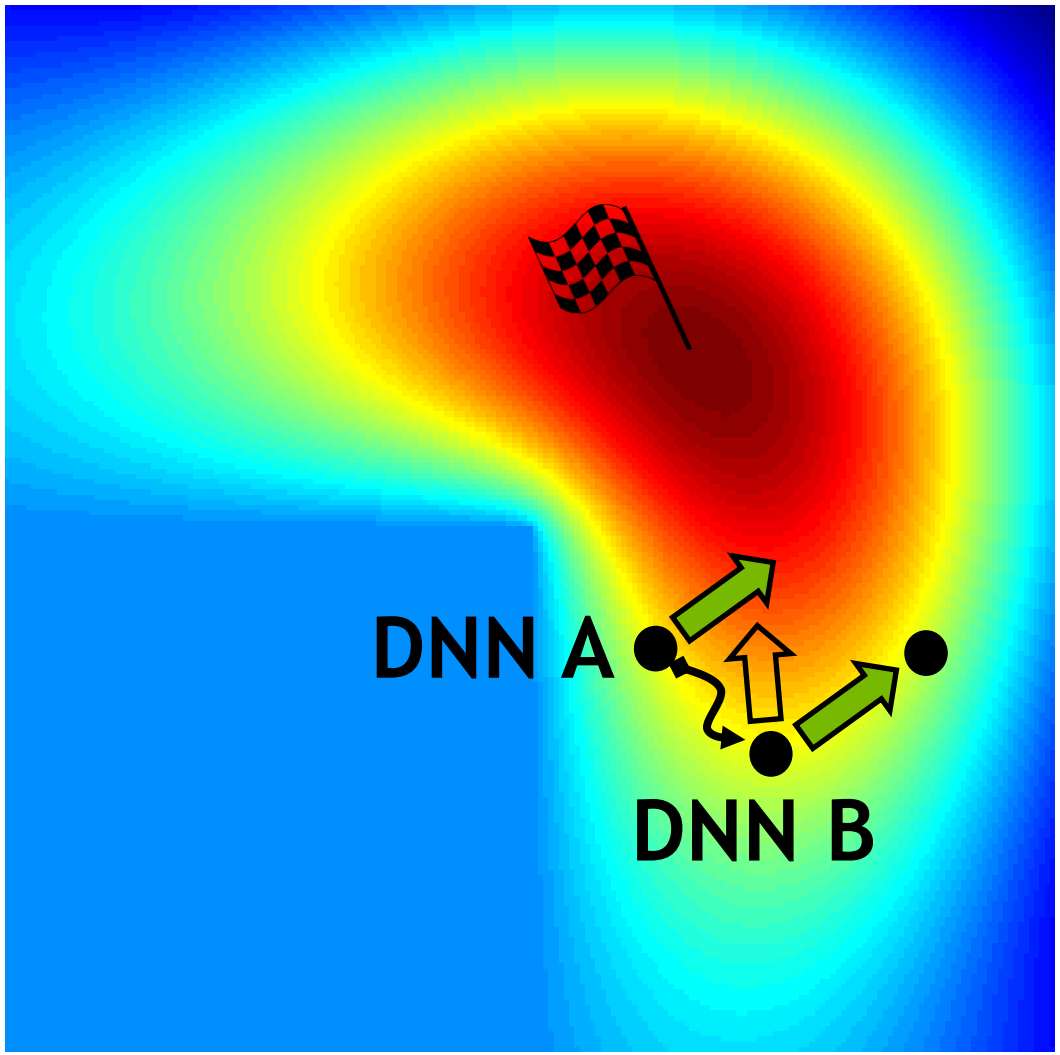
CPU & GPU UTILIZATION IN GA3C

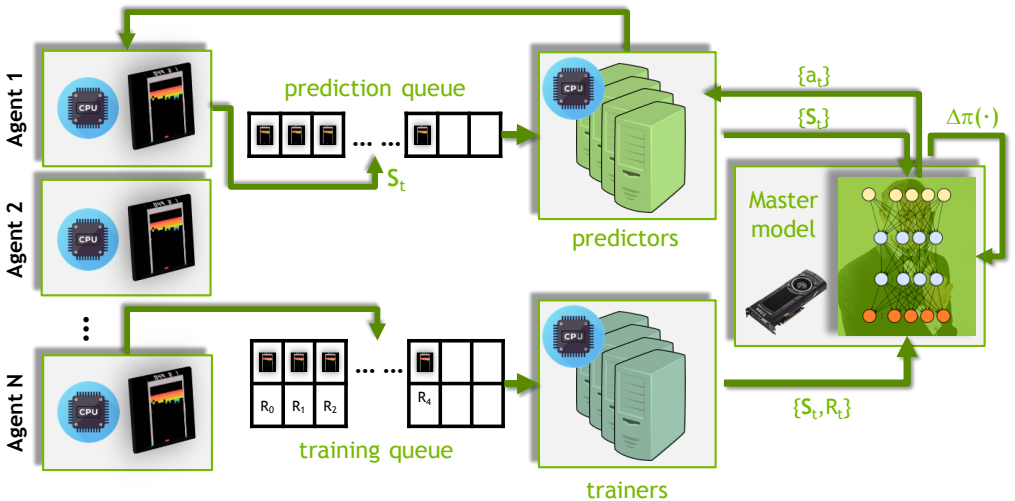
For larger DNNs



GA3C POLICY LAG

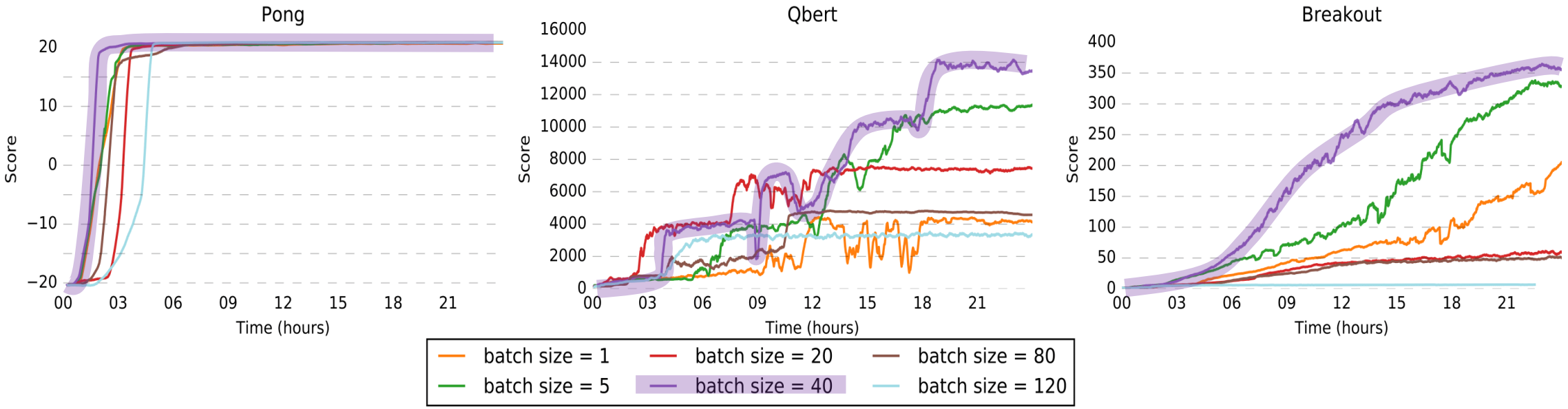
Asynchronous playing and training





STABILITY AND CONVERGENCE SPEED

Reducing policy lag through min training batch size

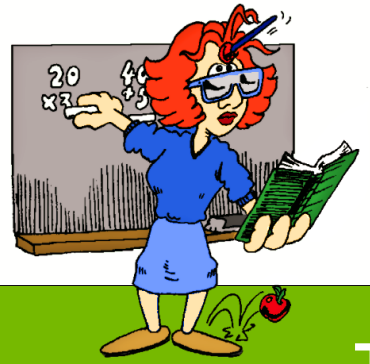


GA3C (45x faster)

Balancing computational resources, speed and stability.

GPU-based A3C

El Capitan big wall, Yosemite Valley



RESOURCES

THEORY

M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, J. Kautz, **Reinforcement Learning through Asynchronous Advantage Actor-Critic on a GPU**, ICLR 2017 (available at https://openreview.net/forum?id=r1VGvBcxl&noteId=r1VGvBcxl).

CODING

GA3C, a GPU implementation of A3C (open source at https://github.com/NVlabs/GA3C).

A general architecture to generate and consume training data.

QUESTIONS

QUESTIONS

Policy lag

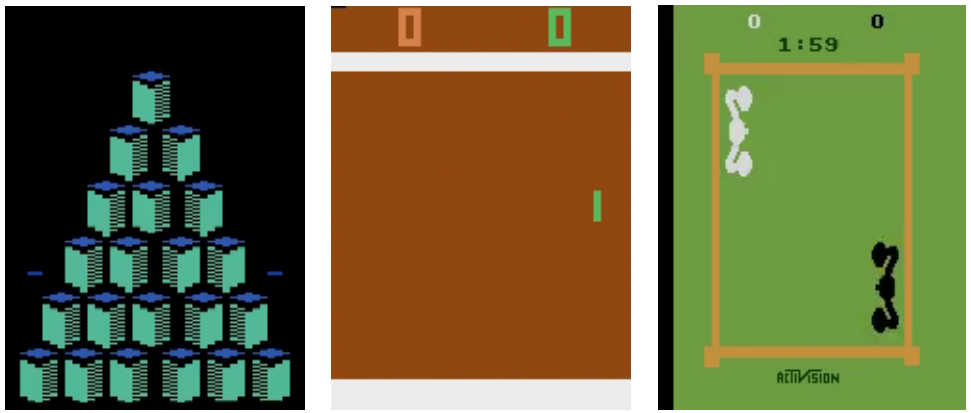
Multiple GPUs

Why TensorFlow

Replay memory

…

ATARI 2600



Github: https://github.com/NVlabs/GA3C

ICLR 2017: Reinforcement Learning through Asynchronous Advantage Actor-Critic on a GPU.

BACKUP SLIDES

POLICY LAG IN GA3C

$$f_\pi(\theta) = \log \pi(a_t|s_t;\theta)(R_t - V(s_t;\theta_t)) + \beta H(\pi(s_t;\theta)) \tag{1}$$

$$f_v(\theta) = (R_t - V(s_t;\theta))^2 \tag{2}$$

$$\nabla_\theta \left[\log \pi(a_{t-k}|s_{t-k};\theta)(R_{t-k} - V(s_{t-k};\theta_t)) + \beta H(\pi(s_{t-k};\theta)) \right] \tag{3}$$

$$\nabla_\theta \left[\log \left(\pi(a_{t-k}|s_{t-k};\theta) + \epsilon\right)(R_{t-k} - V(s_{t-k};\theta_t)) + \beta H(\pi(s_{t-k};\theta)) \right] \tag{4}$$

Potentially large time lag between training data generation and network update

SCORES

	Atari Game Scores									Attributes		
	AMIDAR	BOXING	CENTIPEDE	NAME THIS GAME	PACMAN	PONG	QBERT	SEAQUEST	UP-DOWN	Time	PPS*	System
Human [2]	1676	10	10322	6796	15375	16	12085	40426	9896	–	–	–
A3C-1 [9]	284	34	3773	5614	3307	11	13752	2300	54525	1 day	352	CPU
A3C-4 [9]	264	60	3756	10476	654	6	15149	2355	74706	4 days	352	CPU
GA3C	218	92	7386	5643	1978	18	14966	1706	8623	1 day	1080	GPU

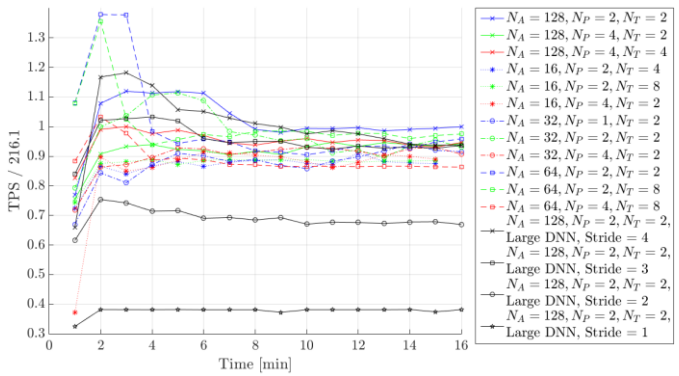
Table 3: Average scores on a subset of Atari games achieved by: a human player [2]; A3C after one and four days of training on a CPU [9]; and GA3C after one day of training. (*Note: predictions per second (PPS) is measured on our TensorFlow CPU and GPU implementations of A3C, while scores are from the cited publications.)

Training a larger DNN

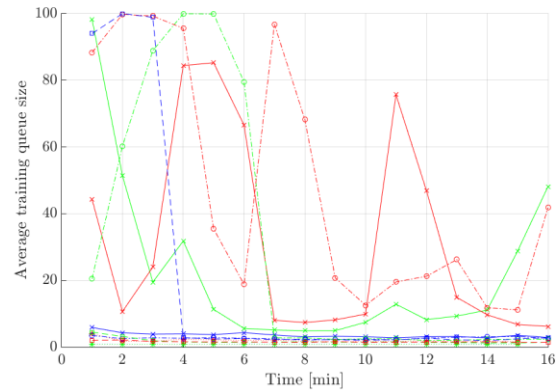
		PPS			Utilization (%)	
System	**DNN**	**A3C**	**GA3C**	**Speed up**	**CPU**	**GPU**
System I	small	352	1361	4×	32	56
	large, stride 4	113	1271	11×	27	68
	large, stride 3	97	1206	12×	27	77
	large, stride 2	43	874	20×	26	82
	large, stride 1	11	490	45×	17	90
System II	small	116	728	6×	62	33
	large, stride 1	12	336	28×	49	78
System III	small	300	1248	4×	31	60
	large, stride 1	38	256	6×	17	82

BALANCING COMPUTATIONAL RESOURCES

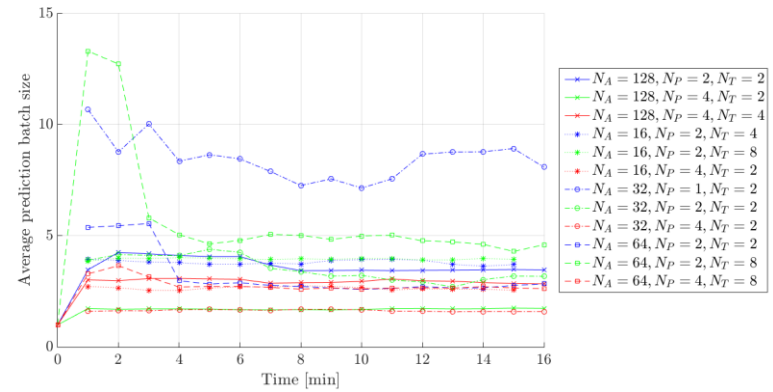
The actors: CPU, PCI-E & GPU



Trainings per second



Training queue size



Prediction queue size

Generating and consuming data with constant queue sizes maximizes speed