Customisable Control Policy Learning for Robotics

Ce Guo[†], Wayne Luk[†], Stanley Loh Qing Shui[†]

Alexander Warren[‡] and Joshua Levine[‡]

[†] Imperial College London [‡] Intel

ASAP'19, Cornell Tech, New York, 15 July 2019

Background: modern robot

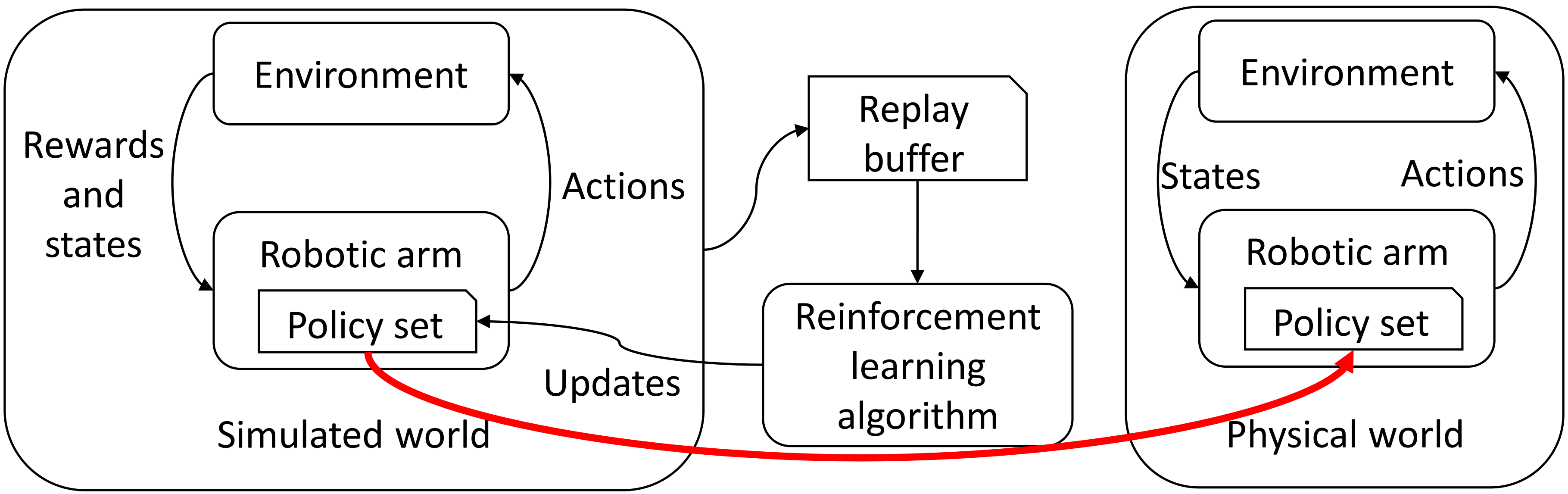
- Complicated and unpredictable
- Equipped with modern sensors: lidars, cameras, etc.
- Hard-code programming: tedious and error-prone

Background: robot learning

- Robot learning
  - No hard-code action rules
  - Train robots by rewarding proper actions
- Training on physical robots
  - Slow training process
  - Potential physical damage during training

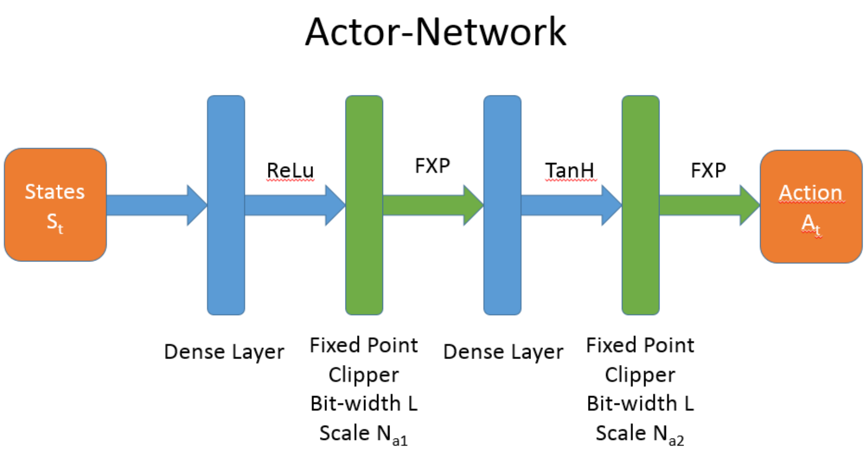
Background: sim-to-real learning

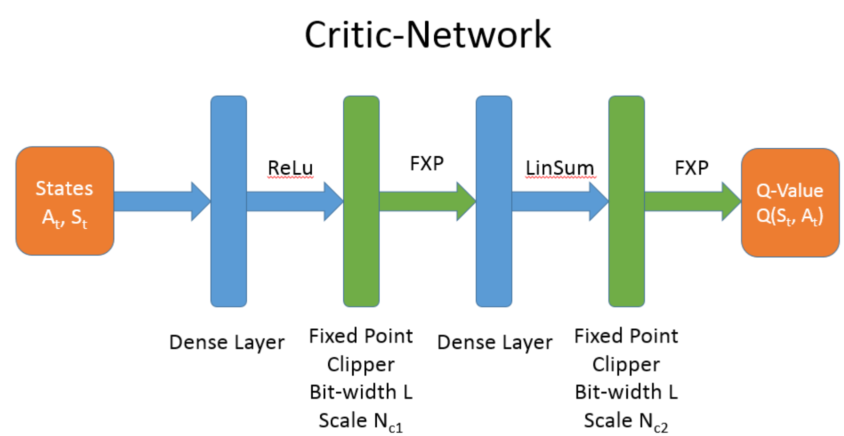
- Sim-to-real robot learning
  - Train robot with simulation for improved efficiency
  - Simulated robot learns a policy set during training
  - Transfer policy set to physical robot after training



Deep deterministic policy gradients

- DDPG: a reinforcement learning technique
- Continuous action space supported
- Deep networks used as function approximators
- Efficiency bottleneck: gradient computation





Contributions

1. Customisable hardware architecture for DDPG
   - Back-propagation via odd layers and even layers
   - Policy learned and encoded with fixed-point numbers

2. Sim-to-real policy learning platform
   - Customised 3D printed robotic arm
   - Simulated robotic arm and environment

3. Evaluation: accelerated policy learning
   - Stratix V at 200MHz versus i7-6700 at 3.4GHz
   - Faster gradient computation: up to 18.7 times speedup
   - Better convergence: fewer training episodes

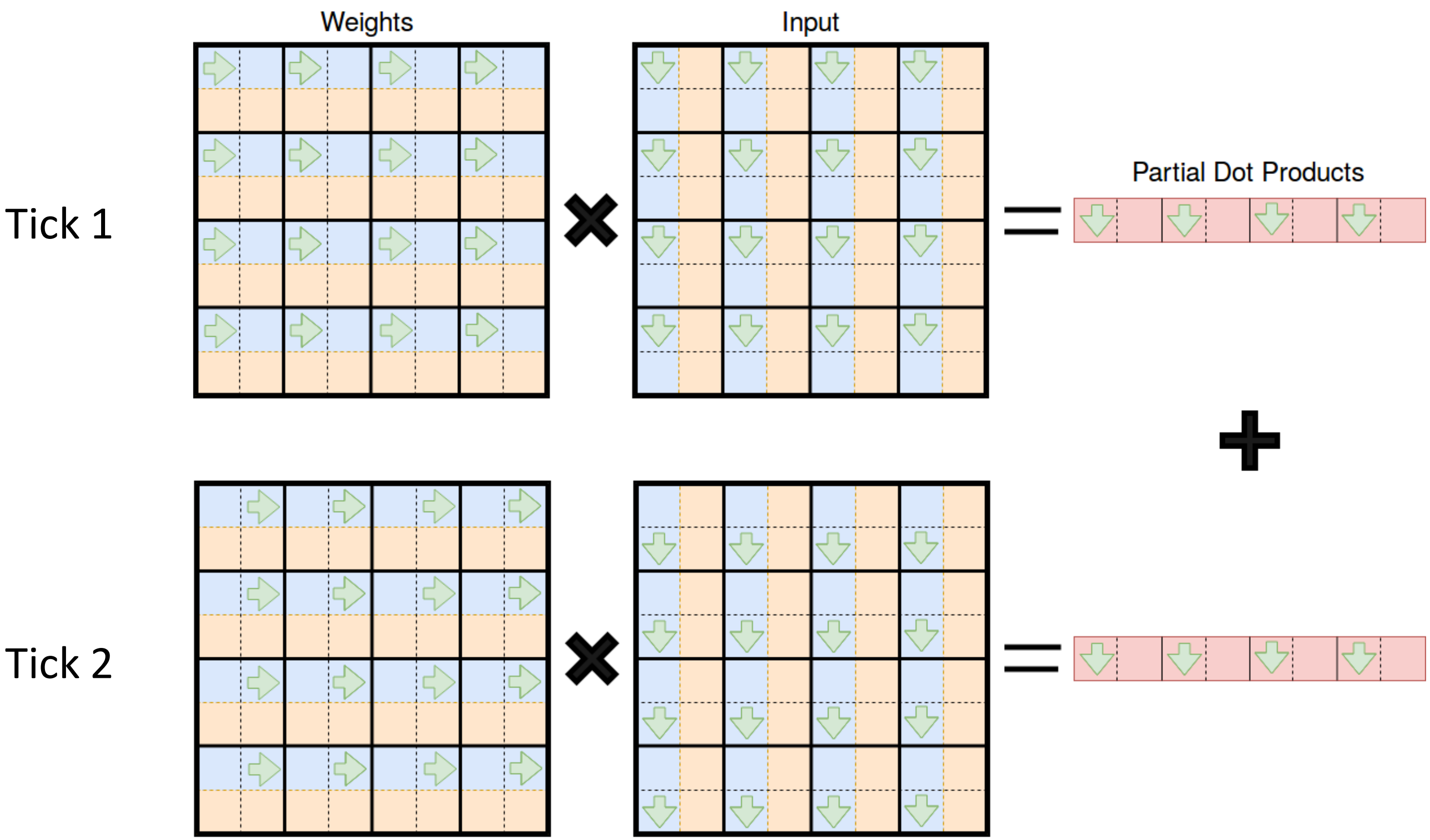
1. Customisable DDPG architecture

- Task partitioning
  - Software on CPU: simulation and weight update
  - Customised hardware on FPGA: gradient computation
- Gradient computation via backpropagation
  - Forward pass for decisions
  - Backward pass for feedbacks and gradients
- Hardware resources shared by both passes
  - Parallel elements for **odd** and **even** layers
  - Alternating between even and odd layers
- Policy set encoded in fixed-point numbers

Odd layer

- A new input is available every tick
- An output becomes ready every K ticks

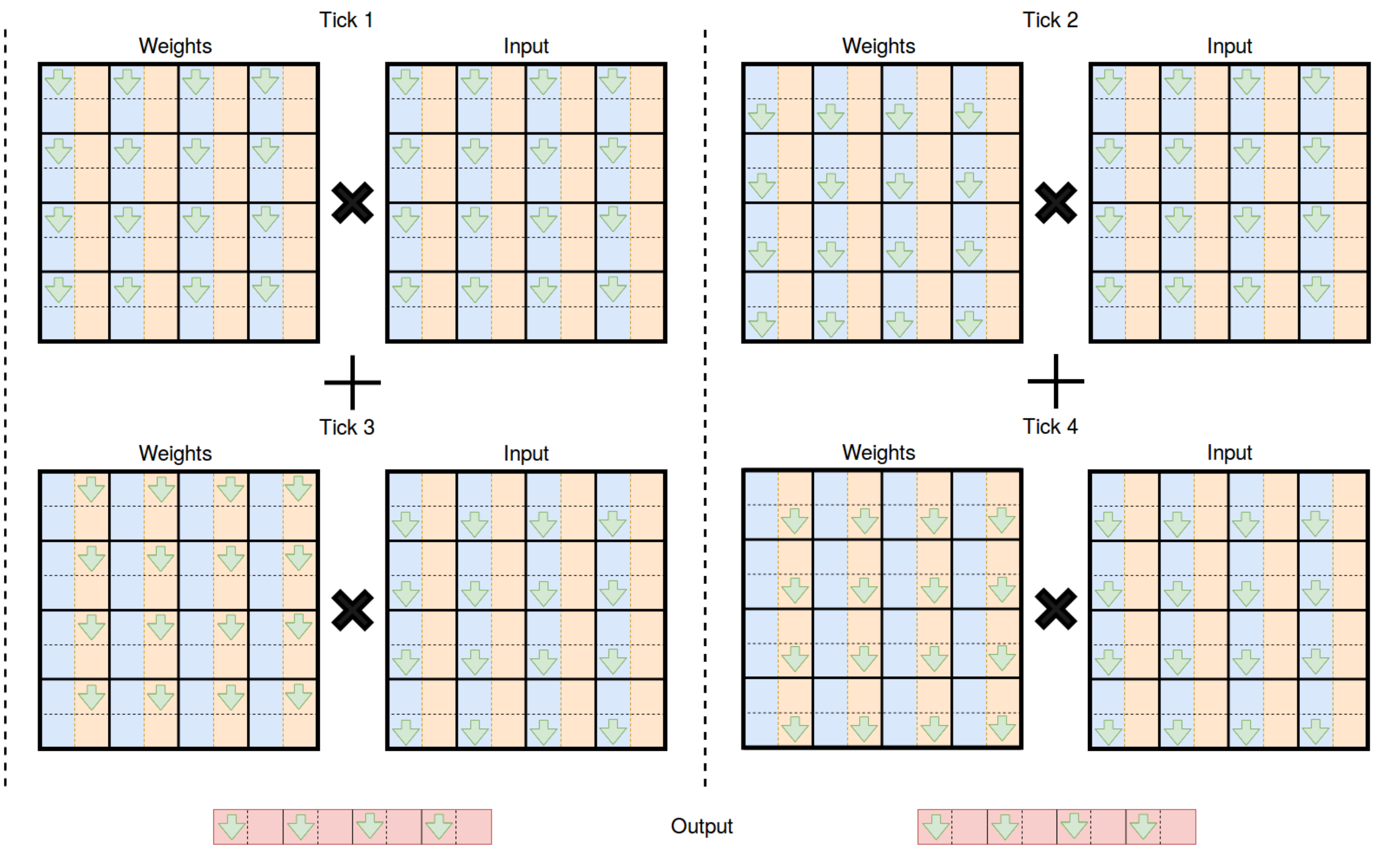
Inputs streamed in continuously
Outputs streamed out at regular intervals



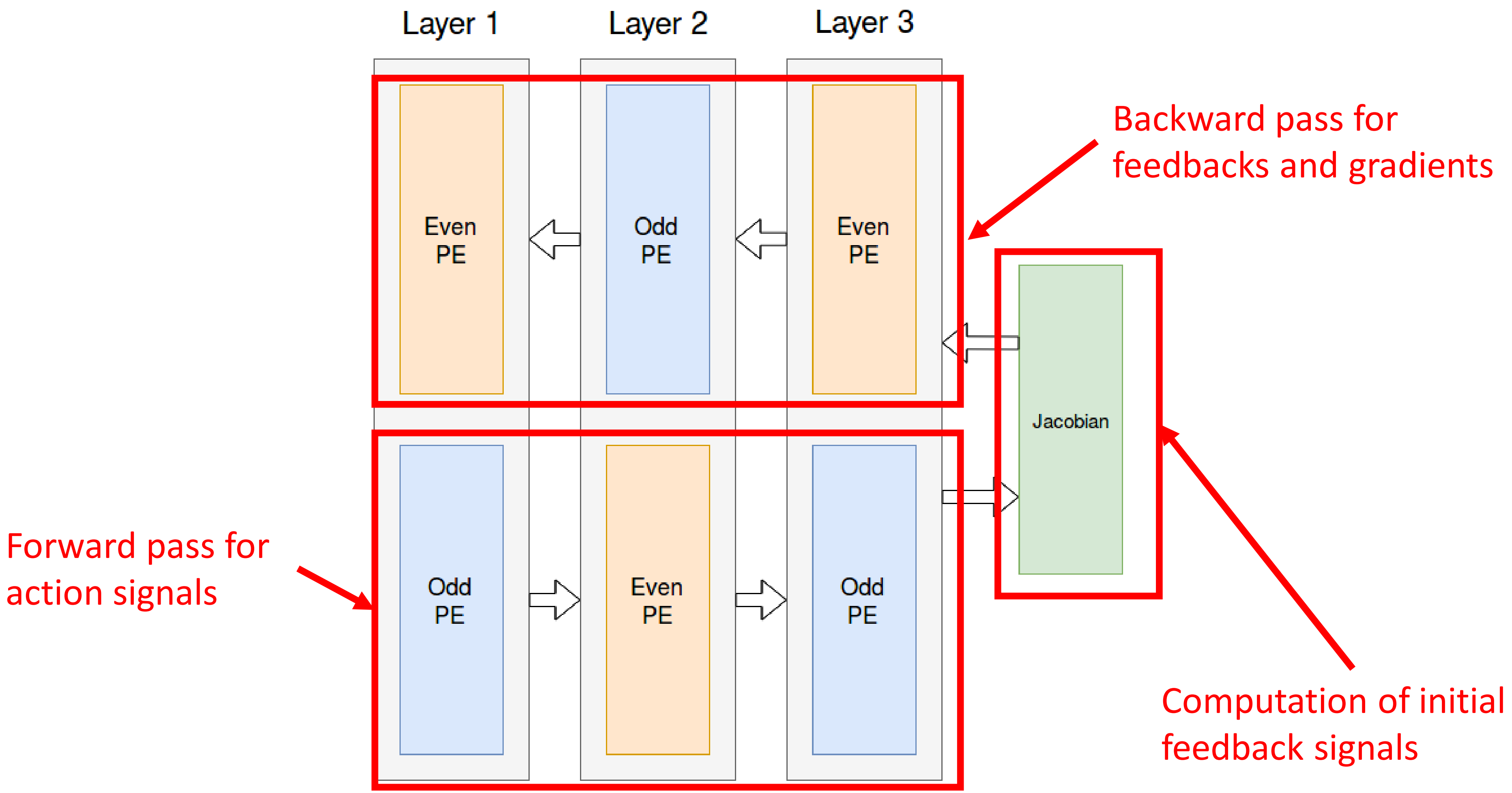
Even layer

- A new input is available every K ticks
- An output becomes ready every tick

Different IO stream pattern from odd layers



Backpropagation

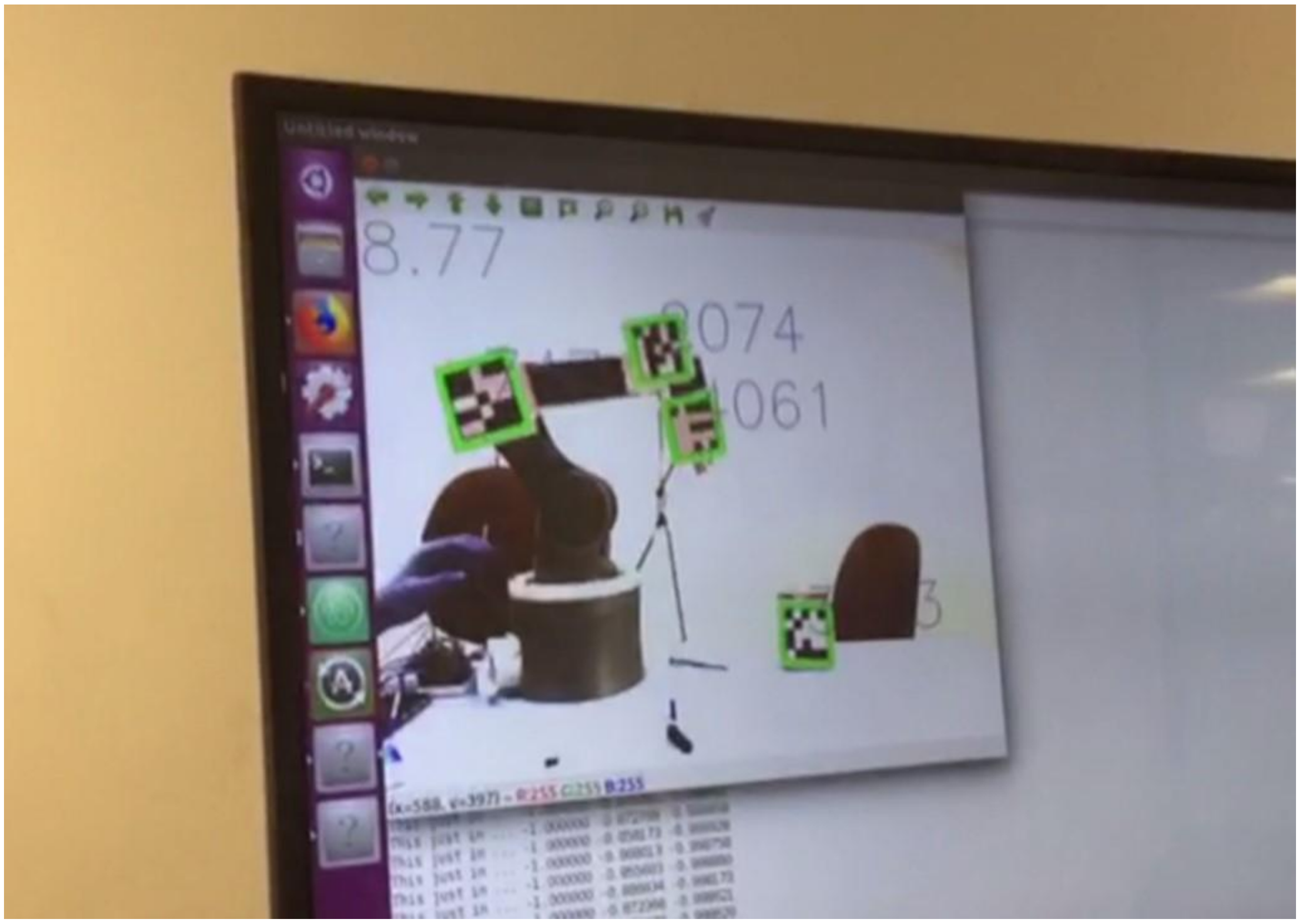


2. Platform for sim-to-real learning

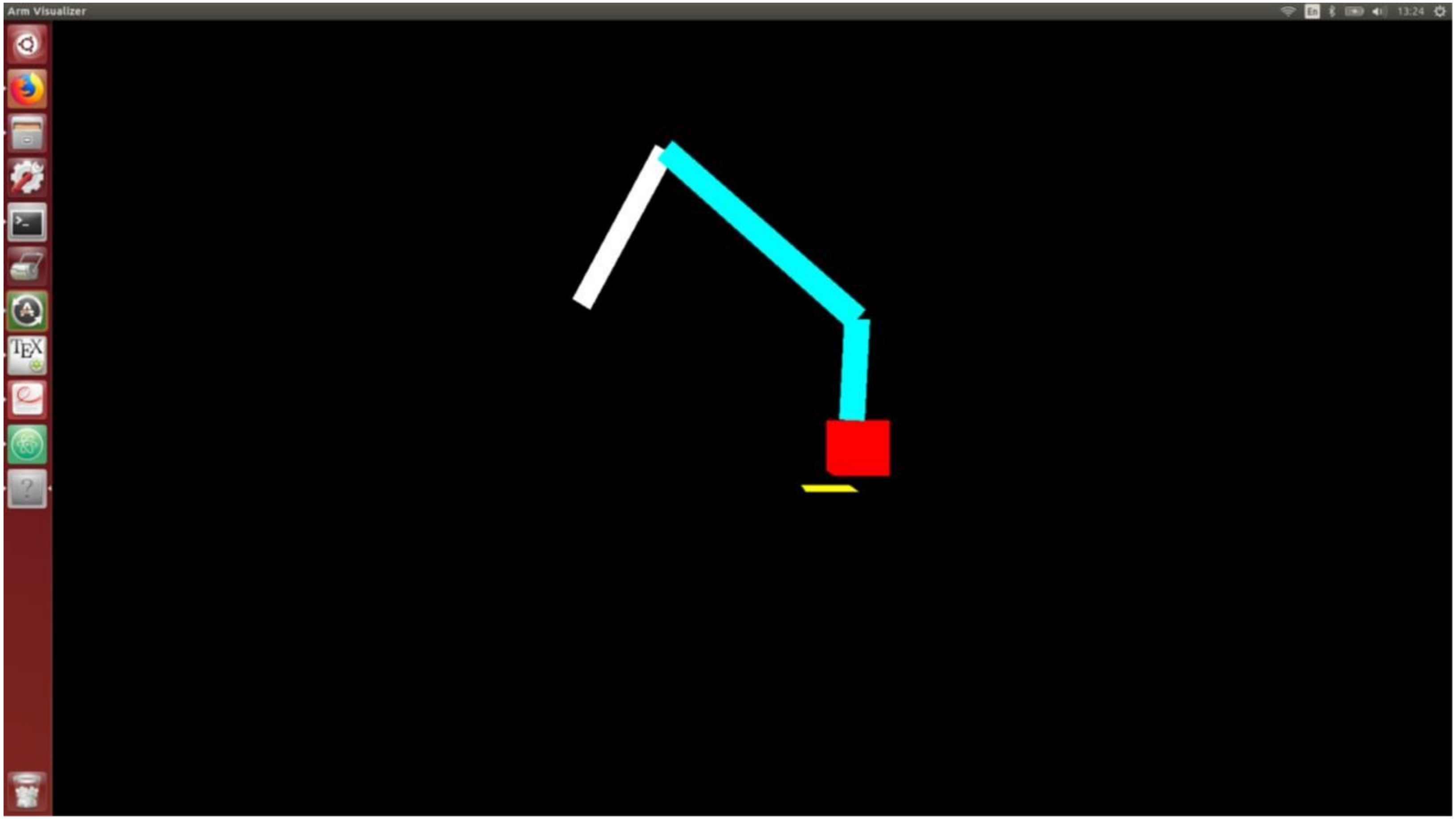
- Major components:
  - Customised hardware for gradient computation
  - 3D printed robotic arm with electromagnet
  - Simulated robotic arm and environment
- State space: 23 components
  - 12 for joint-box and goal-box distances
  - 9 for positions of end effector, box and goal
  - 2 for progress monitoring
- Action space: 5 components
  - 4 rotation angles for motors
  - 1 control signal for end effector

Physical robot

State computation



Simulation



3. Evaluation: accelerated policy learning

- Aspects to evaluate
  - Execution time for gradient computation in each episode
  - Number of episodes before convergence
- Software on CPU
  - Intel Core i7-6700 CPU (14nm, 4 cores, 3.4 GHz)
  - Single-precision floating point arithmetic using NumPy
- Hardware on FPGA
  - Intel Stratix-V FPGA (28nm, 200 MHz)
  - 32-bit fixed-point arithmetic (8 integer bits; 24 fractional bits)
- Models
  1. FPGA-based DDPG
  2. FPGA-based DDPG with expanded action space
  3. Deeper model

Reward function

Distance between end effector and box

Sum of four components:

$$r_1(S) = -w_1 \boxed{\sqrt{(x_{link3} - x_{box})^2 + (y_{link3} - y_{box})^2 + (z_{link3} - z_{box})^2}}$$

$$r_2(S) = -w_2 \boxed{\left(\frac{\pi}{2} - cos^{-1} \frac{\sqrt{(x_{link2} - x_{link3})^2 + (y_{link2} - y_{link3})^2}}{\sqrt{(x_{link2} - x_{link3})^2 + (y_{link2} - y_{link3})^2 + (z_{link2} - z_{link3})^2}}\right)}$$

Elevation angle of last segment

$$r_3(S) = \boxed{\begin{cases} 0, & \text{if } (S_{link3} - S_{box})^2 < \epsilon \\ -w_3 * max(a5, 0) & \text{otherwise} \end{cases}}$$

Energy saver for electromagnet

$$r_4(S) = -w_4 \boxed{\sqrt{(x_{box} - x_{goal})^2 + (y_{box} - y_{goal})^2 + (z_{box} - z_{goal})^2}}$$

Distance between box and goal

Gradient computation

Gradient computation and transmission

M	Connection	FPGA exe. time (ms)	Speedup	Norm. speedup
1	PCIe	0.250	2.57	5.14
2	PCIe	0.250	3.68	7.36
3	Infiniband	2.311	1.33	2.66

Technology
CPU: 14nm
FPGA: 28nm

Speed bottleneck: IO bandwidth

Gradient computation without transmission

M	FPGA exe. time (ms)	Speedup	Norm. Speedup
1	0.0492	13.1	26.2
2	0.0492	18.7	37.3

Gradient computation

- Execution time estimation for DDPG learning
  - Gradient computation + gradient update
- Assumptions in estimation
  - No IO bottleneck
  - Extra resources for weight optimiser
  - Design runs at 200MHz

Higher than the speedup for gradient computation

Theoretical maximum acceleration for policy learning

M	Cycles	Time (ms)	Speedup	Norm. speedup
1	5450	0.028	23	47
2	5450	0.028	33	67
3	17000	0.085	35	71

Policy transfer

- Task specification
  - A: (400,0) → (400,150)
  - B: (400,0) → (200,0)
  - C: (300,0) → (400,150)
  - D: (300,0) → (200,150)
  - E: (200,0) → (400,150)
  - F: (200,0) → (400,0)
- Subtasks
  - Attach: electromagnet attaches box
  - Put: robotic arm moves box to goal position

Proposed hardware:
first to support this subtask

Policy transfer

Number of episodes to achieve goal

	TRPO [1] Attach	DDPG Attach	DDPG Put	F-DDPG Attach	F-DDPG Put
A	×	600	×	800	×
B	Yes	800	**800***	800	1000*
C	×	600	1200	600	**600**
D	×	800	1200	**600**	**1000**
E	×	600	1000	600	1000
F	Yes	800	**800***	**600**	1000*

F-DDPG gives better policies

Fixed-point arithmetic reduces number of episodes

* Goal achieved without lifting

TRPO: Trust Region Policy Optimisation

DDPG/F-DDPG: DDPG model on CPU/FPGA

[1] S. Shao *et al.* “Towards hardware accelerated reinforcement learning for applicationspecific robotic control,” ASAP’18

New direction for robot training!

1. Customisable hardware architecture for DDPG
    - Back-propagation via odd layers and even layers
    - Policy learned and encoded with fixed-point numbers

2. Policy learning platform
    - Customised 3D printed robotic arm
    - Simulated robotic arm and environment

3. Evaluation: accelerated policy learning
    - Stratix V at 200MHz versus i7-6700 at 3.4GHz
    - Faster gradient computation: up to 18.7 times speedup
    - Better convergence: fewer training episodes

A short video…

- https://youtu.be/bKnkJPQcyIM