Generalized Computation Graphs

presented by Jason-TOKO

Outline

- Generalized Computation Graphs(GCG)
  - Background
  - Framework
  - Experiment
- Generalization through Simulation(GtS)
  - Background
  - Algorithm
  - Experiment
- Conlusion
- Reference

Generalized Computation Graphs(GCG)

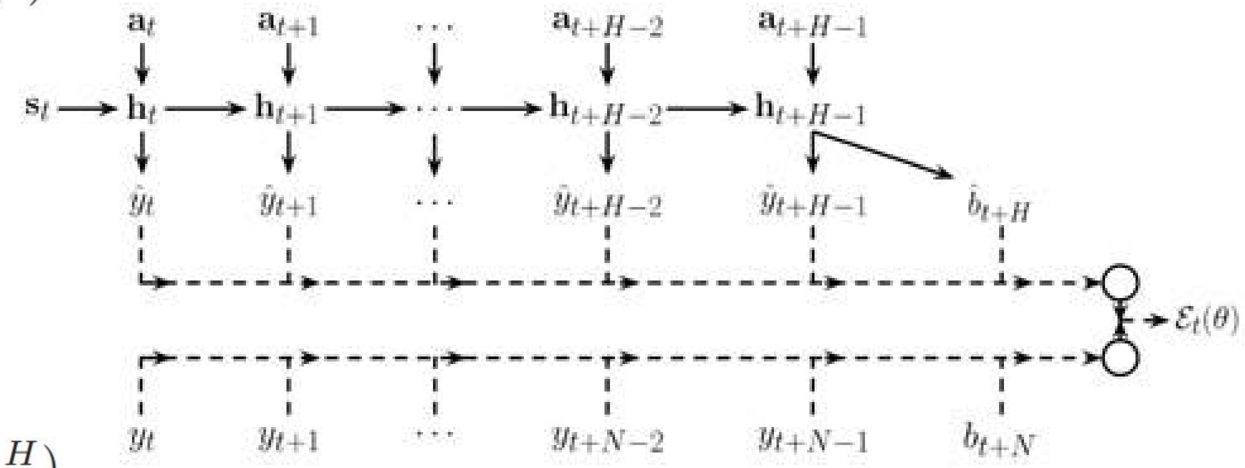
- Background
  - Model-free VS Model-based: sample efficiency, stability, and final performance.
  - These three metrics suffer when the state space is high-dimensional (e.g.image).
- Idea:
  - Subsume both value-based model-free methods and model-based algorithms.

Generalized Computation Graphs(GCG)

- Framework
  - the computation graph: $G_\theta(\mathbf{s}_t, \mathbf{A}_t^H)$
  - input: $\mathbf{s}_t$

    $\mathbf{A}_t^H = (\mathbf{a}_t, ..., \mathbf{a}_{t+H-1})$
  - output: $\hat{Y}_t^H = (\hat{y}_t, ..., \hat{y}_{t+H-1})$

    $\hat{b}_{t+H}$
  - error function: $\mathcal{E}_t(\theta)$
  - policy evaluation function: $J(\mathbf{s}_t, \mathbf{A}_t^H)$



Generalized Computation Graphs(GCG)

Algorithm 1 Reinforcement learning with generalized computation graphs

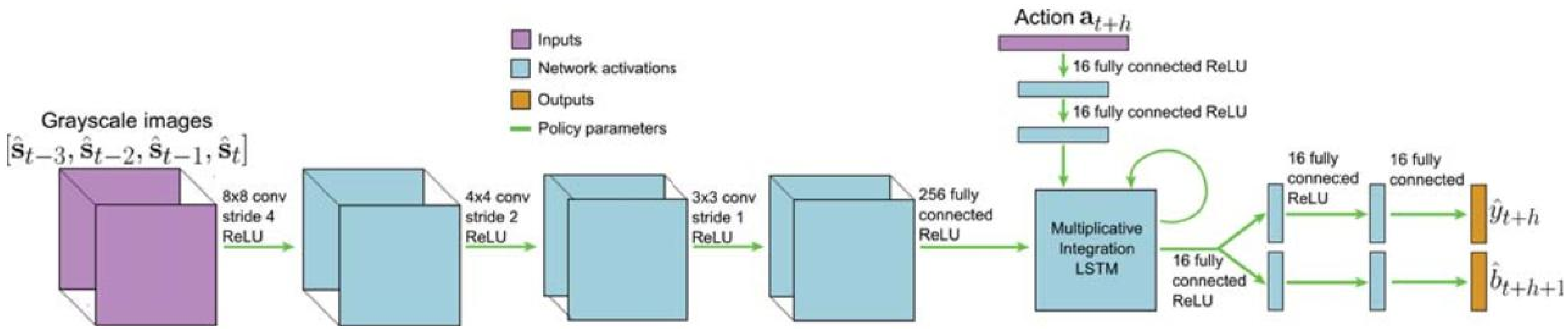
input: computation graph $G_\theta(\mathbf{s}_t, \mathbf{A}_t^H)$, error function $\mathcal{E}_t(\theta)$, and policy evaluation function $J(\mathbf{s}_t, \mathbf{A}_t^H)$
initialize dataset $\mathcal{D} \leftarrow \emptyset$
for $t = 1$ to T **do**
  get current state $\mathbf{s}_t$
  $\boxed{\mathbf{A}_t^H \leftarrow \arg\max_{\mathbf{A}} J(\mathbf{s}_t, \mathbf{A})}$ **e.g. MPC, CEM**
  execute first action $\mathbf{a}_t$
  receive labels y_t and b_t
  add $(\mathbf{s}_t, \mathbf{a}_t, y_t, b_t)$ to dataset $\mathcal{D}$
  update G_θ by $\theta \leftarrow \arg\min_\theta \mathcal{E}_{t'}(\theta)$ using $\mathcal{D}$
end for

Generalized Computation Graphs(GCG)

- GCG instantiation:
  - Model: deep RNN
  - Output:
    - $\hat{Y}_t^H$: reward ; $\hat{b}_{t+H}$: future value-to-go
    - $\hat{Y}_t^H$: probability of collision; $\hat{b}_{t+H}$: the best-case future likelihood of collision
  - Policy evaluation function:
    - $J(\mathbf{s}_t, \mathbf{A}_t^H) = \sum_{h=0}^{H-1} \gamma^h \hat{y}_{t+h} + \gamma^H \hat{b}_{t+H}$
    - $J(\mathbf{s}_t, \mathbf{A}_t^H) = \sum_{h=0}^{H-1} -\hat{y}_{t+h} - \hat{b}_{t+H}$
  - Policy evaluation: MPC, CEM ......
  - Model horizon:
    - H = 1 --> fully model-free
    - H = ∞ --> fully model-based
    - H = constant --> interpolating between model-free and model-based

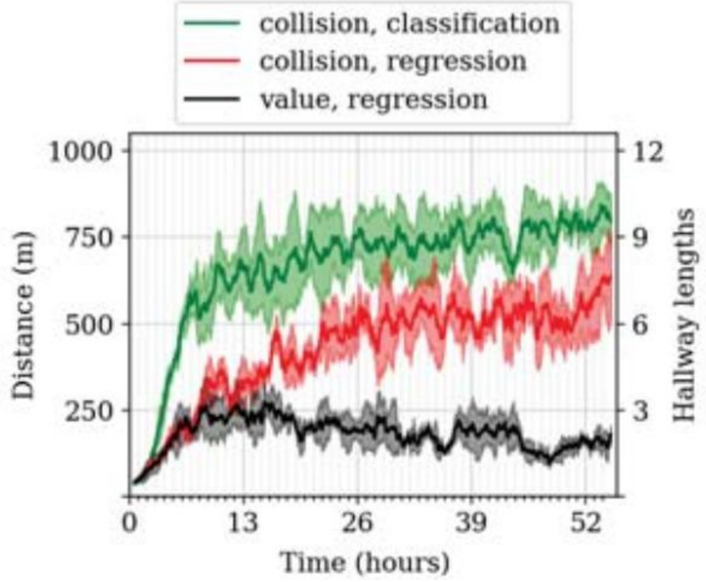
Generalized Computation Graphs(GCG)

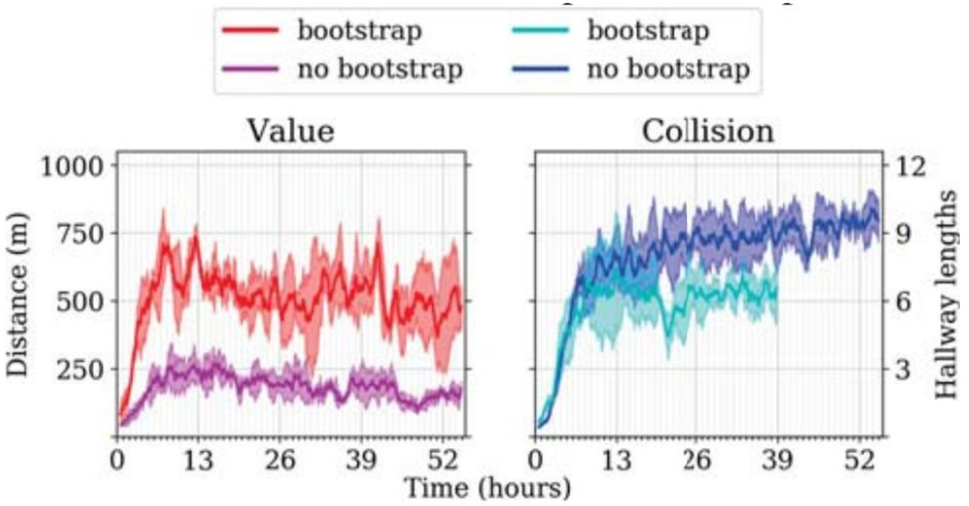
- GCG instantiation
  - Boostrapping: can increase H but cause bias and instability
  - Loss function: Bellman error / cross entropy loss
  - Network struture:

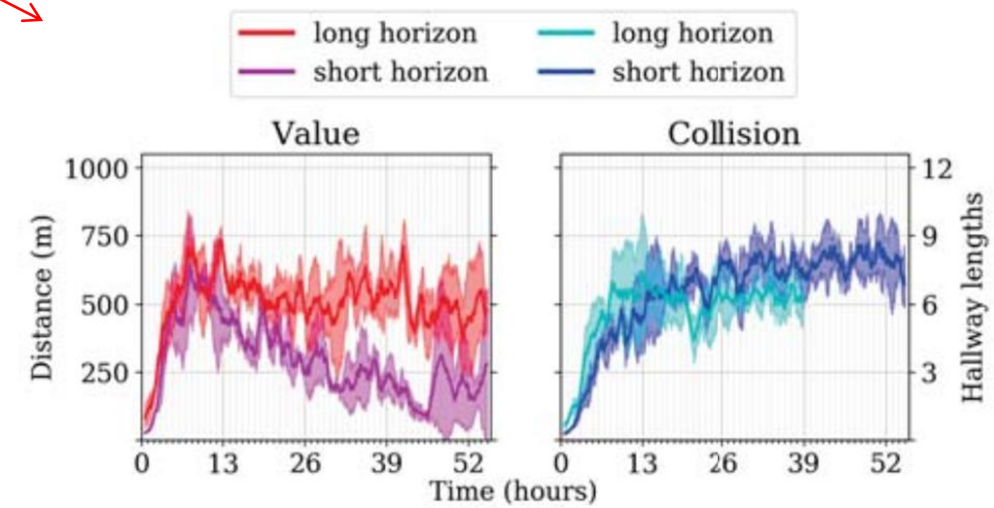


Generalized Computation Graphs(GCG)

- Experiment:
  - 1. Model outputs and loss function
  - 2. Model horizon
  - 3. Bootstrapping

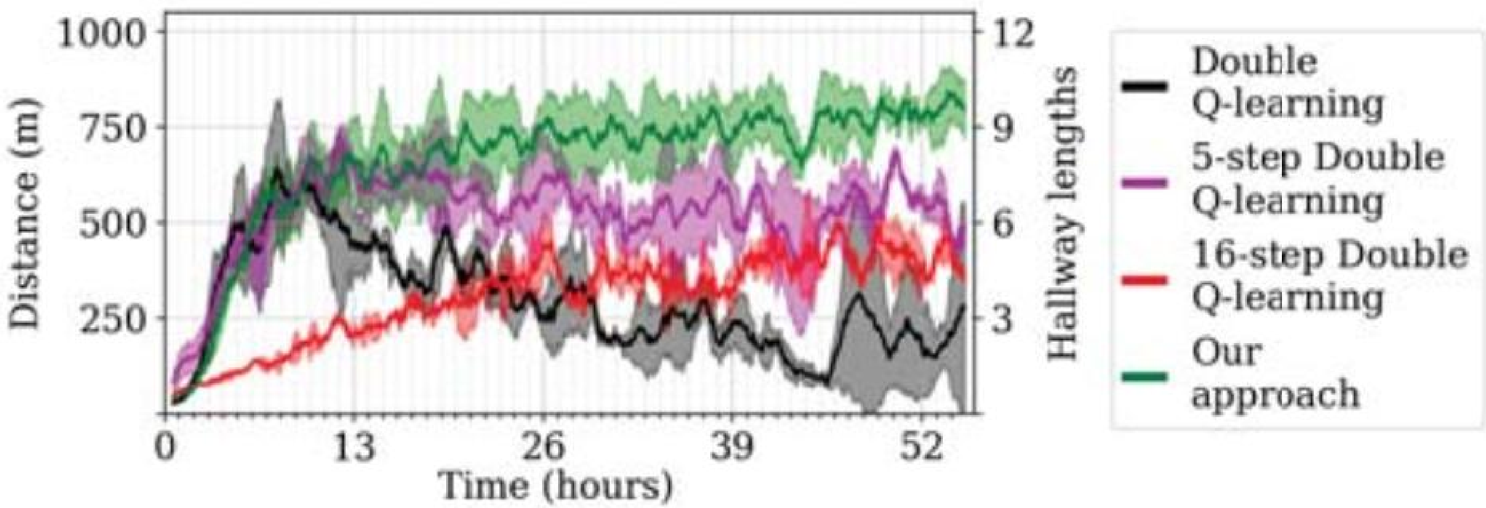






Generalized Computation Graphs(GCG)

- Experiment:
  - 4. Comparion to prior work

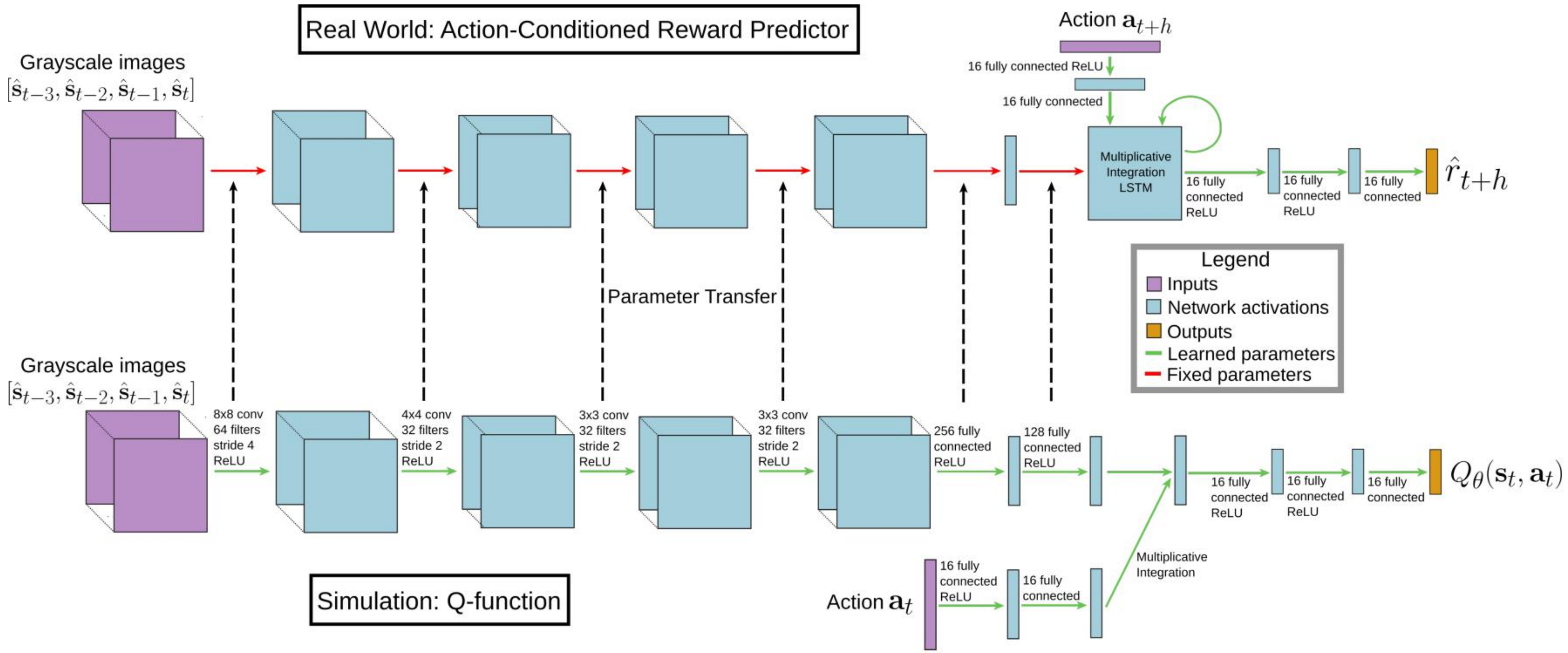


Generalization through Simulation(GtS)

- Background:
  - Complex realworld physical and visual phenomena are difficult to simulate accurately.
  - The systematic differences between simulation and reality are typically impossible to eliminate.
- Idea:
  - use real-world experience to learn how to fly(control) and use simulated experience to learn how to generalize(perception).
- The main contribution:
  - Combining large amounts of simulated data with small amounts of real-world experience to train real-world collision avoidance policies for autonomous flight with DRL

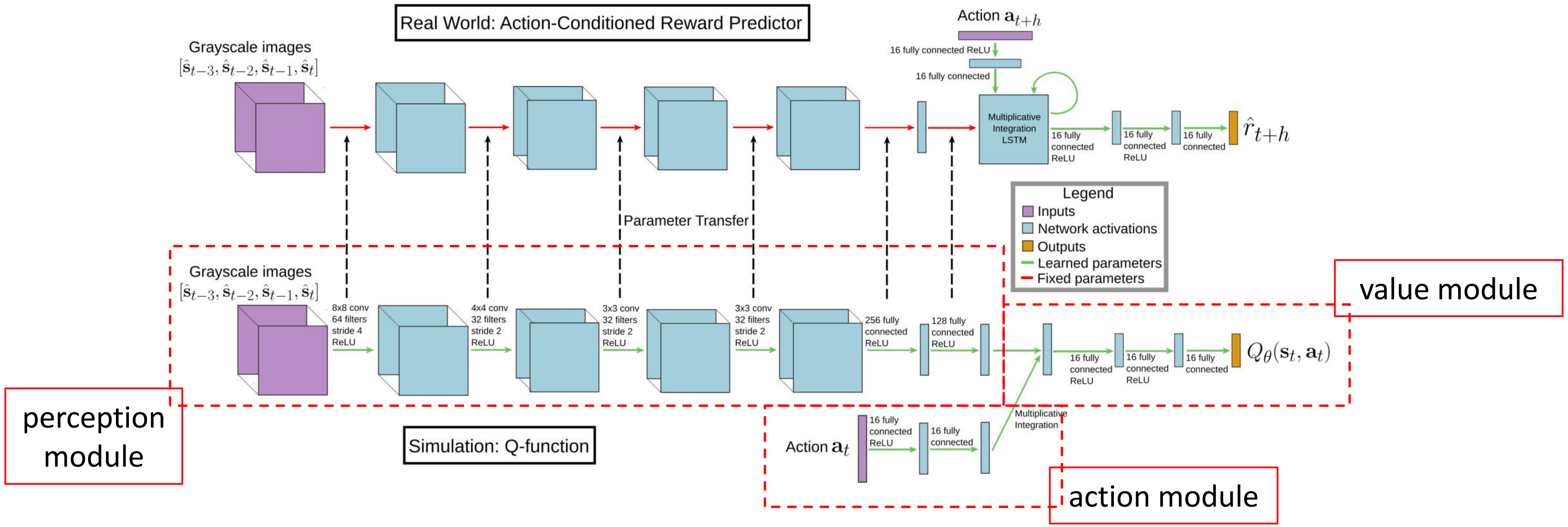
Generalization through Simulation(GtS)

- Network Struture



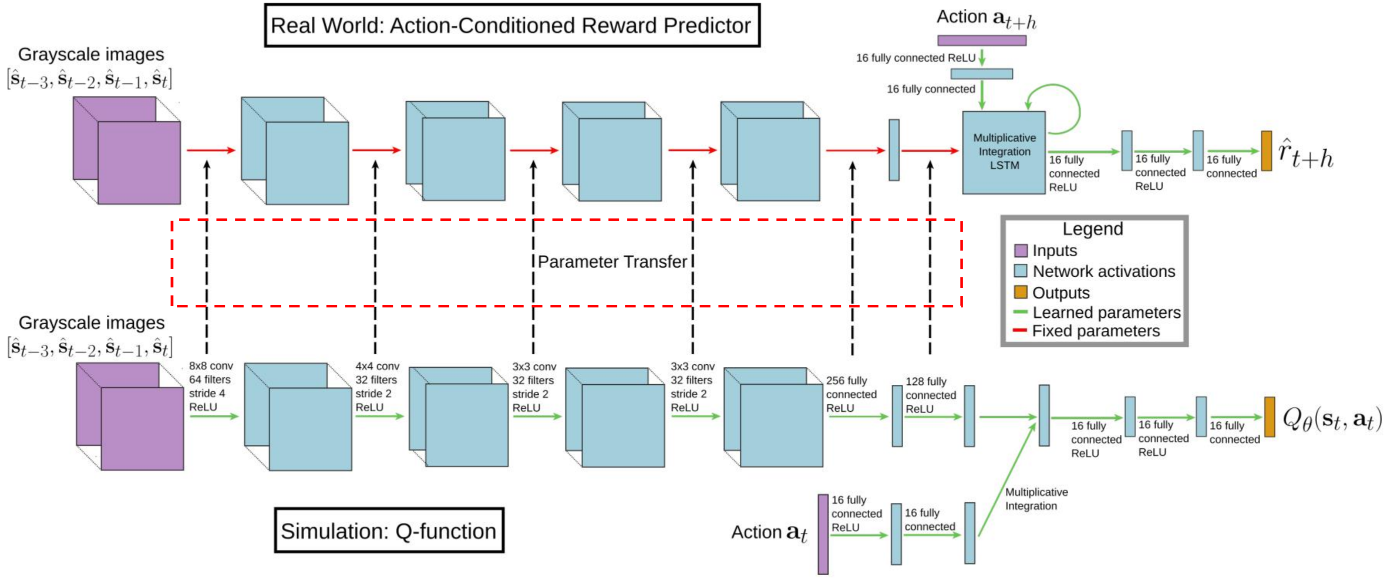
Generalization through Simulation(GtS)

- Learning a task-specific model:
  - Train a DNN Q-function $Q_\theta(s_t, a_t)$ using Q-learning:
    - ① We have access to large amounts of data in simulation, which is a requirement for deep Q-learning.
    - ② Q-learning can learn long-horizon tasks, which may improve the visual features that it learns.



Generalization through Simulation(GtS)

- Visual perception system transfer:
  - Initialize the weights of real-world policy 's perception layers using the layers from the Q-function learned in simulation.
  - Hold these perception layers fixed during real-world policy training.



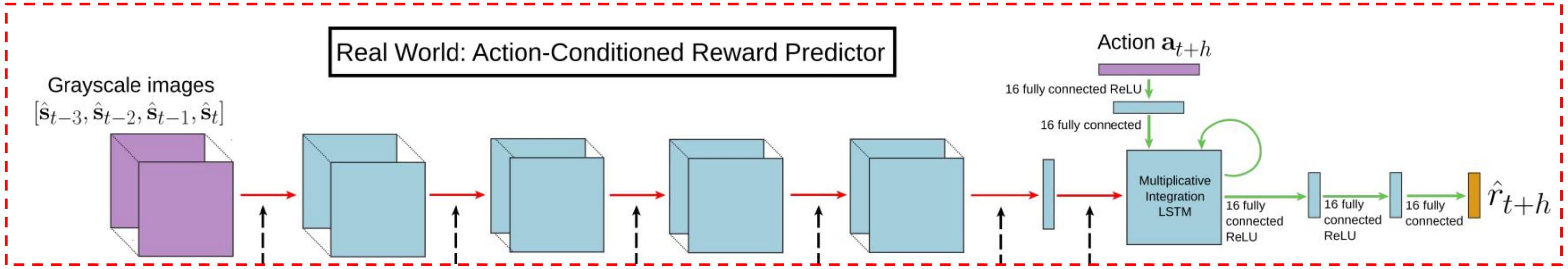
Generalization through Simulation(GtS)

- Real-world policy learning:
  - Action-conditioned reward predictor: $G_\theta(\mathbf{s}_t, \mathbf{A}_t^H)$
    - input: $\mathbf{s}_t$ & $\mathbf{A}_t^H = (\mathbf{a}_t, ..., \mathbf{a}_{t+H-1})$
    - output: $\hat{R}_t^H = (\hat{r}_t, ..., \hat{r}_{t+H-1})$
  - At training time, minimize the loss (supervised learning):

$$\theta^* = \arg\min_\theta \sum_{(\mathbf{s}_t, \mathbf{A}_t^H, R_t^H) \in \mathcal{D}^{\text{RW}}} \| G_\theta(\mathbf{s}_t, \mathbf{A}_t^H) - R_t^H \|^2$$

  - At test time, solve the optimal action sequence(MPC, CEM etc):

$$\mathbf{A}^* = \arg\max_{\mathbf{A}} \sum_{h=0}^{H-1} \gamma^h \hat{r}_{t+h}$$



Generalization through Simulation(GtS)

- Algorithm overview:
  - First, train a DQN in a visually diverse set of simulated environments.
  - Then, transfer the perception layers from the DQN to the action-conditioned reward predictor(ACRP).
  - Next, train the ACRP using real-world data gathered by the robot(hold the perception layers fixed).
  - At test time, use the learned ACRP by MPC etc to select an optimal action sequence, and executing the first action.

Generalization through Simulation(GtS)

- Expreiment:
  - Does including real-world data improve performance?
  - Does the ACRP lead to better real-world policies compared to Q-learning?
  - Is a task-specific or task-agnostic simulation-trained model better for real-world transfer?
  - Does transferring the perception module from the simulation-trained model improve real-world performance?

	Simulation Model	Perception System Transferred	Real-World Learned Model	Uses Real-World Data	Perception Layers Trained with Real-World Data	Time Until Collision (seconds, max 86)	Percentage Hallway Traversed
sim only	Task-specific	N/A	N/A	✗	N/A	16.5 (0.5)	19
sim fine-tuned	Task-specific	✗	Q-function	✓	✓	6.0 (28.5)	7
sim fine-tuned perception fixed	Task-specific	✗	Q-function	✓	✗	6.5 (66.5)	8
real-world only	N/A	✗	ACRP	✓	✓	7.8 (30.0)	9
supervised (ImageNet) transfer	N/A	✓	ACRP	✓	✗	9.5 (4.5)	11
unsupervised (VAE) transfer	Task-agnostic	✓	ACRP	✓	✗	21.0 (19.3)	24
GtS (ours)	Task-specific	✓	ACRP	✓	✗	**85.8 (2.5)**	**100**

Conclusion

- GCG:
  - A generalized framework that is suited for MB & MF.
  - Avoid predicting (high dimensional) future state, make it easier to learn.
  - The main limitation is horizon(H)
- GtS:
  - Reduce the real-world learning to supervised learning which need less data.
  - Simulation learning with Q-learning may improve the visual (task-relevant) features that it learns.
  - Hold the perception layers fixed to prevent overfitting to the real-world data.

Reference

- Self-Supervised Deep Reinforcement Learning with Generalized Computation Graphs for Robot Navigation
- Generalization through Simulation: Integrating Simulated and Real Data into Deep Reinforcement Learning for Vision-Based Autonomous Flight