Sergey Levine: https://www.youtube.com/watch?v=17NrtKHdPDw&t=1&ab_channel=RAIL

Introduction

- Generative AI
  - Generate content using models
  Why should we care about RL?
- Modern GenAI is impressibe because it looks like something humans would do.
- RL: Get results that people would likely not get.
  - Use techniques that goes beyond human

Why should we run RL in real world?

- Simulators can be hard to build
  - Not very important according to Sergey
- Emergence is way more exciting in **complex** environments
  - Need powerful algos interacting with complex environments
- If RL can produce optimised policies, it's much better to "overfit" to the real world
  - Find policies to world in which it will be deployed, compared to policy in simulators for a general use-case

A quiet revolution in real-world deel RL

conventional wisdom

- Deep RL requires millions(billions?) of trials
- Only runs in simulation
- Completely divorced from data-driven learning

Reality: above view is outdated

- Sample complexity is now very good
  - We can learn locomotion skills in just 6 minutes
- Real-world RL is more practical
- We are figuring out how to use prior data

Learning locomotion in the real-world

- In 2016:
  - Learn bipedal location

- 6 days of real time(training data)
- .. robot can run on infinite plane

We can do better

Ingredeitns of sample-efficient real-world RL

- Sample efficient **off-policy** algorithm(eg: soft actor-critic)
  - Use q-function to recycle old data via replay buffer
- Good **regularization** to enable gradient steps
  - Ideally, number of gradient steps >>> number of time steps
  - As num of gradient steps increase, chance of overfitting increases
  - This is why we need good regulaization.
  - Many approaches -> they all work -> Key is to use regularization, not skip it
- Take as many gradients steps per time step as possible

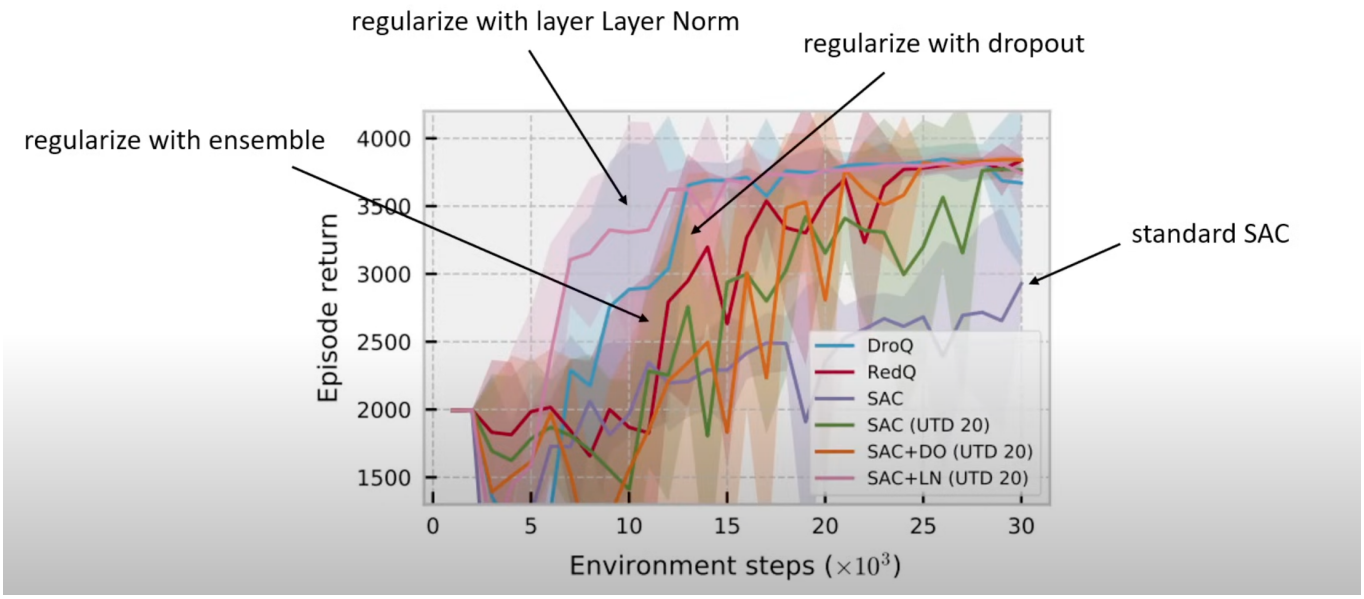
Further components to make it work

- Multi-task formulation to enable rests
- Avoid catastrophic failures(eg: failing) wherever possible
  - Main problem: It slows down learning
  - Avoiding it will help learn faster

How much faster are latest RL methods?

- Quite a bit faster.
- Robot learning on it's own in garden
  - Learns it about 20 minutes
- Because it's fast, it can be trained on many different terrains

What makes this fast?

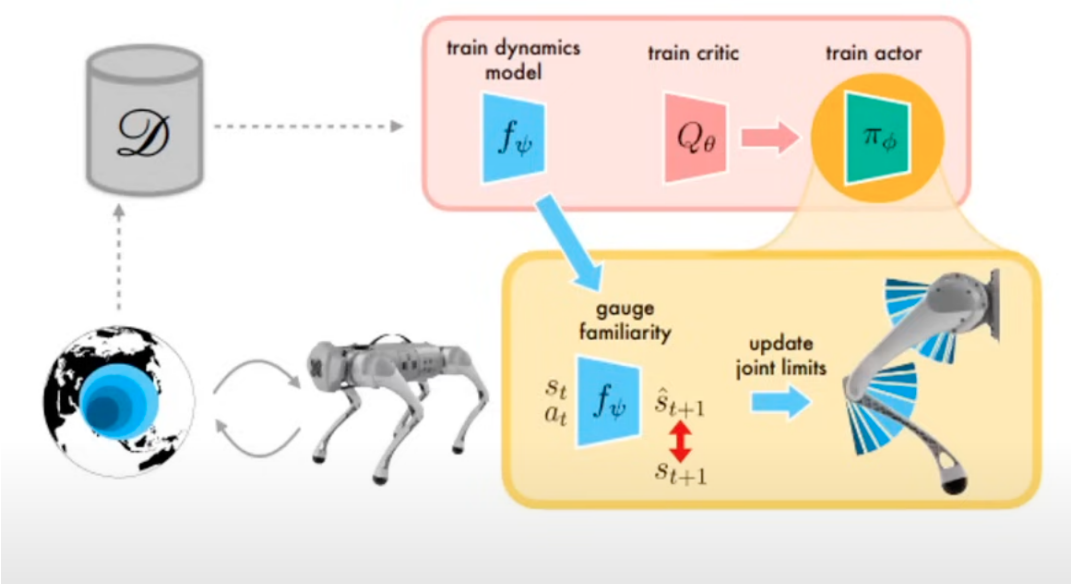


- Steps outline before
- Regularization with ensemble
  - More graidents update per time step
- Regularize with dropout is better
- Regularize with layer norm
  - Works best

Can we learn to walk even faster?

- Relates to how the algorithm is deployed
- Difference is how exploration is done:
  - APRL Exploration
  - Incentive more aggresive behaviour. But ideally not reach novel behaviour that causes catastrophic failure
  - **Solution**: Prevent the robot to NOT explore too much novel behaviour
    - Limit actions robot can take until the robot is "ready" for novel actions
    - "Ready" -> It's familiar enough with the current situation using old actions, such that it can use more aggresive actions appropriately without doing all the crazy stuff
    - Train a dynamics model
      - Train next state from current state
      - Usually used to find novel states
      - HERE, constrain the robot more when the dynamics model in a novel state i.e sees more error

- Don't let robot take aggresive actions when it's in a state that it's not familiar with
- How is this enforced: Add harsher/stronger **joint** limits i.e joint movements
- Tell robot: If you're not sure what's going on, BE CAREFUL AND SLOW DOWN



- Once it's sure, then take more aggresive actions.
- Allows spending real-world time learning more useful actions

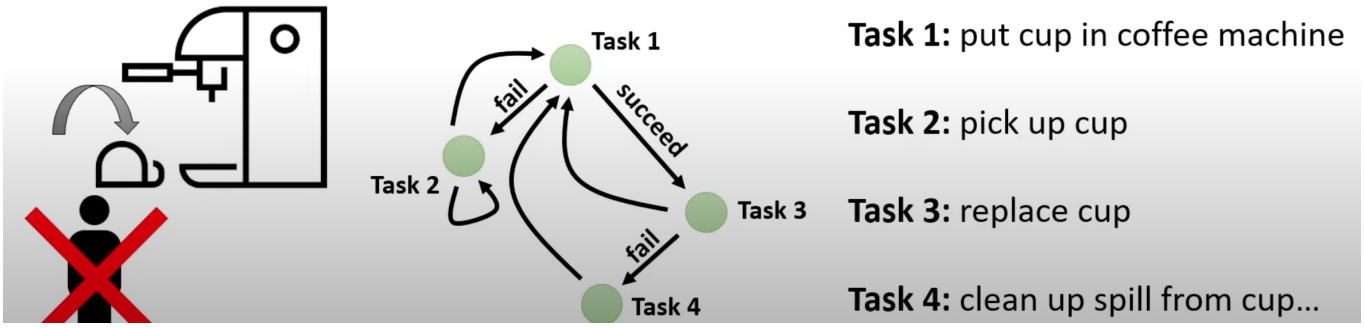
- **Why is this so important**
  - Small actions = "safe" exploration
  - safe exploration = fewer falls
  - fewer falls = much faster learning

- **General Lesson**
  - Exploration is not about always training new things
  - Instead, cautiously explore new situations.

Learning manipulation Skills

What might go wrong in real-world RL

- If you fail a task, maybe its an opportunity to try solving a differen task.
- So long as tasks keeping reseting to each other, this is a continous loop that can be used to train the robot on multiple tasks, without human interference/another robot

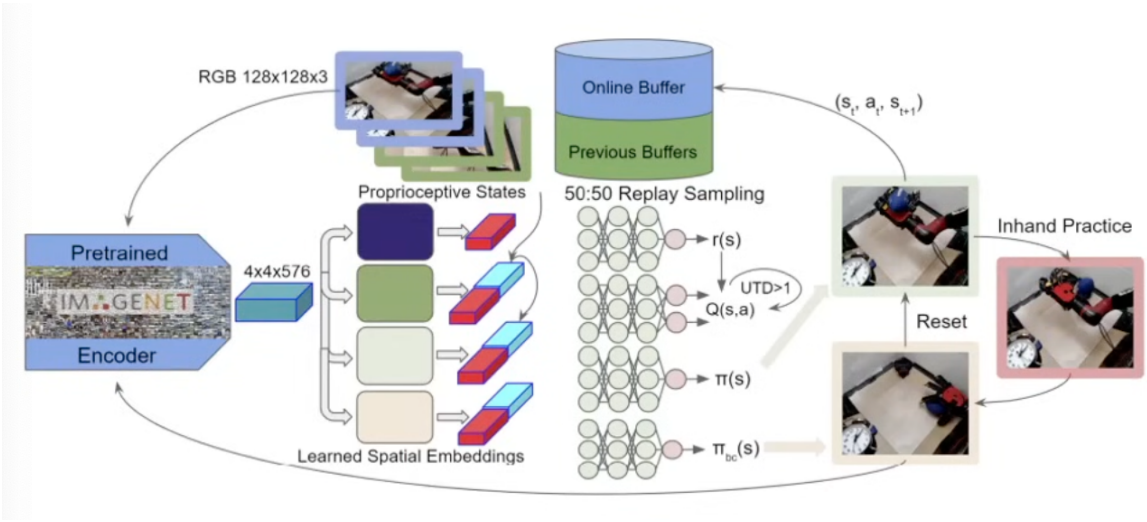


How do we learn without human intervention?

- Setup graph of tasks such that they are all linked.

Scaling it up with vision and prior data

- Paper: REBOOT: Reuse Data for Bootstrapping Efficient Real-world Dexterous Manipulation, 2023
- Instead of learning from scratch, robot might have learned other tasks.
  - We have prior data
  - Preload data in buffer
  - We'll learn this new task "reset free"?



- Most skills are trained in 6-8 hours. 10x improvement
- 2x improvement in efficiency
  - Prior data helped, even though prior data was for a different task

Learning with hetrogenous prior data

- Instead of using carefully selected and small prior datasets, might want to use big generative models that are trained on huge amounts of prior data
- This is kinda how people learn!

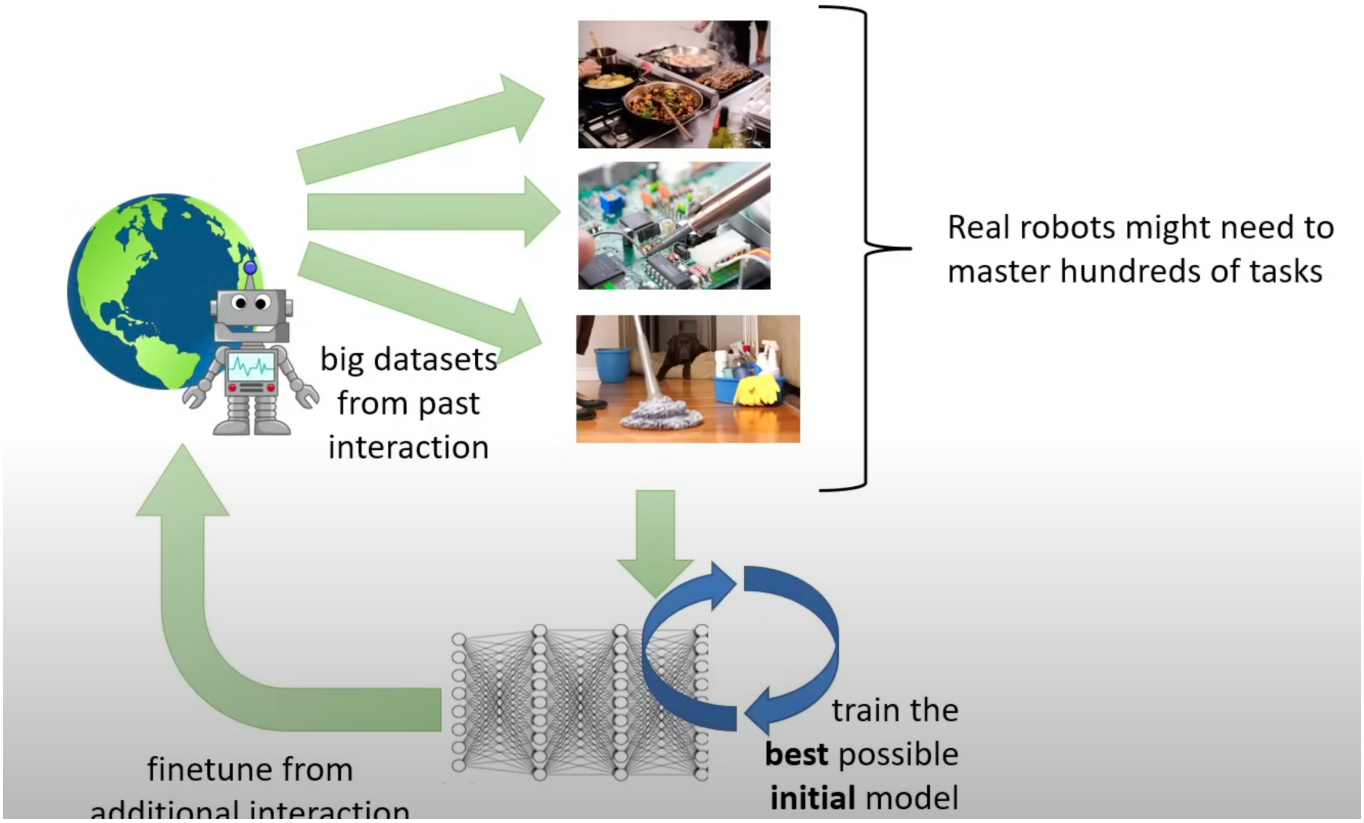
Pretraining from hetrogenenous data

- Paper: GNM: A general navigation model to drive and Robot
- Pretty powerful generalizable model
- Not doing finetuning yet

Now finetune with sample-efficient RL

- Ran offline RL training with IQL
- Use the above as init data for a sample efficient RL algorithm, similar to the one's already described(slow actor-critic(SAC))
- Results in robots that can be trained in about 30 minutes, with performance comparable to huamans. This is tested specfically in robot bar driving scenarios

MOAR SCALING: Can we do all of this at a larger scale?



- Does RL work a **large** scale?

Large-scale RL with language instructions

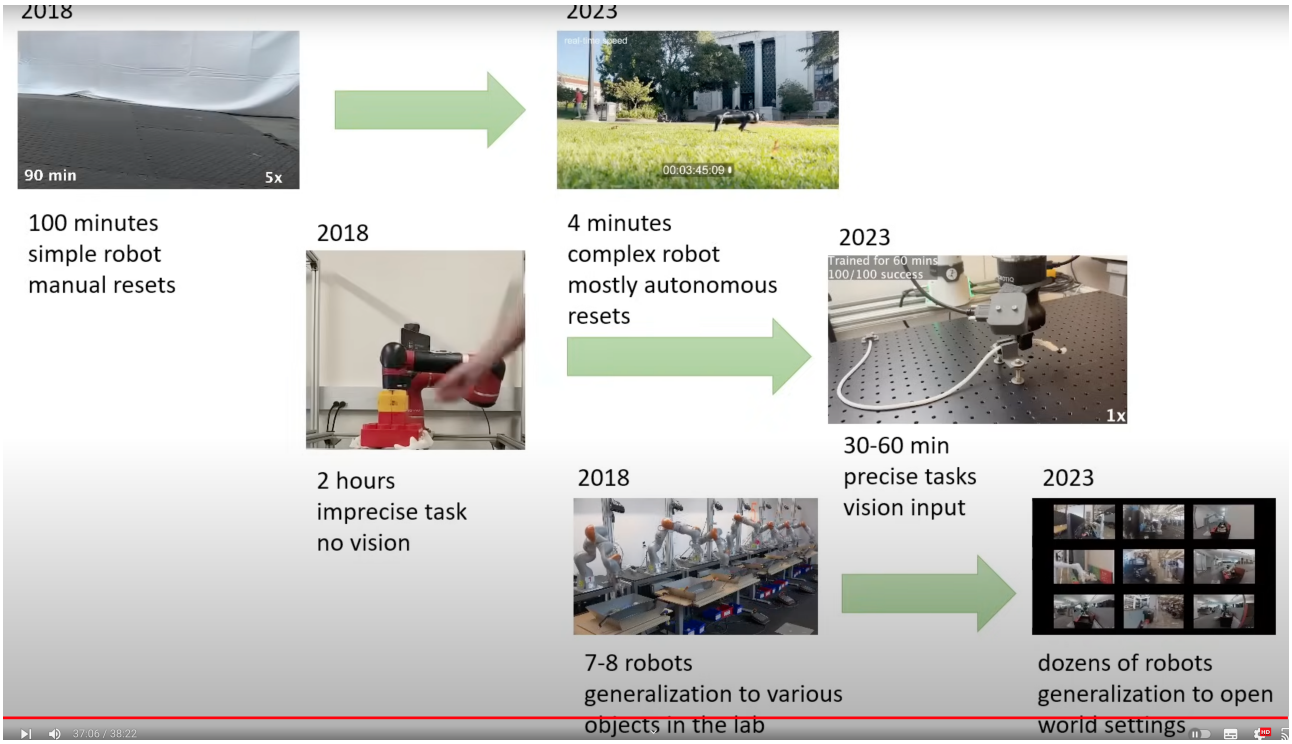
- Paper: Q-Transformer
- Based on **conservative Q-learning(CQL)**

- Success rate is more than doubled(56% compared to accuracy of RT-1 robot at 25%)

Robotic RL at huge scale

- Paper: Deep RL at scale
- Deployed in Google
- Task: Sort garbage in bins
- Multiple levels of pretraining and finetuning
- Final success rate was about 80%

Summary:

- Last 5 years have seen many improvements in efficacy of real-world RL
- 

What is left?

- A lot more juice left in **prior data**
  - How well can we accelerate exploration by leveraging prior data?
  - Can we enable safe and robust learning in the real world by leveraging prior data?
- Continual and lifelong learning has a ton of potential for real-world RL
  - Distinguishing feature of RL methods: They can **continually** and **perpetually** improve after deployment
- Stability, efficiency and reliability