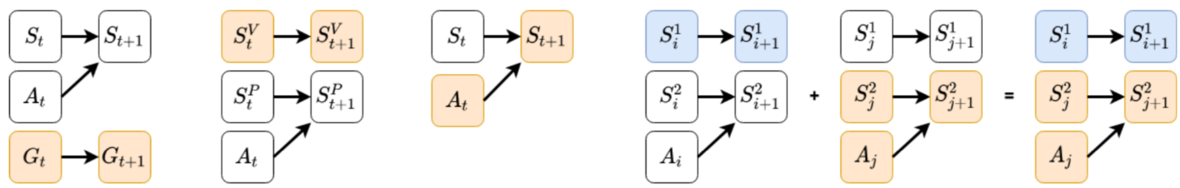
0.1 Counterfactual Data Augmentation (CoDA)

- dynamic processes involve a set of interacting subprocesses
- interactions between subprocesses are sparse
- introduces **local causal models** included from a global causal model by conditioning on a subset of state space
- local structures + experience replay to generate counterfactual experiences
- locally (during time between interaction) factor dynamics and model subprocesses independently
- underlying processes are difficult to model precisely
- factored subspaces of observed trajectory pairs are swapped
- CoDA is a data-augmentation strategy
- use attention to discover local causal structure
- improves sample-efficiency in batch-constrained and goal-constrained RL
- model time slice $(t, t+1)$ using a structural causal model (SCM)
- $\mathcal{M}_t = \langle V_t, U_t, \mathcal{F} \rangle$ with a DAG $\mathcal{G}$
- V_t is the set of state, action and next states, U_t is a set of noise variables and $\mathcal{F}$ is the set of structure equations that maps noise $\times$ current state to next state
- assume structural minimality allows us to think of edges in $\mathcal{G}$ as representing global causal dependence
- $P(S^i_{t+1}|S_t, A_t) = P(S^i_{t+1}|Pa(S^i_{t+1}))$ and S is independent of all nodes that isn't its parent
- often exists large subspace ($\mathcal{L}^{(j \perp\!\!\!\perp i)} \subset \mathcal{S} \times \mathcal{A}$) such that the next state is independent
- e.g. two-armed robot, there are many states such that the two arms are too far apart to influence each other
- consider a *local causal model* ($\mathcal{M}_t^{\mathcal{L}^{(j \perp\!\!\!\perp i)}}$) whose DAG is strictly sparser than the global DAG

Figure 1: Different instances of CoDA



(a) Goal relabeling **(b)** Visual augmentation **(c)** Dyna (model-based) **(d)** Counterfactual Data Augmentation by swapping locally independent subspaces **(ours)**

- Dyna augments real states with new actions and resamples the next state using a learned dynamics model
- can generate an exponential amount of data with CoDA (n independent samples from subspace $\mathcal{L}$ whose graph has m connected components $\implies n^m$ samples)
- CoDA can be used to mix and match data across timesteps

Figure 2: CoDA algorithm

Algorithm 1 Mask-based Counterfactual Data Augmentation (CoDA)

function $\textsc{CoDA}$(transition $\texttt{t1}$, transition $\texttt{t2}$):
- $\texttt{s1}, \texttt{a1}, \texttt{s1'} \leftarrow \texttt{t1}$
- $\texttt{s2}, \texttt{a2}, \texttt{s2'} \leftarrow \texttt{t2}$
- $\texttt{m1}, \texttt{m2} \leftarrow \textsc{Mask}(\texttt{s1}, \texttt{a1}), \textsc{Mask}(\texttt{s2}, \texttt{a2})$
- $\texttt{D1} \leftarrow \textsc{Components}(\texttt{m1})$
- $\texttt{D2} \leftarrow \textsc{Components}(\texttt{m2})$
- $\texttt{d} \leftarrow$ random sample from $(\texttt{D1} \cap \texttt{D2})$
- $\tilde{\texttt{s}}, \tilde{\texttt{a}}, \tilde{\texttt{s}}\texttt{'} \leftarrow \text{copy}(\texttt{s1}, \texttt{a1}, \texttt{s1'})$
- $\tilde{\texttt{s}}\texttt{[d]}, \tilde{\texttt{a}}\texttt{[d]}, \tilde{\texttt{s}}\texttt{'[d]} \leftarrow \texttt{s2[d]}, \texttt{a2[d]}, \texttt{s2'[d]}$
- $\tilde{\texttt{D}} \leftarrow \textsc{Components}(\textsc{Mask}(\tilde{\texttt{s}}, \tilde{\texttt{a}}))$
- **return** $(\tilde{\texttt{s}}, \tilde{\texttt{a}}, \tilde{\texttt{s}}\texttt{'})$ **if** $\texttt{d} \in \tilde{\texttt{D}}$ **else** $\emptyset$

function $\textsc{Mask}$(state $\texttt{s}$, action $\texttt{a}$):
Returns $(n+m) \times (n)$ matrix indicating if the n next state components (columns) locally depend on the n state and m action components (rows).

function $\textsc{Components}$(mask $\texttt{m}$):
Using the mask as the adjacency matrix for $\mathcal{G}^{\mathcal{L}}$ (with dummy columns for next action), finds the set of connected components $C = \{C_j\}$, and returns the set of independent components $D = \{\mathcal{G}_i = \bigcup_k \mathcal{C}_k^i \,|\, \mathcal{C}^i \subset \text{powerset}(C)\}$.

- specify a ratio between observed to counterfactual data to control for selection bias
- inferring local fatorization, how to approximate causal model
- use a global network mask (MADE) for autoregressive distribution modeling and GraN-DAG for causal discovery
- locally conditioned network mask by taking matrix product of locally conditioned layer masks
- MLP and single-head set transformer
- Inferring local factorization
  - SANDy (Sparse Attention Neural Dynamics)
  - it learns a function or mask that represents the adjacency matrix of the local causal graph
  - SANDy transformer performs better than MLP in AUC score
  - SANDy mixture is only sufficient for the simple synthetic MP environments, does not work for Spriteworld
  - the transformer has stronger inductve bias and more reliably infer local interaction patterns
- Some experiment details and notes:
  - working with the original TD3 codebase
  - mask function trained using 42k samples from random policy

- increase agent batch size from 256 to 1000, more batch size allows agent to see more of its on-policy data in the face of many off-policy CoDA samples
- batch-RL experiment in the Pong environment
- trained a transformer to learn a mask function
- goal-conditioned RL experiments on OpenAI gym FetchPush environment
- these experiments use a hand-coded heuristic designed with domain knowledge
- e.g. action is entangled with gripper and gripper + objects are disentangled when they are more than 10cm apart

- Other experimental insights:
  - tried augmenting the buffer by sampling from a dynamics model similar to model-based RL
  - good at next-state prediction, but fail to capture collisions and long-term dependencies

0.2 Differentiable Causal Discovery from Interventional Data (DCDI)

- learning a causal directed acyclic graph from data
- reformulates as continuous constrained optimization problem, solved using the augmented Lagrangian method
- this work proposes a NN model that can leverage interventional data
- using only observational data is challenging because the *faithfulness assumption* states that the true DAG is only identifiable up to a *Markov equivalence class*
- this can be improved by considering interventional data
- when observing enough interventions, the DAG is exactly identifiable
- there are score-based and constraint-based optimization methods, but they are computationally expensive and rely on parametric assumption
- perfect interventions remove the dependencies of a node on its parents, e.g. gene knockout in biology
- two classes of methods in causal structure learning
  - constraint-based methods
    * PC algorithm works with observational data
    * rely on conditional independence
    * COmbINE and HEJ support interventional data
    * JCI supports latent cofounders and can deal with interventions with unknown targets

- score-based methods
  - formulate problem of estimating DAG G^* by optimizing a score function $\mathcal{S}$ over the space of DAGs

$$\mathcal{G} \in \underset{\mathcal{G} \in \text{DAG}}{\text{argmax}} \mathcal{S}(\mathcal{G})$$

  - common choice in the purely observational setting is the regularized ML

$$\mathcal{S}(\mathcal{G}) := \max_{\theta} \mathbb{E}_{X \sim P_X}[log f_{\theta}(X) - \lambda |\mathcal{G}|]$$

  - the space of DAGs is super-exponential in the number of nodes
  - these methods rely on greedy combinatorial search algorithms
  - e.g. GIES (adaptation of GES), assumes a *linear* gaussian model
  - CAM uses greedy search, nonlinear, additive noise model
- hybrid-methods
  - combines both score-based and constraint, e.g. IGSP

- *weighted adjacency matrix* and acyclicity constraint: $\text{Tr}e^{A_\theta} - d = 0$
- shown that graph is acyclic iff the above constraint is satisfied
- problem is solved approximately using an augmented Lagrangian procedure
- performance is assessed by two metrics comparing estimated graph to the ground-truth graph
- *structural Hamming Distance* (SHD) = number of edges that are different between the two DAGs
- *structural interventional distance* (SID) how the two DAGs differ wrt to their causal inference statements
- DCDI relies on stochastic gradient descent and thus is scalable to graphs with hundreds of nodes even though the augmented Lagrangian procedure requires computing marix exponential which is $O(d^3)$
- Synthetic datasets
  - first sample a DAG using the Erdos-Renyi scheme
  - perfect v.s. imperfect interventions
  - different types of causal mechanisms: linear, additive noise model (ANM), and nonlinear with non-additive noise (NN)
-

0.3 Off-Policy Deep Reinforcement Learning without Exploration (BCQ)

- Extrapolation error: unseen state-action pairs are estimated to have unrealistic values
- Error can be attributed to mismatch in distribution of data induced by policy and data contained in the batch
- BCQ uses state-conditioned generative model to produce only previously seen actions
- Extrapolation error can be attributed to several causes:
  - Absent data: (s, a) is unavailable
  - Model bias
  - Training mismatch
- Off-policy algorithms (e.g. DDPG) deteriorate in perfomance when data is uncorrelated and value estimate produced by the Q-net diverges
- Off-policy algorithms are ineffective when learning *truly off-policy*
- Three different batch tasks: final buffer, concurrent, and imitation
- Behavioral agent consistently outperformed the off-policy agent trained with DDPG
- Batch-constrained policies trained to select actions with respect to three objectives:
  - Minimize distance of selected actions to data in the batch
  - Lead to states where familiar data can be observed
  - Maximize the value function
- BCQ generates plausible candidate actions with high similarity to the batch and selects the highest valued action using a learned Q-network
- Actions outputted using a generative model $G_w(s)$
- A perturbation model $\xi_\phi(s, a, \Phi)$ which is used to increase the diversity of the actions
- The choice of the number of actions to sample n and Φ creates a trade-off between imitation learning and RL
- Also uses a modified Clipped Double Q-Learning to estimate value by taking a convex combination of two Q-networks
- Experiments and results
  - MuJoCo environments in OpenAI gym
  - Baselines: DDPG, DQN, BC, and VAE-BC
  - Tasks: HalfCheetah, Hopper, Walker2d
  - BCQ is the only one that succeeds at all tasks, matching or outperforming BC

Algorithm 1 BCQ

Input: Batch $\mathcal{B}$, horizon T, target network update rate τ, mini-batch size N, max perturbation Φ, number of sampled actions n, minimum weighting λ.
Initialize Q-networks $Q_{\theta_1}, Q_{\theta_2}$, perturbation network ξ_ϕ, and VAE $G_\omega = \{E_{\omega_1}, D_{\omega_2}\}$, with random parameters θ_1, θ_2, ϕ, ω, and target networks $Q_{\theta'_1}, Q_{\theta'_2}$, $\xi_{\phi'}$ with $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2, \phi' \leftarrow \phi$.
for $t = 1$ **to** T **do**
  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  $\mu, \sigma = E_{\omega_1}(s, a), \quad \tilde{a} = D_{\omega_2}(s, z), \quad z \sim \mathcal{N}(\mu, \sigma)$
  $\omega \leftarrow \mathrm{argmin}_\omega \sum (a - \tilde{a})^2 + D_{\mathrm{KL}}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1))$
  Sample n actions: $\{a_i \sim G_\omega(s')\}_{i=1}^n$
  Perturb each action: $\{a_i = a_i + \xi_\phi(s', a_i, \Phi)\}_{i=1}^n$
  Set value target y (Eqn. 13)
  $\theta \leftarrow \mathrm{argmin}_\theta \sum (y - Q_\theta(s, a))^2$
  $\phi \leftarrow \mathrm{argmax}_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$
  Update target networks: $\theta'_i \leftarrow \tau\theta + (1 - \tau)\theta'_i$
  $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
end for

  - Achieves high performance in very few iterations, able to disentangle poor and expert actions
  - Suggests that extrapolation error has been successfully mitigated, BCQ is able to accurately estimate the true Q-value

0.4 IRIS: Implicit Reinforcement without Interaction at Scale for Learning Control from Offline Robot Manipulation Data

- IRIS is a framework that addresses the problem of offline policy learning from a large set of diverse and suboptimal demonstrations
- Use demonstrated data in lieu of reward function, avoids the problem of exploration
- Conventional IL methods assume that demonstration data is near-optimal and unimodal
- High-level controller selects a new goal state s_g for T timesteps, low-level controller is conditioned on s_g and tries to reach that state in the T timesteps
- High-level controller has two parts
  - conditional VAE (cVAE): predicts the distribution of states $p(s_{t+T}|s_t)$, used to sample goal proposals
  - value function $V(s)$ used to select the most promising goal proposal
  - value function is trained using a simple variant of BCQ
- Low-level controller:
  - a goal-conditioned RNN that outputs an action a_t given s_t and s_g

– trained on trajectory sequences from dataset, the last observation in each sequence is treated as the goal

– trained using a simple Behavioral Cloning loss

– learns to copy action sequence that resulted in a particular observation

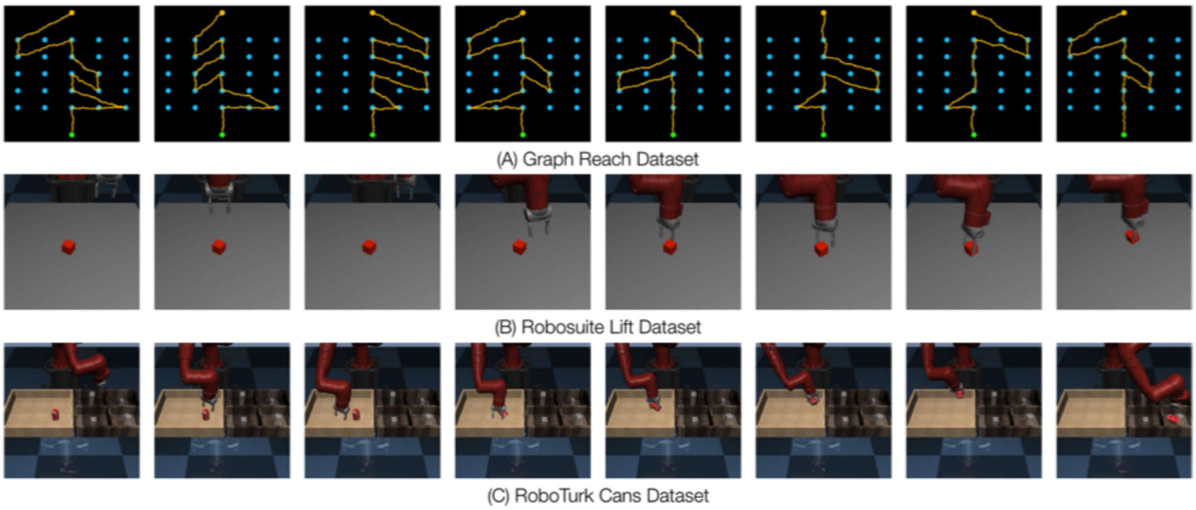
- This induces *selective imitation*
- Experiments:

– Graph reach is 2D navigation domain, contains 250 demonstrations, demonstration paths can take detours

– Robosuite Lift: actuate Sawyer robot to grasp and lift cube from table

– RoboTurk Can Pick and Place task and Can Image task

Figure 3: Tasks



– Baselines: BC, BC-RNN, BCQ, IRIS w/o cVAE, IRIS w/o Value Network

– All baselines achieve perfect performance on Graph Reach, IRIS 81.3% on Lift, and 28.3% on Cans dataset

0.5 Building Machines That Learn and Think

- desire to build systems that learn and think like people
- machines should build causal models of the world that support explanation and understanding and not just pattern recognition
- ground learning in intuitive physics
- harness compositionality and learning-to-learn, rapidly acquire knowledge to new tasks and situations
- deep learning models may be solving the problems differently than people do
- Prediction versus explanation

- Developmental start-up software, cognitive capabilities that are present early in development
  - 1) intuitive physics
  - primitive object concepts that allow them to track objects over time and discount physically implausible trajectories
  - new task, but physics still works the same way
  - 2) intuitive psychology
  - understanding that people have mental states like goals and beliefs
- model-building is the basis of human-level learning
- compositionality and learning-to-learn can make rapid model learning possible
- we are incredibly fast at perceiving and acting
- neural networks are designed for pattern-recognition rather than model-building
- integrate NN with rich model-building mechanisms can explain how human minds understand the world so well, so quickly
- humans can learn a lot more from a lot less
- single example of a new visual concept can be enough information to support: classification of new examples, generation of new examples, parsing object into parts and relations, and generation of new concepts
- DQN, simple model-free RL algorithm, that learns to play the game Frostbite
- visual system and policy are highly specialized to the games that it was trained on
- DQM plays games at human-level performance, but it is doing so in a way different than humans
- DQN trained on 200 million frames, 924 days, about 500 times more training experience as a human, not sample efficient
- Fundamental differences in representation and learning between people and DQN
- DQN relies on some reward function, otherwise take random actions
- DQN is inflexible to changes in inputs and goals, e.g. changing color of object will be harmful to performance
- People can understand the game + goals quickly. Moreover, people understand enough to invent new goals, generalize changes to input, and explain the game to others
- How do we bring to bear rich prior knowledge to learn new tasks and solve new problems quickly?
- Intuitive physics-engine approach to scene understanding
- PhysNet used DCNN to predict stability of block towers from simulated images
- Could neural networks be trained to emulate a general-purpose physics simulator?

- Integrating intuitive physics and DL could be important to more human-like learning algorithms
- Intuitive psychology can allow us to infer the beliefs, desires, and intentions of others (e.g. avoiding bird in Frostbite game)
- Injecting inductive bias can boostrap reasoning about abstract concepts
- One-shot learning is innate characteristic of humans from a young age
- Compositionality is the idea that new representations can be constructed from the combination of primitive elements
- Object-oriented reinforcement learning, representing a scene as a composition of objects
- compositionality is important at the level of goals and subgoals
- causality is a subclass of generative models that resemble how the data are actually generated
- causality can glue features together by relating them to a deep underlying cause
- perception without key ingredients and absence of causal glue can lead to errors
- network can get the objects correctly but fail to understand the physical forces at work
- learning to learn, learning a new task can be accelerated through previous or parallel learning of related tasks
- people transfer knwoledge at multiple-levels, learn compositionally structured causal models of a game
- there is evidence that suggests our brain has a model-based learning system, building a map of the environment and using it to plan action sequences for complex tasks
- Intrinsically motivated learning
- Responses to common questions
  - It is unfair to compare learning speeds of human and machine because of apriori knowledge / experience
  - Neuroscience in the long run will place more constraints on theories of intelligence
  - Language is essential to human intelligence and goes hand in hand with other key ingredients
  - Language facilitates more powerful learning-to-learn and compositionality, allow people to learn more quickly and flexibly
- AI has made incredible progress: beat chess masters, Jeopardy champions, facial recognition, speech understanding
- More exciting applications to come: self-driving, medicine, drug design, genetics, and robotics

- Recent advancements in incorporating psychological ingredients with deep networks: selective attention, augmented memory, experience replay
- Attention allows the model to focus on smaller sub-tasks rather than solving whole problem in one-shot
- Memory incorporates classical data structures into gradient-based learning systems
- AI systems like AlphaGo are trained on millions of self-play games whereas world champion probably only played 50,000 games in his lifetime
- Proposed goal: systems should see objects rather than features, build causal models and not just recognize patterns, recombine representations without retraining, and learning-to-learn rather than starting from scratch

0.6 A Review of Robot Learning for Manipulation

- Autonomous manipulation has manyyy applications: hospitals, elder and child care, factories, outer space, restaurants, service, and home
- robots perceive latent object properties by observing outcome of manipulation - interactive perception
- interactive perception is the basis for self-supervised learning
- manipulation tasks exhibits highly hierarchical structure!!
- this structure enables modularity for skills to be mixed and matched together
- tasks that are similar enough to not be considered unique skills is known as a task family
- exploit similarity between tasks to perform them more efficiently
- object-centric representations
- ability to handle novel concepts and unforeseen situations, robot must generalize knowledge
- robot learning can be formulated as an MDP
- common to model environment as a collection of object states
- skills are modelled after the options framework (HRL framework)
- option, $o = I_o, \beta_o, \pi_o$

$I_o =$ initiation set, indicator function describing when the option may be executed
$\beta_o =$ termination condition, describes the probability that an option finishes when reaching stat
$\pi_o =$ option policy, maps states to actions, motor skill controller

- discovering reusable skills
- robot needs to learn policies as a function of a context vector τ which encodes extra task-specific information (possibly factored into object)

- representations should capture within- and across task variations
- types of object variations: pose, shape, material properties, interactions / relative properties
- object models can be hierarchical, geometric and non-geometric properties
- point-level representations: point cloud, pixel, voxels
- they are flexible, each element can be associated with additional info
- part-level representations: can lead to better generalization, objects have similar parts that afford similar interactions
- object-level representations: define relative poses, forces, and constraints between objects
- can also be used to define different types of interactions or relations between objects
- passive perception (perceiving environment without exploiting physical interactions), useful for estimating position, shape, and material properties
- interactive perception (robot physically interacts with surroundings), e.g. push object to estimate its constraints or weight, estimate dynamic properties
- allows robot to reduce uncertainty
- robots combine multiple sensor modalities (vision + touch and haptics) and incorporate task information (e.g. instructions)
- transition models can be reused between different tasks, but the new task needs to share the same state, action, and context spaces
- transfer depends on overlap between data distributions
- covariate shift (input varies) and dataset shift (input and output varies)

0.7 Lessons from Amazon Picking Challenge

- Amazon Picking Challenge tests ability of robotic system to fulfill orders by picking items from a warehouse shelf
- Build tightly integrated systems and modularize the system by breaking it down into simpler subproblems
- In manipulation it is important to consider alternative embodiments
- Integrate planning with feedback from physical interactions, interactions help reduce uncertainty
- Finding general solutions is desirable but may be infeasible. Try to search for reasonable and useful assumptions to simplify problem
- Challenges: narrow bins and objects were barely visible and partially obstructed, floor made of reflective metal, poor lighting conditions
- Use joint- and task- space feedback controllers

- Use a mobile base to allow the arm to reposition itself to generate easier grasps, but increased the dimensionality of the configuration space
- Simple end-effector that using a suction cup, can pick up most objects, thin shape reduces need to consider complex collision avoidance
- Proper embodiment simplifies the overall solution
- Use a hybrid automaton where states correspond to a feedback controller and state transitions are triggered by sensor events
- Most of the failure cases occurered because of perception inability to discriminate objects properly
- There is recurring underlying structure in robotics problems, making suitable assumptions helps alleviate difficulties of general purpose solutions
- Adding explicit knowledge about physics makes problem more tractable

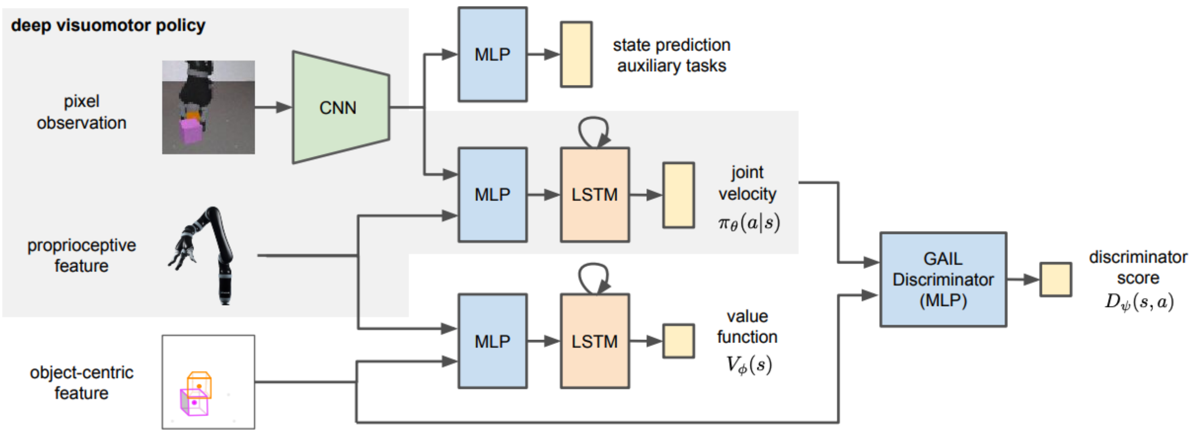
0.8 Reinforcement and Imitation Learning for Diverse Visuomotor Skills

- Model-free DRL method that leverages small amount of demonstration data to assist RL agent
- Applied to robotic manipulation tasks, end-to-end visuomotor policies that map RGB camera inputs to joint velocities
- RL for robotics, policies must map multi-modal and partial information to control of many DOFs
- Real tasks have contact-rich dynamics and vary along many dimensions, generalization challenge
- Exploration is challenging due to high-dimensional and continuous action space
- Techniques for exploiting priviledged + task specific information to accelerate + stabilize training
- Combine IL with RL using a hybrid reward (imitation reward based on GAIL)
- Also use demonstrated data to create a curriculum by randomizing the start state distribution
- Learn policy and value in separate modalities
- Value function is used in PPO for estimating the advantage to compute policy gradient, instead of using pixels, they use low-level physical states to train value function
- Auxiliary tasks for visual modules
  - improve learning efficiency
  - state-prediction layer used to predict locations of objects from camera observation

- use fully-connected layer to regress 3D coordinates, minimize l_2 loss

- GAIL discriminator uses object-centric representations (positions of objects), requires some domain knowledge
- Diversify training conditions such as visual appearance, object geometry, and system dynamics to sim2real transfer
- Deep visuomotor policy takes as input RGB observation and proprioceptive features (joint positions and angular velocities)
- GAIL has two networks: a policy network and a discriminator network and uses a min-max objective
- Policy is trained using policy gradient methods to maximum discounted sum of reward
- Employing shaping reward as a trick to facilitate exploration. Task rewards given as a sparse reward at different stages of the tasks, e.g. block stacking - reaching, lifting, stacking
- This is better than hand-crafting a dense shaping reward
- Hybrid reward: $r(s_t, a_t) = \lambda r_{gail}(s_t, a_t) + (1 - \lambda) r_{task}(s_t, a_t)$
- There is a balanced contribution between RL and IL rewards

Figure 4: Deep Visuomotor Policy



- Experiments
  - Block lifting, block stacking, pouring liquid, order fulfillment, clearing table
  - Kinova Jaco arm has 9 DOF: 6 arm joints and 3 actuated fingers
  - Used various objects ranging from basic geometric shapes to 3D objects made from primitive shapes
  - Sim2real is still a challenge, large domain gap, transfer is hindered by visual discrepancies, arm dynamics, and physical properties of the environment
  - Certain level of degradation when running on a real robot, zero-shot sim2real transfer

– On a real robot, there is often a delay in execution of action which is detrimental to robot's performance

– Fine-tuned agent in simulation subjected to a random chance of dropping actions

0.9 Making Sense of Vision and Touch

- Contact-rich manipulation tasks require haptic and visual feedback
- Use self-supervision to learn multimodal representation of inputs that are used to improve sample efficiency for policy learning
- Evaluated method on peg insertion for different geometry, configurations, and clearances
- Contact-rich tasks: peg insertation, block packing, edge following
- Diverse set of modalities including vision, range, audio, haptic, proprioceptive data and language and often these are complementary of each other
- DL usually requires lot of high-dimensional training data and self-supervision does not rely on having human annotated data
- Their model encodes 3 types of data: RGB, haptic feedback from F/T sensor, and proprioceptive data from joint encoders
- Heterogenuous nature requires domain-specific encoders for each modality
- Generate labels automatically through self-supervision
- The model has to predict 1) the optical flow generated by action and 2) whether end-effector will make contact with environment
- To exploit concurrency between data streams, use a third objective that predicts whether two streams are temporally aligned (binary classification)

Figure 5: Multimodal fusion model

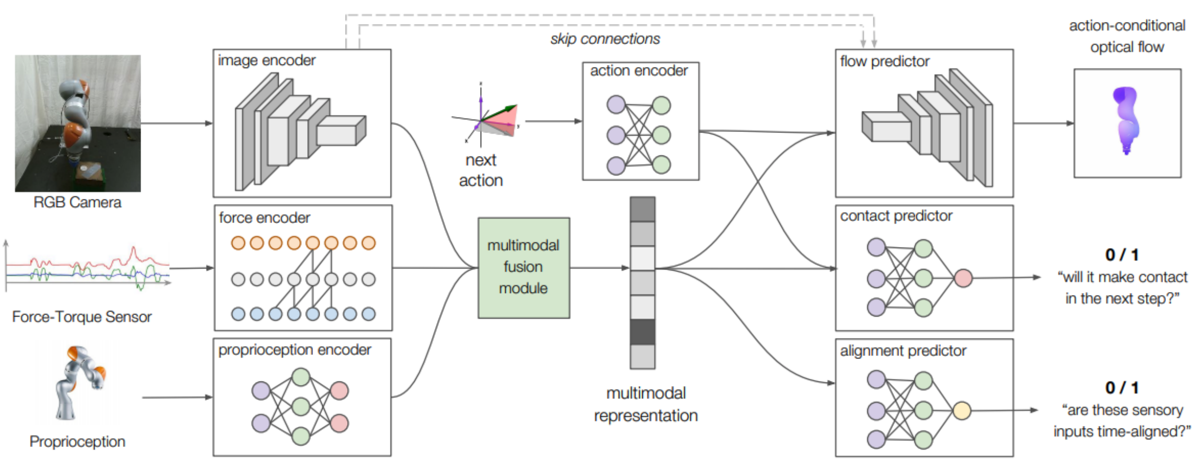
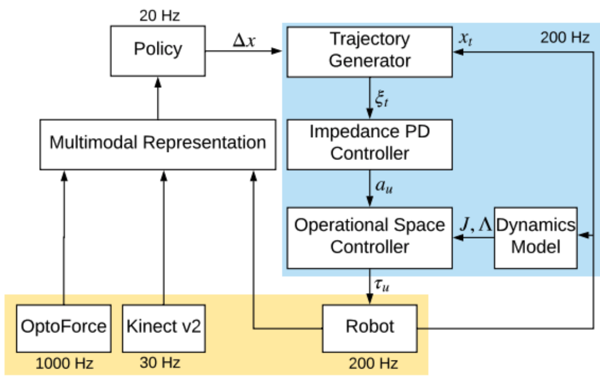


Figure 6: Controller

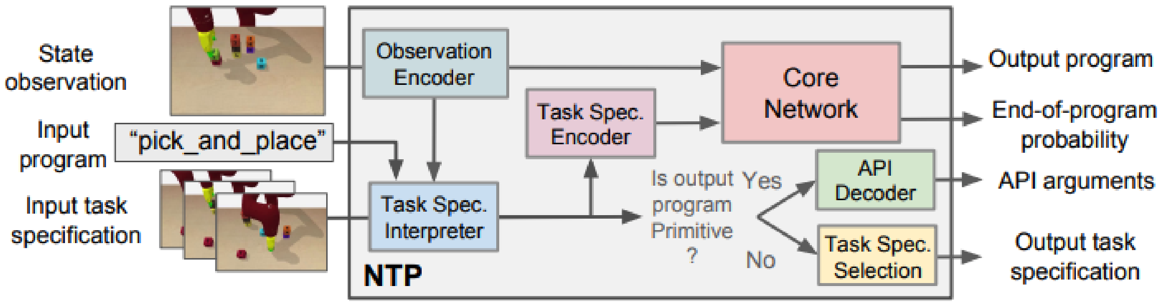


- Model-free removes need for an accurate dynamics model (hard to obtain in presence of rich contacts), use TRPO for policy learning
- Policy network is 2-layer MLP that takes in 128-d multimodal rep and produces a 3D displacementof end-effector
- Policy outputs Cartesian-control commands instead of joint-space commands
- Use direct torque control because it gives robot compliance making it safer
- Also make use of a staged-reward function for subtasks to simplify the challenge of exploration
- Conduct ablation study to learn about importance of each modality, also study robustness of policy in presence of sensor noise and external perturbation (e.g. pushing robot arm)
- Lots of other challenges in real-world: sensor synchronization, variable delays, real-world dynamics, etc

0.10 Neural Task Programming: Learning to Generalize Across Hierarchical Tasks

- few-shot learning from demonstration and neural program induction
- inputs a task specification (e.g. video specification) and recursively decomposes it into finer sub-tasks
- Complex manipulation tasks: object sorting, assembly, de-cluttering, etc
- NTP interprets a task specification and instantiates a hierarchical policy as a neural program
- Task specification can either be a task demonstration recorded as a state-trajectory or even a list of language instructions
- NTP generalizes to 3 kinds of variations in task structure: task length, task topology, and task semantics

Figure 7: NTP



- NTP is a meta-learning algorithm, decomposes final objective into simpler objectives recursively and each subtask is assigned a neural program
- NTP extends upon NPI (Neural Programmer-Interpreter)
- NPI is a type of neural program induction in, core of NPI is an LSTM which selects at every timestep the next program to run
- NTP has three parts: Task Specification Interpreter (f_{TSI}), Task Specification Encoder (f_{TSE}), and a core network (f_{CN})

Figure 8: NTP Algorithm

Algorithm 1 NTP Inference Procedure

Inputs: task specification ψ, program id i, and environment observation o

function RUN(i, ψ)

  $r \leftarrow 0$, $p \leftarrow M^{prog}_{i,:}$, $s \leftarrow f_{ENC}(o)$, $c \leftarrow f_{TSE}(\psi)$

  while $r < \alpha$ **do**

    $k, r \leftarrow f_{CN}(c, p, s)$, $\psi_2 \leftarrow f_{TSI}(\psi, p, s)$

    $i_2 \leftarrow \arg\max_{j=1 \ldots N}(M^{key}_{j,:} k)$

    if program i_2 is primitive **then** ▷ if i_2 is an API

      $\mathbf{a} \leftarrow f_{TSI}(\psi_2, i_2, s)$ ▷ decode API args

      RUN_API(i_2, $\mathbf{a}$) ▷ run API i_2 with args $\mathbf{a}$

    else

      RUN(i_2, ψ_2) ▷ run program i_2 w/ task spec ψ_2

    end if

  end while

end function

- NTP vs NPI: NTP can interpret task specifications and perform hierarchical decomposition, NTP uses APIs as primitive actions and it uses a reactive core network instead of a RNN
- APIs are subroutines for learning at an abstract level, APIs take in specific arguments (e.g. object category or end-effector position)
- APIs used were move_to, grip, and release

0.11 ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks

- Action Learning From Realistic Environments and Directives (ALFRED)
- Mapping natural language instructions and vision to actions for household tasks
- Contains long, compositional tasks to bridge gap between research and real-world applications
- Language directives have high-level goals and low-level language instructions
- Baseline model performs poorly on the ALFRED dataset, more room for grounded visual language models
- Lot of platforms for language-driven navigation, embodied QA
- 8k demonstrations, average of 50 steps
- Interact with objects by specifying a pixelwise interaction mask, unlike other settings where they treat localization as a solved problem
- Evaluate on seq2seq model like VLN tasks which is not effective achieving less than 5% success rate
- Need models that can address challenges of language, action, and vision for accomplishing household tasks
- Objects contain multiple visual variations different shapes, textures, and colors
- Tasks are parameterized by object of focus
- Generate expert demonstrations by encoding agent and environment dynamics into PDDL rules
- Split validation and test into *seen* and *unseen* environments to test models generalization to new spaces and novel object classes
- Baseline model: seq2seq architecture
  - CNN encodes visual input
  - bi-LSTM encodes the language directive
  - decoder LSTM infers low-level actions and attends over encoded language
- Model is trained using imitation learning, DAgger-style is non-trivial
- High-level language is concatenated with low-level language with a seperator
- Tasks require reasoning over long sequences of images and instructions, proposed two auxillary losses to add temporal information (progress monitoring)
- Progress monitoring is where the agent maintains an internal estimate of their progress towards the goal and subgoals
- Evaluate both Task Success and Goal-Condition success
- Seq2seq model only achieved 8 percent goal-conditioned success rate

- Also conducted some ablations to investigate unimodal ablation, found that both language and vision was necessary and a single modal is not sufficient to complete the task
- Possible directions: models that exploit hierarchy, are modular, and support structured reasoning and planning

0.12 Causal Discovery with Reinforcement Learning

- Propose to use RL to search for a DAG with the best scoring
- Encoder-decoder takes observable data and generates a graph adjacency matrix
- Reward = predefined score function + two penalty terms to enforce acyclicity
- GES checks acyclicity one edge at a time
- Problem is to use observed data X to infer the underlying causal DAG $\mathcal{G}$
- Modified BIC for score function, used same acyclicity constraint as in NOTEARS
- Overall reward function is $-[\mathcal{C}(\mathcal{G}) + \lambda_1 I(\mathcal{G} \notin \text{DAGs} + \lambda_2 h(A)$