10-703: Quiz 3 Review II

Slides by Athiya Deviyani and Alex Kireeff

General Tips

- Read the paper
  - You don’t have to remember a lot from the paper, just remember which sections discuss what

Problem 1.4: Conceptual Questions (10 pts)

1. Describe the main differences between AlphaZero and MuZero. Give an example for a problem where you would prefer MuZero over AlphaZero.
2. In the MuZero paper the authors describe an additional technique called Reanalyze. Describe what Reanalyze does and how it can improve sample efficiency. How would you implement this in the given code?
3. MuZero has no constraints to learn a consistent hidden state representation. Specifically consider a state o_t, and a state o_{t+1}. When embedded with the initial inference, this gives hidden states s_t and s_{t+1}. When we apply our dynamics to s_t, we should expect that the result $\hat{s}_{t+1}$ aligns with s_{t+1}. How might we enforce this constraint during training?

Appendix A Comparison to *AlphaZero*

MuZero is designed for a more general setting than *AlphaGo Zero* [40] and *AlphaZero* [39].

In *AlphaGo Zero* and *AlphaZero* the planning process makes use of two separate components: a simulator implements the rules of the game, which are used to update the state of the game while traversing the search tree; and a neural network jointly predicts the corresponding policy and value of a board position produced by the simulator (see Figure 1 A).

Specifically, *AlphaGo Zero* and *AlphaZero* use knowledge of the rules of the game in three places: (1) state transitions in the search tree, (2) actions available at each node of the search tree, (3) episode termination within the search tree. In *MuZero*, all of these have been replaced with the use of a single implicit model learned by a neural network (see Figure 1 B):

1) State transitions. *AlphaZero* had access to a perfect simulator of the true dynamics process. In contrast, *MuZero* employs a learned dynamics model within its search. Under this model, each node in the tree is represented by a corresponding hidden state; by providing a hidden state s_{k-1} and an action a_k to the model the search algorithm can transition to a new node $s_k = g(s_{k-1}, a_k)$.

2) Actions available. *AlphaZero* used the set of legal actions obtained from the simulator to mask the prior produced by the network everywhere in the search tree. *MuZero* only masks legal actions at the root of the search tree where the environment can be queried, but does not perform any masking within the search tree. This is possible because the network rapidly learns not to predict actions that never occur in the trajectories it is trained on.

3) Terminal nodes. *AlphaZero* stopped the search at tree nodes representing terminal states and used the terminal value provided by the simulator instead of the value produced by the network. *MuZero* does not give special treatment to terminal nodes and always uses the value predicted by the network. Inside the tree, the search can proceed past a terminal node - in this case the network is expected to always predict the same value. This is achieved by treating terminal states as absorbing states during training.

In addition, *MuZero* is designed to operate in the general reinforcement learning setting: single-agent domains with discounted intermediate rewards of arbitrary magnitude. In contrast, *AlphaGo Zero* and *AlphaZero* were designed to operate in two-player games with undiscounted terminal rewards of ± 1.

General Tips

- Read the paper
  - You don't have to remember a lot from the paper, just remember which sections discuss what

Problem 1.4: Conceptual Questions (10 pts)

1. Describe the main differences between AlphaZero and MuZero. Give an example for a problem where you would prefer MuZero over AlphaZero.
2. In the MuZero paper the authors describe an additional technique called Reanalyze. Describe what Reanalyze does and how it can improve sample efficiency. How would you implement this in the given code?
3. MuZero has no constraints to learn a consistent hidden state representation. Specifically consider a state o_t, and a state o_{t+1}. When embedded with the initial inference, this gives hidden states s_t and s_{t+1}. When we apply our dynamics to s_t, we should expect that the result $\hat{s}_{t+1}$ aligns with s_{t+1}. How might we enforce this constraint during training?

Appendix A Comparison to *AlphaZero*

MuZero is designed for a more general setting than *AlphaGo Zero* [40] and *AlphaZero* [39].

In *AlphaGo Zero* and *AlphaZero* the planning process makes use of two separate components: a simulator implements the rules of the game, which are used to update the state of the game while traversing the search tree; and a neural network jointly predicts the corresponding policy and value of a board position produced by the simulator (see Figure 1 A).

Specifically, *AlphaGo Zero* and *AlphaZero* use knowledge of the rules of the game in three places: (1) state transitions in the search tree, (2) actions available at each node of the search tree, (3) episode termination within the search tree. In *MuZero*, all of these have been replaced with the use of a single implicit model learned by a neural network (see Figure 1 B):

1) State transitions. *AlphaZero* had access to a perfect simulator of the true dynamics process. In contrast, *MuZero* employs a learned dynamics model within its search. Under this model, each node in the tree is represented by a corresponding hidden state; by providing a hidden state s_{k-1} and an action a_k to the model the search algorithm can transition to a new node $s_k = g(s_{k-1}, a_k)$.

2) Actions available. *AlphaZero* used the set of legal actions obtained from the simulator to mask the prior produced by the network everywhere in the search tree. *MuZero* only masks legal actions at the root of the search tree where the environment can be queried, but does not perform any masking within the search tree. This is possible because the network rapidly learns not to predict actions that never occur in the trajectories it is trained on.

3) Terminal nodes. *AlphaZero* stopped the search at tree nodes representing terminal states and used the terminal value provided by the simulator instead of the value produced by the network. *MuZero* does not give special treatment to terminal nodes and always uses the value predicted by the network. Inside the tree, the search can proceed past a terminal node - in this case the network is expected to always predict the same value. This is achieved by treating terminal states as absorbing states during training.

In addition, *MuZero* is designed to operate in the general reinforcement learning setting: single-agent domains with discounted intermediate rewards of arbitrary magnitude. In contrast, *AlphaGo Zero* and *AlphaZero* were designed to operate in two-player games with undiscounted terminal rewards of ± 1.

- If you are running out of revision time, **don't skip the paper**, watch the video or read the website/blog associated with the paper (linked to this slide)

Today

M **11/27**	Lecture #23 : **Visual Imitation Learning** [slides]	• Hahn et al. No RL, No Simulation:Learning to Navigate without Navigating • Aytar et al. Playing hard exploration games by watching YouTube • Peng et al. SFV Reinforcement Learning of Physical Skills from Videos • Baker et al. Video PreTraining (VPT):Learning to Act by Watching Unlabeled Online Videos
W **11/29**	Lecture #24 : **Language and Robot Control** [slides]	• Radford et al. Learning Transferable Visual Models From Natural Language Supervision • Shridhar et al. CLIPort What and Where Pathways for Robotic Manipulation • Brown et al.Language Models are Few-Shot Learners • Liang et al.Code as Policies:Language Model Programs for Embodied Control
M **12/04**	Lecture #25 : **Self-Supervised Visual Learning** [slides]	• Chen et al. A Simple Framework for Contrastive Learning of Visual Representations • Srinivas et al. CURL Contrastive Unsupervised Representations for Reinforcement Learning • Khandelwal et al. Simple but Effective CLIP Embeddings for Embodied AI

Visual Imitation Learning

No RL, No Simulation: Learning to Navigate without Navigating (Hahn et al.)

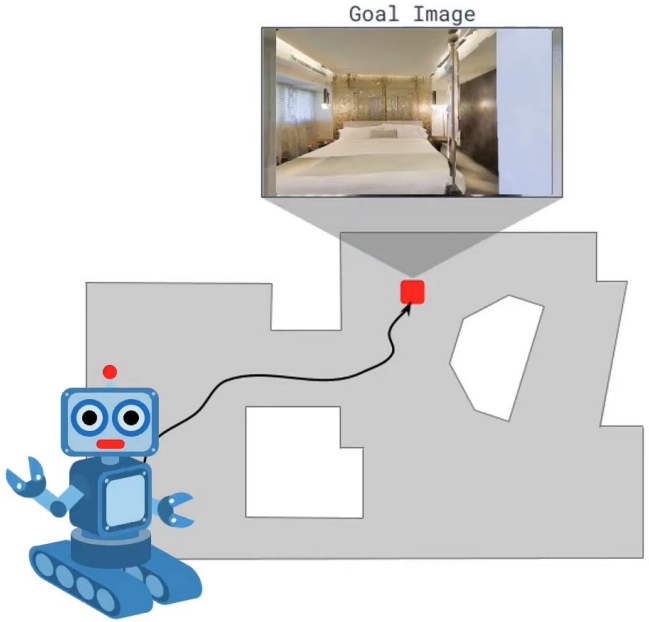
Imagine going to a friend's house and they ask you to get water from the kitchen.

This is the first time you are visiting, so you don't know where it is. Looking at the house, we can see that there are 3 possible paths that may take you to the kitchen.

Humans use semantic priors and understanding of commonalities in environments to navigate in unseen environments.

Source: https://meerahahn.github.io/nrns/

No RL, No Simulation: Learning to Navigate without Navigating (Hahn et al.)



Embodied agents have difficulty performing the same task, as the goal location is unknown and the agent needs to make intelligent and efficient exploration decisions to find and reach the goal.

Agents need to learn the semantic priors of the environment, which require large scale and diverse datasets.

Bottlenecks

1. Scalability: RL is sample inefficient
2. When goals are unknown, exploration is needed and samples needed increases
3. Diversity: Simulators are limited by the number of environments
4. Environments are expensive to create

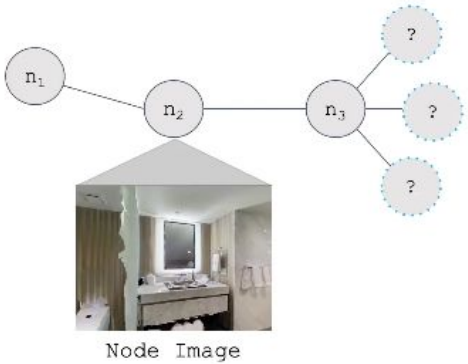
Source: https://meerahahn.github.io/nrns/

No RL, No Simulation: Learning to Navigate without Navigating (Hahn et al.)

Solution proposed: learn a navigation method directly from passive data trajectories of indoor environments, for the task of image goal navigation

RL simulation is not needed to do navigation, as it is a structured problem

- No need to do a credit assignment (use distance; predict distance to goal in the exploration frontier)
- No need to learn a policy (use greedy; selects direction which minimizes predicted distance to goal location)
- No need for interaction data (use depth to avoid local obstacles)



1. Predict Non-Collision Future Nodes
2. Predict Goal-Distance from Future Nodes
3. Be Greedy!

Source: https://meerahahn.github.io/nrns/

No RL, No Simulation: Learning to Navigate without Navigating (Hahn et al.)

Task: Image Goal Navigation

- An agent is placed in a novel environment with no map of the environment
- Agent is given an image from goal location with limited field of view, no coordinates
- The agent is tasked to navigating within 1m of the location where the image is taken within a set number of steps
- Agent builds and maintains a topological map of the area
  - Contains explored and unexplored nodes (explored nodes: exploration frontier)
- Uses the topological graph to decide where to go next to find the goal image
  - Selects the unexplored node that minimizes the agent's distance to goal image
  - Agent learns a GNN over the topological map which uses the visual features of each node to predict the distance to the goal image
- Agent use a heuristic depth-based policy to choose low level actions; uses the same policy to expand graph with the exploration frontier
- Then the agent predicts whether it should stop exploring, and predicts the goal location (use a CNN + MLP)

Source: https://meerahahn.github.io/nrns/

No RL, No Simulation: Learning to Navigate without Navigating (Hahn et al.)

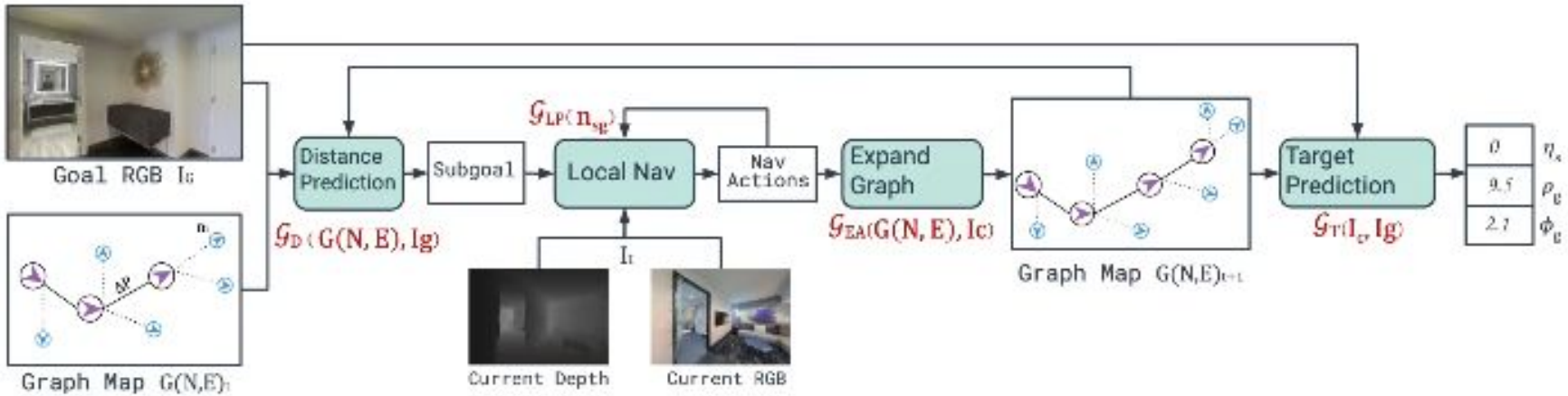
Task: Image Goal Navigation

- An agent is placed in a novel environment with no map of the environment
- Agent is given an image from goal location with limited field of view, no coordinates
- The agent is tasked to navigating within 1m of the location where the image is taken within a set number of steps
- Agent builds and maintains a topological map of the area
  - Contains explored and unexplored nodes (explored nodes: exploration frontier)
- Uses the topological graph to decide where to go next to find the goal image
  - Selects the unexplored node that minimizes the agent's distance to goal image
  - Agent learns a GNN over the topological map which uses the visual features of each node to predict the distance to the goal image **(distance prediction)**
- Agent use a heuristic depth-based policy to choose low level actions **(local policy)**; uses the same policy to expand graph with the exploration frontier **(graph expansion)**
- Then the agent predicts whether it should stop exploring, and predicts the goal location (use a CNN + MLP) **(target prediction)**

Source: https://meerahahn.github.io/nrns/

No RL, No Simulation: Learning to Navigate without Navigating (Hahn et al.)

Overview



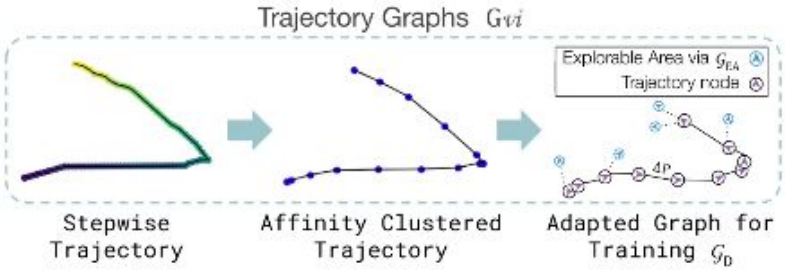
Source: https://meerahahn.github.io/nrns/

No RL, No Simulation: Learning to Navigate without Navigating (Hahn et al.)

Methodology

- Learn directly from rGBD videos of indoor trajectories to learn the distance prediction and target prediction function
- Use the habitat simulator to generate passive RGBD videos of trajectories
  - Generate ~1K videos per training environment
- Transfer the videos to trajectory graphs using affinity clustering over the pose and visual features of the stepwise trajectory (without use of actions)
  - Use the explorable area function to add unexplored nodes to video graph
    - Select multiple subgraphs for training the distance prediction function

RGBD Video (MP3D)



Source: https://meerahahn.github.io/nrns/

Playing hard exploration games by watching YouTube (Aytar et al.)

Motivation: one successful method of guiding exploration in these domains is to imitate trajectories provided by a human demonstrator, however these demonstrations are collected under artificial conditions (access to agent's exact environment setup and demonstrator's action and reward trajectories)

Approach:

1. Learn to map unaligned videos from multiple sources to a common representation using self-supervised objectives constructed over both time and modality (vision and sound)
2. Embed a single YouTube video in this representation to construct a reward function that encourages an agent to convincingly exceed human-level performance on difficult exploration games such as Montezuma's revenge, pitfall!, and private eye for the first time, even if the agent is not presented with any environment rewards

Playing hard exploration games by watching YouTube (Aytar et al.)

Problem: “hard exploration”, i.e. sparse environmental rewards, 100s of environment steps to even reach the first reward in Montezuma’s Revenge

Possible solutions

- Intrinsic motivation: create an auxiliary reward to encourage trying new trajectories (this doesn’t solve the problem of unknown-unknowns)
- Imitation learning: observe some demonstrations of others playing the game, then imitate their trajectories
  - Source: YouTube videos; humans can learn by watching somebody do something, regardless of significant differences in timing, lighting, background, sounds, etc.
  - Challenges: different colors, aspect ratios, location within the frame and artifacts such as the avatar, etc. (**domain gap**)
    - It is a problem since most methods expect clean demonstrations, as well as complete action-reward sequences from them

Source: https://students.mimuw.edu.pl/~jj385649/um-sem/Playing%20hard%20exploration%20games%20by%20watching%20YouTube.pdf

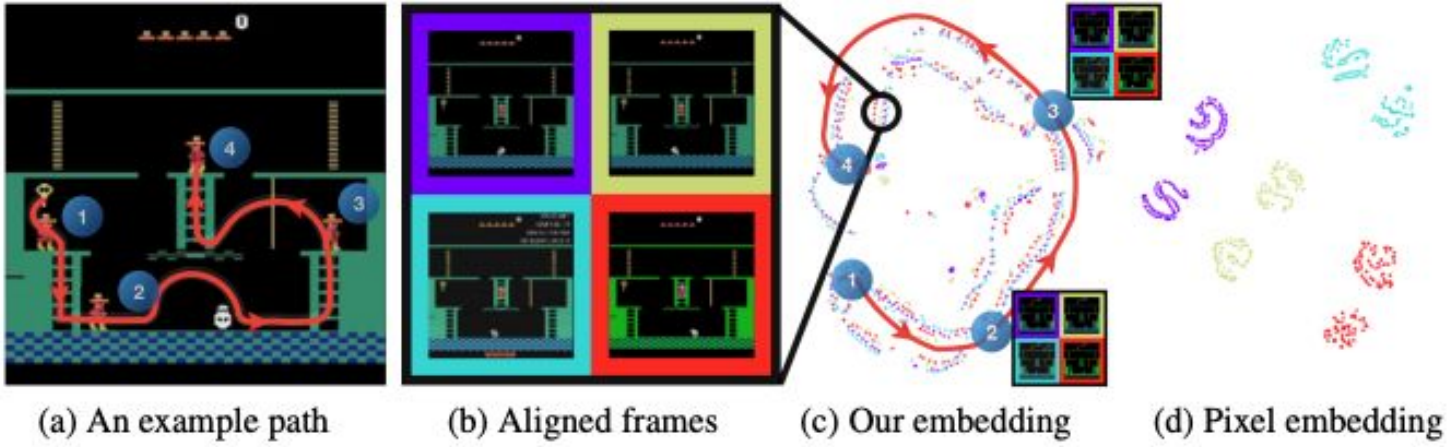
Playing hard exploration games by watching YouTube (Aytar et al.)

Approach

Deep reinforcement learning methods traditionally struggle with tasks where environment rewards are particularly sparse. One successful method of guiding exploration in these domains is to imitate trajectories provided by a human demonstrator. However, these demonstrations are typically collected under artificial conditions, i.e. with access to the agent's exact environment setup and the demonstrator's action and reward trajectories. Here we propose a two-stage method that overcomes these limitations by relying on noisy, unaligned footage without access to such data. First, we learn to map unaligned videos from multiple sources to a common representation using self-supervised objectives constructed over both time and modality (i.e. vision and sound). Second, we embed a single YouTube video in this representation to construct a reward function that encourages an agent to imitate human gameplay. This method of one-shot imitation allows our agent to convincingly exceed human-level performance on the infamously hard exploration games MONTEZUMA'S REVENGE, PITFALL! and PRIVATE EYE for the first time, even if the agent is not presented with any environment rewards.

Source: https://students.mimuw.edu.pl/~jj385649/um-sem/Playing%20hard%20exploration%20games%20by%20watching%20YouTube.pdf

Playing hard exploration games by watching YouTube (Aytar et al.)

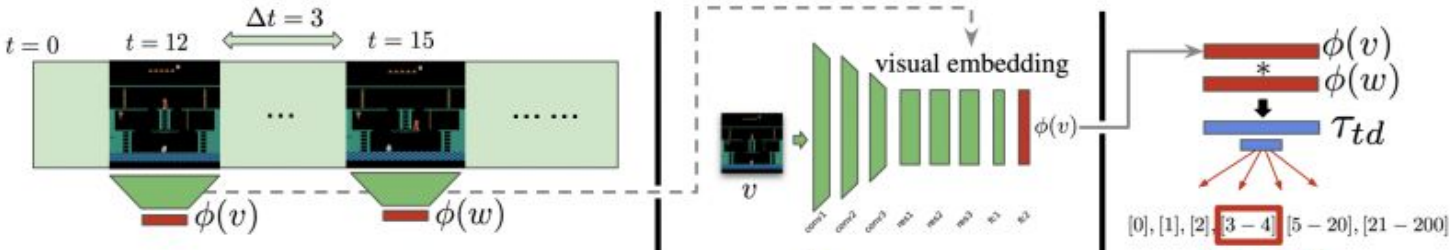


Goal: using three different training videos per game, produce a common representation for them.

They train an embedder to solve an auxiliary task which is self-supervised and encourages a desirable embedding, which is to predict the temporal distance between two frames from the same demonstration using visual-visual embedding and visual-audio embedding.

Source: https://students.mimuw.edu.pl/~jj385649/um-sem/Playing%20hard%20exploration%20games%20by%20watching%20YouTube.pdf

Playing hard exploration games by watching YouTube (Aytar et al.)

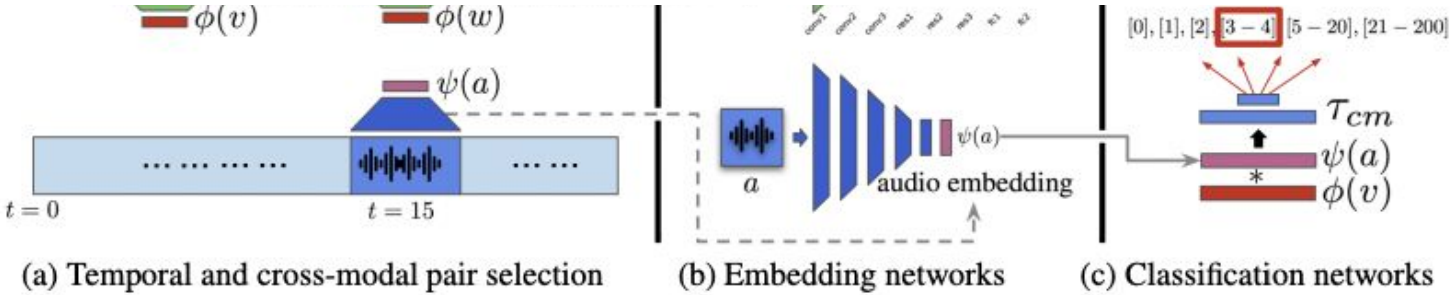


Temporal distance classification (TDC): look at the embeddings of two frames from one demonstration, determine the number of steps between them

Source: https://students.mimuw.edu.pl/~jj385649/um-sem/Playing%20hard%20exploration%20games%20by%20watching%20YouTube.pdf

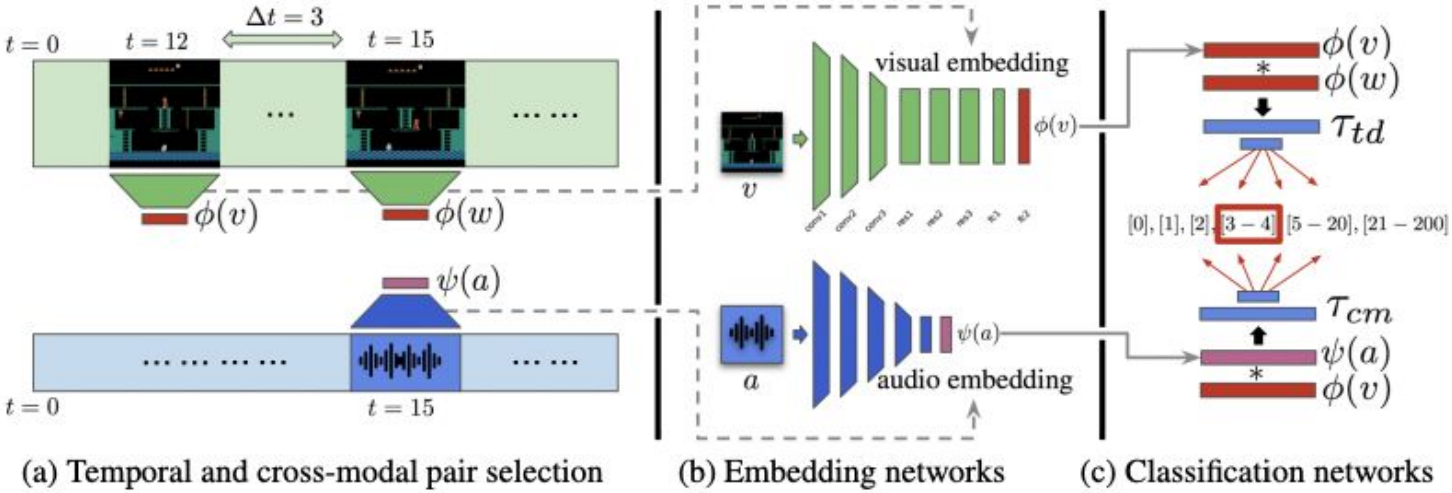
Playing hard exploration games by watching YouTube (Aytar et al.)

Cross-modal classification (CMC): looking at an embedded frame and a sound snippet, determine the time between them



Source: https://students.mimuw.edu.pl/~jj385649/um-sem/Playing%20hard%20exploration%20games%20by%20watching%20YouTube.pdf

Playing hard exploration games by watching YouTube (Aytar et al.)



Train to minimize the weighted sum of cross-entropies.

Source: https://students.mimuw.edu.pl/~jj385649/um-sem/Playing%20hard%20exploration%20games%20by%20watching%20YouTube.pdf

Playing hard exploration games by watching YouTube (Aytar et al.)

Approach

Deep reinforcement learning methods traditionally struggle with tasks where environment rewards are particularly sparse. One successful method of guiding exploration in these domains is to imitate trajectories provided by a human demonstrator. However, these demonstrations are typically collected under artificial conditions, i.e. with access to the agent's exact environment setup and the demonstrator's action and reward trajectories. Here we propose a two-stage method that overcomes these limitations by relying on noisy, unaligned footage without access to such data. First, we learn to map unaligned videos from multiple sources to a common representation using self-supervised objectives constructed over both time and modality (i.e. vision and sound). Second, we embed a single YouTube video in this representation to construct a reward function that encourages an agent to imitate human gameplay. This method of one-shot imitation allows our agent to convincingly exceed human-level performance on the infamously hard exploration games MONTEZUMA'S REVENGE, PITFALL! and PRIVATE EYE for the first time, even if the agent is not presented with any environment rewards.

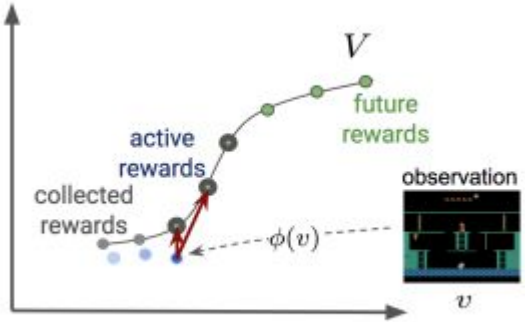
Source: https://students.mimuw.edu.pl/~jj385649/um-sem/Playing%20hard%20exploration%20games%20by%20watching%20YouTube.pdf

Playing hard exploration games by watching YouTube (Aytar et al.)

Dataset for embedder: three training videos per game, where one pair of training frames is obtained by sampling one of three videos, sampling a time interval, and randomly selecting two frames separated by that interval

One-shot imitation

- Combine a standard RL agent, the trained embedder, another YouTube video
- Goal: imitate the video
- Every 16 frames make a checkpoint, add an auxiliary reward for visiting checkpoints in the right order

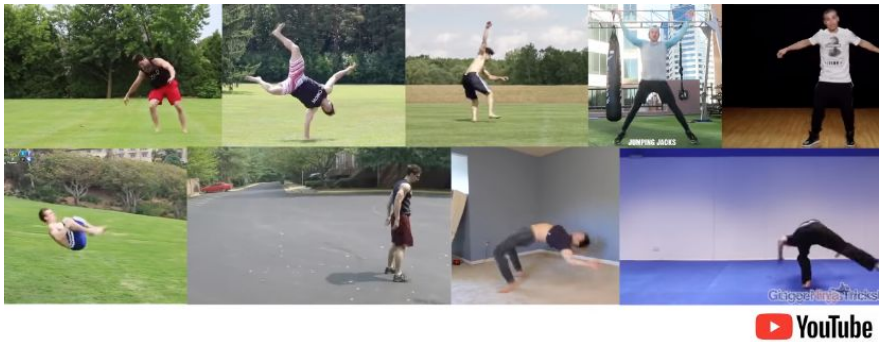


(b) One shot imitation

Source: https://students.mimuw.edu.pl/~jj385649/um-sem/Playing%20hard%20exploration%20games%20by%20watching%20YouTube.pdf

SFV Reinforcement Learning of Physical Skills from Videos (Peng et al.)

Motivation: training simulated characters to imitate mocap data can be highly effective for producing natural motions, but mocap data can be difficult to acquire and often requiring heavy instrumentation. Video clips offer a much more accessible and abundant source of data!

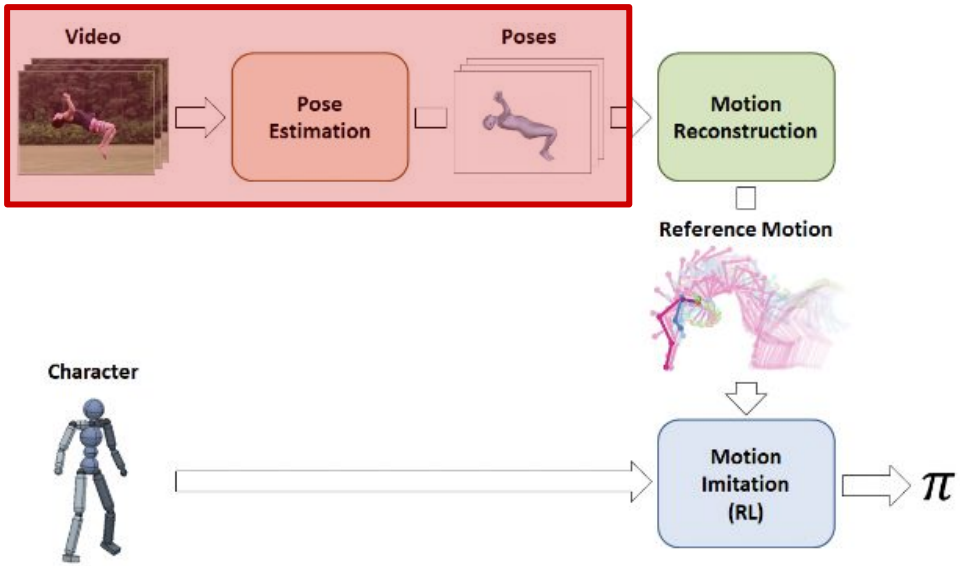


Source: https://www.youtube.com/watch?v=4Qg5I5vhX7Q&t=0s

SFV Reinforcement Learning of Physical Skills from Videos (Peng et al.)

Contribution: a framework that enables simulated characters to learn skills directly from video, which consists of three stages:

1. **Pose estimation:** given input video, predict the pose of the actor in each frame

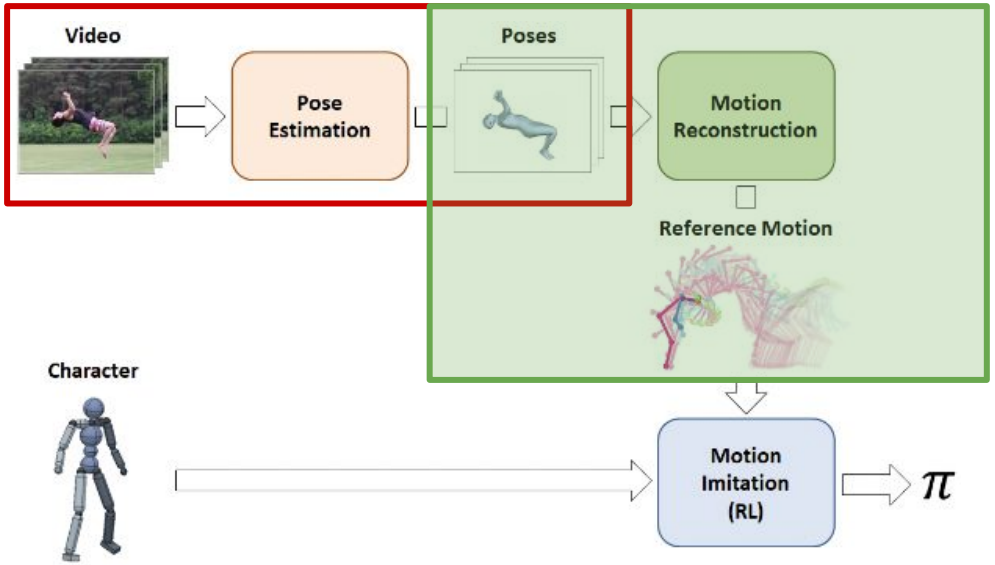


Source: https://bair.berkeley.edu/blog/2018/10/09/sfv/

SFV Reinforcement Learning of Physical Skills from Videos (Peng et al.)

Contribution: a framework that enables simulated characters to learn skills directly from video, which consists of three stages:

1. **Pose estimation:** given input video, predict the pose of the actor in each frame
2. **Motion reconstruction:** consolidated the pose predictions into a reference motion and fixes artifacts that might have been introduced by the pose predictions

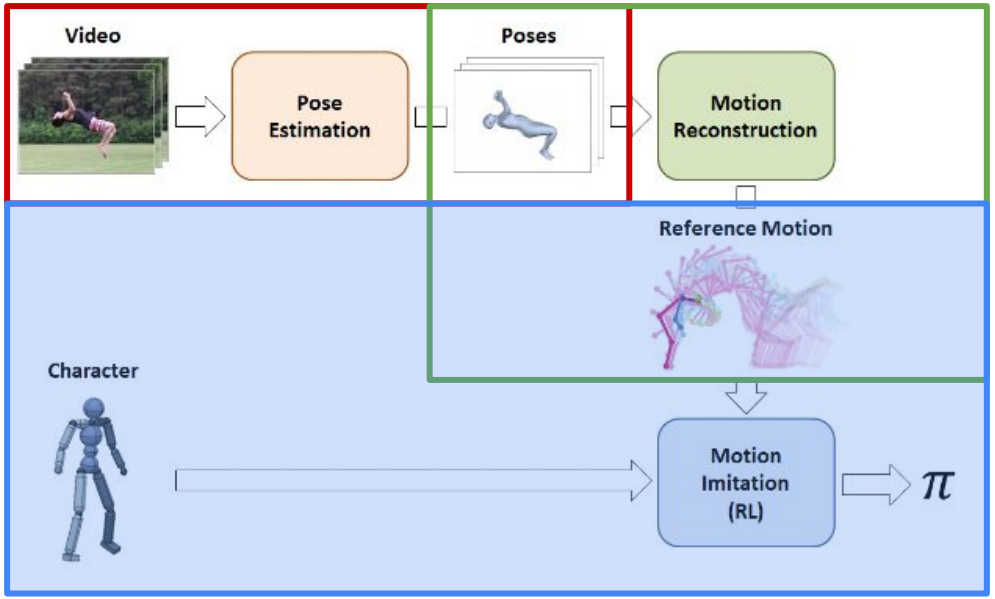


Source: https://bair.berkeley.edu/blog/2018/10/09/sfv/

SFV Reinforcement Learning of Physical Skills from Videos (Peng et al.)

Contribution: a framework that enables simulated characters to learn skills directly from video, which consists of three stages:

1. **Pose estimation:** given input video, predict the pose of the actor in each frame
2. **Motion reconstruction:** consolidated the pose predictions into a reference motion and fixes artifacts that might have been introduced by the pose predictions
3. **Motion imitation:** reference motion is passed here, where a simulated character is trained to imitate the motion using reinforcement learning



Source: https://bair.berkeley.edu/blog/2018/10/09/sfv/

SFV Reinforcement Learning of Physical Skills from Videos (Peng et al.)

Pose Estimation

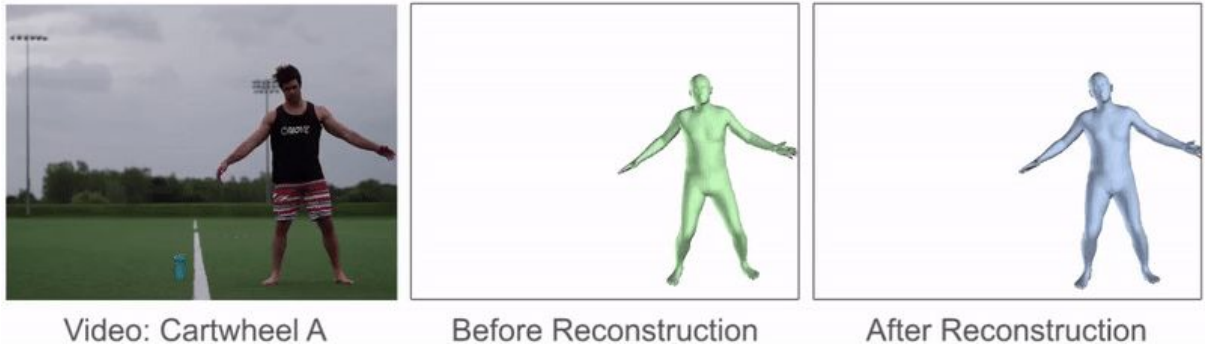
Video: Backflip A | 2D Estimator | 3D Estimator

- An ensemble of 2D and 3D pose estimators is used to predict the actor's pose in each frame
- Training the pose estimators with rotation augmentation substantially improves accuracy for acrobatic motions

Source: https://www.youtube.com/watch?v=4Qg5I5vhX7Q&t=0s

SFV Reinforcement Learning of Physical Skills from Videos (Peng et al.)

Motion Reconstruction



- Since the poses are independently predicted for each frame, they may not be temporally consistent
- Reconstruct a temporally smooth pose trajectory that consolidates the 2D and 3D pose predictions

Source: https://www.youtube.com/watch?v=4Qg5I5vhX7Q&t=0s

SFV Reinforcement Learning of Physical Skills from Videos (Peng et al.)

Motion Imitation via RL

- Each policy is modeled with a feedforward network and trained with RL to imitate the reference motion
  - Reward function encourages the policy to minimize the difference between the pose of the simulated character and the pose of the reference motion at each frame
- The simulation can cleanup non-physical behaviors in the reference motion (using tracking errors)
- This approach works well, and the characters are able to learn a diverse repertoire of challenging acrobatic skills, where each skill is learned from a single video demonstration

Cartwheel A

Frontflip

Source: https://bair.berkeley.edu/blog/2018/10/09/sfv/

Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos (Baker et al.)

Motivation: aims to produce behavioral priors from commonly available but unlabeled area, in domains where an agent needs to act but reward signal is so sparse that RL is virtually impossible

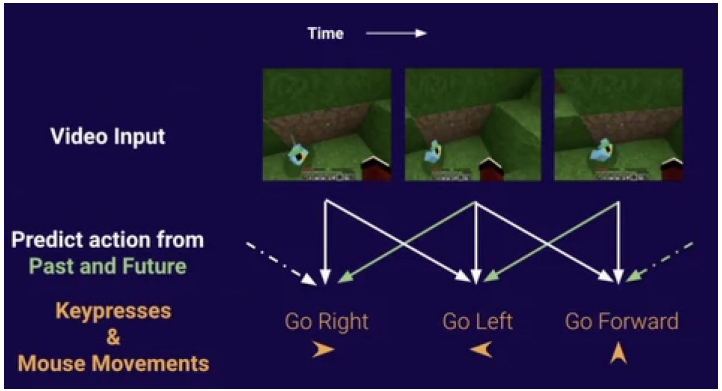
- VPT follows powerful GPT playbook: pretrain on large, noisy, internet-scale datasets, then use the resulting model for downstream task (such as computer-using agents and video games where there are a lot of videos online but are unlabeled)

Source: https://slideslive.com/38991776/video-pretraining-vpt-learning-to-act-by-watching-unlabeled-online-videos?ref=speaker-17019

Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos (Baker et al.)

What to do with unlabeled demonstrations?

- Action labels associated with each video frame is unavailable
- Use inverse-dynamic models to label: given a sequence of video frames, IDM predicts actions by looking at past and future frames (BC: looks at past frames only, infer player intent for future frames)

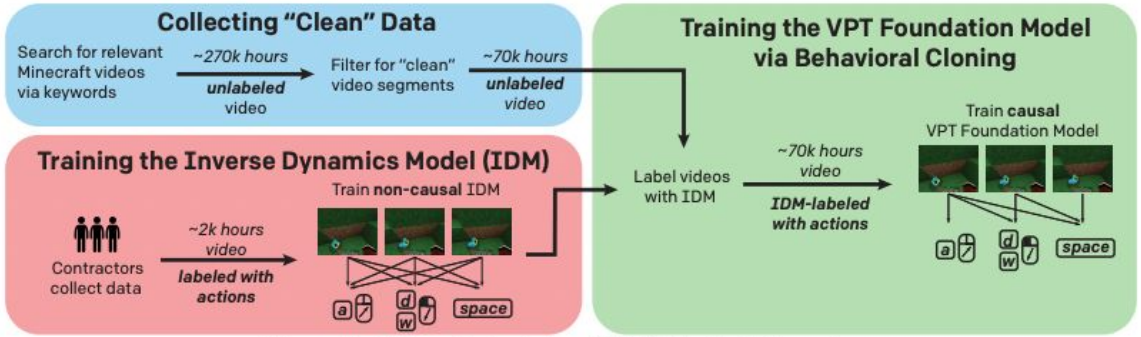


Source: https://slideslive.com/38991776/video-pretraining-vpt-learning-to-act-by-watching-unlabeled-online-videos?ref=speaker-17019

Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos (Baker et al.)

VPT pipeline

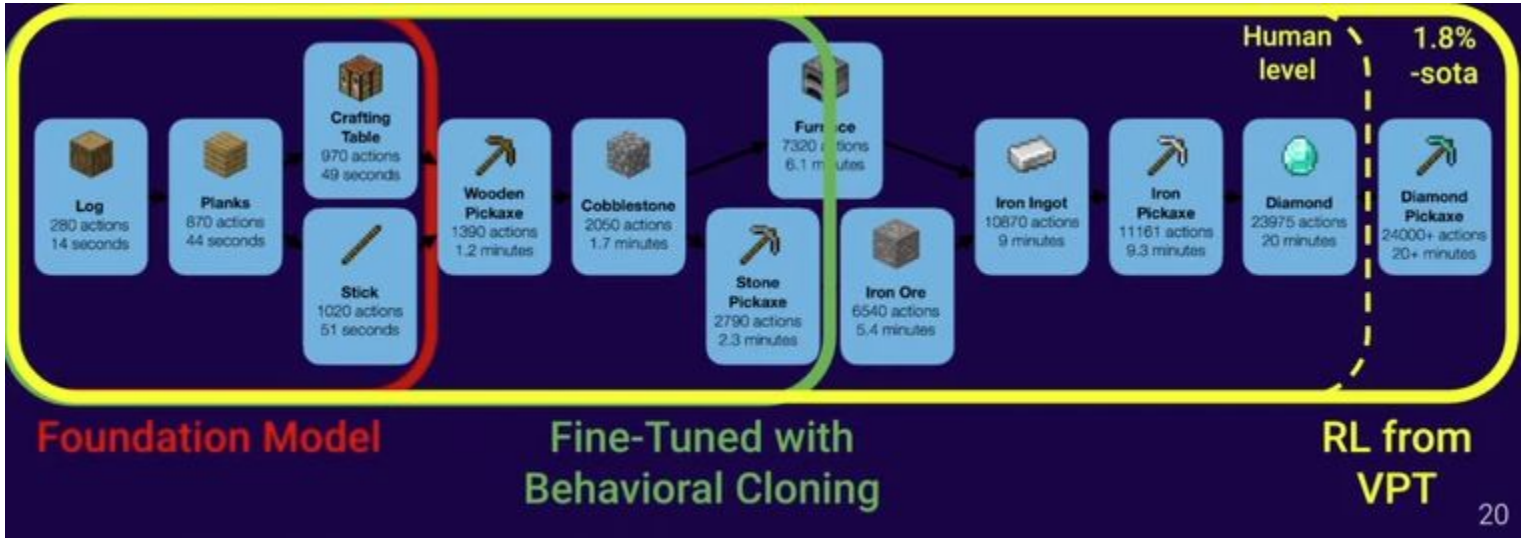
- Search internet for minecraft videos, filter for clean video segments for mods, overlays, and other noise to **collect clean data**
- In parallel, hire contractors to play minecraft and record 2000 hours labeled trajectories, and **train IDM** on these trajectories
- Label clean videos using the IDM and **train the model via behavioral cloning**



Source: https://slideslive.com/38991776/video-pretraining-vpt-learning-to-act-by-watching-unlabeled-online-videos?ref=speaker-17019

Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos (Baker et al.)

Fine-tuning for specific tasks with reinforcement learning, e.g. collecting a diamond pickaxe



Source: https://slideslive.com/38991776/video-pretraining-vpt-learning-to-act-by-watching-unlabeled-online-videos?ref=speaker-17019

Language and Robot Control

Learning Transferable Visual Models From Natural Language Supervision (Radford et al.) a.k.a the CLIP paper

Motivation: GPT-3 shows that models pre-trained on high web-scale collections of text surpassed high-quality crowd-labeled NLP datasets, so **could scalable pre-training methods which learn directly from web text result in a similar breakthrough in computer vision?**

Approach: learning perception from supervision contained in natural language

Strengths of learning from natural language:

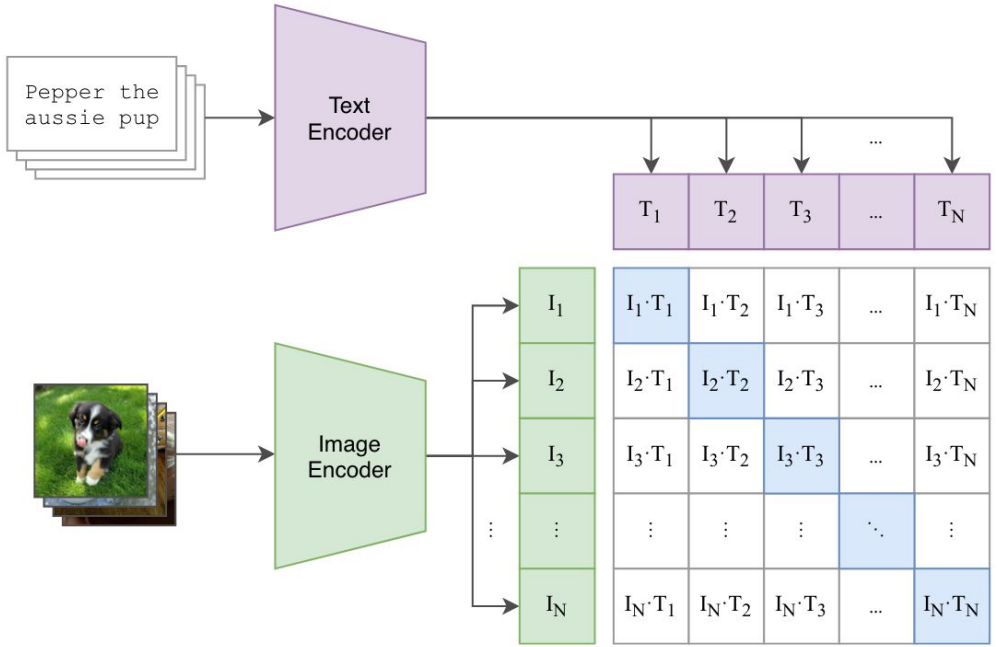
- Easier to scale natural language supervision compared to standard crowdsourced labeling for image classification
- Doesn't just learn a representation but also connects that representation to language which enables flexible zero-shot transfer

Source: https://amaarora.github.io/posts/2023-03-06_Understanding_CLIP.html

Learning Transferable Visual Models From Natural Language Supervision (Radford et al.) a.k.a the CLIP paper

CLIP architecture

1. Pass images through image encoder and texts through text encoder to get image and text features
2. Have two projection layers for the text and image features to project them to the embedding dimension
   a. Get the joint multimodal embeddings by doing a dot product
3. Get the cosine similarity between the joint embeddings
4. Use contrastive loss during training (we want the cross-entropy loss of the items in the diagonal to be high and low elsewhere)



Source: https://amaarora.github.io/posts/2023-03-06_Understanding_CLIP.html

CLIPort: What and Where Pathways for Robotic Manipulation (Shridhar et al.)

Figure 1. **Language-Conditioned Manipulation Tasks:** CLIPORT is a broad framework applicable to a wide range of language-conditioned manipulation tasks in tabletop settings. We conduct large-scale experiments in Ravens [2] on 10 simulated tasks (a-j) with 1000s of unique instances per task. See Appendix A for challenges pertaining to each task. CLIPORT can even learn one multi-task model for all 10 tasks that achieves better or comparable performance to single-task models. Similarly, we demonstrate our approach on a Franka Panda manipulator with one multi-task model for 9 real-world tasks (k-o; only 5 shown) trained with just 179 image-action pairs.

Source: https://cliport.github.io/

CLIPort: What and Where Pathways for Robotic Manipulation (Shridhar et al.)

Motivation

1. End-to-end networks can learn dexterous skills that require precise spatial reasoning, but methods fail to generalize to new goals or quickie learn transferable concepts across tasks
2. Great progress in learning generalizable semantic representations for vision and language by training on large-scale internet data, but they lack spatial understanding

CLIPort is a two-stream architecture with semantic and spatial pathways for vision-based manipulation; a language-conditioned imitation-learning agent that combines the broad semantic understanding of CLIP with the spatial precision of TransporterNets.

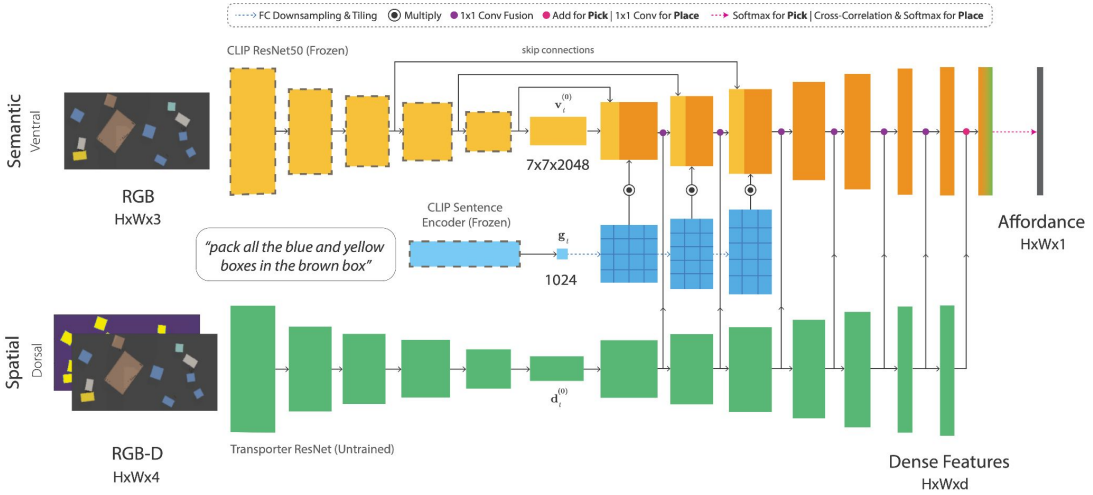
Outcome: solve variety of language-specified tabletop tasks without any explicit representations of object poses, instance segmentations (e.g. “fold the cloth in half”, what is *fold*?)

Source: https://cliport.github.io/

CLIPort: What and Where Pathways for Robotic Manipulation (Shridhar et al.)

Two-Stream Architecture

- The **semantic stream** uses a pre-trained CLIP model to encode RGB and language-goal input
  - CLIP is trained with large amounts of image-caption pairs from the internet: acts as a powerful semantic prior for grounding visual concepts such as colors, shapes, parts, texts, and object categories
- The **spatial stream** is a tabula rasa fully-convolutional network that encodes RGB-D (depth) input



Source: https://cliport.github.io/

CLIPort: What and Where Pathways for Robotic Manipulation (Shridhar et al.)

Transporter Networks (https://transporternets.github.io/)

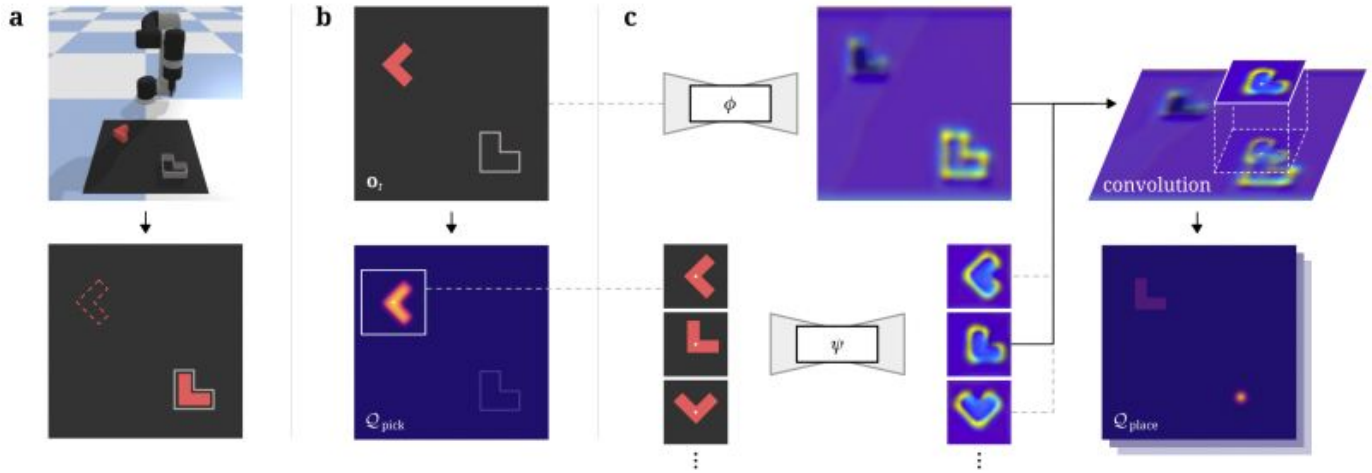
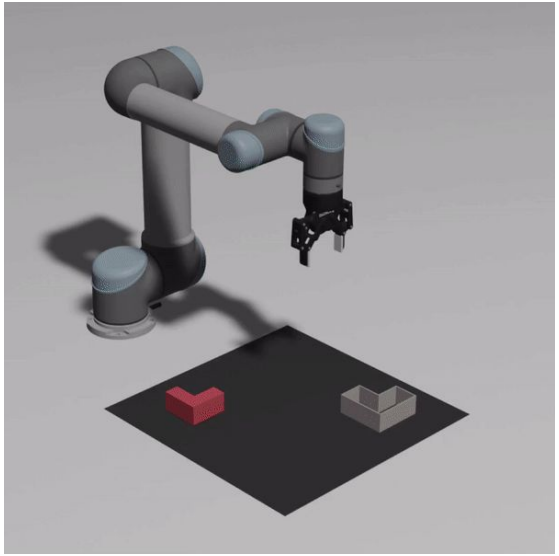


Figure 3. In this setting (a) where the task is to pick up the red block with an immobilizing grasp (e.g., suction) and place it into the fixture, the goal of Transporter Networks is to recover the distribution of successful picks (b), and distribution of successful placements (c) conditioned on a sampled pick. For pick-conditioned placing (c), deep feature template matching occurs with a local crop around the sampled pick as the exemplar. Rotations of the crop around the pick are used to decode the best placing rotation. Our method preserves rotation and translation equivariance for efficient learning.

Source: https://cliport.github.io/

CLIPort: What and Where Pathways for Robotic Manipulation (Shridhar et al.)



The two-stream architecture is used in all three networks of Transporter Networks to predict and pick and place affordances at each timestamp.

The TransporterNet first attends to a local region to decide where to pick, then computes a placement location by finding the best match for the picked region through cross-correlation of deep visual features.

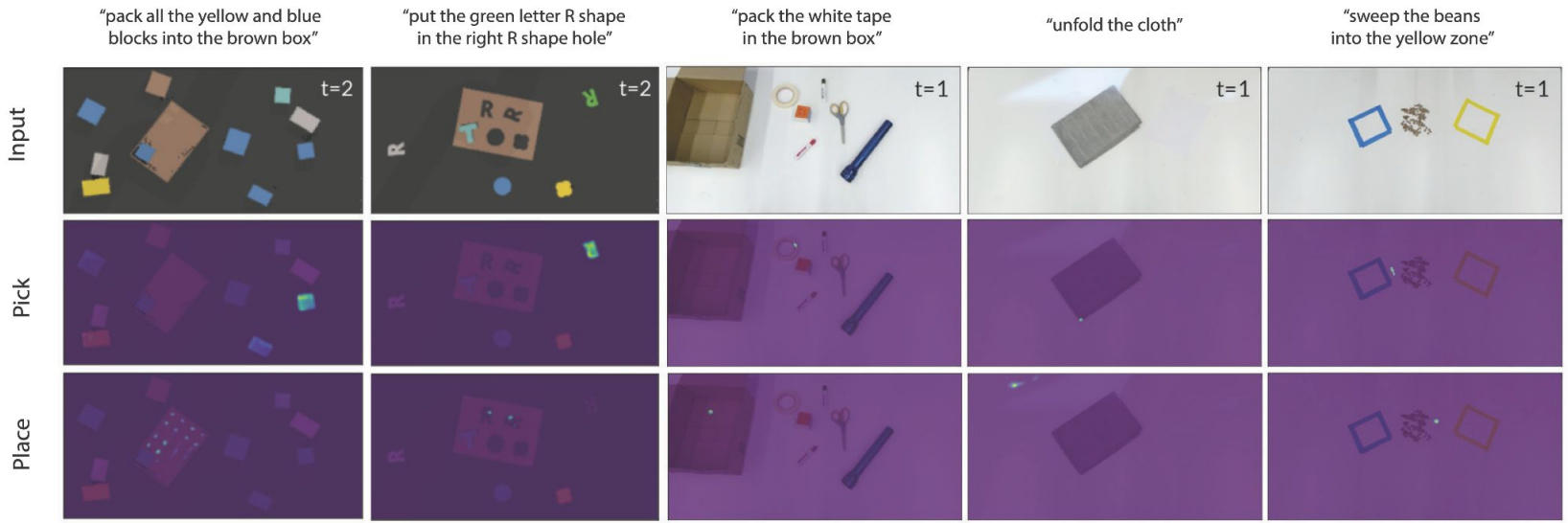
- This structure serves as a powerful inductive bias for learning roto-translationally equivariant representations in tabletop environments.

Source: https://cliport.github.io/

CLIPort: What and Where Pathways for Robotic Manipulation (Shridhar et al.)

Affordance Predictions

Examples of pick and place affordance predictions from multi-task CLIPort models



Source: https://cliport.github.io/

Language Models are Few-Shot Learners (Brown et al.)

a.k.a. The GPT-3 paper

Architecture: A decoder-only transformer model (similar to its predecessor GPT-2), with alternating dense and locally banded sparse attention patterns used in the layers of the Transformer (similar to the SparseTransformer)

Dataset: CommonCrawl (410B)++

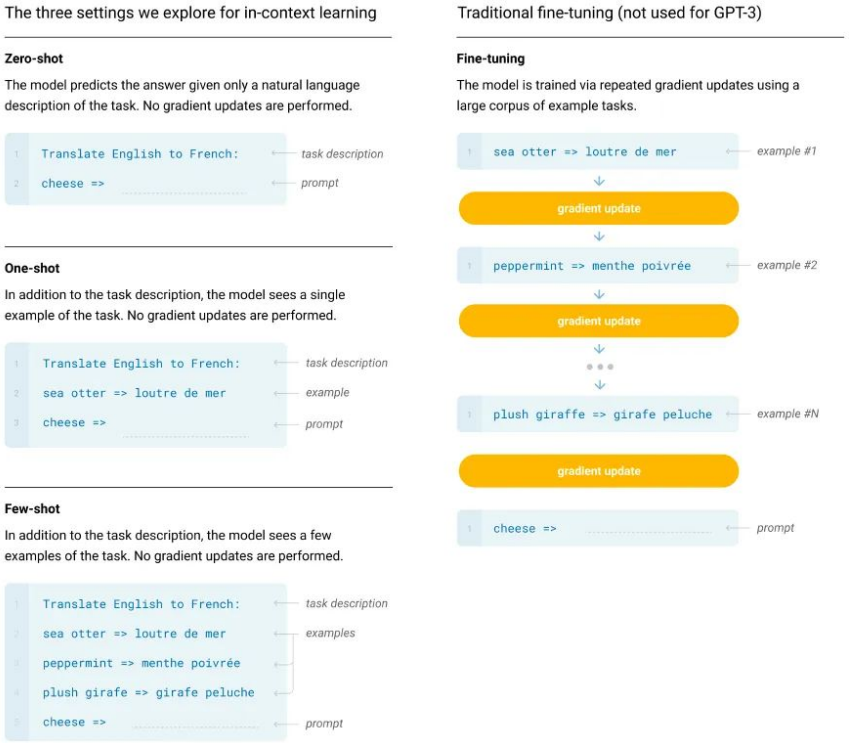
- CommonCrawl data is downloaded and filtered based on similarity to a range of high-quality reference corpora
- Fuzzy deduplication is performed at the document level
- Known high-quality reference corpora are added to the training mix to augment CommonCrawl and increase its diversity: WebText2 (19B), Books1 (12B), Books2 (55B), English Wikipedia (3B)

Pretraining task: (G)enerative (PT) is a technique that involves training a language model on a large corpus of text data in an unsupervised manner (self-supervised), where the primary goal is to generate text that closely resembles human-written text by predicting the next word in a given sequence.

Source: https://sh-tsang.medium.com/review-gpt-3-language-models-are-few-shot-learners-ff3e63da944d

Language Models are Few-Shot Learners (Brown et al.)

a.k.a. The GPT-3 paper

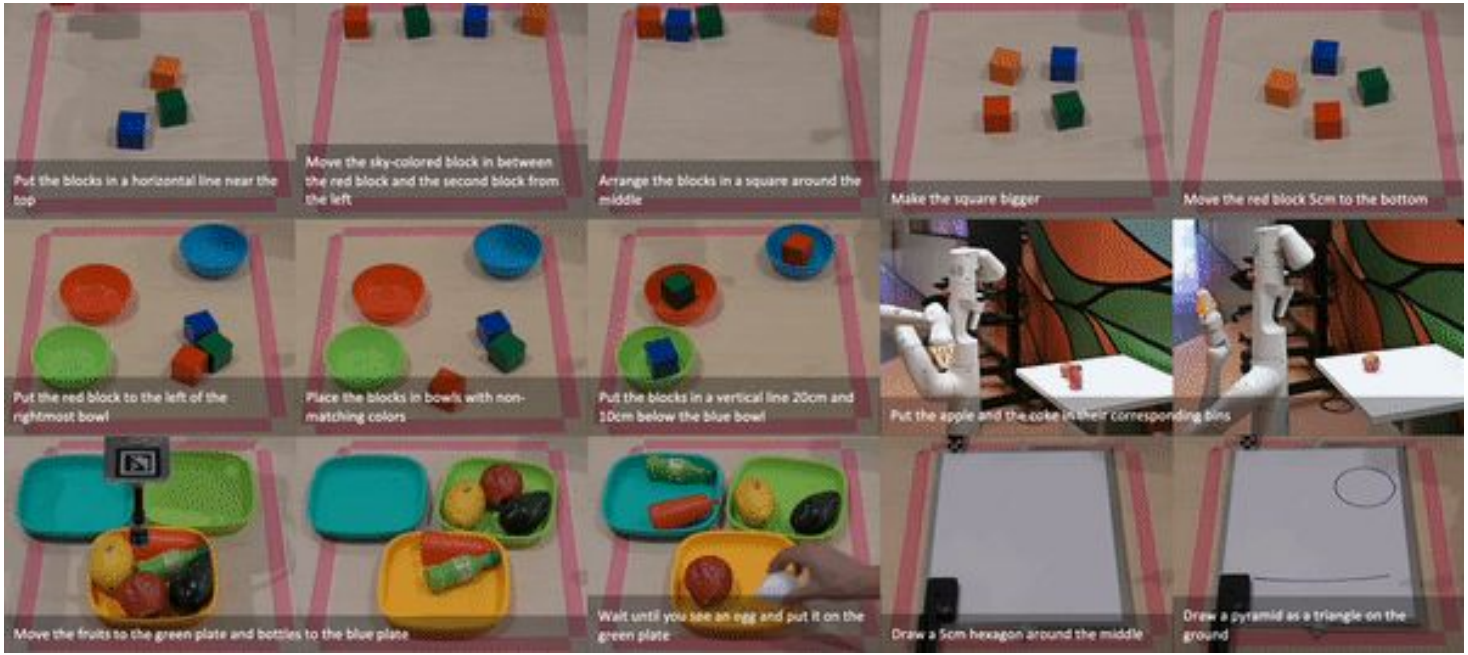


The GPT-3 model is **task-agnostic**: can perform tasks with very few (one- or few-shot) or no examples (zero-shot)

Source: https://sh-tsang.medium.com/review-gpt-3-language-models-are-few-shot-learners-ff3e63da944d

Code as Policies: Language Model Programs for Embodied Control (Liang et al.)

How can robots perform a wide-variety of tasks specified by language?



Source: https://code-as-policies.github.io/

Code as Policies: Language Model Programs for Embodied Control (Liang et al.)

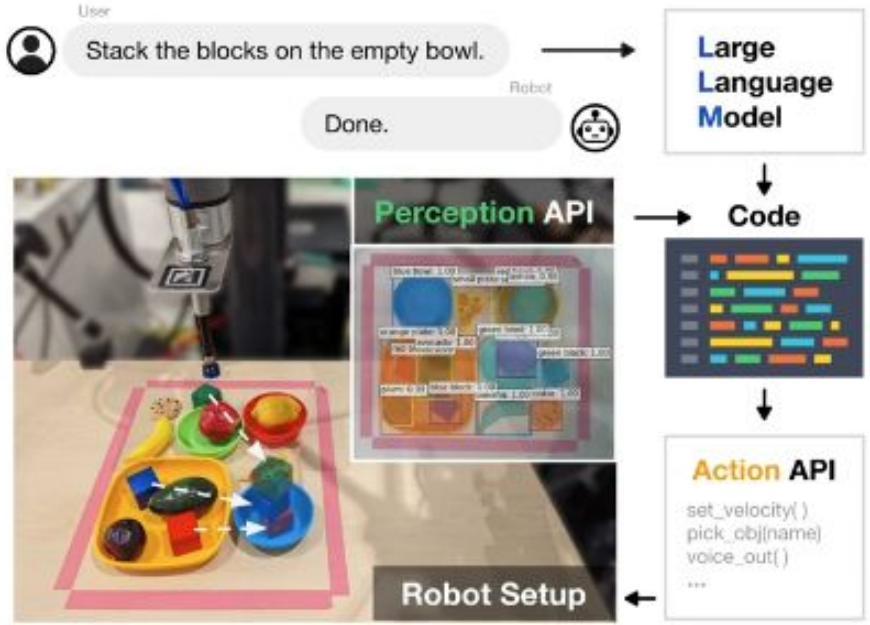
Motivation: recent works have shown success with using large language models (LLMs) to plan robot tasks, where the input is a natural language description of the task, and the output is a natural language description of a sequence of skills the robot needs to execute in order to complete the task

Research question: can robots leverage LLMs beyond high-level planning to also perform low-level reasoning and control?

Approach: use a code-writing language model that, when prompted with hints (i.e. import statements that inform which APIs are available) and examples (instruction-to-code pairs that present few-shot ‘demonstrations’ on how instructions should be converted into code), writes new code for new instructions, i.e. hierarchical code generation.

Source: https://code-as-policies.github.io/ , https://blog.research.google/2022/11/robots-that-write-their-own-code.html

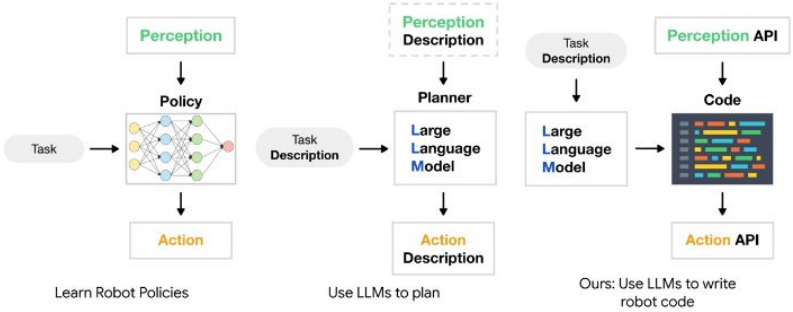
Code as Policies: Language Model Programs for Embodied Control (Liang et al.)



Proposal: Language Model Programs to Generate Code as Policies

Benefits

- Expressive inputs and outputs
- Solve tasks with few-shot prompting and zero-shot training
- Improved generalization to unseen tasks



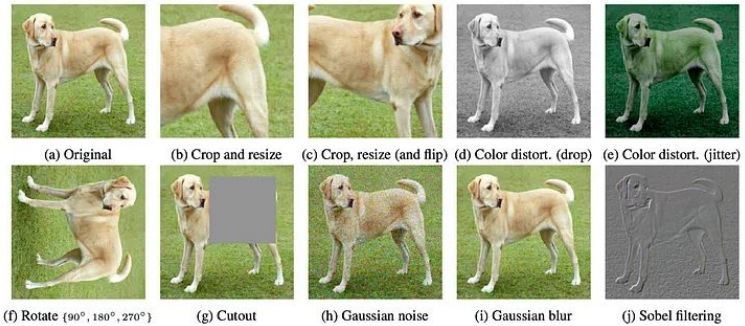
Source: https://code-as-policies.github.io/

Self-Supervised Visual Learning

A Simple Framework for Contrastive Learning of Visual Representations (Chen et al.)

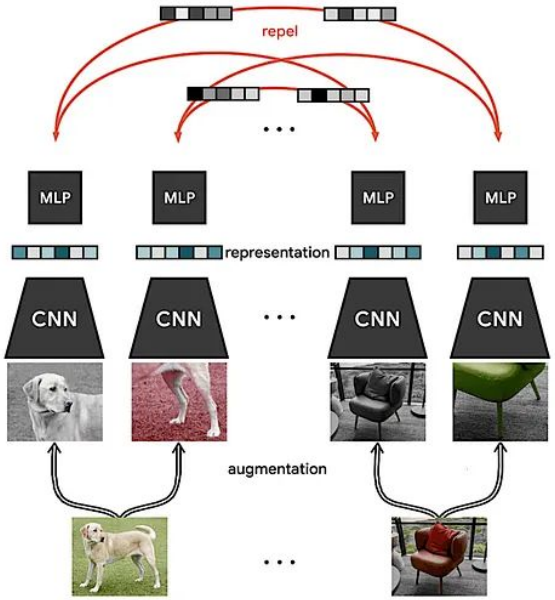
Self-supervised learning: model learns to supervise itself; we augment a single datapoint and let the model learn that these data points contain the same information, thus leading the model to learn a similar latent representation for the same objects

- This leads to the model learning a similar latent representation (an output vector) for the same objects
- Data augmentation in SimCLR: create pairs of images to learn the similarity from by applying augmentations or transformations to a single datapoint



Source: https://towardsdatascience.com/paper-explained-a-simple-framework-for-contrastive-learning-of-visual-representations-6a2a63bfa703

A Simple Framework for Contrastive Learning of Visual Representations (Chen et al.)



Learning image similarity with SimCLR

1. After augmenting the image, you have a positive pair (both augmented images contains the same object)
2. Pass the pair to a CNN to create a feature representation for each image (ResNet was used)
3. Output is passed to a projection head for further processing (MLP with one hidden layer; only used during training and refining feature representation of input images)
4. Learning goal: maximize agreement between different augmentations of the same image
    a. **Contrastive learning:** minimize the distance between images that contain the same object and maximize the distance between images that contain vastly different objects
    b. NT-Xent loss: normalized temperature-scaled cross entropy loss; different examples are weighted effectively allowing the model to learn much more effectively from vector representations that are far away from each other even though their origin is the same image (hard negatives); achieves an attraction of similar images

Source: https://towardsdatascience.com/paper-explained-a-simple-framework-for-contrastive-learning-of-visual-representations-6a2a63bfa703

A Simple Framework for Contrastive Learning of Visual Representations (Chen et al.)

Goal: learn a good latent representation for some images

Process

- x_i and x_j are augmented images, if they are the same we want to maximize similarity, if not we want to minimize similarity
- Pass them through the f network to get our intermediate representation and then pass the results through the g network so we can compare them

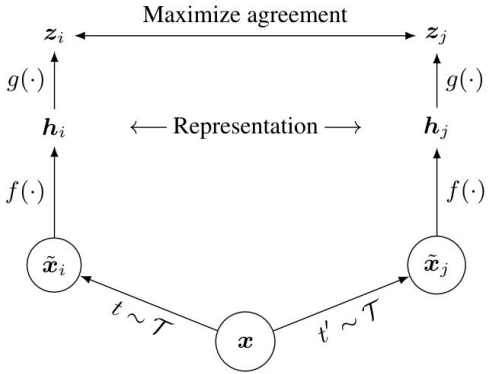
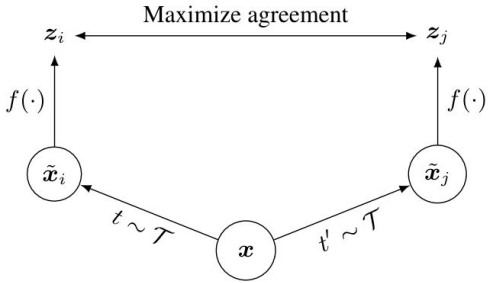


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation $\boldsymbol{h}$ for downstream tasks.

Source: https://towardsdatascience.com/paper-explained-a-simple-framework-for-contrastive-learning-of-visual-representations-6a2a63bfa703

A Simple Framework for Contrastive Learning of Visual Representations (Chen et al.)

Why are we projecting for what seems like no reason?

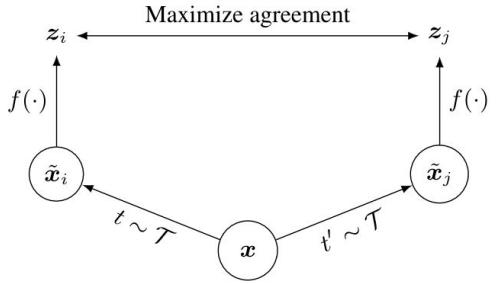


Source: https://towardsdatascience.com/paper-explained-a-simple-framework-for-contrastive-learning-of-visual-representations-6a2a63bfa703

A Simple Framework for Contrastive Learning of Visual Representations (Chen et al.)

Why are we projecting for what seems like no reason?

> “We conjecture that the importance of using the representation before the nonlinear projection is due to loss of information induced by the contrastive loss.”
>
> “To verify this hypothesis, we conduct experiments that use either h or g(h) to learn to predict the transformation applied during the pretraining.”



What to predict?	Random guess	Representation h	Representation $g(h)$
Color vs grayscale	80	99.3	97.4
Rotation	25	67.6	25.6
Orig. vs corrupted	50	99.5	59.6
Orig. vs Sobel filtered	50	96.6	56.3

Table 3. Accuracy of training additional MLPs on different representations to predict the transformation applied. Other than crop and color augmentation, we additionally and independently add rotation (one of $\{0°, 90°, 180°, 270°\}$), Gaussian noise, and Sobel filtering transformation during the pretraining for the last three rows. Both $\boldsymbol{h}$ and $g(\boldsymbol{h})$ are of the same dimensionality, i.e. 2048.

Source: https://towardsdatascience.com/paper-explained-a-simple-framework-for-contrastive-learning-of-visual-representations-6a2a63bfa703

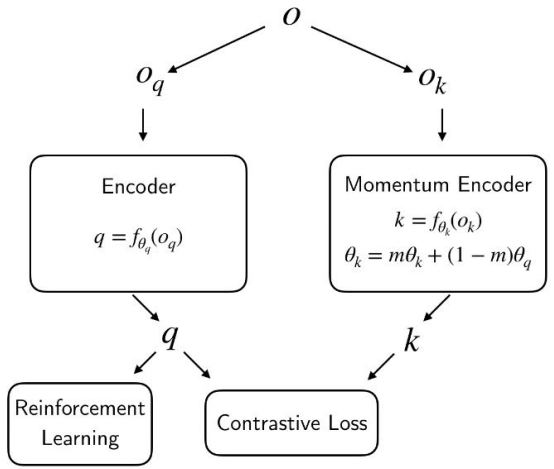
CURL: Contrastive Unsupervised Representations for Reinforcement Learning (Srinivas et al.)

Motivation: RL is becoming limited by the amount of data that is present in simulated words

Proposed solution: improve the efficiency of RL techniques that operate in extremely high dimensional spaces, ultimately allowing RL methods to simulate a more realistic world

Methodology

- Collect transitions and store in a replay buffer (e.g. videos used to train RL algorithm to play a game)
- For each transition, do data augmentation to produce a key and query
- Encode key and query through 2 separate encoders which feed it to an unsupervised algorithm, while only key encoder feeds them to the RL algorithm
- Use multitask learning to structure the loss function and learn mapping from high dimensional stack of image frames into a lower-dimensional representation



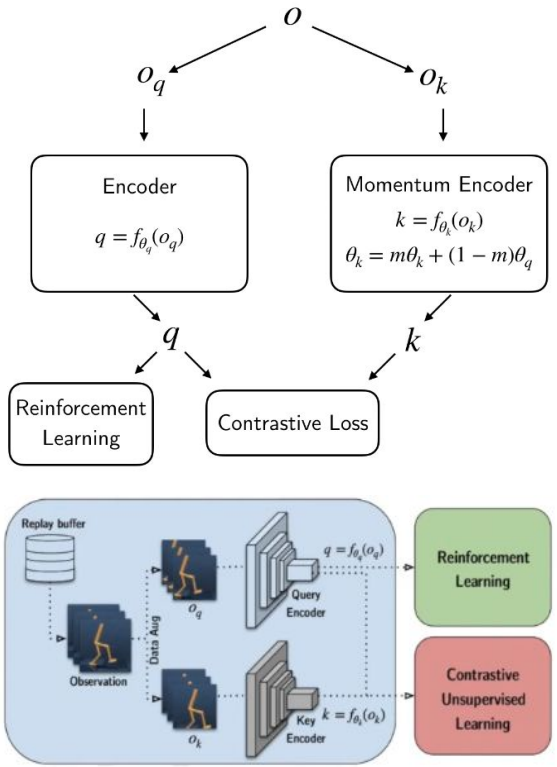
Source: https://www.youtube.com/watch?v=-Drowt9r4zY , https://towardsdatascience.com/openai-curl-reinforcement-learning-meets-unsupervised-learning-b038897daa30

CURL: Contrastive Unsupervised Representations for Reinforcement Learning (Srinivas et al.)

Same idea as SimCLR, of using contrastive loss on augmented data.

But this we use a stack of sequential frames from our environment and we are learning latent representations to increase sample efficiency.

We are also using an exponentially weighted encoder instead of two of the same encoder, similar to how some RL calculates its target.

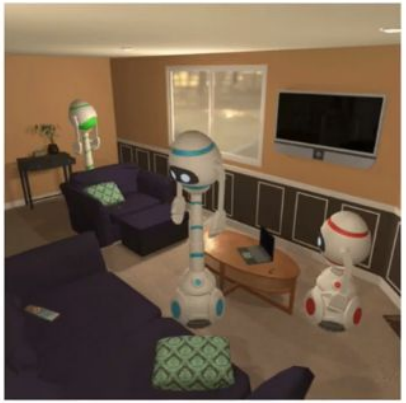


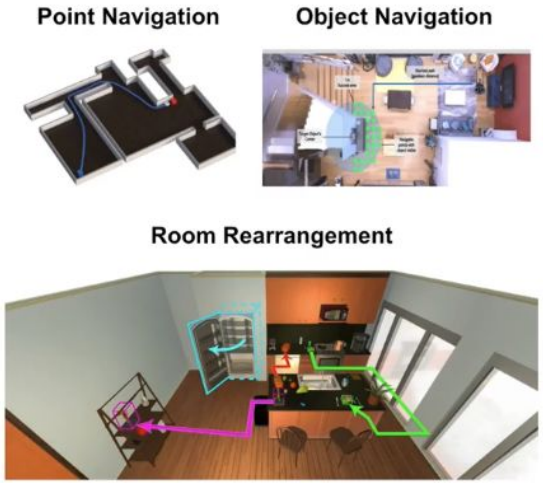
Source: https://www.youtube.com/watch?v=-Drowt9r4zY

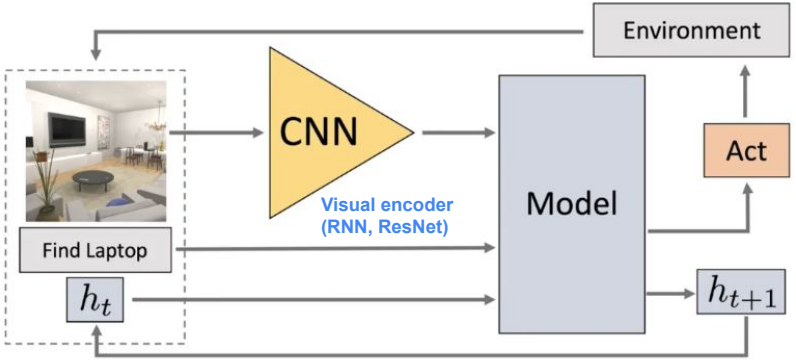
Simple but Effective: CLIP Embeddings for Embodied AI (Khandelwal et al.)

Embodied AI: agents are trained to solve tasks in physical and simulated environments

- Learn to interact in their environments given sensor inputs (e.g. walking around a scene based on what it sees with the camera)





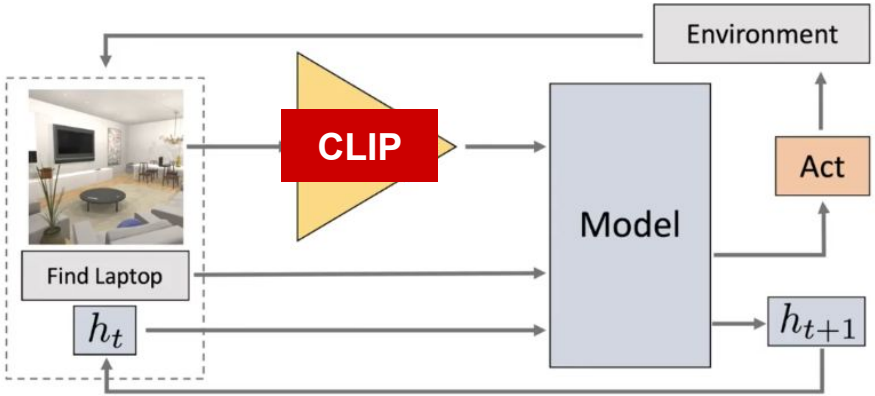


Source: https://www.youtube.com/watch?v=bnzgmlLePxw

Simple but Effective: CLIP Embeddings for Embodied AI (Khandelwal et al.)

Methodology: conduct a series of experiments, train baseline agents using dd-ppl with frozen ImageNet-pretrained ResNet-50 encoders and compare them against frozen ResNet-50 CLIP encoders

- Compare the models across four experiments covering three different navigational tasks and across two simulators (THOR and Habitat)

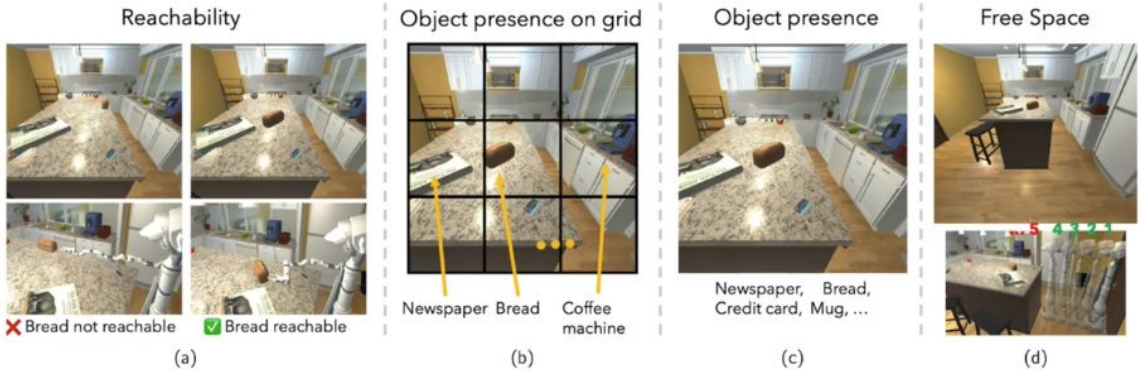


Source: https://www.youtube.com/watch?v=bnzgmlLePxw

Simple but Effective: CLIP Embeddings for Embodied AI (Khandelwal et al.)

Why are CLIP representations much better than ImageNet representations?

- Conduct linear probing experiments measuring how well their visual representations encode the following semantic primitives:



- CLIP encodes all four of these primitives more effectively!

Source: https://www.youtube.com/watch?v=bnzgmlLePxw