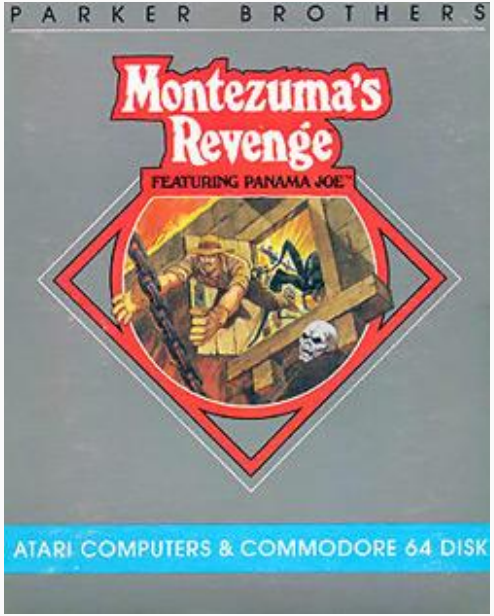
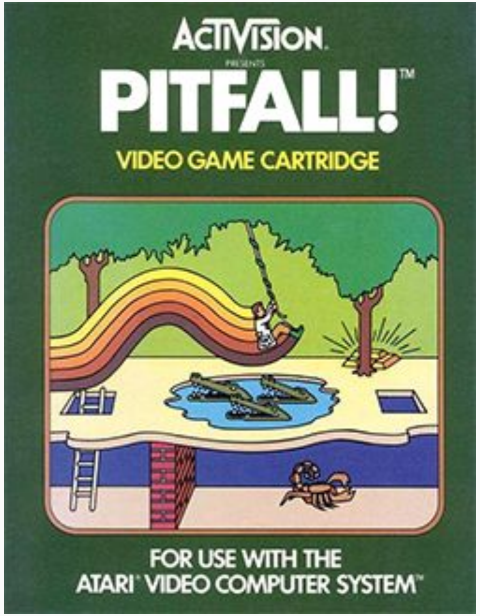
Playing hard exploration games by watching YouTube

Yusuf Aytar, Tobias Pfaff,
David Budden, Tom Le Paine, Ziyu Wang, Nando de Freitas

arXiv:1805.11592

Three Atari games from the early 80s

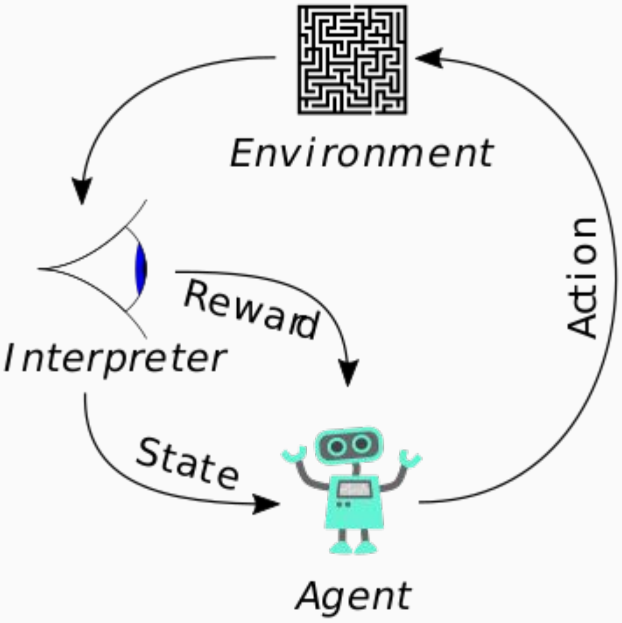




How do we teach our agent to play them?

Can we use **trial-and-error**?

Unfortunately, there's a problem



How do we teach our agent to play them?

The problem:

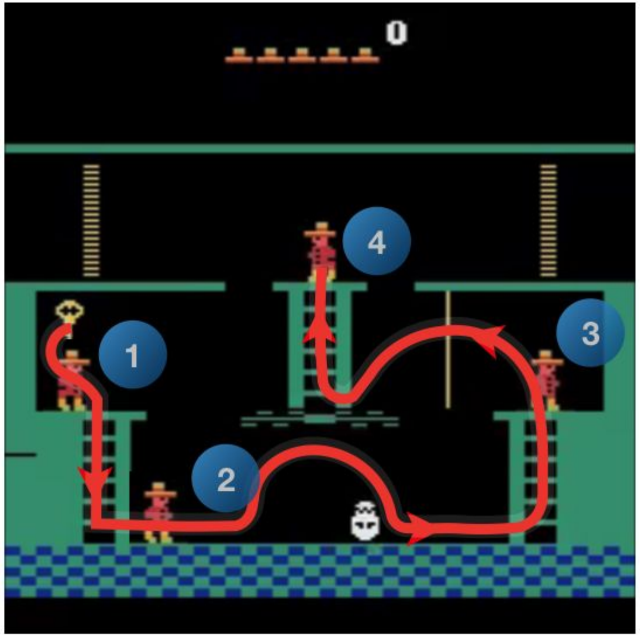
“hard exploration”

sparse environmental rewards

100s of environment steps

to even reach the first reward

in Montezuma’s Revenge

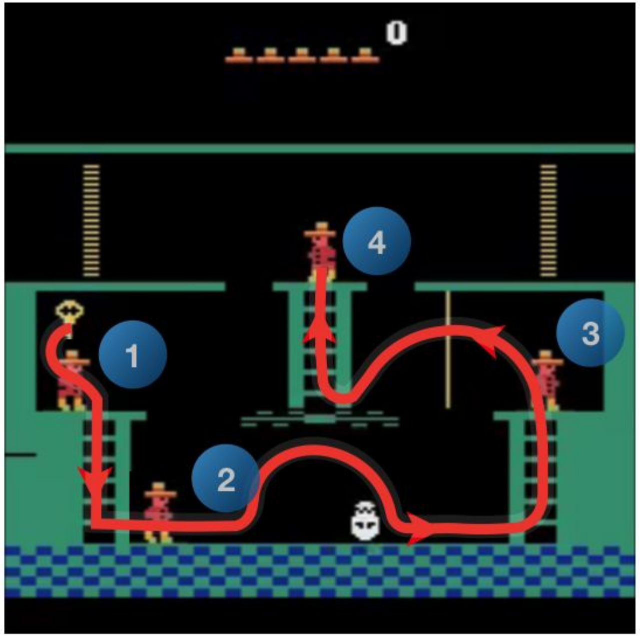


How do we teach our agent to play them?

One solution:

intrinsic motivation

create an auxiliary reward to encourage trying new trajectories; doesn't solve the problem of unknown-unknowns



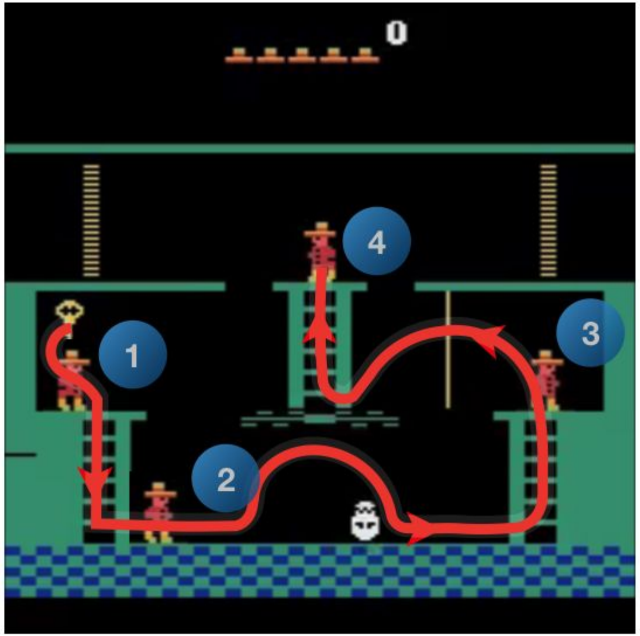
How do we teach our agent to play them?

Another solution:

imitation learning

observe some demonstrations of others playing the game, then imitate their trajectories

This can work!



What demonstrations would we use?

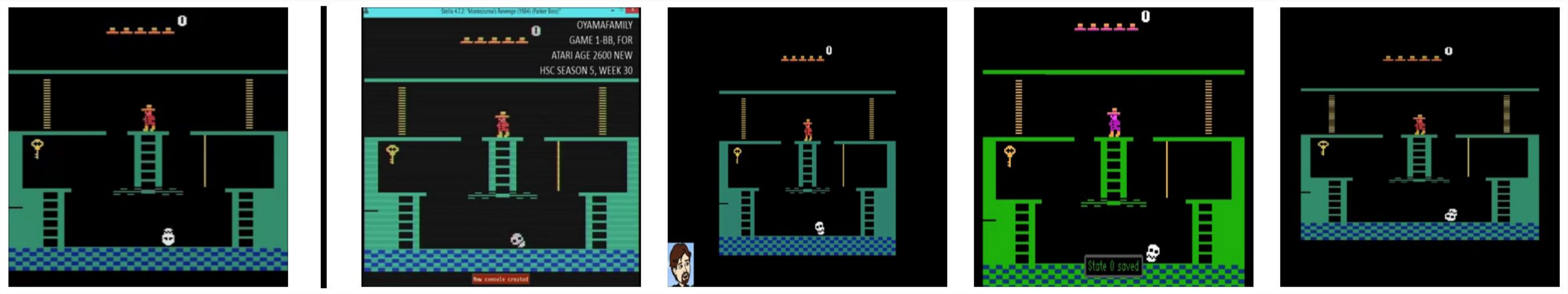
YouTube videos: humans can absorb knowledge easily by just watching somebody else do it

What demonstrations would we use?

YouTube videos: humans can absorb knowledge easily by just watching somebody else do it **despite significant differences** in timing, lighting, background, sounds, body characteristics etc.

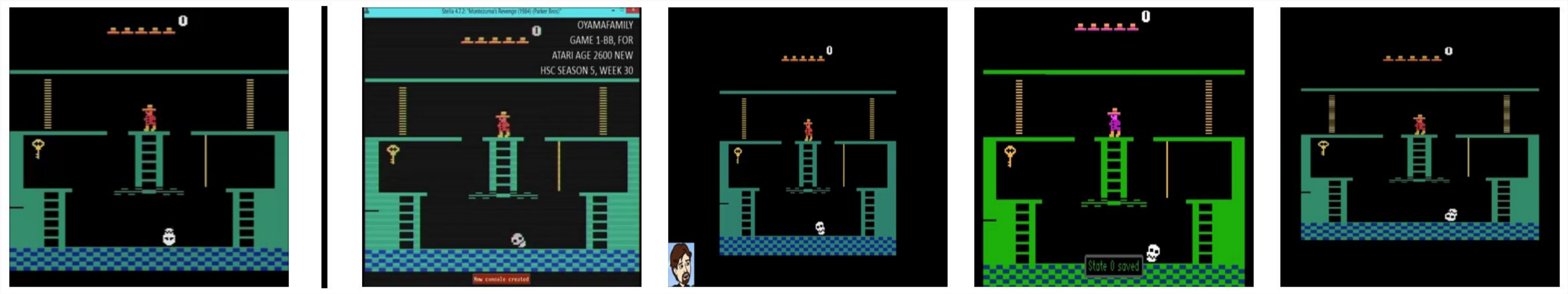
The challenges of using YouTube videos

First frame of Montezuma's Revenge on:
the Atari Learning Environment (on the left)
vs. four demonstration videos



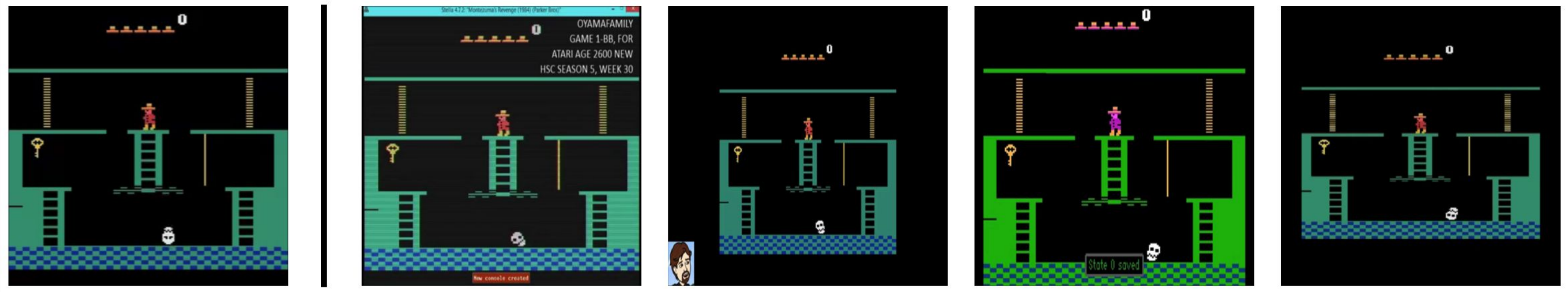
The challenges of using YouTube videos

Notice the different colors, aspect ratios, location within the frame + artifacts like the emulator window, avatar etc.
This is the **domain gap**.

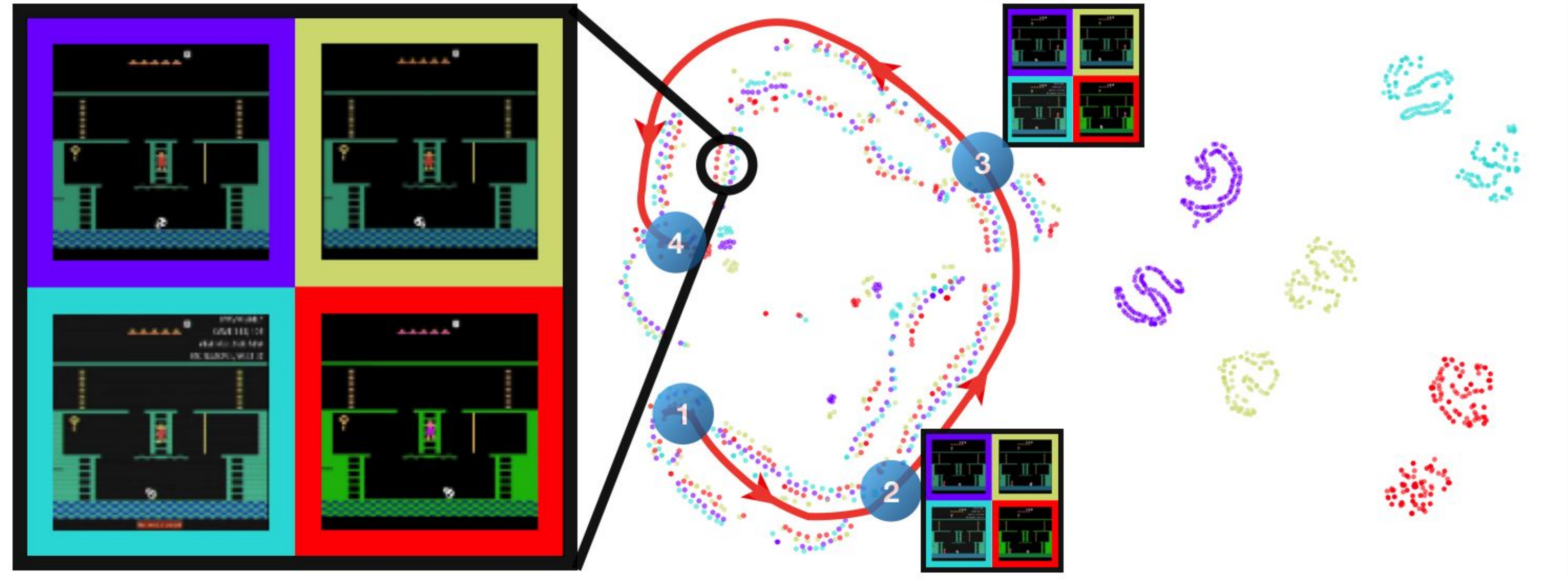


The challenges of using YouTube videos

The **domain gap** is a problem, since most methods, including the then-SotA, expect clean demonstrations, as well as complete **action-reward sequences** for them



Embedding: overcoming the domain gap



Training the embedder

There are three different training videos per game

The goal is to produce a **common representation** for them

Training the embedder - auxiliary task

The embedder can be trained by using it to solve an **auxiliary task**, which:

- is self-supervised
- encourages a desirable embedding

Training the embedder - auxiliary task

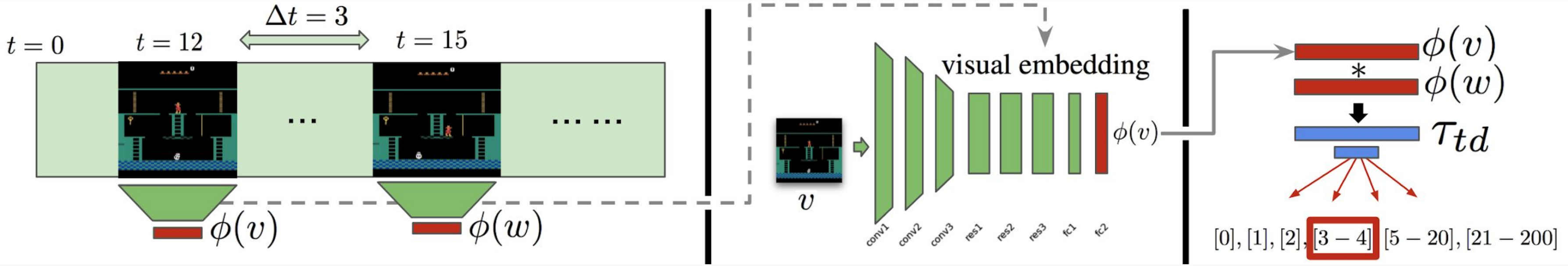
The auxiliary task:

Predicting the **temporal distance** between two frames from the same demonstration using:

- visual-visual embedding
- visual-audio embedding

Temporal distance classification (TDC)

Looking at the embeddings of two frames from one demonstration, determine the number of steps between them



Temporal distance classification (TDC)

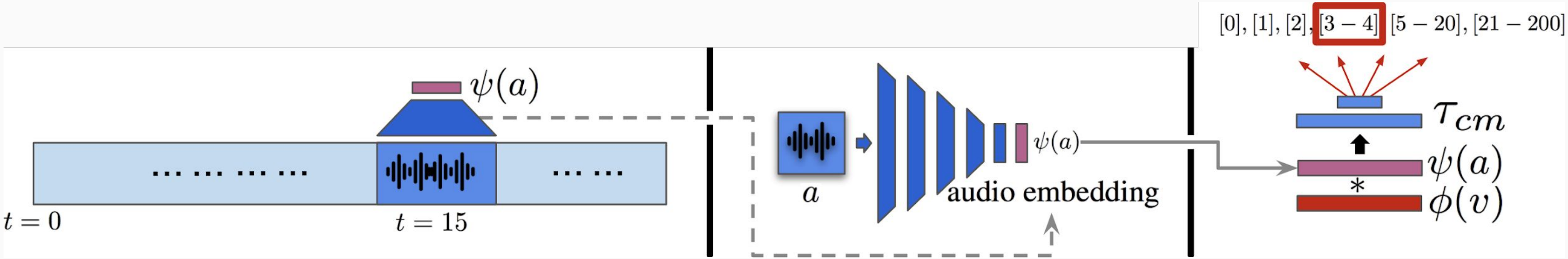
$$d_k \in \{[0], [1], [2], [3-4], [5-20], [21-200]\}$$

$$\phi : I \rightarrow \mathcal{R}^N$$

$$\tau_{tdc} : \mathcal{R}^N \times \mathcal{R}^N \rightarrow \mathcal{R}^K$$

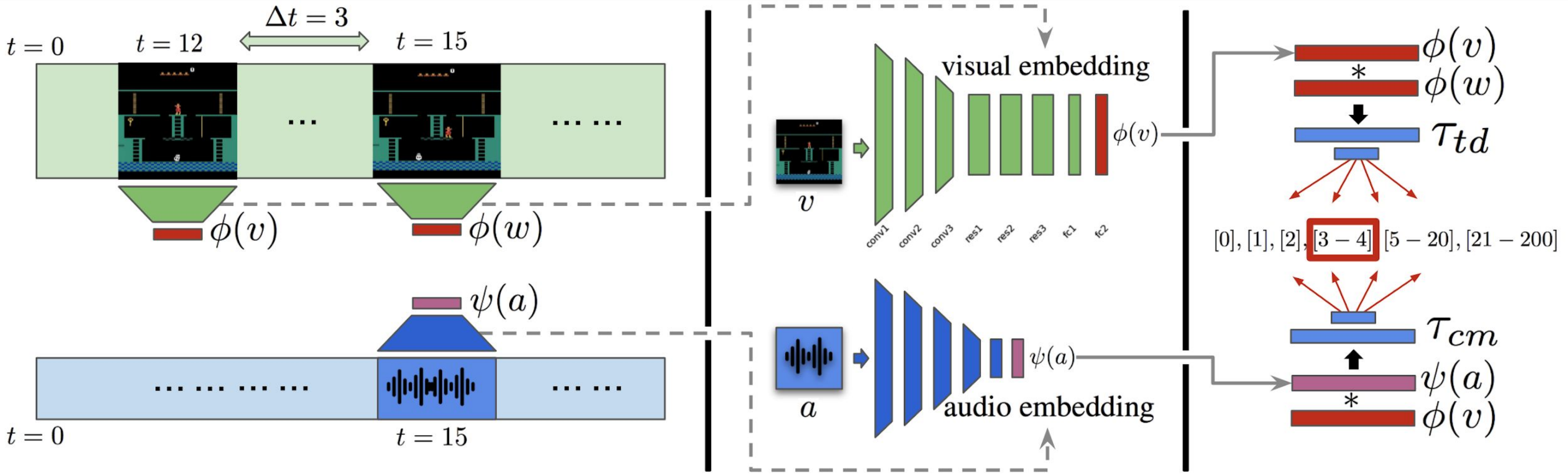
Cross-modal classification (CMC)

Looking at an embedded frame and a sound snippet, determine the time between them (eg. they're synchronized)



Complete classification problem

Training: minimize the weighted sum of cross-entropies.



The dataset for training the embedder

There are three training videos per game.

To get one pair of training frames:

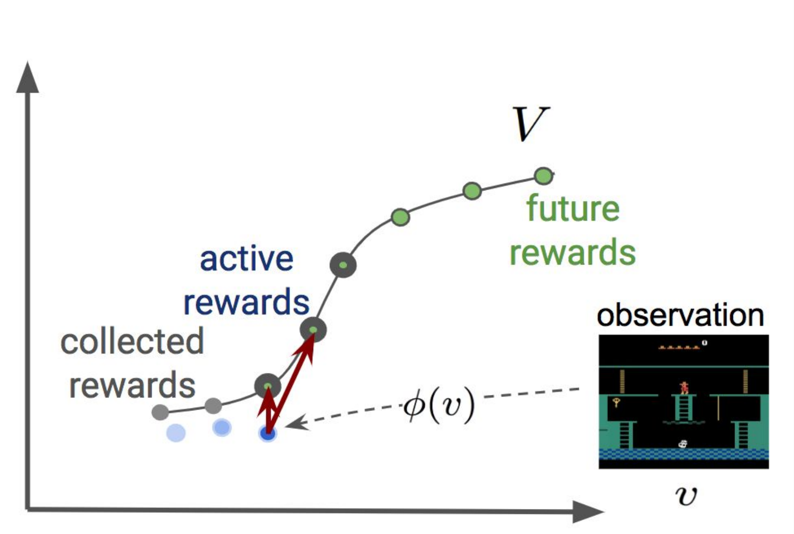
- sample one of three videos
- sample a time interval
- randomly select two frames separated by that interval

The last step - one-shot imitation

Combine:

- a **standard RL agent**
- the trained **embedder**
- another **YouTube video**

The goal: **imitate the video**.



The last step - one-shot imitation

Every 16 frames make a checkpoint, add an auxiliary reward for "visiting" the checkpoints in the right order.

$$r_{\text{imitation}} = \begin{cases} 0.5 & \text{if } \bar{\phi}(v_{\text{agent}}) \cdot \bar{\phi}(v_{\text{checkpoint}}) > \alpha \\ 0.0 & \text{otherwise} \end{cases}$$

$$\alpha = 0.5$$

$$v_{\text{checkpoint}} \in \{v^{(n+1)}, \ldots, v^{(n+1+\Delta t)}\}$$

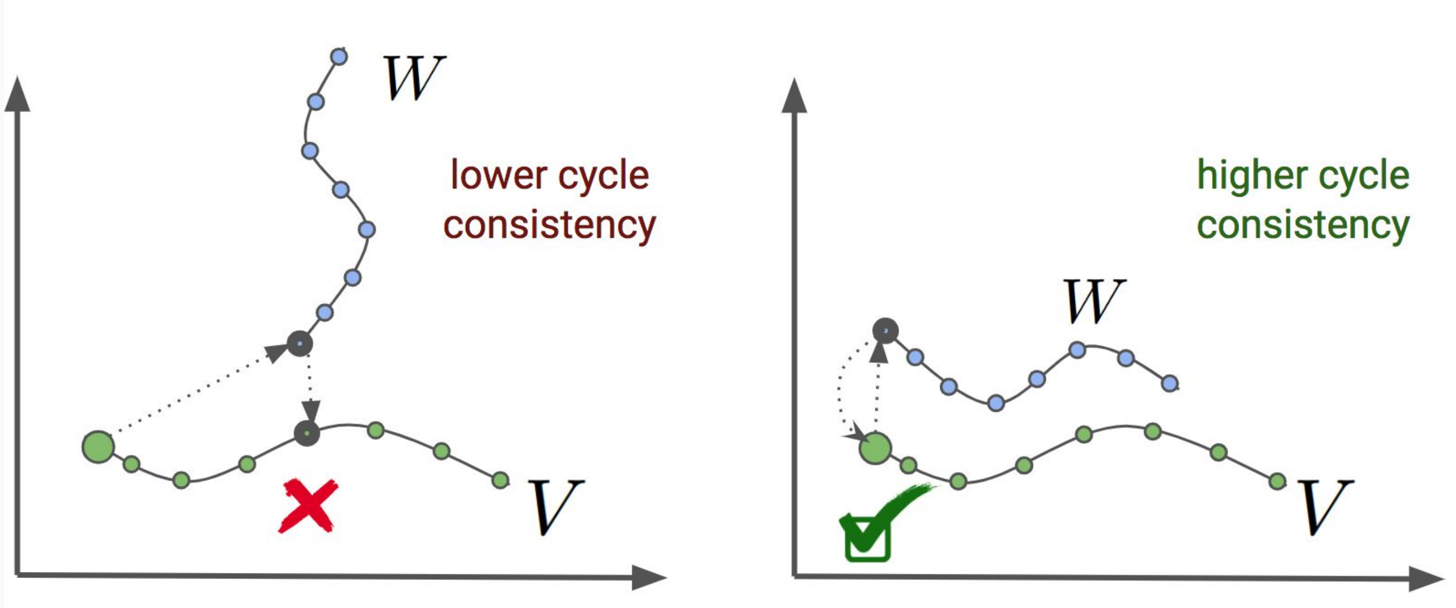
$$\Delta t = 1$$

Evaluating the embedder

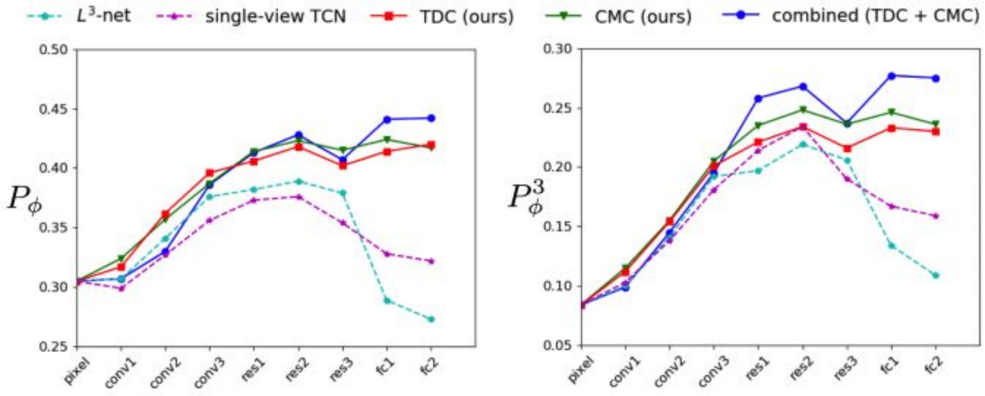
To successfully use YouTube videos as demonstrations, the embedder should exhibit:

- cycle-consistency and alignment capabilities
- meaningful abstractions of the game state

Cycle-consistency



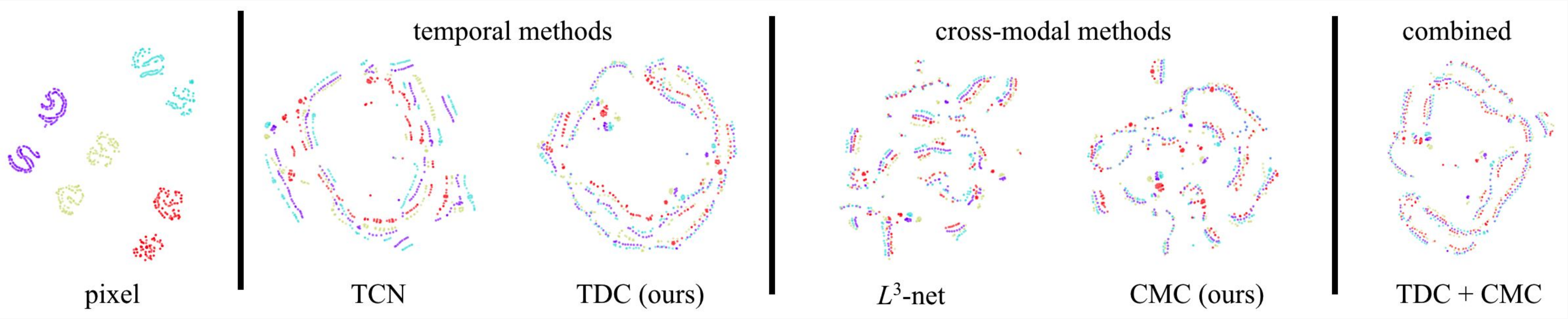
Cycle-consistency



Embedding Method	P_ϕ	P_ϕ^3
l_2 pixel distance	30.5	08.4
single-view TCN [34]	32.2	15.9
TDC (ours)	**42.0**	**23.0**
L^3-Net [3]	27.3	10.9
CMC (ours)	**41.7**	**23.6**
combined (TDC+CMC)	**44.2**	**27.5**

Embedding and alignment

https://youtu.be/RyxPAYhQ-Vo



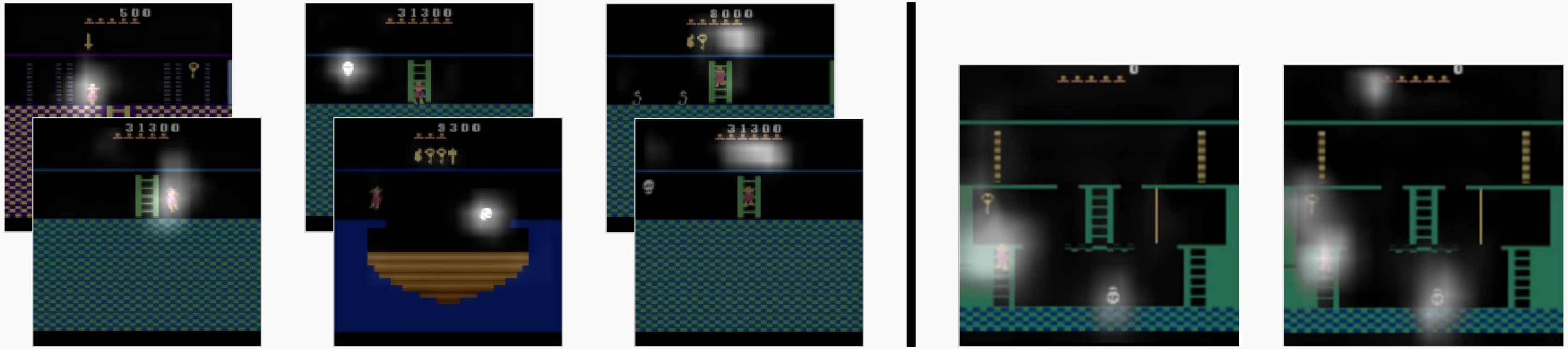
Embedding and alignment

t-SNE visualization shows:

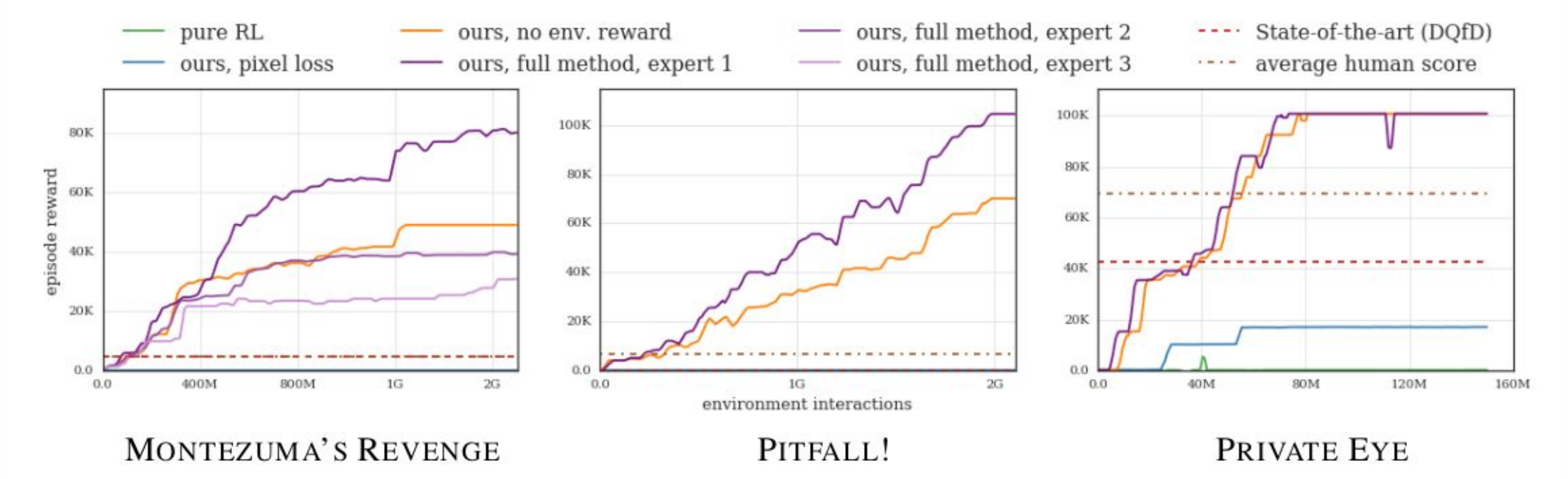
- different videos are recognized as following a similar path
- clear step-by-step trajectory
- not one cycle but two - the level requires “there and back again”, long-range dependency

Embedding and abstractions

The neurons focus on important things (inventory, player and enemy location); audio helps shift attention to inventory.



Overall results



Overall results

	Montezuma's Revenge	Pitfall!	Private Eye
Rainbow [19]	384.0	0.0	4,234.0
ApeX [22]	2,500.0	-0.6	49.8
DQfD [20]	4,659.7	57.3	42,457.2
Average Human [43]	4,743.0	6,464.0	69,571.0
Ours ($r_{\text{imitation}}$ only)	37,232.7	54,912.4	98,212.5
Ours ($r_{\text{imitation}} + r_{\text{env}}$)	**58,175.1**	**76,812.5**	**98,763.2**

Thank you for your attention