Knowledge-free domain-independent planning for video games

Raphael Lopes Baldi
Advisor: Felipe Meneguzzi
Computer Science
✉ raphael.baldi@acad.pucrs.br

Pontifical Catholic University of Rio Grande do Sul

Motivation

- Classical planning requires us to build symbolic representations of our domains, usually in a description language such as STRIPS or PDDL. Building those models represent a knowledge acquisition bottleneck, as we need to build and verify it manually [1];
- The AI community is interested in video games for their complexity. Describing a video game as a PDDL domain is a hard task for a human to perform, and results in extensive and incomplete planning definitions;
- Agents capable of playing Atari games usually rely on reinforcement learning algorithms or running look-ahead techniques to collect states for using as input for a classical planner using traditional search algorithms - like A*, and Greedy Best-First Search (GBFS).
- In this work, we evaluate a new technique to use classical planning algorithms in domains containing a large number of states and transitions - Atari's games - without the need to build a domain formalization in a domain definition language (like PDDL or Strips).

Solution

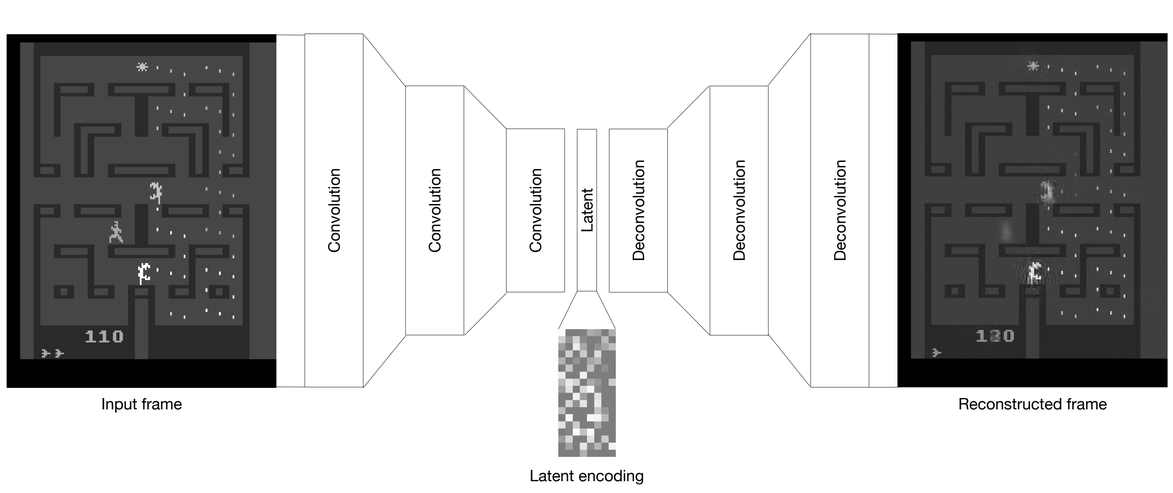

Figure 1: State Autoencoder.

- Our solution uses the Arcade Learning Environment [2] to run Atari games in a computer, and extract frames, and transitions from it. We perform random and human controlled actions, repeating each of those actions across several frames to improve diversity (we skip three frames for each one of the actions we perform).
- We use the grayscale frames to train an autoencoder (the State Autoencoder - SAE) from which we obtain a state encoding. This method allows us to reduce the dimensionality of our states, and get an artificial neural network (ANN) capable of encoding unseen states, and reconstruct states given its encodings (see Figure 1).
- With the encodings for known states we train three additional artificial neural networks:
  1. *State Discriminator (SD)*: an ANN that learns to tell valid and invalid state's encodings apart.
  2. *Action Autoencoder (AAE)*: an ANN that learns a transition function, given the encoding of states and the encoding of its successors (Figure 2).
  3. *Command Classifier (CC)*: an ANN that learns Atari commands from the encodings of states and the encoding of the states successors.
- We obtain encodings for the initial and goal states from the images that represent them.
- Finally, we run a classical planner, using the GBFS search algorithm, using image difference as the distance heuristic. Given the current state, for each action encoding, we reconstruct the possible successor state using the AAE. We prune the resulting state using the SD and testing the reconstruction with our SAE. When we get to the goal state, we have a sequence of state encodings, that we use in our command classifier to play the game.

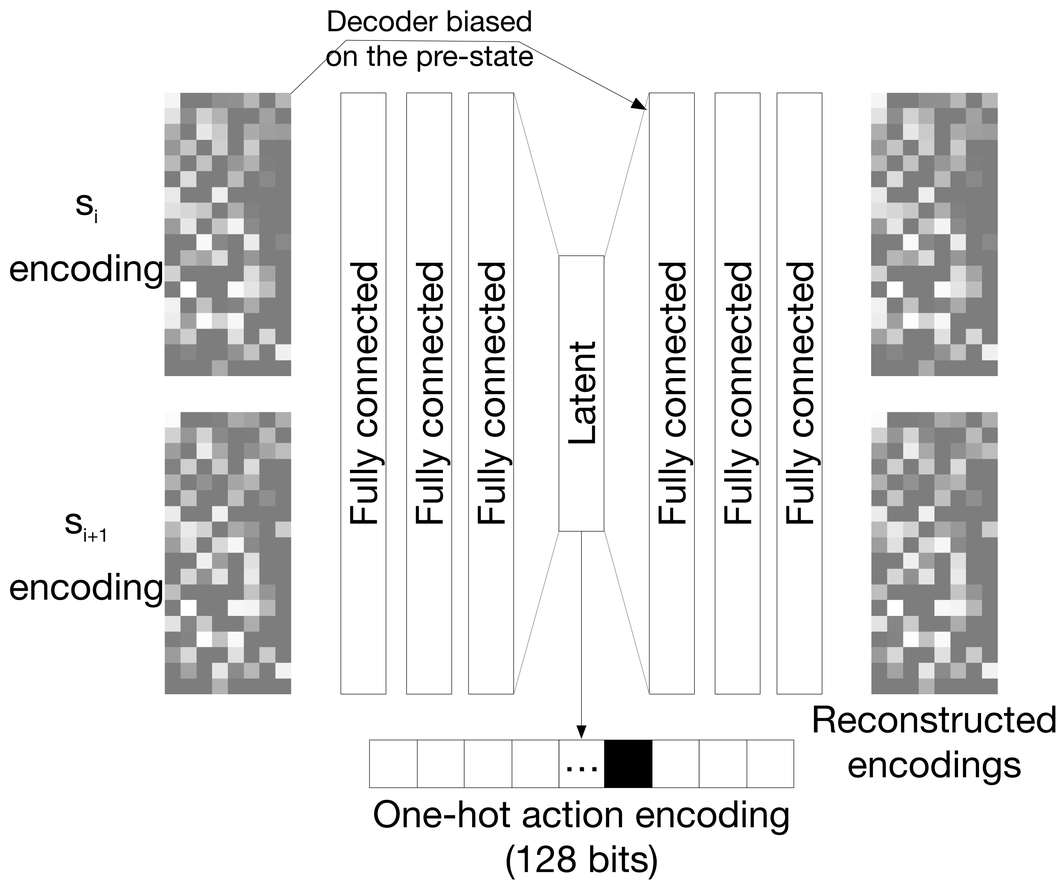

Figure 2: Action Autoencoder.

Experiments and results

- We evaluate three architectures for the state autoencoder, and tried to get the latent layer to converge into a categorical representation using the Gumbel-Softmax reparameterization trick [3];
- Our state discriminator, which is crucial to prune out invalid states during planning, produces too many false positives (it recognizes 3% of our fake states as invalid);
- Since we cannot prune states during plan expansion - due to the inaccuracy of our state discriminator -, and since we have 128 action encodings to inflate for each state the planner needs to check, our branching factor makes finding plans slow, and the planner cannot find plans deeper than four frames long. For shorter plans, the planner fails to detect the goal due to reconstruction error: the frame we reconstruct from the encoding is not close enough to the original goal frame for us to detect it (see Figure 1);
- We use two different autoencoders, and the accumulation of reconstruction losses makes the planner fail to detect the goal state. The noise in the reconstruction process makes detecting state similarities hard for the planner. If we reduce the state difference threshold, we get invalid plans.

Conclusion

- We evaluated a novel technique to use classical planning in video games and found negative results. Reconstruction loss in our networks makes the planner unable to find the goal state.
- With an accuracy of 36%, our command classifier leads to a substantial number of invalid actions during plan execution. Our future work includes:
  1. Design a single neural network to generate successor states, and action labels, and evaluate combining it with the *SAE*, to reduce the reconstruction loss accumulation effect;
  2. Evaluate other methods to generate the fake states we need to train the state discriminator.

References

[1] Masataro Asai and Alex Fukunaga. Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary. *AAAI Conference on Artificial Intelligence*, 2018.

[2] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Int. Res.*, 47(1):253–279, May 2013.

[3] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *CoRR*, abs/1611.01144, 2016.