Key Frame Plan Expansion With Machine Learning

Guillermo Carvalho Borges
PUCRS

Abstract

This paper proposes the use of machine learning techniques used for image processing to expand high level plans to a lower level of abstraction, taking inspiration from animation and how key frames or drawings are used as reference to fill in the gaps in between and deliver smooth motion.

Introduction

Machine learning has grown significantly over the past decade, and with the advance of computing power it is more accessible than ever. Every day solutions to new, interesting problems are found through processing data in new and different ways. It is an inspiring field of study, and through extensive research I have found an idea worth researching.

Continuous planning is a challenging subject. Plans are inherently discrete, but that doesn't stop a cleverly programmed robot or a human from filling in the gaps between steps. A person knows to open the door when going from one room to another, and a robot knows to keep the motors engaged as it travels from point A to checkpoint B.

But humans have an advantage over their mechanical brethren. Seemingly effortlessly, a person can take care of the minor details without much thought. Take for example a robot using A* search to navigate an environment: it has a detailed plan down to every single little twist and turn from beginning to destination. A person intuitively walks towards their destination, through any number of checkpoints on the way. Little thought is given to a small obstacle in the way.

Could a robot learn to fill in the gaps between more abstract instructions the way a person does? That's the question this research proposal seeks to answer.

The key frame approach

While considering the applications image processing techniques could have on planning algorithms, the process of animation instantly came into mind. Traditionally in animation production, a lead artist would draw the key frames representing the start and end of a character or object in motion. Meanwhile, a team of junior artists are tasked to fill in the gaps and draw in-between frames that when paired with the key frames, produce the illusion of motion.

Computing interpolated, in-between images is a classic problem in image and video processing, and is a necessary step in numerous applications such as frame rate conversion (Meyer et al. 2015). The same machine learning techniques can be applied to planning, by using state instead of pictures and training the network to interpolate "in-between" states.

This is particularly useful for agents whose actions are continuous rather than discrete. For example, should this article's approach work as intended, a classical planner could be used to create the positions a robotic arm should reach to accomplish a task, while a machine learning algorithm would fill in the gaps for smooth operation. A PDDL planning algorithm could work at a higher level of abstraction, with simpler predicates that are easily understood by a human maintainer, while a neural network would perform the low level planning required to take the agent through each step of the process.

Neural Agent Model

This section proposes a model by which to represent, and train, a machine-learning agent using the classical concepts. An agent is any entity responsible for their own behavior, with individual knowledge of their environment and the power to interact with it in some way. (Wooldridge and Jennings 1995)

Our model defines similar concepts within a machine learning perspective:

- Cognition: an agent's understanding of the environment and its place in it. A projection of perceived state into a set of features that accurately describe it.
- Reflexes: instinctive recognition of it's abilities and limitations. The neural agent has to learn how to interact with the environment and recognize the effect of its actions.
- Intuition: the agent's ability to infer an arbitrary number of in-between state transitions.

Cognition

The first of the three elements in the neural agent is the simplest one to implement. An auto-encoder is a neural network

which can compress an arbitrary input into a vector of features, which it can then decompress back into the original input with varying levels of accuracy. (Baldi 2011)

We can interpret this compressed state as the agent's understanding, or projection of the environment, but its main purpose is to filter, and compress all inputs into a smaller set of more meaningful variables. The training process is simple: to collect various samples of data the agent perceives and use that data set to train the network.

Reflexes

While the name is still subject to change, reflexes are also a straightforward implementation. Every action the agent can execute has a distinct set of features:

- The expected consequence of executing the action. In other words, the transition function between initial and final state.
- A confidence factor. This value is the probability the transition is valid.

Given the above, an agent can make an informed decision when choosing what action to take. Thus, those are the outputs of the reflex network, while the inputs are the agent's cognition of the initial state, and the parameters that define the action to execute.

Intuition

Training a network to infer frames through intuition involves several challenges. The first is to ensure the state transitions being generated are valid. This is done by training the reflex network separately. The task of the intuition component is not to predict the next in-between frame, but to predict the parameter vector that is then passed on to the reflex component. This process is repeated K times where K is the number of frames to be inferred. The intuition network is structured as a chain of reflex and intuition layers.

The objective of the intuition component is to chose X such that the resulting state will be "closer" to the final state than the previous one, and the final layer of the network has to produce the next key frame. This mimics the process of drawing in-between frames.

Finding how close a state is to another is a problem on its own. Fortunately, Generative Adversarial Networks are a suitable solution to this problem. The adversarial arrangement pits two networks against each other. The generator network tasked with generating fake data, and the discriminator has to estimate the probability that said data is real or not (Goodfellow et al. 2014). For our purposes, the intuition network acts as the generator, while the discriminator is tasked with determining how likely the generated and final state are to be equal.

Thus, to infer the key frames between states 'A' and 'F', the intuition network must generate a new state 'b' which has a higher probability to be 'F' than 'A' when tested against the generator. In other words, somewhere between A and B.

Without a fixed number of frames (determining the depth of the network), it has no incentive to ever reach B. It could very well get stuck generating minuscule improvements and never reach the goal within the allocated number of frames.

Therefore, we set the number of key frames to a fixed number K. That way, the network can be trained as a chain of intuition and reflex layers, up to 2K total layers. The output of such a network has to be close enough to the final state that the discriminator function reports a high enough probability that they are equal.

Limitations

Since the final layer of this intuition network has to approximate the final frame, back-propagation will adjust the network to "stretch" its frames accordingly. Training has to be done in such a way the weights of the reflex network are not altered in the process, or the validity of the frames will be compromised. For this approach to work, the environment has to be continuous; an arbitrary number of valid state transitions can be created in between the initial and final state. Additionally, the quality of the in-between frames might limit its applications.

Project Plan

Should it be approved, this project aims to achieve the following:

- Establish a testable environment for the agent that highlights use cases for the key frame approach.
- Generate data sets and train the described neural agent model.
- Test the effectiveness of the trained system, and perform corrections to the model as necessary.
- Should the presented model produce satisfactory results, look for alternatives for a variable frame count approach.
- Write a follow up article with the intent of future publication.

References

Baldi, P. 2011. Autoencoders, unsupervised learning and deep architectures. In *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27*, UTLW'11, 37–50. JMLR.org.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, 2672–2680. Cambridge, MA, USA: MIT Press.

Meyer, S.; Wang, O.; Zimmer, H.; Grosse, M.; and Sorkine-Hornung, A. 2015. Phase-based frame interpolation for video. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 1410–1418.

Wooldridge, M., and Jennings, N. R. 1995. Intelligent agents: Theory and practice. *Knowledge Engineering Review* 10:115–152.