Using Deep Learning to Automate Feature Modeling in Learning by Observation: A Preliminary Study

Michael W. Floyd[1], JT Turner[1], and David W. Aha[2]
[1]Knexus Research Corporation; Springfield, Virginia; USA
[2]Navy Center for Applied Research in AI; Naval Research Laboratory (Code 5514); Washington, DC; USA
{michael.floyd, jt.turner}@knexusresearch.com | *david.aha@nrl.navy.mil*

Abstract

A primary advantage of learning by observation is that it allows non-technical experts to transfer their skills to an agent. However, this requires a general-purpose learning agent that is not biased to any specific expert, domain, or behavior. Existing domain-independent learning by observation agents generalize a significant portion of learning but still require some human intervention, namely, modeling the agent's inputs and outputs. We describe a preliminary evaluation of using convolutional neural networks to train a learning by observation agent without explicitly defining the input features. Our approach uses the agent's raw visual inputs at two levels of granularity to automatically learn input features using limited training data. We describe an initial evaluation with scenarios drawn from a simulated soccer domain.

1. Introduction

Learning by observation (LbO) agents are trained to perform specific behaviors by observing an expert demonstrate the behaviors. Whereas traditional methods for training an agent may involve computer programming or knowledge modeling competency, LbO only requires the expert to be able to perform the behavior. By shifting the knowledge-acquisition task from the expert to the agent itself, the agent is provided with the opportunity to learn from a variety of non-technical experts (e.g., healthcare professionals, military commanders). However, for an agent to learn an unknown behavior without any prior knowledge of the expert or domain, it should learn in a general, non-biased manner.

We describe our preliminary approach to overcome the limitations of existing general-purpose learning by observation agents. Specifically, we remove the need for input features to be manually modeled for each domain. Instead, we use deep learning (DL) techniques (LeCun, Bengio, and Hinton 2015) to learn a feature representation from the agent's raw visual inputs. Our approach trains two DL models: one uses the agent's complete visual inputs (i.e., everything it can currently observe) while the other uses close-range visuals. The output of the two models are used to select actions to perform in response to novel visual input (i.e., what the agent can see as it attempts to replicate the expert's behavior).

Our preliminary evaluation examines the feasibility of our approach under common learning by observation conditions. More specifically, these conditions include limited observations (i.e., due to limited expert availability), noisy or erroneous observations (e.g., errors by the expert or incorrect observations by the agent), and partial observability in the environment. We discuss related research in Section 2, followed by a description of our approach in Section 3. We evaluate our approach using scenarios defined in a simulated soccer domain in Section 4, and conclude with a discussion of future work in Section 5.

2. Related Work

Learning by observation has been used in a variety of domains, including poker (Rubin and Watson 2010), Tetris (Romdhane and Lamontagne 2008), first-person shooter games (Thurau, Bauckhage, and Sagerer 2003), helicopter control (Coates, Abbeel, and Ng 2008), robotic soccer (Grollman and Jenkins 2007), simulated soccer (Floyd, Esfandiari, and Lam 2008; Young and Hawes 2015), and real-time strategy games (Ontañón et al. 2007). However, most of these approaches were designed to learn in a single domain, so the agents cannot be directly transferred to new environments. Two domain-independent approaches for LbO have been proposed (Gómez-Martín et al. 2010; Floyd and Esfandiari 2011), both of which separate the agent's learning and reasoning from how it interacts with the environment. This is advantageous because the observation, learning, and reasoning components are general-purpose and are not biased to any specific expert, behavior, or domain. However, they both require the inputs (i.e., what objects the agent can observe) and outputs (i.e., the actions the agent can perform) to be modeled. Although the modeling only needs to be performed once (i.e., before the agent is deployed in a new environment), it still requires some human intervention. Floyd, Bicakci, and Esfandiari (2012) use a robot architecture that allows sensors to be dynamically added or removed, with each change modifying how the LbO agent represents inputs. While this does not require human intervention before deployment in a new domain, it does require human intervention for each new type of sensor. Our approach differs in that it does not require any human intervention to model the environment; the only requirement is that the domain provides a visual representation of the environment.

Deep learning by observation is used for initial training of AlphaGo (Silver et al. 2016). However, their learning methodology has several limitations that may make it unsuitable for some LbO tasks. First, they trained their system with over 30 million observations. Large datasets may be available for established games like Go, but less popular games or novel behaviors may not have any existing observation logs. Second, such a large dataset requires months of training using datacenters composed of state-of-the-art hardware. If models need to be trained rapidly with limited computational resources, alternative learning approaches are necessary. Finally, LbO is performed using images of a turn-based board game. This minimizes the influence of object occlusion (i.e., each Go piece is on its own square), observation error (e.g., due to erroneous or delayed responses by the expert), and provides the learning agent with full observability. We instead examine the feasibility of using DL for LbO tasks with limited observations and limited training time in complex, real-time domains.

Our feature learning method is inspired by the deep reinforcement learning work of Mnih et al. (2015). They use raw visual inputs to learn to play a variety of Atari 2600 games. A primary difference from our work, in addition to the amount of training time required to train their agents, is they use reinforcement learning rather than LbO. Reinforcement learning requires a reward function to be defined for each domain (e.g., based on the game score), thereby adding additional knowledge engineering before an agent can be deployed in a new environment. Deep reinforcement learning has also been used in simulated soccer (Hausknecht and Stone 2016), with the reward functions partially encoding the desired behavior (e.g., *move to ball* reward and *kick to goal* reward). Although reinforcement learning approaches are beneficial in that they do not require labeled training data, they require explicitly encoding reward functions which may bias the agents to learning specific behaviors.

3. System Design

In real-time computer games, agents typically receive sensory inputs in the form of periodic messages from the game. These messages can include information about the state of the game (e.g., elapsed time, score), the agent's properties (e.g., player number, team name, resource levels), and observable objects. The observable objects are particularly important for an agent's decision making because they provide information about the physical state of the environment. For example, in a soccer game the observable objects would include the location of the ball, other players, goal nets, and boundary markers. While most games explicitly define the set of observable objects in the game (e.g., in a user manual), deploying an agent in a new game still requires some level of knowledge engineering to model these objects (i.e., converting the object definition into a format that is understandable by the agent).

To remove the need for modeling the observable objects, our approach uses the raw visual representation of the environment. For example, Figure 1 shows a player's view of the field in a soccer game. The left side of Figure 1 shows the player's entire field of vision, which we will refer to as the *full* visual representation, whereas the right side shows an enlarged view of the objects close to the player (i.e., a fixed-sized region surrounding the player), which we refer to as the *zoomed* visual representation. Both representations contain only a partial view of the environment (i.e., what is currently within the player's field of vision, not the entire field), with the full representation giving a larger view of the field than the zoomed representation. The agent is not explicitly given information about what is contained in the images (e.g., it does not know that the white circle is the soccer ball). Each of the visual representations is stored as a 256×256 RGB image.

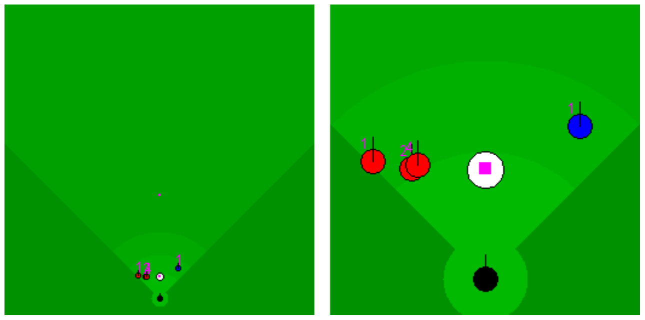
Figure 1: The full visual representation (left) and zoomed visual representation (right) in a simulated soccer game

During observation, the learning agent records the expert's current visual inputs, both the full version V_{full} and zoomed version V_{zoomed}, as well as the action A performed by the expert. Each input-action pair is stored in the corresponding observation set, $\mathcal{O}_{full}$ or $\mathcal{O}_{zoomed}$ ($\mathcal{O}_{full} \leftarrow \mathcal{O}_{full} \cup \langle V_{full}, A \rangle$ and $\mathcal{O}_{zoomed} \leftarrow \mathcal{O}_{zoomed} \cup \langle V_{zoomed}, A \rangle$).

Learning is performed using two convolutional neural networks (CNN) (Krizhevsky, Sutskever, and Hinton 2012), with one trained on the full observations (i.e., $\mathcal{O}_{full}$) and a second trained on the zoomed observations (i.e., $\mathcal{O}_{zoomed}$). These models represent the environment at two levels of granularity and are used in combination to overcome limited training data. For example, a nearby ball would be easier to detect in the zoomed image because objects appear larger, whereas the full image would be necessary to detect a goal net on the other side of the field.

We use a modification of the CaffeNet architecture (Jia et al. 2014): an input layer, five convolution layers, five pooling layers, two fully connected layers, and one softmax loss layer. The network takes as input the pixel values using all three color channels (i.e., red, green, and blue), resulting in $256 \times 256 \times 3$ inputs. The outputs of the network represent the confidence in each of the possible actions (i.e., the confidence that each action should be selected in response to the input image). In the soccer example, three actions[1] are used: *kick*, *dash* (i.e., move), and *turn*.

Rather than training the entire network, our approach uses several layers that are pretrained on other data sources. The convolution and pooling layers are extracted from an existing network trained on ImageNet data (Jia et al. 2014), whereas the fully connected layers and softmax loss layer are trained using observation data. This approach has two primary advantages. First, the pretrained ImageNet layers can identify many visual features already (e.g., lines, curves, shapes, objects). This removes the need to relearn these common features. Second, the limited number of observations makes it impractical to train the entire network. Instead, the network learns how to use existing features to classify the observation data. Although some layers are pretrained, they do not bias the learning to any particular domain or task since the ImageNet dataset contains millions of images across a variety of topics (i.e., they are not soccer-specific images). During learning, both the full and zoomed models use an identical architecture but are trained independently.

During deployment, the learning agent attempts to replicate the expert's behavior and uses its own visual input as input to the CNNs. For each input the agent receives, the CNNs output six confidence outputs (i.e., both networks output confidence values for all three actions). The maximum of the six confidence values is selected and its associated action is used by the agent (i.e., the agent performs the action in the environment). By using this combined approach, the agent leverages the strengths of each individual model during action selection. For example, we would expect the zoomed model to perform better when important objects are near the agent, whereas the full model should perform better when information from the entire field of vision is necessary. The primary goal of deployment is for the agent to select similar actions to the expert when presented with similar sensory inputs.

4. Evaluation

To evaluate the performance of our DL LbO system we collected data from the RoboCup Simulation League (RoboCup 2016). The matches were 5 vs 5 soccer games with each player controlled by a scripted AI agent. The specific agent used, *Krislet*, performs simple soccer behaviors that involve locating the ball, running towards the ball, and kicking the ball towards the opponent's goal. In each match, a single player was used as the expert (i.e., its inputs and actions were recorded). The learning agent observed 10 full soccer matches, with each game being 10 minutes in length. In total, this resulted in approximately 40,000 observations for both the full and zoomed observation sets. However, the dataset is highly imbalanced (73% dash, 26% turn, 1% kick), so a balanced training set was created such that each action was equally represented (1617 total observations in each observation set). A balanced test set of 1029 observations was created by observing additional soccer matches.

The CNNs were trained using a base learning rate of 0.01, polynomial rate decay with a power of 3, and 13,000 training iterations. Table 1 shows the F_1 score (i.e., harmonic mean of precision and recall, with 1.0 being the maximum possible performance) when the test set was used to evaluate the trained models. In addition to our combined approach, we also evaluated performance when only the full or zoomed model was used for action prediction.

Table 1: Results of trained CNNs on RoboCup test data

Model	**F_1 Kick**	**F_1 Dash**	**F_1 Turn**	**F_1 Overall**
Full	0.84	0.56	0.59	0.67
Zoomed	0.93	0.57	0.57	0.69
Combined	0.92	0.61	0.61	0.71

These results, while preliminary, show that the agent can learn a suitable model for action selection. While both the

[1] Soccer actions can also be parameterized (e.g., how hard to kick, turn direction) but for simplicity our initial evaluation only examines action classification.

full and zoomed models perform reasonably well, the best performance was achieved when the *Combined* model was used. This demonstrates that using multiple representations of the visual data is preferable since these models have varying strengths and weaknesses.

5. Conclusions and Future Work

We described a preliminary study of how well a learning by observation agent can learn without explicitly modeling the objects it observes. Our approach uses an expert's raw visual inputs at two levels of granularity to train a pair of CNNs. In our study, the agent reproduced the expert's action selection decisions reasonably well in tasks drawn from a simulated soccer domain. This indicates that even with limited training observations, noisy observations, and partial observability, it is possible to create an agent that can learn an expert's behavior without being provided an explicit object model.

Although our approach removes the need to model observable objects, it still requires modeling the possible actions. An area of future work will be to identify methods for learning the actions an expert performs based on observations. Additionally, we have only examined a single two-model architecture (i.e., selecting the most confident prediction from two CNNs). In future work we will examine if added benefit can be achieved by training additional models (e.g., other levels of granularity) or by modifying how the model outputs are combined (e.g., inducing a decision tree from their output). Our preliminary evaluation has only measured the performance from a single experiment from a single expert in a single domain. We plan to perform a more thorough evaluation of the learning performance involving numerous experimental trails. This will not only allow us to show the benefit of our approach, but it will also allow for a thorough comparison with other LbO agents that learn in RoboCup (Floyd, Esfandiari, and Lam 2008; Young and Hawes 2015). To determine whether our approach is truly domain-independent, we plan to conduct additional studies with different experts in different environments. Finally, we plan to examine how this approach can be extended to learn from state-based experts since the RoboCup expert we examined is purely reactive (i.e., the expert's action is based entirely on its current visual inputs).

References

Coates, A., Abbeel, P., and Ng, A. Y. 2008. Learning for control from multiple demonstrations. In *Proceedings of the 25th International Conference on Machine Learning*, 144-151. Helsinki, Finland: ACM.

Floyd, M. W., Bicakci, M. V. and Esfandiari, B. 2012. Case-based learning by observation in robotics using a dynamic case representation. In *Proceedings of the 25th International Florida Artificial Intelligence Research Society Conference*, 323-328. Marco Island, USA: AAAI Press.

Floyd, M. W., and Esfandiari, B. 2011. A case-based reasoning framework for developing agents using learning by observation. In *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence*, 531-538. Boca Raton, USA: IEEE Computer Society Press.

Floyd, M. W., Esfandiari, B., and Lam, K. 2008. A case-based reasoning approach to imitating RoboCup players. In *Proceedings of the 21st International Florida Artificial Intelligence Research Society Conference*, 251-256. Coconut Grove, USA: AAAI Press.

Gómez-Martín, P. P., Llansó, D., Gómez-Martín, M. A., Ontañón, S., and Ram, A. 2010. MMPM: A generic platform for case-based planning research. In *Proceedings of the International Conference on Case-Based Reasoning Workshops*, 45-54. Alessandria, Italy.

Grollman, D. H., and Jenkins, O. C. 2007. Learning robot soccer skills from demonstration. In *Proceedings of the IEEE International Conference on Development and Learning*, 276-281. London, UK: IEEE Press.

Hausknecht, M., and Stone, P. (2016) Deep reinforcement learning in parameterized action space. In *Proceedings of the International Conference on Learning Representations*. San Juan, Puerto Rico.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R. B., Guadarrama, S., and Darrell, T. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, 675-678. Orlando, USA: ACM.

LeCun, Y., Bengio, Y. and Hinton, G. E. 2015. Deep learning. *Nature*, 521, 436-444.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. 2012. Classification with deep convolutional neural networks. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*, 1106-1114. Lake Tahoe, USA.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature*, 518, 529-533.

Ontañón, S., Mishra, K., Sugandh, N., and Ram, A. 2007. Case-based planning and execution for real-time strategy games. In *Proceedings of the 7th International Conference on Case-Based Reasoning*, 164-178. Belfast, UK: Springer.

RoboCup. 2016. *RoboCup Official Site*. Retrieved from [http://www.robocup.org]

Romdhane, H., and Lamontagne, L. 2008. Forgetting reinforced cases. In *Proceedings of the 9th European Conference on Case-Based Reasoning*, 474-486. Trier, Germany: Springer.

Rubin, J., and Watson, I. 2010. Similarity-based retrieval and solution re-use policies in the game of Texas Hold'em. In *Proceedings of the 18th International Conference on Case-Based Reasoning*, 465-479. Alessandria, Italy: Springer.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529, 484-503.

Thurau, C., Bauckhage, C., and Sagerer, G. 2003. Combining self organizing maps and multilayer perceptrons to learn bot-behaviour for a commercial game. In *Proceedings of the 4th International Conference on Intelligent Games and Simulation*, 119-123. London, UK: EUROSIS.

Young, J., and Hawes, N. 2015. Learning by observation using qualitative spatial relations. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems,* 745-751. Istanbul, Turkey: ACM.