Deep Models and Artificial Intelligence for Military Applications:
AAAI Technical Report FS-17-03

A Framework Using Machine Vision and Deep Reinforcement Learning for Self-Learning Moving Objects in a Virtual Environment

Richard Wu, Ying Zhao, Alan Clarke, Anthony Kendall
rwu@umassd.edu, yzhao@nps.edu, ajclarke@nps.edu, wakendal@nps.edu
Naval Postgraduate School, CA, USA

Abstract

In recent artificial intelligence (AI) research, convolutional neural networks (CNNs) can create artificial agents capable of self-learning. Self-learning autonomous moving objects utilize machine vision techniques based on processing and recognizing objects in digital images. Afterwards, deep reinforcement learning (Deep-RL) is applied to understand and learn intelligent actions and controls. The objective of our research is to study methods and designs on how machine vision and deep machine learning algorithms can be implemented in a virtual world (e.g., a computer game) for moving objects (e.g., vehicles or aircrafts) to improve their navigation and detection of threats in real life. In this paper, we create a framework for generating and using data from computer games to be used in CNNs and Deep-RL to perform intelligent actions. We show the initial results of applying the framework and identify various military applications that may benefit from this research.

Introduction

Training a computer to identify adversary weapons or hostile vehicles in complex backgrounds, as one of the machine vision applications, would bring values to the military especially in real-time and post-mission processing. Real-time applications include assisting human sensor operators in analyzing footage or scanning for threat indications, missile launches, or navigation cues when the sensor is not actively being monitored by an operator. Combining the visual scene (optical flow) with other sensor data (position, pose, control inputs) might provide valuable insight in training autonomous systems to identify, emulate and augment human performance. The first step is to filter large volumes of data collected from electro-optical (EO) sensors, which reduces the human-manageable load for fast analysis. The second step is to analyze the filtered data sets to perform interesting discovery. For example, intelligent machine vision analysis can be used to find crash wreckage in the open ocean, or to spot deviations in the behavior of flocking birds that could indicate the bird disturbances caused by troop movements.

Although most autonomous aerial navigation techniques use a combination of global positioning system (GPS) and inertial navigation system (INS) information, they cannot perform the interesting discovery described above. New sensor technologies such as LIDAR-based sensors, which can detect and avoid obstacles, evaluate landing zones, and plan routes to perform tasks (AACUS 2017), can only generate more image data but cannot perform behavior-based discovery.

Machine vision with machine learning implemented in moving objects such as cars, helicopters, drones, and many other unmanned aerial vehicles (UAV) has potential to overcome the limitations of INS, GPS and sensors, which not only improves orienting and sensing the environment, but also performs advanced self-learning and discovering to accomplish various tasks such as monitoring its surroundings, detecting objects, mapping terrain, and for autonomous controls.

There are three types of machine learning as follows:

- Supervised learning: requires training data to correlate input (e.g., images) and output (e.g., object classes) so that it can predict output for a new data. The convolutional neural networks (CNNs) method used in machine vision is a supervised learning method.
- Reinforcement learning: requires the feedback reward data from the environment through exploration to modify the parameters in a system so it learns to reinforce the right actions and avoid the wrong ones. Reinforcement learning is important for self-learning, real-time learning and adaptation in large-scale applications (DeepMind 2017).
- Unsupervised learning: requires only input data to discover patterns and anomalies. It can be used to

spot the deviation behavior such as the bird deviation combined with machine vision.

Machine vision and machine learning are important methods and components in modern AI designs and systems. When the components of machine vision and machine learning are implemented in military sensors, they have the potential to increase the value of the sensors, for example, to have the sensors working towards better understanding and learning the environment instead of just collecting data. Commercial companies have already been working in developing machine vision and machine learning applications supported by sensor footage libraries (FLIR 2017). For example, traffic camera systems can use machine learning to improve their ability to detect and discern people either riding bicycles or cars. Military applications such as automatic target detection could reduce operator work load and improve detection probabilities. Other applications related to the military mission include vehicle mounted cameras to assess features in the field of view to identify weapons, weapon signatures, or other items of interest. Machine vision and machine learning would be useful for such military applications as:

- Performing AI-guided human attention management: Single-seat aircraft so that the sensor can perform while the pilot is flying the aircraft or doing other tasks in real-time.
- Optimizing human supervisor to remote systems ratios: A single operator is monitoring multiple cameras in a security role in real-time.
- Automating analysis/flagging of post-mission data by providing human analysts with highlights.
- Distributing tasks of persistent multi-constraint, correlation and processing with an AI assistance.
- Cross-correlating target ID with other (e.g., electromagnetic) sensors to increase ID confidence.
- Recognizing landscape/terrain, sidereal scene features to validate navigation in GPS-denied environment.
- Performing critical intervention: recognition of hazard conditions, imminent collisions by optical flow.
- Identifying hand-held weapons at an airport checkpoint or large vehicles such as commercial aircraft.

Another analysis of data is to discover patterns over time in the scene such as identifying 1) people moving much more quickly than usual (e.g., running away from something); 2) people meeting at time that they don't usually meet; or 3) places devoid of people that are normally occupied, which is outside the realm of the purely optical techniques and more in the data realm.

The machine vision applications discussed above require large amount of training data - e.g., images and their classes due to the nature of supervised machine learning. Although the training data is difficult and expensive to generate in reality, previous research demonstrated that it's much easier to obtain the training data from a virtual environment – e.g., video games (Knight 2016) (Shafaei 2016).

Overview of Methods and Concepts

The objective of our research is to study methods and designs on how machine vision and deep machine learning algorithms can be implemented in a virtual world (e.g., a computer game) for moving objects (e.g., vehicles or aircrafts) to improve their navigation and detection of threats in real life.

In order to understand the methods and framework used, we provide an overview of important concepts in machine vision, machine learning and artificial intelligence.

Machine Vision Using Convolutional Neural Networks (CNNs)

An artificial neural network is a computer system that models the structure and functions of the brain that can be used for a machine vision task such as recognizing the object class of an input image. A CNN has groups of nodes interconnected with each other and organized into multiple layers as shown in Fig. 1. The input layer is known as the convolutional layer, in which digital images are preprocessed and many feature detectors are applied. Multiple convolutional layers may be added to the network to improve its training and learning rate. The second layer is the pooling layer, which ensures spatial invariance in the images. In other words, if an object in the image tilts, twists, or is oriented differently, then the feature detector will still be able to recognize them. The third layer is the flattening layer where the nodes are organized into a column of values for the final layer. The final layer is the fully connected layer, where all features are processed through the network and an image class is determined (Eremenko 2017). To summarize, a CNN will recognize images and classify them. Such a neural network is trained through forward and backward propagation algorithms, which will adjust the assigned weights on each connection. By executing many iterations of that process, a CNN will improve its performance and learn based on the data it is trained on.

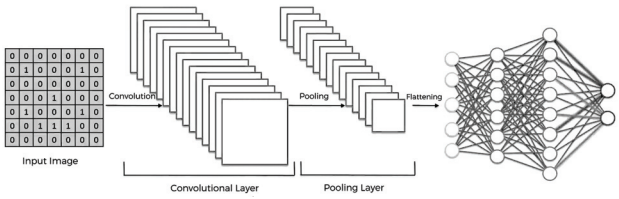

Fig. 1 Convolutional Neural Network Architecture

CNNs have become an effective technique for machine vision because of the architecture of pre-processing layers (i.e., convolutional layer and pooling layer) and improved computational power of parallel processing.

Reinforcement Learning

Reinforcement learning is a field of artificial intelligence that focuses on connecting situations to actions to maximize a numerical reward signal (i.e., Q-value). In other words, the learning model does not have a set of instruction to take specific actions, but instead it seeks to find out which actions produce the most reward through trial and error. Q-learning, a reinforcement learning algorithm, is used to learn an optimal policy by estimating the Q-value function using a multi-layer perceptron with one hidden layer (DeepMind 2017). A reward function maps each state-action pair to a single number, or reward. The learning agent's main purpose is to maximize the total reward. For example, a drone may receive a reward of plus one (+1) for finding and landing on a flat, safe landing zone. However, it may receive punishment of minus one (-1) for landing on rocky, unstable surfaces. A Q-value function specifies the total amount of reward a learning agent can receive overtime within a certain state (Dwibedi).

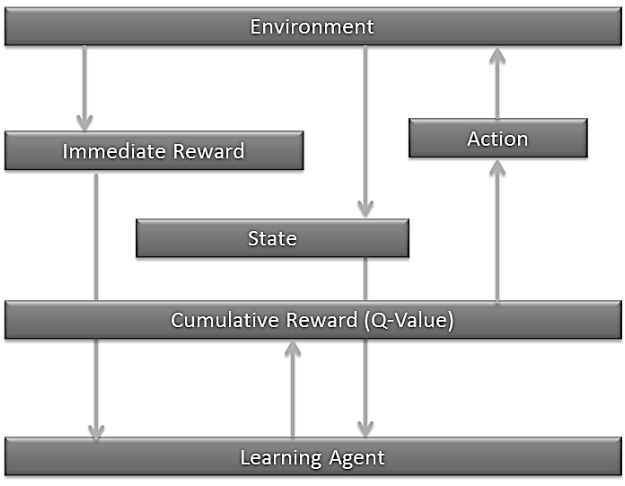

Fig. 2 Basic Reinforcement Learning Cycle

As shown in Fig. 2, the learning cycle of RL is 1) an agent observes some information (a state, e.g., images taken from the first-person view of a driving car); 2) the agent takes an action (e.g., turning left, right or going straight) based on its current state and internal knowledge models; 3) then the environment gives an immediate reward or penalty (e.g., continuing driving or crash). Reinforcement learning involves algorithms to learn, update or modify a cumulative reward Q-value. These cumulative rewards are learned by using the Q-learning algorithm (Watkins 1989).

Self-learning Using Deep Reinforcement Learning Techniques (Deep-RL)

Q-learning includes data structures consisting of environment states, actions, discount factor, reward, and state transition probabilities (Egorov 2016). Q-learning tries to learn the value of being in a given state and action. Deep Q-learning or Deep-RL use deep neural networks such as CNNs to learn Q-value and allow learning agents (Sallab), for example, a CNN can take a set of images as input and try to estimate a state-action value function.

Reinforcement learning differs from standard supervised learning because correct input/output pairs, i.e., ground truth, are never presented. Instead, learning is conducted through reinforced immediate rewards. Also it is a so-called self-learning based on reward from environment (Sutton 2017).

Initial Design of the Framework

Fig. 3 shows the initial design of our work. It contains a block diagram of the concepts of reinforcement learning (RL) and includes the Q-learning algorithm. The Q-learning algorithm does not require a model of the environment, instead, the Q-value measures a preference of a state-action pair at a given time step, in which the solution decides on the most preferred pair. The Q-value can be approximated with a deep neural network, which is known as a deep Q-network (DQN). Our design is to integrate multiple CNNs into a RL learning paradigm:

- CNN 1: serves as machine vision classifier, learning relations between input images and output object classes or relations between input images and output actions. For example, in Fig. 3, CNN 1 takes state (e.g., images, sensor data such as longitude, latitude, altitude and heading) and output object class (e.g., dogs, cats, trees or object labels).
- CNN 2 (DQN): serves as cumulative reward (Q-value) predictor, learning relations between input state, action and output cumulative reward. For example, in Fig. 3, CNN 2 takes state and action (e.g., turn left, right or go straight, stop, speed up or slow down, avoid collision) and output a cumulative reward (e.g., winning or losing of the whole games).

- CNN 3: serves as immediate reward predictor, learning relations between input state, action and output immediate reward. In Fig. 3, CNN 3 takes state, action and output immediate reward (e.g., crashes, continuation of flying or driving).

As explained earlier, CNNs require training data of input and output pairs. As shown in Fig. 3, examples of input and output training data for CNN 1, CNN 2 and CNN 3 need to be collected in a virtual environment such as in a video game. A standard CNN classifies input images into object classes. In RL, the Q-learning algorithm allows learning total reward (Q-value) from immediate rewards from the environment and the current state (observations).

The following steps and methods in our research framework are summarized as follows:

Step 1: Research basic CNN models that can be deployed as CNN 1, CNN 2, and CNN 3

Step 2: Collect training data for CNN 1, CNN 2 and CNN 3 from a virtual environment such as a video game

Step 3: Train and test CNN 1, CNN 2 or CNN 3 to the self-learning of an autonomous moving object in a virtual environment such as a video game.

Step 4: Research basic Q-learning models that can be deployed in this problem

Step 5: Apply Q-learning together with CNN 1, CNN 2 and CNN 3 to the self-learning of an autonomous moving object in a virtual environment such as a video game

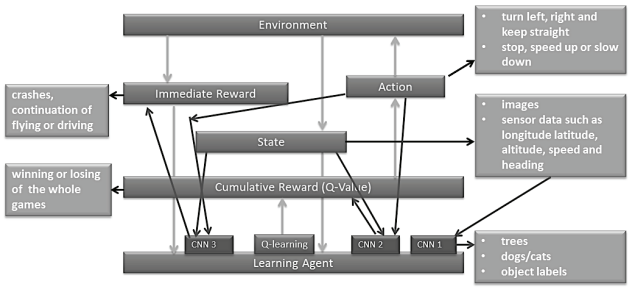

Fig. 3 Initial Design of Machine Vision and Deep-RL

Initial Results for Step 1: Basic CNNs Models

Supervised learning was used to train a learning agent –e.g. a moving object. In this section, we described basic CNNs can be trained using historical data and machine vision tools that can be useful in the framework. Data sets are images and corresponding classes. In previous research, datasets have been collected from Grand Theft Auto V in order to rapidly create pixel-accurate semantic class maps for these extracted screenshots from computer games (Ritcher 2016).

Once an image dataset is created, a CNN can be trained from these images. The CNN was created in a Python using Spyder, TensorFlow and Keras as machine learning libraries implemented in Python. Two datasets were tested.

- One dataset contained two image classes – i.e. cats and dogs. The CNN produced accurate results above 95% after 25 iterations during testing. This means that the CNN was consistently predicting new images of cats or dogs.
- Another test data is about the Electro-optical (EO) sensors. We tested the CNN with an unclassified sample data of EO images. The sample data contains a large collection of visible and IR imagery collected by the US Army Night Vision and Electronic Sensors Directorate (NVESD). It contains 207 GB of IR imagery and 106 GB of EO images along with an image viewer, ground truth data, meteorological data, photographs of the objects. The EO dataset contained nine image classes from Electro-optical (EO) sensors. Each image class contains 500 images. The CNN was able to classify the images at an accuracy of 99% accurate after the second iteration. A total of 25 iterations were performed and the test accuracy reached 99.53%. 75% of the dataset was used for the training set and 25% was used for the test set. To be specific, 375 images of each image class for training set, and 125 images of each image class for the test set. The baseline object recognition for this data was given using the method of representation learning through topic models (Flenner 2015). We also obtained similar test accuracy for the IR data set. The CNNs used for EO/IR classifications were used as the foundation and bases for other CNN applications in the framework.

Additionally, we also explored an open source machine vision library known as OpenCV to extract and pre-process image frames from a driving video. OpenCV could also detect lane lines in the road (Collie 2017). Before the images are used in the CNN, it must be preprocessed. To preprocess images, grayscale and Gaussian blur filters are applied to the image. Following the OpenCV documentation (OpenCV 2017), a region of interest was defined to focus on a specific area in the image. Next, the canny edge detection algorithm was used to convert the original image into binary image, which reduces the noise and highlights the edges around objects. Finally, the Hough line transform algorithm is applied to detect and highlight straight lines, which overlaps the lane lines in the road. The purpose of applying the edge detections is to correlate them to the initial actions in RL for autonomous control.

Initial Results for Step 2: Collecting Training Data

To collect training data from a virtual environment, we have researched the following video games:

- Grand Theft Auto 5 (GTA V) (Rockstar Games 2017) is a popular game and can be modified to represent military and security environment for data collection. A memory-less computer vision AI is shown playing GTA V (Sentdex 2017). The workflow and software the researchers used to capture the data are documented in (Roberson 2017).
- The military game with the helicopter is Arma 3 and VBS 3 (Bohemia Interactive 2017).
- Marine Corps and Naval Aviation training use the Aeschelon Image Generator (Aechelon Technology 2017).
- Neurala is the AI spotting poachers and elephants (Collie 2017).
- A google video tool running an analysis on videos and returning keywords was shown in (Ritman 2017). It could potentially tie into Lexical Link Analysis (Zhao 2015).
- Many of the sensors for military aircrafts and ground station would potentially have knowledge and data sets to share (FLIR 2017).
- The video game engine Unity can also create training data with ground truth based on video creation and human participation.

For our initial tests, Unity was used to create the environment. To apply machine vision and Deep-RL, we explored Udacity, an online educational organization, which open sourced a self-driving car simulator made from Unity for the purpose of machine learning. The self-driving car contains three cameras with perspectives from the center, left, and right of the car. The simulator contains a training mode and an autonomous mode. The training mode allows a user to virtually drive the car and generate data based on the user's driving. During data generation, images are captured from center, left, and right cameras correlated to steering angle, speed, throttle, and brake. The user manually controls the car with four keys (W, A, S, D) from the keyboard, "W" for forward acceleration, "A" for left, "S" for reverse, and "D" for right. The user must manually drive around the track for multiple laps up to five, which produces data for supervised learning (regression) of a CNN.
The CNN here is a regression model instead of a classification model that output is a predicted steering angle, instead of an object class. When the recording is finished, the data is saved as a csv file, which can be used in training the CNN. In supervised learning, a CNN model requires training data collected by recording images and the related measures, i.e., steering angles, throttles, speed and brake, of the car driving for five laps around the track. The user manually controls the car throughout those five laps.

Initial Results for Step 3: Training and Testing a CNN

In this example, the CNN only learns steering angles. The CNN was trained to use the images to predict steering angles based on the five laps of the training data. After the autonomous model (i.e., the CNN) has been trained, the autonomous mode allows the user to test the model.

During the training phase, the training data is loaded and split into training and validation set. The csv file is read into a single data frame variable, which allows the user to select rows and columns by their file names. The car images from the center, left, and right perspectives are saved as the input data. The output data is the steering angles.
The training and testing process are implemented in a server and client architecture. The simulator stays in the server and the CNN stays in the client. In the testing phase, the simulator sends test images (i.e., images taken from the test driving) to the CNN, the CNN predicts the steering angles based on the trained CNN, and sends predicted steering angles, together with specified speed (throttles depend on the speed) to the simulator. The simulator uses the predicted steering angles to simulate the test driving (Udacity 2017). It would allow for real-time data augmentation on images on the CPU in parallel to training the model on the GPU. Fig. 4 shows the Unity car drives in the autonomous mode using the predicted steering angles from the trained CNN.

Fig. 4 The Unity Autonomous Car Driving Using Trained CNNs and Predicted Steering Angles

Step 4 to 5: Discussion and Future Work

In this paper, we focus on virtual environments for these applications. While rewards from real-world environment are often expensive to measure or delayed, virtual environments typically will produce better data because a custom simulation allows the user to vary different weather conditions, types of terrain, landscapes, and civilization. Another advantage is that there would be no issues with camera equipment. An artificial moving object trained in a virtual environment would be safer and cost efficient.

Additionally, machine vision suggests the ability to search for specific objects in videos (Ritman 2017). The tool that is used to analyze videos and to make its contents searchable can be valuable for military applications, for example, searching for weapons or potential threats from surveillance videos. While part of our design is to generate machine learning training data from commercial simulation engines capable of simulating realistic environments such as day, color, and visible light scenes, the artifacts of blooming, blurring and latency are less well simulated in EO/IR scenes. Better EO/IR scenes might be obtained in the sensor footage libraries.

When the model is tested and drives autonomously, the model would be identical to the user's driving at best. We will perform Step 4 to 5 in the future. With Deep-RL, a virtual moving object might be able to fully self-learn from its own reward collected from its environment and drive better than a human operator.

Conclusion

The contribution of this paper is that we developed a framework using machine vision and deep reinforcement learning for self-learning moving objects in a virtual environment. We showed the initial results of applying the framework and created initial CNNs to classify EO images with high test accuracy, later the CNNs were adapted to train a Unity car learning from data collected from a human driver in a virtual environment. We also discussed various military applications that may benefit from the results of this research.

References

AACUS – Aurora Flight Sciences. (2017). Retrieved June 14, 2017, from http://www.aurora.aero/aacus/

Aechelon Technology. (2017, April 28). Retrieved August 1, 2017, from http://aechelon.com/

Bohemia Interactive. Retrieved June 5, 2017, from https://www.bistudio.com/

Bohemia Interactive Simulations. Retrieved June 5, 2017, from https://bisimulations.com/

Collie, S. (2017, May 22). Drones and AI combine to combat poaching in southern Africa. Retrieved June 20, 2017, from http://newatlas.com/neurala-lindbergh-foundation-drone-ai-poaching/49627/

DeepMind. Retrieved August 11, 2017, from https://deepmind.com/

Dwibedi, D. and Vemula, A. *Vision-based Deep Reinforcement Learning*. Carnegie Mellon University.

Egorov, M. (2016). *Multi-Agent Deep Reinforcement Learning*. Stanford University.

Eremenko, K., & Ponteves, H. D. Deep Learning A-Z™: Download Practice Datasets. Retrieved July 15, 2017, from https://www.superdatascience.com/deep-learning/

Flenner, A. 2015. Representation learning through topic models. NFCS NAWCWD, China Lake, in the 2015 National Fire Control Symposium.

FLIR Systems | Thermal Imaging, Night Vision and Infrared Camera Systems. Retrived July 28, 2017, from http://www.flir.com/surveillance/content/?id=63482

Knight, W. (2016). "Self-Driving Cars Can Learn a Lot by Playing Grand Theft Auto". MIT Technology Review.

OpenCV-Python Tutorials. Retrieved July 10, 2017, from http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html

Richter, S., Vineet, V., Roth, S., and Koltun, V. (2016). *Playing for Data: Ground Truth from Computer Games*. TU Darmstadt

Ritman, J. (2017, May 23). A new Google tool actually lets you search videos for specific objects. Retrieved July 28, 2017, from https://www.airsassociation.org/services-new/airs-knowledge-network-n/airs-articles/item/19686-a-new-google-tool-actually-lets-you-search-videos-for-specific-objects

Roberson, M., Barto, C., Mehta, R., Sridhar, S., Rosaen, K., and Vasudevan, R. (2017). *Driving in the Matrix: Can Virtual Worlds Replace Human-Generated Annotations for Real World Tasks?*

Rockstar Games. Grand Theft Auto V. Retrieved June 5, 2017, from http://www.rockstargames.com/V/

Sallab, A, Abdou, M, Perot, E, and Yogamani, S. *Deep Reinforcement Learning framework for Autonomous Driving*.

Sentdex, H. Self-Driving Car in GTA (Convolutional Neural Network). Retrieved August 2, 2017, from https://www.twitch.tv/sentdex

Shafaei, A., Little, J., and Schmidt, M. (2016). *Play and Learn: Using Video Games to Train Machine vision Models*. Cornell University.

Sutton, R. and Barto, A. (1998, 2017) *Reinforcement Learning: An Introduction*. MIT Press. 1998.

Udacity. Open Source Self-Driving Car. Retrieved August 5, 2017, from https://www.udacity.com/self-driving-car

Watkins, Christopher J.C.H.. Learning from Delayed Rewards (PDF) (PhD thesis). Kings College, Cambridge, UK. 1989.

Zhao, Y., Mackinnon, D. J., Gallup, S. P., Big data and deep learning for understanding DoD data. Journal of Defense Software Engineering, Special Issue: Data Mining and Metrics, 2015.