Deep Learning Triggers a New Era in Industrial Robotics

O
ne reason deep learning has attracted the attention of so many researchers and engineers, even outside of the AI community, is because it can capture abstract features and recognize patterns in ways many once thought impossible for computers. The breakthrough was exemplified by the emergence of AlphaGo. Prior to the successes of AlphaGo, experts had thought that the abstract strategical thoughts and theories that a human Go player develops over time through training and experience were not replicable by a computer—that is, without some technological breakthrough. AlphaGo proved that deep learning is indeed that breakthrough.

The pattern recognition capabilities of deep learning have pushed the limits in various fields—and industrial robotics is no exception. Deep learning will arguably not solve all of the problems we encounter in industrial robotics, but it will improve the perception capabilities of robotics systems, given its power to recognize complex real-world patterns robustly. Here, I examine some deep learning applications in robotics.

Automated Bin Picking

Let’s first consider the application of automatically picking steel cylinders with a suction hand, as shown in Figure 1a. Given a depth camera image as input, the robot system is expected to figure out a point within the bin at which the suction hand will most reliably suck a cylinder. Conventional systems usually tackle this task by matching the image with predefined photographs or CAD data (see Figure 1b). If the object’s appearance or shape cannot be predefined, conventional systems will often look for a planar surface larger than a certain area. Either way, experienced human operators must carefully tune parameters to achieve reliable results.

Researchers and engineers at Preferred Networks and FANUC have teamed up to demonstrate that deep learning could offer an alternative solution for this task. We use a depth camera image around a given suction hand position as input, together with the output representing whether that suction was successful. Through trial and error, we collect this input and output pair with the actual robot, initially starting with a random policy (see Figure 2). By training the deep neural network with thousands of these inputs and outputs, we have achieved 90 percent accuracy, which is comparable to a conventional system whose parameters are tuned by experienced operators. Furthermore, our deep learning approach doesn’t require us to redefine the object’s appearance or shape.

Amazon Picking Challenge

Another, more complex application that might benefit from deep learning involves picking items of various shapes and forms stored on shelves. The Amazon Picking Challenge is a competition held by Amazon that aims to “strengthen the ties between the industrial and academic robotic communities to promote shared and open solutions to some of the big problems in unstructured automation” (www.robocup2016.org/en/events/amazon-picking-challenge).

For the 2016 Challenge, competitors were instructed to pick items off a shelf and place them back again. As easy as this task is for humans, the industry has yet to see automated robots replace all human pickers. Of the 15
teams that advanced to the finals, two-thirds used deep learning\(^3\) in their system, including the top scoring teams. As an example of how deep learning is integrated into these competitive systems, I look at the work of last year’s winner, Team Delft—a joint team from the TU Delft Robotics Institute and the company Delft Robotics.

I spoke with former Team Delft member Wilson Ko to learn more about how they used deep learning in their object detection component to classify objects in a camera image and output the bounding box for each object (Figure 3). Although Team Delft’s success cannot be solely attributed to the use of deep learning, Ko asserted that properly integrating deep learning into conventional systems will bring us closer to fully autonomous, robust picking systems. In the following, each module in Team Delft’s pipeline (the process from recognizing the object to sending a motor command to the robot) is explained in detail, based on information provided by Ko and his former teammate Mihai Morariu.

Object Detection
Because deep learning has shown impressive performance during the last few years in addressing object detection problems, Morariu said that the team decided to use it over algorithms that rely on hand-crafted features. The Faster Region-based Convolutional Neural Network (Faster-RCNN) system was one of the most popular deep neural-network-based object detection systems when the team started working on their system. It had shown state-of-the-art accuracy when it was released on datasets such as Pascal’s visual object challenge (VOC) 2007 and 2012 and the Microsoft Common Objects in Context (MS COCO) dataset.\(^4\) It also had the advantage of running at near real-time frame rates, which was essential for developing a fast robotic system. The underlying idea behind Faster-RCNN is to use a fully convolutional network that generates high-quality region proposals (that is, bounding boxes that enclose an object and their “objectness score”).

Object Pose Estimation
The next step in the pipeline is object pose estimation. According to Morariu, the Team Delft system estimated the 6D pose of the detected object and matched a premade CAD model of the object against the real-time point cloud retrieved from the camera. Using the bounding box obtained from Faster R-CNN, only the relevant part of the captured point cloud was being matched. For deformable items, however, a premade CAD model is meaningless because the objects change in shape. Instead, object pose estimation was skipped and the system generated grasps directly on the filtered point cloud.

For the rigid items, Morariu explained that the team used the Super 4-Points Congruent Sets (Super 4PCS) algorithm\(^5\) to do the matching. The system can use the transformation that is found at the end of this process to estimate the object’s pose with regard to the robot. Finally, the system refines the object pose using the Iterative Closest Point (ICP) algorithm.

Grasp Generation
According to Ko, after the system determined the object’s pose, it used this information to generate the grasp poses, and the grasps were predefined for the CAD model in the

![Figure 1. A bin picking application of steel cylinders: (a) Given a depth camera image as input, the robot system is expected to figure out a point within the bin at which the suction hand will most reliably suck a cylinder. (Photo courtesy of Eiichi Matsumoto; used with permission.) (b) Conventional systems usually tackle this task by matching the image with predefined photographs or CAD data. (Photo courtesy of FANUC Corporation; used with permission.)](image-url)
The team used shape primitives to describe the object geometry and predefine the grasps. Because an estimate of the object pose was available, the predefined grasp poses could be transformed with the object pose, providing the robot with poses to move into so it could pick up the item. The system then scored the grasps and eliminated some based on reachability and robustness. For deformable items, the system used surface normals of the segmented point cloud to generate grasp poses.

**Motion Planning**

Ko explained that the team distinguished between two types of trajectories: offline and online motions. Offline motions were used for motions outside of the shelf, which could be pre-generated using RRT-Connect of MoveIt (http://moveit.ros.org), based on Rapidly exploring Random Trees, RRTs, and still be collision free, because it was assumed that the environment outside the shelf would remain static.

The robot used online motions to move inside the shelf, and such motions are variable because they depend on the target’s location and orientation. Online motions were split into different parts: approach, contact, lift, and retreat. Again, MoveIt was employed here for collision checking, with Trac-IK and RRT-Connect plugins for inverse kinematics and path planning.

To execute the motions, the team used the MotoROS driver from Robot Operating System (ROS)-Industrial and enhanced it to fit their needs.

**Inspiring Research**

Here, I introduce some promising research that will likely impact how industrial robot systems are designed in the near future.

**End-to-End Training**

Previously the “best practice” for designing robotics systems was to connect modular system components as a pipeline. For example, if...
you were to design a robot that serves coffee, you might build system components, such as a state estimator that realizes which phase of pouring the coffee the robot is in, a planner to designate the next action the robot should take, a controller to actuate the motors in the robot so that the desired action is achieved, and so on.

A similar approach might be applied to program a computer to play video games. Let’s say you want a computer to play the game of Pong. The program receives only the video image of the game, so you could build a module to figure out the abstract state of the player (for example, where is the ball and paddle?). You could build another module to plan what the player should do (in which direction should the ball be hit back, and where should the paddle be to achieve that?), and yet another module to decide the actual commands to input into the game (which button should the player press, if any?). Finally, you’d connect all these modules.

A group from DeepMind showed that you don’t need to go through the trouble of building all of these modules—by combining deep learning and some techniques from a field called reinforcement learning, you can directly train a computer program to output the game commands given visual images as input. DeepMind’s trained computer program has outperformed humans in some games.

The terminology “end-to-end” refers to methods that don’t require intermediate components in this sense. Naturally enough, the transition from “pipelines” to “end-to-end” is happening in the robotics field as well. One example is the work from the University of California, Berkeley, in which a robotic arm was trained end-to-end to perform tasks like inserting toy blocks into boxes.

Admittedly, with the current technology, end-to-end trained systems will not be as precise and accurate as conventional systems that are tweaked and tuned for a very specific task, such as controlling the position of a robotic arm. However, being able to teach abstract tasks (opening the cap of a bottle, for example) to robots end-to-end was not something experts thought was practical until a few years ago. With new technology, we might see robotic system designs radically reshaped in the near future.

Toward Robust Grasping
As exciting and astonishing as the state-of-the-art achievements of deep learning are, many will point out the caveats of the technology. For example, deep learning requires a large dataset for training, which is why many of those working on automated grasping have welcomed the release of Dex-Net 2.0 from the University of California, Berkeley. Dex-Net 2.0 contains
millions of datapoints to train a deep learning network\(^8\) that tells you the quality of a grasp with a parallel jaw gripper.

Collecting large amounts of data for deep learning reminds me of the work at Google Research, where they ran as many as 14 robots simultaneously over the course of two months to collect 800,000 grasp attempts.\(^9\) In contrast, Dex-Net does not rely on time and a vast number of robots to collect data. Rather, it exploits physics-based models so that grasping attempts can be synthesized instead of experimenting with the grasp in the real world.

For example, let’s say you want to pick up a cube with two fingers. You immediately choose to place your fingers on the two facing sides and not on any other combination of sides. Your intuition about your grasp quality is in line with what grasp analytics predict. Therefore, instead of spending time and using numerous robots to predict the outcome of every possible grasp, it makes sense to use our knowledge about grasp quality and inject it into the training dataset to train a deep neural network.

One impressive trait of these methods that use deep learning is their ability to adapt to data they didn’t see in the training phase—a capability is referred to as “generalization.” Even without deep learning, you could program a robot arm to grasp a specific object. However, that robot will usually have difficulties adapting to other objects of various shape, friction, or appearance without reprogramming. Deep learning relaxes that restriction, and it is exactly that capability that makes it a promising method with which to tackle many of the unsolved problems and challenges in industrial robotics.

Conventional industrial robotics was all about controlling and reducing the variance of the environment so that unintelligent robots could do their repetitive work. Perhaps this practice is acceptable if the system is for mass production and the initial investment in programming/teaching the robot to perform its repetitive work will likely be recovered in the foreseeable future. However, there are two issues that need to be addressed. First, there is a growing demand for automation in mass customization solutions (as opposed to mass production). Second, even in some mass production factories that will benefit from automation, we still see human workers performing repetitive tasks that are technologically challenging (requiring dexterity, for example) or not worth the investment to automate. In either case, deep learning is a promising technology for cultivating undeveloped areas and providing us with more robust, adaptive, and reliable systems.

References


Ryo Miyajima is an engineer at Preferred Networks. Contact him at ryo@preferred.jp.

This series of in-depth interviews with prominent security experts features Gary McGraw as anchor. *IEEE Security & Privacy* magazine publishes excerpts of the 20-minute conversations in article format each issue.

www.computer.org/silverbullet

*Also available at iTunes*