German Journal of Artificial Intelligence manuscript No.
(will be inserted by the editor)

Learning by enhancing half-baked AI projects

Ken Kahn · Niall Winters

July 2020

Abstract We have developed thirty sample artificial intelligence programs in a form suitable for enhancement by non-expert programmers. The projects are implemented in the Snap! blocks language and can be run in modern web browsers. These projects have been designed to be modifiable by school students. The projects involve speech synthesis, speech and image recognition, natural language processing, and deep machine learning. They illustrate a variety of AI capabilities, concepts, and techniques. The intent is to provide students with hands-on experience with AI programming so they come to understand the possibilities, problems, strengths, and weaknesses of AI today.

Keywords Project-based learning · Blocks programming · Snap! · Artificial Intelligence · Machine Learning

1 Declarations

1.1 Funding

The eCraft2Learn project is funded by the European Union's Horizon 2020 Coordination and Research and Innovation Action under Grant Agreement No 731345.

The Go Girl project is funded by the University of Oxford IT Innovation Seed Fund and has received subsequent funding from Goldman Sachs Gives.

1.2 Conflicts of interest/Competing interests

None

Department of Education University of Oxford 15 Norham Gardens Oxford, OX2 6PY E-mail: toontalk@gmail.com · niall.winters@education.ox.ac.uk

1.3 Availability of data and material (data transparency)

https://github.com/ecraft2learn/ai

1.4 Code availability

https://github.com/ecraft2learn/ai

2 Introduction

One very effective way of acquiring a deep understanding and appreciation of AI is by building AI programs. However, building AI programs can be difficult and time-consuming. These shortcomings can be significantly reduced, however, by providing learners with high-level building blocks and associated guides [6]. Here we present an additional way to enable learners to experience the construction of AI programs despite a lack of experience and a limited amount of time.

The idea is to provide a range of half-baked or minimal AI programs designed to be enhanced by learners [7]. As part of the eCraft2Learn project [5] and subsequently, we developed several such projects. All the projects were built upon the Snap! [3] programming blocks designed to support AI programming [6]. The programs illustrate several different AI concepts, techniques, and capabilities. While each one is able to perform a simple task or two, they can be enhanced by non-expert programmers to be more capable. In doing so, we expect that by engaging in how the programs work to improve them, the students will learn about the abilities, strengths, and weaknesses of AI today. At least one or two layers of technology underlying AI programs will stop being magical black boxes to the stu-

dents. We are currently at an early stage of evaluating these expectations with empirical studies.

Some of the projects illustrate ways of creatively using speech synthesis, speech and image recognition, natural language processing, and deep machine learning. Others explore innovative ways of applying pre-trained deep learning models to tasks. The variety of projects maximises the chance that a project will fit with a student's interests and goals.

3 Some half-baked projects

The complete list of thirty projects can be found at the eCraft2Learn AI home page [4]. Descriptions of a sample of these projects follows.

3.1 Speech commands

We have developed a variety of projects that illustrate using speech to control an application. Typically these commands control the movement of graphical sprites. The simplest projects respond to single words or fixed phrases. One project creates the illusion that one can give full sentence commands. It works by searching for a keyword and a number somewhere in the sentence. E.g. "Could you please go forward 25 steps" is interpreted as "MOVE 25". Several of these projects also use speech synthesis to support a pure voice interface.

The mechanism of speech recognition varies between projects. One project uses the Web Speech API [9] supported by several browsers. It is the most reliable and flexible method for recognising speech. This reliance upon a web service, however, prevents the recognition functionality from being more than a black box to students. Furthermore, it requires a fast reliable Internet connection and raises privacy issues. Another project instead relies upon Snap! blocks that we provide for training the system to recognise a handful of words. Yet another takes advantage of Google's Teachable Machine [2] where students can create and train audio recognition models that are then imported into their Snap! Projects. Both of these approaches rely only upon the student's computer to train and run the model and hence protect their privacy.

When students train their own speech recognition models they are quickly exposed to the imperfections of recognition. If trained with samples only from one person it may not be very reliable with other speakers. They can discover that with more and varied samples the recogniser becomes more robust. If they increase the number of words their models can recognise they will discover the system's accuracy drops. Working with their trained model they become exposed to the notion of "confidence scores" that indicate the relative certainty of correctly classifying audio input. Finally, they need not limit their projects to speech but can explore the recognition of different sounds.

3.2 Interactive sentence and story generators

Two projects use speech input and output to interactively fill-out a sentence or story template. Experience with older textual versions of this is that students discover that without careful attention to parts of speech and grammar rules ungrammatical sentences result. These projects give students first-hand experience with the construction of voice-only interfaces.

3.3 Training a system to distinguish between different categories of images

Projects include a rock, paper, and scissors game and a drawing program that is driven by gestures. As with the speech recognition projects, the students encounter the need for a sufficient number of varied inputs for trained models to be accurate enough for their projects. They may acquire some degree of understanding how popular apps that recognise people, objects, and gestures work internally. And reasons why these apps make mistakes or exhibit biases.

3.4 Using body pose and segmentation to provide augmented reality apps

This includes a project that detects if someone is touching his or her face and another where virtual balloons are popped with real video hands. These projects may contribute to an understanding of how gesture-based games such as those that use Microsoft's Kinect [11] work and how filters are added to images in apps such as Instagram and Snapchat.

3.5 An app that generalises mathematical relationships based on examples

This is the simplest of the projects that rely upon creating, training and evaluating deep neural networks. Unlike projects that rely upon real-world data the goal here is to build a trained model that can approximate a mathematical relationship as simple as doubling or the more challenging square root. While enhancing this project students may learn the need for larger and deeper

models for certain kinds of relationships. And how the number of samples and training cycles affects the quality of the predictions. Students may be surprised to discover that neural networks can quickly learn to approximate very well a relationship like doubling and yet fail to produce exact outputs.

3.6 A program that learns to play better Tic Tac Toe

This app is an example of how a neural network can be created, trained, and then used to guide game play. Learning to play a game typically involves more advanced machine learning methods such as deep reinforcement learning. As a much simpler alternative we provide a framework where the training is in order to predict the probabilities of winning given the current board for each possible move. One can engage with fundamental concepts such as the trade-off between exploiting one's current knowledge and exploring new things to increase one's knowledge. This project also illustrates issues in how to encode something like a game board into a list of numbers a neural network can process. One can learn the strengths and weaknesses of learning from self-play.

Note that by providing students with a functioning Tic Tac Toe implementation students can focus their energies and attention on getting the computer to play it well.

3.7 An app to predict how one rates abstract art

It uses deep machine learning to predict how a user will rate randomly generated images. Each image is generated from a dozen random numbers and the program learns to associate lists of image generating numbers with user provided ratings. The trained model can then be used to "recommend" new images similar to how many familiar recommender systems work.

3.8 Three very different ways to create a question answering app

One project fetches answers from web services, another extracts answers from passages of text, and the third one is trained to recognise paraphrases of questions with known answers. Experience enhancing any of them should help demystify conversational agents such as Alexa and Siri. Exploring three approaches to question answering should reveal their different strengths and weaknesses.

3.9 An app that learns to determine how much confidence is revealed in a text passage

This is an example of emotion detection that relies upon sentence encoding. This simple example of sentiment analysis can be revised by students to categorise other sentiments such as worry, fear, optimism, etc.

3.10 An app that learns to give names to random colours

This app provides a simple example of a categorical deep learning classifier. Given three numbers corresponding to red, green, and blue it tries to match it to one of the colour names it was trained with. Students can learn how the number of colour names, number of samples, model architecture, and training regime influence accuracy.

3.11 An app that transfers the styles of famous artists to new images

A fun example of combining the learned style of an artist with a new image. It provides an example of AI generated art by mashing up two different kinds of inputs in a high-level manner.

3.12 Word guessing games that rely upon word embeddings

Word embeddings map words into a high-dimensional space (300 in our implementation). Words that are closely related are close in this space and unrelated words are far apart. Here a very simple game uses word embeddings to provide clues to a player trying to guess a word. This and other uses of word and sentence embeddings expose students to the idea of distributed meanings. Other projects could be created to illustrate very different uses such as finding word analogies or doing translations.

4 Preliminary results from trials

We have introduced our half-baked projects to over a hundred students in several workshops. However, with only three hours for introducing the tools, libraries, and sample projects, little time is left for deep engagement or significant enhancements. While most of our projects have yet to be enhanced by students, a few that have include:

1. Taking a spoken command app and a pose detection app and creating a drawing program that responds to spoken commands to change the pen colour and line thickness. Drawing is controlled by moving one's hand in the air.
2. Another student changed the spoken command app into a spoken version of the classical textual adventure games.
3. Starting with an app that controlled the movement of a sprite to the left or right by pointing, a young student added two more directions.

Students were proud of what they created even when most of their project was constructed by us. In some cases, with young students (8 to 12 years old) in a 3-hour workshop, it is hard to imagine how they could complete a project in any other manner. A few of the projects were multiple-day efforts by older students (14 to 16 years old) who became very engaged in changing the seed project to make something that was authentically theirs.

5 An online synchronous course to enhance the Don't Touch Your Face project

We recently completed teaching a course consisting of 13 one-hour Skype sessions. The participating three young women from non-traditional academic backgrounds are being supported by the Go Girl Project [10]. They all had some experience with Scratch [8].

Each session consisted of the girls presenting their homework followed by one of the authors (Kahn) introducing them to some new technology and concepts followed by the students doing an exercise. The first half of the course involved a combination of a general introduction to AI and an introduction to the new AI blocks we have added to Snap!.

The Don't Touch Your Face app was inspired by donottouchyourface.com which uses a webcam to determine if the user is touching his or her face and if so issues a warning. Our project relies upon Posenet [1] which quickly reports the locations of up to 17 face and part parts. The half-baked version issues a warning when either wrist (hands are not tracked) comes close to either eye or the nose (the mouth isn't tracked).

All the students fine-tuned the threshold for deciding what "close" means as well as the threshold for the minimum confidence score that a body part's location is known by the model. They changed the artwork for the hands, eyes, and nose. They changed the message when touching to include how many seconds since the start of the touch. They added sounds that they recorded to the app. One student added a mouth, ears, and the rest of the face despite the fact that the system did not report their locations. So she made the mouth's location be an offset from where the nose is.

Fig. 1 The Don't Touch Your Face app turned into a "filters" app

The students discovered that the app had been tuned to work well only when someone was the appropriate distance from the camera. Because the threshold was defined in terms of pixel distance it produced false positives when one is far from the camera and false negatives when close. The students followed the suggestion that they compute the threshold as the distance between the eyes times a tunable parameter.

A surprising outcome happened after they enhanced the app to display the camera feed in the background. The sprites for the nose and eyes were displayed on top of the video. They drew funny eye glasses, earrings, and noses that became "filters".

6 Conclusion

We have designed thirty half-baked AI projects in the Snap! programming system. They were designed to be explored and enhanced by students. We hope to learn more about the pedagogical effectiveness of half-baked AI projects as we continue our research. We encourage others to make use of our AI learning resources in their teaching and research. It is open source and Creative Commons license. Over the coming years we hope to see many creative remixes of our projects. And we expect that the students who created these enhancements will, in the process, have acquired an appreciation and understanding of AI.

References

1. Google. Posenet, 2020.
2. Google. Teachable machine, 2020.

3. Brian Harvey, Daniel D Garcia, Tiffany Barnes, Nathaniel Titterton, Daniel Armendariz, Luke Segars, Eugene Lemon, Sean Morris, and Josh Paley. Snap!(build your own blocks). In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 759–759, 2013.
4. Ken Kahn. ecraft2learn ai home page, 2020.
5. Ken Kahn, Calkin Suero Montero, and Christian Voigt. Steam learning in formal and informal settings via craft and maker projects. In *Proceedings of the 17th ACM Conference on Interaction Design and Children*, pages 728–733, 2018.
6. KM Kahn and Niall Winters. Ai programming by children. In *Proceedings of Constructonism 2018 Conference*. Constructionism 2018, 2018.
7. Chronis Kynigos et al. Half-baked logo microworlds as boundary objects in integrated design. *Informatics in Education-An International Journal*, 6(2):335–359, 2007.
8. John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, 10(4):1–15, 2010.
9. Mozilla. Web speech api, 2020.
10. University of Oxford. Go girl, 2020.
11. Wikipedia. Kinect, 2020.