Oui, Chef!!: Supervised Learning for Novel Gameplay with Believable AI

Gabriele Cimolino
Queen's University
Kingston, ON K7L 3N6 Canada
gabriele.cimolino@queensu.ca

T.C. Nicholas Graham
Queen's University
Kingston, ON K7L 3N6 Canada
nicholas.graham@queensu.ca

Sam Lee
Queen's University
Kingston, ON K7L 3N6 Canada
12sl5@queensu.ca

Quentin Petraroia
Queen's University
Kingston, ON K7L 3N6 Canada
13qp2@queensu.ca

CHI PLAY EA '19, October 22–25, 2019, Barcelona, Spain.

ACM ISBN 978-1-4503-6871-1/19/10.
https://doi.org/10.1145/3341215.3358247

Abstract
The use of artificial intelligence for the creation of game agents is fundamental to digital game development, enabling the design of new styles of games offering novel play experiences. Machine learning has long been used in the creation of digital games, most often for the purpose of creating game agent controllers that are trained off-line and do not learn during play. Little attention has been given to the possibility of game designs or mechanics that use machine learning to allow the player to train game agents on-line. This document outlines the design and implementation of *Oui, Chef!!*, a supervised learning game in which the player trains neural networks to map recipes to the ingredients necessary to make dishes in a restaurant. The game demonstrates that training game agents in a supervised manner provides a fun and engaging player experience wherein the effects of training are easily recognizable, and sometimes delightfully surprising.

CCS Concepts
•**Computing methodologies** → **Supervised learning;**
•**Applied computing** → **Computer games;**

Author Keywords
Game design; Machine learning; Machine learning game; Supervised learning game

Introduction

Since their inception, digital games have employed artificial intelligence (AI) to provide the player with seemingly intelligent opponents or allies. Initially, AI was developed to competently play classic games, such as Chess, as a substitute for a human opponent [7]. This enabled the creation of digital games wherein a single human player competes or collaborates with AI agents. Agents developed for this purpose rely heavily on early game AI techniques, often producing predictable behaviour with little sophistication. These are good qualities for agents whose purpose is to serve as a problem for the player to solve, but bad for providing richer interactions that give the illusion of a believable intelligence that is fallible and adaptable. Novel game designs featuring agents that learn from the player could provide more immersive player experiences by changing agents' behaviour in believable ways.

For instance, in real-time strategy games, players can typically issue orders to their AI confederates of the form "**Go here** *and* **do this**, *then* **go there** *and* **do that**." These agents might use pathfinding algorithms to efficiently navigate between destinations, but lack any understanding of the player's goals. This necessitates the player laboriously micromanaging agents' behaviour, creating a gameplay experience that could be considered tedious and boring. The development of more sophisticated game agents, and game designs that enable their use, has the potential to provide the player with richer AI interactions via more robust and less predictable behaviour.

There has long been a desire to bridge the gap in sophistication between academic AI and games industry AI, but few games have attempted to apply these techniques to game agents [7]. Machine learning algorithms in particular provide a means to create believable AI [7], although there has been little exploration of this possibility due to the lack of well-known game designs that incorporate learning agents. This is partly due to restrictions imposed by stock gaming hardware, which can make the on-line training of game agents impractical. Approaches that have been tried involve the off-line training of game agent controllers that remain static at runtime [3], doing no further learning and producing behaviour that may be predictable by players. Although some of the more popular methods of producing game agent controllers can be adapted to learn on-line, responding to situations brought about as a result of the game actions taken by the player, there has been little exploration of this possibility. This is likely because using machine learning to produce game agent controllers requires many training samples that describe a wide range of possible situations and appropriate responses in high-dimensional observation and action spaces [3]. On-line learning also requires sufficient computational resources to train the agent in real time, and requires the agent's behaviour to change rapidly enough to suit the pace of gameplay.

If these issues could be overcome, then new styles of digital games focused on training agents would become possible, and games in established genres could benefit from game agent controllers that adapt to the user's play to produce richer player experiences due to the depth in the agent's responses. There has long been a desire among researchers and gamers for believable agents that appear to adapt to user interaction, behaving in ways that permit the user to suspend their disbelief in the agent's consciousness and attribute human emotions and reasoning to them [4]. In search of a method to incorporate real time training of game agents in digital games, we have designed and implemented a supervised learning game called *Oui, Chef!!*,

demonstrating how machine learning can afford novel gaming experiences.

Related work

It was not until the release of *Creatures* [1] in 1997 that digital games began to incorporate machine learning techniques as a means of adapting game agent behaviour to respond to user input [4]. *Creatures* employed a reinforcement learning strategy using deep networks of integrate-and-fire neurons, organized into Lobes responsible for different cognitive functions, to train the game's Creatures to exhibit the player's desired agent behaviour when presented with an Object in the environment or a command given by the user [5]. This approach enables the game's Creatures to learn correct behaviour given feedback from the environment and from the player, in the form of stroking or slapping for a positive or negative reward signal respectively.

The game *Black & White* [6], released soon after in 2001, features a similar operant conditioning mechanic where the player is responsible for training a game agent called a Creature to take desired actions toward the game's villagers and enemy agents [2]. The behaviour of these agents is determined using a hybrid representational architecture that makes use of simple perceptrons and decision trees to create Desires and Opinions as part of a Belief-Desire-Intention model. Much like *Creatures*, *Black & White*'s Creatures are trained using a reinforcement learning approach that modifies the agent's desires and opinions according to feedback from the environment and the player.

These approaches to the on-line training of subordinate game agents create separation between the player and the world. This leads to a novel gameplay experience in which the player can see the effects of their influence without requiring direct control or the communication of formal intentions for their minions' behaviour. Minions appear to have their own goals, informed by their handler's training.

Supervised learning as a novel game mechanic

What work has been done is still far from a complete exploration of the possible approaches to integrating machine learning into digital games as a central game mechanic. Recent advances in machine learning techniques and computer hardware have made it possible to design agents that have fewer limitations than those used in earlier machine learning games. This has enabled us to create a supervised learning game that uses a richer form of learning than has been seen previously, providing a more engaging training experience for the player.

Oui, Chef!! is a supervised learning game, with a restaurant management theme, that tasks the player with training cooks, each with their own neural network, to correctly select the ingredients necessary to prepare customer orders. The player's goal is to make enough money each week to pay their cooks' wages and the restaurant's operational costs.

During kitchen management (Figure 1), the player has 5 minutes to make as much money as possible by fulfilling customer orders. After an order is delegated to a cook, the player has the opportunity to reject their cook's preparation, if their ingredient selections are not satisfactory, or send the order out, whether it is correct or not.

Order delegation

Upon receiving an order, the cook navigates to the fridge to gather ingredients before preparing their selections at the various preparation stations. The kitchen features three of these preparation stations used to prepare different ingredients: the Mixer, the Cutting Board, and the Stove. Once the

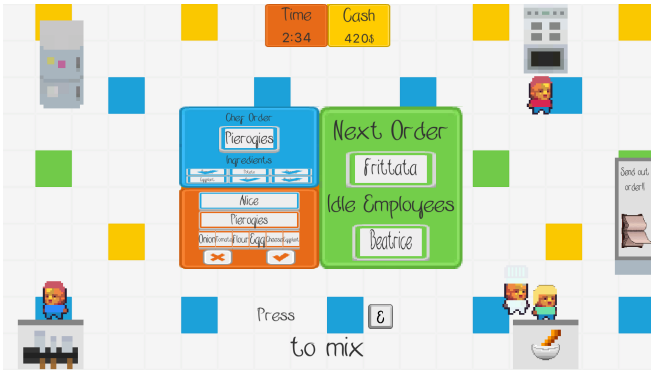

Figure 1: The *Oui, Chef!!* kitchen management screen. The player character, Master Chef, and a cook can be seen at the bottom right, next to the Mixer. Two other cooks can be seen using the Cutting Board and the Stove. The cook closest to the player, Alice, is presenting her preparation of pierogies, in the orange panel in the center, for verification. The player must decide whether to send the order out as is or train Alice to make better ingredient selections and remake the order.

cook's selected ingredients have been prepared, the completed order is then presented to the player for verification.

Order verification

At this restaurant, if a correct ingredient is substituted for something else, the customer pays only part of the menu price for their order. This forces the player to consider whether it is preferable to have a cook who has incorrectly chosen some ingredients remake the order, taking twice as much time or more, or send the order out as is, losing the opportunity to improve its performance. The player must strike a balance between training their cooks, investing in their long term usefulness, and generating revenue, contributing to their short term goal of staying open. The verification mechanic puts the player in the role of supervisor by providing a simplified and intuitive interface for correcting their cooks' behaviour. This leads to a novel form of play, wherein the player is engaged in both delegating tasks and overseeing their execution.

Figure 2: The Cookbook encodes the synthetic data set that cooks are tasked with learning. It has 8 recipes in total, organized into 3 tiers. Lower tier recipes are combined to make recipes in higher tiers. For instance, the recipe for Lasagna is Spaghetti and Eggplant Parmesan.

Practical implementation of supervised learning as a central game mechanic

The fully playable *Oui, Chef!!* provides an example of a performant implementation of supervised learning in a digital game. It requires a data set to learn, a model to train, and a means of training.

Cookbook

The Cookbook (Figure 2) lists the menu options available at the player's restaurant and the synthetic data set that the player's cooks are tasked with learning. It is structured into three tiers of menu options with 2, 4, or 6 ingredients required to prepare the dish. Dishes from tiers 2 and 3 are comprised of combinations of the ingredients required to make dishes from lower tiers. This enables the player's

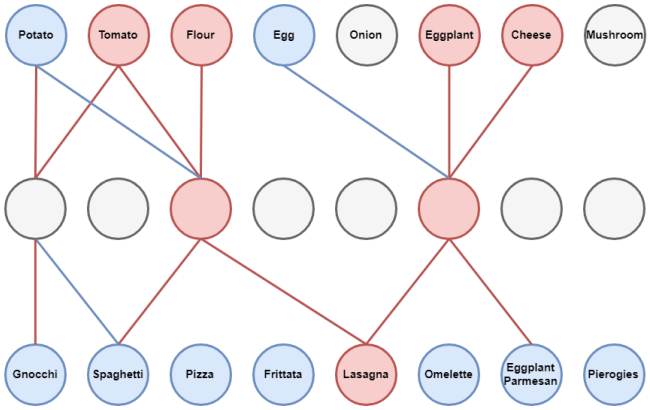

Figure 3: The cook network structure and example weights for a cook who has been trained to prepare several of the first tier recipes and lasagna, a second tier recipe which is comprised of two of the known lower tier recipes. Connections and neurons are colored to indicate the value of weights and activations, with red for positive and blue for negative.

cooks to exploit previously learned associations between commonly paired ingredients to quickly learn more complex recipes.

Cook agent architecture
Each cook's ingredient selection network is structured as a homogeneous simple feedforward neural network with 3 layers of 8 neurons (8-8-8). The network's weights are unbounded within $\mathbb{R}$ and units in the network's hidden layer use hyperbolic tangent for their activation function, while units in the output layer use sigmoid.

A cook selects ingredients by feeding the order's one-hot vector encoding through its neural network. Ingredient selection is a multi-label classification problem wherein the desired number of labels is known to the agent and that number of its network's most activated output neurons is taken to be its classification. The correct subset of available ingredients required to prepare a dish must be learned by modifying the cook's network's weights during training. This means that initially a cook will make random ingredient selections, determined by the randomly initialized network weights or any prior training, when presented with an unknown order vector encoding.

Training
Completed orders are presented to the player with the option to accept or reject the ingredient selections. When the player finds that a cook's preparation of a given dish is not satisfactory, they can choose to reject it,training the cook to make better ingredient selections next time, and causing it to remake the order, rather than send out the imperfect preparation. To train the cook, their ingredient selection network is corrected using backpropagation to minimize the sum of squared errors between a vector encoding of the cook's ideal selections and the network's actual output. Depth in the cooks' network structure, as well as the Cookbook's design, enable cooks to build up associations between commonly paired ingredient selections in the weights leading to a single hidden unit.

While depth enables cooks to leverage previously learned associations to quickly learn new recipes, learning has the potential to cause cooks to forget other recipes. Just as a novice player might forget the ingredients required to prepare an order, training an inexperienced cook may cause it to forget what it has previously learned. Similarly, an expert player may never forget the Cookbook's recipes; and a well trained cook may achieve perfect recall across all orders. The similarity between how the player and the cooks learn allows the player to intuitively recognize their cooks abilities and limitations.

Forgetting also makes cooks' behaviour less predictable,

and thereby more believable. In this way, training a cook mimics the path of learning the player follows from complete ignorance to absolute knowledge of the recipes in the Cookbook. The isomorphism between how the problem of learning the Cookbook's recipes is presented to the player and to the cooks allows the player to intuit how training will affect their cooks performance over the course of gameplay.

Experience

The *Oui, Chef!!* design has been implemented using the Unity game engine, and has been demonstrated to hundreds of participants at Queen's University's Creative Computing showcase where attendees had the opportunity to create their own restaurant. The game is available from itch.io (https://anna-mae.itch.io/oui-chef) where it has been played in-browser several hundred times, and can be downloaded as an executable for Windows, Mac, and Linux.

Players expressed their enjoyment of training cooks, largely due to the unpredictable effectiveness of any given instance of training, and their surprise at how quickly their cooks are able to learn the Cookbook. The dynamics of ameliorating a cook's selections through backpropagation creates a gameplay experience that lends itself well to an anthropomorphic understanding of cook behaviour, adding depth to the player's experience of *Oui, Chef!!*.

The playable implementation demonstrates how a game mechanic based on supervised training of agents provides a fun and engaging player experience that would not be possible without the use of machine learning. The simple training interface presented to the player, the choice to accept or reject a cook's preparation, allows players unfamiliar with machine learning to reason about training their cooks in an intuitive way. This enables the player to easily begin training their cooks and quickly see the effects of training, giving the impression that cooks are living beings that learn as a result of the player's guidance.

REFERENCES

1. Creature Labs. 1996. *Creatures*. Game [Windows]. (November 1996). Mindscape, Novato, U.S.. Last played May 2019.
2. Richard Evans. 2002. Varieties of Learning. In *AI Game Programming Wisdom*. Charles River Media, Inc., Rockland, MA, USA, 567–578.
3. Leo Galway, Darryl Charles, and Michaela Black. 2008. Machine learning in digital games: a survey. *Artificial Intelligence Review* 29, 2 (April 2008), 123–161. DOI: http://dx.doi.org/10.1007/s10462-009-9112-y
4. Stephen Grand and Dave Cliff. 1998. Creatures: Entertainment Software Agents with Artificial Life. *Autonomous Agents and Multi-Agent Systems* 1, 1 (March 1998), 39–57. DOI: http://dx.doi.org/10.1023/A:1010042522104
5. Stephen Grand, Dave Cliff, and Anil Malhotra. 1997. Creatures: artificial life autonomous software agents for home entertainment. In *Proceedings of the first international conference on Autonomous agents - AGENTS '97*. ACM Press, Marina del Rey, California, United States, 22–29. DOI: http://dx.doi.org/10.1145/267658.267663
6. Lionhead Studios. 2001. *Black & White*. Game [Windows]. (30 March 2001). Electronic Arts, Redwood City, U.S.. Last played May 2019.
7. Georgios N. Yannakakis and Julian Togelius. 2018. *Artificial Intelligence and Games*. Springer International Publishing. https://www.springer.com/gp/book/9783319635187