5-1-2018

AI-Human Collaboration Via EEG

Adam Noack
College of Saint Benedict/Saint John's University, adamnoack1@gmail.com

AI-Human Collaboration Via EEG

An All-College Thesis

College of Saint Benedict/Saint John's University

by Adam Noack

April 2018

Project Title: AI-Human Collaboration Via EEG

By: Adam Noack

Approved by:

Mike Heroux

Professor, Department of Computer Science

Ben Faber

Professor, Department of Psychology

JA Whitford Holey

Professor, Department of Computer Science

Imad Rahal

Chair, Department of Computer Science

Director, All College Thesis Program

Abstract

As AI becomes ever more competent and integrated into our lives, the issue of AI-human goal misalignment looms larger. This is partially because there is often a rift between what humans explicitly command and what they actually mean. Most contemporary AI systems cannot bridge this gap. In this study we attempted to reconcile the goals of human and machine by using EEG signals from a human to help a simulated agent complete a task.

Contents

1 Introduction

The range of environments in which most modern machine learning algorithms are competent is limited. For example, a machine learning algorithm trained to assign labels to images will fail miserably at chess, mowing the lawn, fetching the newspaper, etc.

The narrowness that has hitherto defined AI algorithms is beginning to dissolve, however. For example, DeepMind's AlphaGo Zero algorithm recently mastered chess and Go with nearly identical initial code bases [19]. AI is becoming ever more general purpose and powerful, and it is only a matter of time before algorithms such as AlphaGo Zero are repurposed to solve problems and operate in the real world. This is exciting. The possibilities are nearly endless.

There may be problems with this new technology, however. Because many of these state of the art machine learning architectures are so complex, their inner workings are difficult if not impossible for humans to fully understand. The only thing one can virtually guarantee about their behavior is that they will attempt to maximize the reward function given to them by their creators.

In that case, it will be made certain that the AI systems are only given good goals. The problem is that creating goals that are good may be incredibly difficult.

Take the following scenario. It illustrates how a seemingly benign directive can lead to unwanted consequences. Imagine that we tell an advanced AI bot to "get us a cup of coffee from the kitchen." The goal of the bot in this case is to bring the coffee to us. Its reward is maximized by successfully bringing us the coffee. Instead of simply walking to the kitchen and grabbing the cup of coffee for us as a human might do, the bot takes some unexpected precautions. To bring the probability of achieving its goal as close to 1 as possible it immediately disables its own stop button (cannot fetch the coffee if it is turned off), kills the pet dog (some small percentage of the time the bot trips over the dog and spills the

coffee), ties us to our chair (the probability that the coffee will successfully be delivered to the human increases if the whereabouts of the human are fixed), etc. This is an absurd example, but it conveys the difficulty present in supplying the bot with a good goal.

Nick Bostrom, a Swedish philosopher and futurist, attributes this difficulty to the fact that "intelligence and final goals are orthogonal"[3]. In other words, the amount of intelligence a system possesses is independent of the quality of goals it has. Therefore it is possible to accidentally or purposely assign a goal to even a *super*intelligent AI system that cuts directly against the interests of humanity.

There is often a rift between what humans explicitly say and what they mean. Humans can easily bridge this gap because we have evolved the ability to efficiently trim away all of the irrelevant information present in any message and extract the meaning. Machines, on the other hand, do not perceive the world in the way we do. Their values and goals are given to them explicitly by humans. Therefore, if we are not careful in defining these values, that is, if we do not think extremely carefully about all of the implicit subgoals we have when giving an AI system its directive, the way in which the AI goes about attaining the goal might run totally contrary to values we forgot to make explicit.

A naive approach to instilling AI systems with our true values involves making everything we care about in any given situation explicit. This would be practically impossible, though. Enumerating every value and subgoal we have for any particular task and translating the list from words into ones and zeros would take a large amount of time [3].

One way to align the goals of an AI system with our true intentions involves including human neurological markers in the reward function of the AI. Our inner state could serve as a compass for the AI system, pointing it away from situations related to states of being that we dislike. Instead of having to articulate how we feel about every given outcome and translate the description into code that the AI system can understand, our emotional state could serve as a proxy for our implicit value structure.

2 Background

The revolutionary AI systems of the modern age have been driven in large part by a few key machine learning paradigms. The following subsections outline some leading approaches: deep learning, reinforcement learning, and deep reinforcement learning.

2.1 Deep learning

In the past 12 years, AI and machine learning have again become popular. Many of the recent successes can be attributed directly to deep learning. The algorithms behind these deep learning successes are not new, however. In fact, much of the theory has been around since the 1940s. Over the years, deep learning has gone by many different names: cybernetics

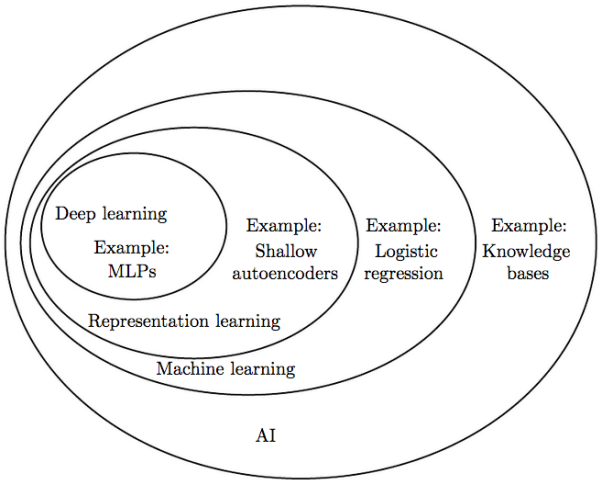

Figure 1: Deep learning's relationship to AI and ML. [4]

in the 1940s through the 1960s, connectionism in the 1980s and 1990s, and finally, deep learning as we know it today since 2006 [4].

Deep learning architectures are a subset of the class of machine learning models that rely on representation learning to master tasks (Figure 1). In general, machine learning algorithms allow a model to learn a solution to a problem without the solution having to be

explicitly coded by the programmer. Unfortunately, most classical machine learning models need handcrafted input features in order for the model to achieve the desired results. This issue arises from their inability to handle input that is high dimensional (i.e. input that has many features).

To get an intuitive idea of why this is so, imagine for a moment how an instructor might teach a student to do a math problem. Most likely, if the teacher just threw a math textbook in front of the student and told him to learn how to integrate a function, he would be lost (assuming he had little prior knowledge of the subject). The size and dimensionality of the input space (the entire textbook) would be too large for him to understand. A more efficient way to teach the student might involve the instructor pointing out specific paragraphs in the text that the student should read, giving him explanatory examples, helping him through a few problems, etc. When performing these actions, the instructor is simplifying the input material by curating it to the student's specific needs and lacks. Viz., the dimensionality of the input space is being lowered. Most classic machine learning algorithms are like this student. Their inputs need to be designed in such a way that make sense to them (and usually, "what makes sense to them" is difficult to determine). This process is called feature creation and it is cumbersome and time-consuming. Deep learning offers a solution to this problem. It relies on representation learning [2] to handle complex input. Representation learning is the process of taking raw, high-dimensional input and composing progressively more abstract, lower-dimensional representations of the original input features while retaining all pertinent information (Figure 2). In other words, it distills the essence from the original data and maps this core meaning to the predicted output. Representation learning allows deep learning models to make sense of incredibly large and complex input without the need for laborious feature-creation.

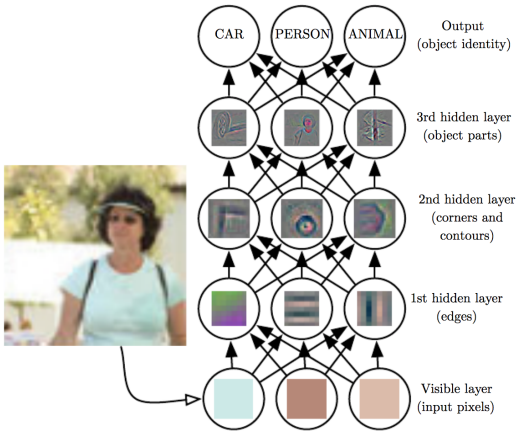

Figure 2: Progressively more abstract representations of the input are learned.

2.2 Reinforcement learning

Reinforcement learning has many pseudonyms as well and has been of interest in a number of different domains such as psychology under the title of operant conditioning, economics with the designation of game theory, and math under the heading of operations research [17]. The goal of reinforcement learning is to learn how to make optimal decisions in a given environment by learning from experience.

The learning process for a reinforcement learning agent can be summed up as follows. An agent takes an action a in a state s and ends up in state s' with reward r relative to some goal (Figure 3). It then stores this experience in a tuple: $e = (s, a, r, s')$ [17]. After gathering enough of these experience [9], and assuming it has experienced all possible state configurations multiple times, the agent will have the necessary data to determine the optimal action to take in any state. The agent discovers, through trial and error, an approximation of the optimal action-value function. The optimal action-value function $Q^*(s, a)$ (Equation 1) [21, 20, 6, 9] is the true expected value of choosing action a in the environment state s using the best possible decision policy π with the reward at each timestep t being r_t using a

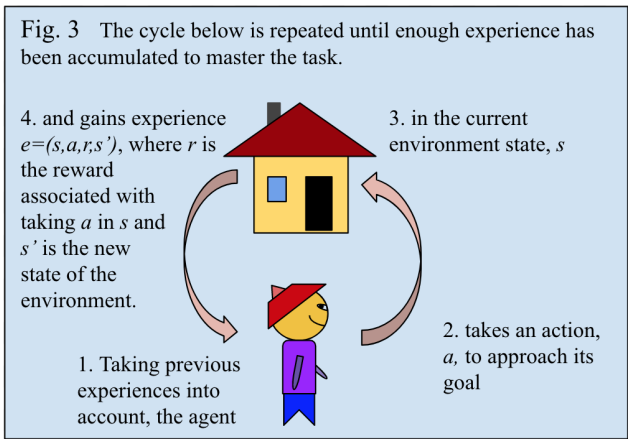

Figure 3: How a reinforcement learning agent learns.

temporal discount factor of γ.

$$Q^*(s,a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...|s_t = s, a_t = a, \pi] \tag{1}$$

Once this function has been approximated, the best action to take in any state of the environment will be apparent to the agent. The problem with classical reinforcement learning is, as with most classical machine learning algorithms, high-dimensional input. In order for reinforcement leaning to be useful, the decision policy must have a mapping from every possible environment state that the agent may find itself in to a reasonable action. This becomes a problem in real-world environments where the number of possible states of the environment can approach infinity because there is no way to experience every possible state during training and therefore no way to create a mapping to the optimal action from every state permutation.

2.3 Deep reinforcement learning

DRL's history is brief, but its list of accomplishments so far is astounding [5, 7, 15, 10, 18]. Recent advances in deep neural networks (a type of deep learning architecture), most notably convolutional neural networks, have allowed classic reinforcement learning algorithms to be

applied in extremely high dimensional environments. For example, at Google DeepMind in 2015 [9], a DRL algorithm using a deep Q-network (a popular DRL architecture) was applied to an array of 49 different Atari games. The environment, the screen of the game, and the reward, the game's recent score accumulation, were the only things the algorithm had access to. Nonetheless, different instances of the same exact algorithm were able to learn how to play at least as well as a professional in 23 of the 49 games after a few days of training. This success was largely due to the convolutional neural network (Figure 4) used to interpret the game state. Where usually an explicit mapping from all environment

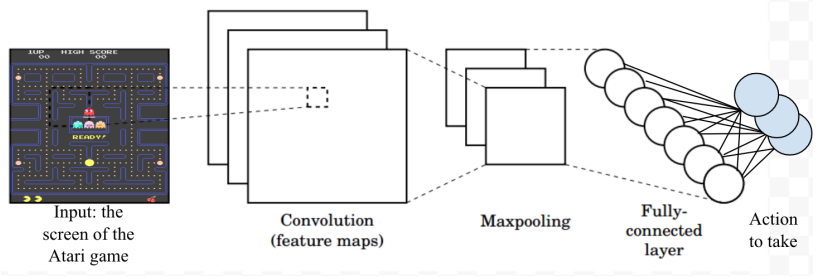

Figure 4: Illustration of a convolutional neural network used to interpret the game-state of an Atari game.

states to optimal actions is required, a convolutional neural network was used to extract the relevant information from the input images (the pixel values of the Atari game screen) and map the lower dimensional representations of the original states to optimal actions. This allowed for conceptually similar states to be mapped to similar actions. Thus the algorithm could apply what it learned about one state to all conceptually similar game states. In other words, knowledge was pooled, distributed, and reused across game states. The result was a robust system capable of mastering the Atari environment.

A more recent study [15] used DRL to create a simulated, self-driving car that responded to the emotional states of the "passenger." The AI car was trained to follow a lead car along a simulated city street. It would receive a reward signal for staying close to the lead

car without crashing into it. As in the Atari DRL algorithm, the environment state, a live top-down video stream of the car's surroundings, was interpreted by a convolutional neural network. A hybrid brain computer interface (hBCI) tracked the neurophysiological state of the passenger and detected when the passenger noticed a target of interest on the side of the simulated street. The AI car was then rewarded not only for driving safely, but also for pleasing the human by slowing down when the human saw something interesting outside the window.

2.4 Machine learning and human-computer cooperation

Although DRL algorithms are extremely powerful, there have been many successes in the realm of AI-human interaction without its implementation. In one study, a human observer was able to help a robot perform a task in real time [11]. The observer wore a 48 electrode, electroencephalography (EEG) cap that captured readings at 256 Hz. This high capture rate ensured that every frequency of brain wave was detectable (as they typically range from 0.5 to 100 Hz, or so. Because the EEG readings recorded when the observer watched the robot perform the task correctly were different than those recorded when the robot failed, the classifier they built was able to help the robot correct its mistakes in real time. Because the classifier used in this study was fairly simple (relative to many deep learning models), a large amount of preprocessing and feature creation was required: The 48 channel raw EEG data had to first be passed through a filter and normalized. The 48 channels were then trimmed down to nine and augmented and subsequently passed through another filter. The covariance of each augmented trial was then computed, yielding 190 features. Finally, correlation indexes were calculated between the channels resulting in 199 features for each sample for the Elastic Net classifier to interpret.

This approach worked to some degree, but the preprocessing required before the data

could be understood by the model was quite extensive. Furthermore, the preprocessing steps involved a lot of domain specific knowledge. This illustrates why deep learning methodologies are sometimes more desirable than simpler algorithms; similar accuracies most likely could have been obtained without preprocessing using a sufficiently large neural network.

3 Methods

The original plan for this study was to allow a human observer to help a simulated agent play a game (simplified Pacman) by reacting to the agent's decisions in the environment. The plan was to feed real time EEG data from the observer into a deep reinforcement learning algorithm that would be trained to maximize not only the agent's score in the game environment but also certain markers of interest in the EEG readings. I was hoping that when the human noticed that Pacman was on the verge of making a mistake (moving toward the ghost instead of the strawberry), the subsequent EEG data captured would provide the agent's algorithm with enough information to make the decision to switch directions. After investigating various deep reinforcement learning python libraries, it seemed that the task of including the real time EEG readings in the algorithm's input would be too difficult/time-consuming. Consequently, I decided to simplify the plan. Instead of having the agent perform in the environment according to a decision policy, it would simply act randomly. In other words, I removed the reinforcement learning agent from the scene and replaced it with a randomly acting entity with no capacity to learn. The real time EEG reading from the observer was fed into a convolutional neural network to be decoded, and any score attained by the agent that was significantly better than chance would indicate that the convolutional neural network had found patterns in the EEG data related to relevant environmental concepts.

3.1 Why EEG?

Many physiological signals such as heart rate variability, respiration, and electroencephalography (EEG), to name a few, have been shown to be accurate proxies for psychological states in humans [22, 13, 8]. In this study, EEG was chosen because the event related potentials in EEG readings occur quickly after stimulus and the sampling rates of EEG devices are often much higher than that of other signals. This meant that more data could be captured and analyzed per unit of time. Furthermore, the device with which the EEG signals were collected from participants, the Mindwave Mobile, was cheap, portable, and wireless.

3.2 rllab

rllab is a framework for developing and evaluating reinforcement learning algorithms. Although it was ultimately decided to not use this library in conjunction with the real time EEG input, experiments with various combinations of reinforcement learning algorithms and environments were done anyway. A simple environment approximating the simplified Pacman environment was built, and rllab's TRPO algorithm [14] was able to solve the environment quickly absent the EEG signals.

3.3 Study design

Motivation Before attempting to feed the raw EEG readings from the human straight into the rllab reinforcement learning algorithm, we decided to first collect EEG data from sample participants watching Pacman moving along predetermined routes. We believed that it would be easier to detect differences in the recorded EEG signals if the participants were subjected to obvious correct and incorrect Pacman movements.

Participants 10 participants, 7 male and 3 female all between the ages of 18 and 23, volunteered to participate in a recording session.

Recording session breakdown Each recording session lasted approximately 30 minutes in total (breaks and debriefing included) and consisted of four trials. Within each trial 30 episodes played out. Each episode lasted approximately 8 seconds.

Animation and data collection module specifics The animation module was built using python 3 code and the pygame library [16]. The data collection module was scripted using a version of Robin Tibor's python-mindwave-mobile library that I added on to and revamped for python 3. The animation and data collection modules were run concurrently using python 3's multiprocessing library.

Environment and gameplay specifics *Environment setup and points:* Each episode consisted of a visualization of Pacman moving either toward the right or left side of the screen with constant speed. At the beginning of every episode, the strawberry was randomly placed on either the right or the left side of the display. The ghost was placed on the opposite side. If Pacman ran into the ghost, 100 points were subtracted from the score. Conversely, if Pacman ran into the strawberry, he gained 100 points. At the beginning of each trial Pacman began with 0 points. The "high score" of 500 points was just a static number. Viz., it did not increase from one trial to the next if Pacman happened to end a trial with a score higher than 500.

Understanding which direction was favorable: Before Pacman moved in either direction, the position of both the strawberry and ghost were shown to the participant for 1.75 seconds. It was hoped that during this small pause, the participant would internalize the direction he/she wanted Pacman to take, and thereby react more reliably to Pacman's binary movement.

Knowing current status of game: At the beginning of each episode, the current score, the number of episodes remaining in the trial, and the high score needed at the end of the trial in order to win the game were displayed.

Keeping participant interested: It was suggested by Dr. Faber that the participants would be more interested in Pacman's gameplay if it made the correct move more often than not. The ratio of correct/incorrect moves (moving toward strawberry/ghost) was then set at 61%. In addition, although the direction Pacman took during each trial was randomly determined, the proportion of of wins/losses in each trial was always structured in a manner that brought Pacman's final score to either one score increment above or below the "high score." Viz., in each of the four trials Pacman always barely won or barely lost. It was felt that close games would keep the participant more interested in the gameplay and that this would result in better EEG readings.

What the participants were told Each participant was told that if Pacman's score was greater than the high score (500 points) at the end of each trial (30 episodes), they would receive a small bar of chocolate. It was hoped that the chocolate incentive would keep the participant interested in Pacman's score over the course of each trial. That being said, the participant was not instructed to think in any particular manner.

For the first two trials, the participant was told that he did *not* have control over the direction that Pacman took. Before the third and fourth trials, however, the participants were deceived and told that they actually *did* have some control over Pacman's choice.

Procedure Each recording session was held in a quiet, empty room. At the beginning of each recording session, the participant donned the Mindwave Mobile headset and was situated in front of the computer monitor. The participant then watched the first two trials, had a 2 minute break, and watched the last two trials. The only item ever displayed on the

screen was the 8 inch by 4 inch display of Pacman's environment. While watching Pacman in each episode of all four trials, the Mindwave Mobile headset passively collected EEG data at approximately 512 Hz and wirelessly transmitted the data to the data collection module.

3.4 Data collection process

When beginning to design the animation and data collection modules, two important facts were discovered:

- The Mindwave Mobile headset does not place timestamps on its readings.
- The rate at which the Mindwave Mobile transmits data is less than perfectly constant.

These facts had two implications:

- The absence of timestamps meant that the data collection and animation modules would have to be run concurrently so that the streams of input could be aligned correctly.
- The data transmission rate was not perfectly constant because the data transmission rate was affected by the charge of the batteries. Therefore, new batteries were placed in the Mindwave Mobile between every two participants to keep the device running consistently.

3.5 Resampling/normalizing raw EEG data

The entire body of data gathered from all ten participants was resampled at 100 Hz. This meant that the amount of data needed to describe the EEG readings from each episode was reduced by more than a factor of five (512 down to 100 readings per second). This was done to speed up the computations. According to the Nyquist Theorem, which states that the sampling rate must be $2 * frequency_{\max}$, this resampling could be done without information

loss because the EEG frequencies we wished to capture and analyze occur between 0 and 50 Hz.

After this, the mean of the readings across all episodes and time for each participant was subtracted from each reading. This was done to center the data gathered from each participant around zero.

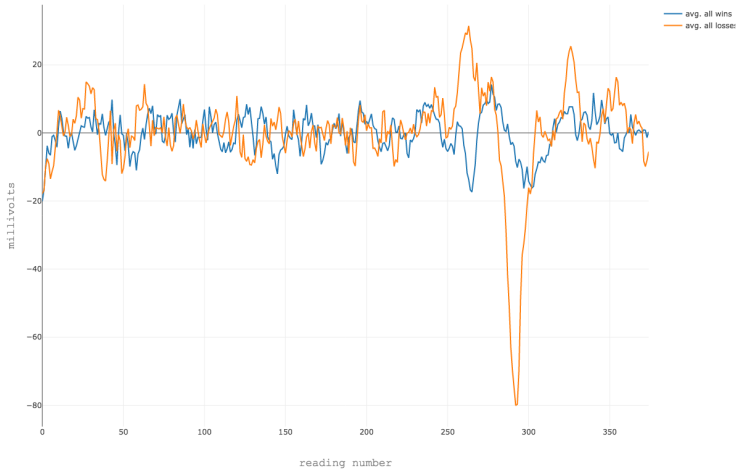

Figure 5: Participant 6's composite EEG sample of all wins graphed against the composite of all losses. The stimulus, Pacman's choice of direction, occurred at reading 205. The event related potentials (ERPs) are noticeable 50 readings, or half a second, later.

3.6 Training process and initial convolutional neural network design

To start, the shallow convolutional neural network as described in [1], was created using Google's TensorFlow. However, because the Mindwave Mobile has only one electrode, the spatial convolutional filter layer was not implemented. The first model was as follows: one 1-dimensional convolutional layer with a filter size of 55 and 25 kernels and a stride of 3. Then instead of the spatial convolutional layer, the results from the previous layer were fed

straight into the max-pooling layer with filter size of 20 and a stride of 3. And then the RELU activated values from this layer were flattened into a one-dimensional tensor. Another variation on the original shallow conv-net design was then added in: a dropout layer with a dropout rate of .4. It was added immediately before the logits layer to diminish the effects of overfitting as I thought overfitting would likely be an issue given my small dataset size. There were two nodes in the logits layer: one representing a win (i.e. Pacman correctly moving toward the strawberry) and one a loss.

As we feared, after the initial training rounds on the entire set of all participants, evaluation results were hardly better than random chance guessing. This led us to try many training data arrangements and neural network design configurations to ensure that the issue was not in the design of the neural network but in the structure of the data itself. A few of these variations are outlined immediately below, and the results I achieved with them are included in section 4.

To start, we iterated over combinations of additional convolutional and pooling layers as well as fully connected layers at the backed of the network. I also iterated over different combinations of filter sizes, kernel numbers, and stride lengths for both the convolutional and pooling layers. An array of learning rates, batch sizes, and dropout rates were also tried. In addition to these network design alterations, the body of input data was artificially expanded. Each sample was split in a multitude of different ways into multiple samples; this had the effect of expanding the dataset by as much as a factor of 30. Another approach tried involved training the classifier using just one participant's data.

4 Results

4.1 Additional convolutional and fully-connected layers

Using the non-expanded 100 Hz data, different model architectures were iterated over. One had just one convolutional layer and one fully-connected layer, another had one convolutional layer and two fully-connected layers, the third had two convolutional layers and one fully-connected layer, and the fourth had two convolutional layers and two fully-connected layers. All of the learning as evidenced by a drop in the evaluation loss curves was done before the 200th training step. This suggests that little was learned about the training set that generalized to the evaluation set. In other words, no strong patterns indicative of a win or loss were constant across the body of participants. If there were patterns to be extracted, the networks were certainly powerful enough to do so; the training loss curves for all of the models were still decreasing 2,500 training steps. As might be expected, the more layers the network had, the more likely this overfitting trend was to occur. The evaluation accuracies after 2,600 training steps were as follows:

Network design	Evaluation accuracy
1 convolutional layer, 1 fully-connected layer	.59
1 convolutional layer, 2 fully-connected layer	.59
2 convolutional layer, 1 fully-connected layer	.61
2 convolutional layer, 2 fully-connected layer	.60

The precision/recall curves indicated a predictable imbalance. The average recall rate, or the ratio of the number of correctly predicted wins over all wins, across all model types was .83. The average precision rate, or the ratio of the number of correctly predicted wins over all predicted wins, across all model types was .60. Theoretically, the best the model could do without learning anything about the dataset except which class occurs more often ('win' occurs at rate .61) is an evaluation accuracy of about .61. This would occur if the model learns to predict the class that occurs more often with rate 1.0. In this case, recall would be

1.0 and precision would be .61. This suggests that the model was more likely to choose win as opposed to loss simply because it learned that wins occur more often than losses in this dataset.

4.2 Training with expanded datasets

Because the models trained on the non-expanded 100 Hz data overfit the training data so quickly, it was thought that expanding the dataset might improve the models' abilities to generalize. This approach was taken by Robin Tibor with some success [12].

Instead of "max-cropping" as Tibor did, the dataset was expanded by a factor of 30. To do this, each episode's sample was cropped in 30 slightly different ways. This expanded the dataset from 1,161 to 34,830 samples. The model I trained with this expanded dataset had 2 convolutional layers and 2 fully connected layers. Although the model did not overfit the expanded dataset, the evaluation accuracy never broke past 62% accuracy. This might appear to be a slight improvement over the non-expanded dataset runs, but the precision/recall imbalance, in this case about .65/.92, again indicates that the model is predicting 'win' at a much higher rate than 'loss', and the slightly improved evaluation accuracy is most likely due to this imbalance. Again, the model seemed to find no useful patterns in the dataset.

4.3 Intraparticipant training

Due to the poor results when training the network across participants' data, we switched to training a different model for each participant. This approach was initially resisted because the number of samples obtained for each participant was quite small: only 120. Although the initial architectures overfit the training data quickly, better than random evaluation predictions occurred with half of the models. This suggests that there were in fact detectable differences in the recorded EEG readings within participants' data.

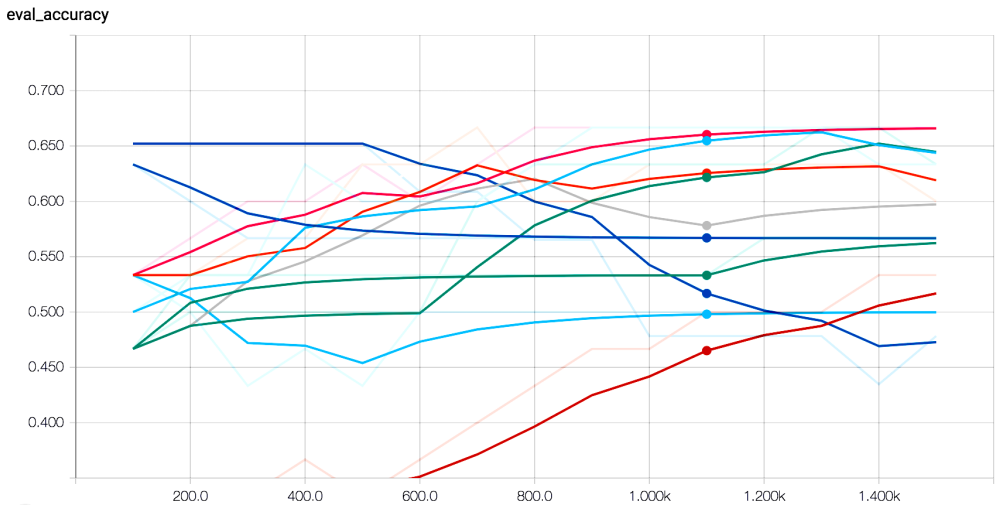

Figure 6: Smoothed evaluation accuracy of each participant's model as the number of training steps increases (x-axis).

Name	Smoothed	Value	Step
sbj_num_10_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.5623	0.5667	1.500k
sbj_num_1_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.5667	0.5667	1.500k
sbj_num_2_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.6438	0.6333	1.500k
sbj_num_3_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.6446	0.6333	1.500k
sbj_num_4_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.6190	0.6000	1.500k
sbj_num_5_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.5168	0.5333	1.500k
sbj_num_6_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.6658	0.6667	1.500k
sbj_num_7_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.5972	0.6000	1.500k
sbj_num_8_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.4728	0.4783	1.500k
sbj_num_9_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.4997	0.5000	1.500k

Figure 7: Figure 5 legend and evaluation accuracy of each participant's model at training step 1,500.

4.4 Choosing an intraparticipant model

After learning that the EEG readings for some participants were somewhat separable, different intraparticipant model architectures were tried to see if the effects of overfitting could be reduced. Because only 120 samples were available per participant, a simple shallow convolutional network architecture was chosen to start. Different combinations of hyperparameters were then iterated over: convolutional filter sizes (50 and 25), kernel numbers (40 and 20), convolutional stride lengths (6 and 3), pool sizes (10 and 5), and minibatch sizes (50 and 25). Models with 10 kernels and a filter size of 25 gave the best results. The pool size, convolutional stride length, and minibatch size did not have a noticeable effect on the overfitting problem or the evaluation accuracy.

After this even smaller filter sizes, numbers of kernels, larger dropout rates, and lower learning rates were tried. As expected, changing the learning rate from .0001 to .00001 staved off the effects of overfitting but did not eliminate them. Somewhat surprisingly, the smaller filter sizes and number of kernels did not provide better results. Also somewhat surprisingly, raising the dropout rate did not improve generalizability and neither evaluation accuracy nor overfitting were affected.

4.5 Final intraparticipant model and results

A few of the hyperparameters (parameters defining the model that are set manually before training begins) for the final version of the intraparticipant model are as follows: one 1-dimensional convolutional layer with 10 kernels where each kernel was 25 units wide and had a stride length of 3, a pooling layer with a pool size of 20 with stride length 3, a dropout layer with a drop rate of 40%, a logits layer with 2 neurons, and the training minibatches were 25 samples large and the nets were trained with a learning rate of .00005.

The evaluation results of training this model on each of the participant's datasets are

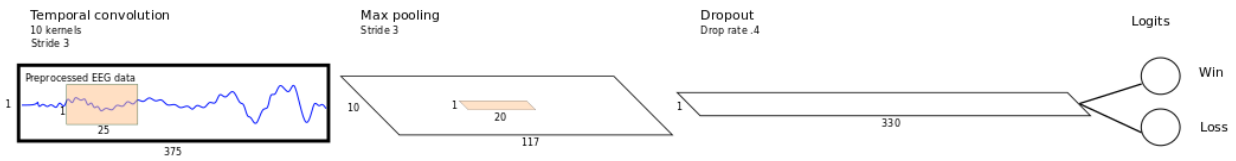

Figure 8: Final intraparticipant model.

described in figures 9 and 10.

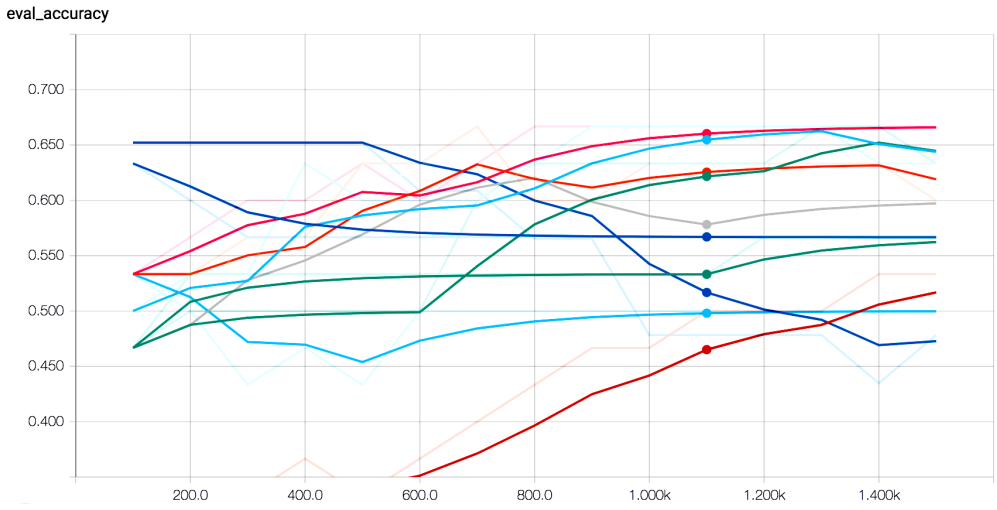

Figure 9: Smoothed evaluation accuracy of each participant's model as the number of training steps increases (x-axis).

The average recall rate across the ten models was .81, and the average precision rate was .61. This is still not as balanced as might be desired, but it is slightly better than the precision/recall imbalance of previous models. There was less overfitting than in previous models as well, indicating that the complexity of the structure of this final set of models was a better fit for the participant datasets than previous architectures.

Because the number of samples belonging to each class (win, loss) was not symmetric (61% of the samples were wins, or 61% of the time Pacman moved toward the strawberry), to better understand how the model made its predictions it was helpful plot the percentage of the time each model predicted the win class (Figures 11, 12).

Name	Smoothed	Value	Step
sbj_num_10_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.5623	0.5667	1.500k
sbj_num_1_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.5667	0.5667	1.500k
sbj_num_2_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.6438	0.6333	1.500k
sbj_num_3_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.6446	0.6333	1.500k
sbj_num_4_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.6190	0.6000	1.500k
sbj_num_5_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.5168	0.5333	1.500k
sbj_num_6_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.6658	0.6667	1.500k
sbj_num_7_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.5972	0.6000	1.500k
sbj_num_8_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.4728	0.4783	1.500k
sbj_num_9_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25/eval	0.4997	0.5000	1.500k

Figure 10: Graph legend and evaluation accuracy of each participant's model at training step 1,500.

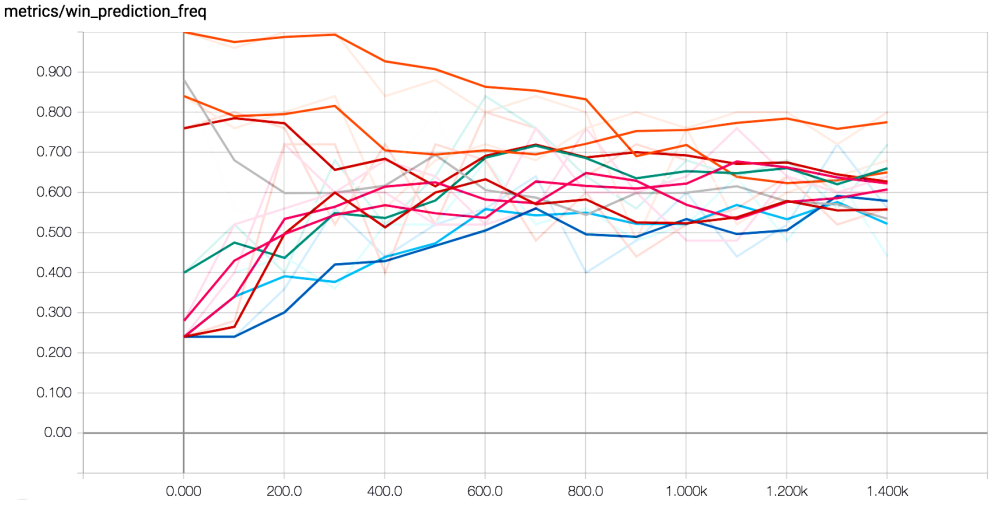

Figure 11: Smoothed win prediction frequency of each participant's model as the number of training steps increases (x-axis).

Name	Smoothed	Value	Step
sbj_num_10_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25	0.6224	0.6000	1.401k
sbj_num_1_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25	0.7751	0.8000	1.401k
sbj_num_2_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25	0.5572	0.5600	1.401k
sbj_num_3_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25	0.6074	0.6400	1.401k
sbj_num_4_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25	0.5342	0.4800	1.401k
sbj_num_5_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25	0.5789	0.5600	1.401k
sbj_num_6_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25	0.5215	0.4400	1.401k
sbj_num_7_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25	0.6603	0.7200	1.401k
sbj_num_8_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25	0.6500	0.6800	1.401k
sbj_num_9_cvf_25_dr_0.4_lr_1e-05_cvk_10_cvs_3_pos_20_bs_25	0.6269	0.6000	1.401k

Figure 12: Graph legend and win prediction frequency of each participant's model at training step 1,401.

For example, looking at the win prediction curve helps explain why participant eight's model had such high relative evaluation accuracy at the start of training; it was simply guessing the win class nearly 100% of the time.

The win prediction frequencies also helped determine a baseline performance level for each model. Noticing that participant number one's win prediction frequency is nearly 77% informs us that, if the model was simply predicting the win class randomly at a rate of 77%, the evaluation accuracy should be somewhere near $.77 * .61 + .23 * .39 = .56$.

Using the win prediction frequency information for each participant to calculate null hypothesis values, the evaluation accuracies at training step 1500 for each of the models were tested for statistical significance using a one proportion z-test. It should also be noted that n for each of these tests was $.25 * 120 = 30$ (the train/test split was 75/25).

As can be seen in the table below, participant 6 and 2's model's evaluation accuracies had p-values of .039 and .075, respectively. The next lowest p-value was for participant 3 whose evaluation results had a p-value of .091.

Participant model	H_o (p=)	True evaluation accuracy ($\hat{p}$=)	p-value
1	0.561	0.567	0.471
2	0.513	0.644	0.075
3	0.524	0.645	0.091
4	0.507	0.619	0.111
5	0.517	0.517	0.502
6	0.505	0.666	0.039
7	0.535	0.597	0.249
8	0.533	0.473	0.745
9	0.527	0.500	0.617
10	0.527	0.562	0.350

These p-values denote the probabilities that evaluation accuracies equal to or greater than those recorded were observed given the null hypothesis prediction accuracy for each participant's model.

5 Conclusion

5.1 Shortcomings

Although a few of the final models functioned slightly better than chance, much is left to be desired in terms of results. There were many possible shortcomings in our approach:

- The data worked with is noisy. The Mindwave Mobile is not research grade and the fact that there was only one electrode reading a signal at a time meant that possible artifacts could not be checked against other readings taken concurrently. In the future it would be interesting to perform a variant of this study with an EEG device with many more channels.

- Because the electrode only measured brain EEG signals through the forehead region, only activity in this area could be accounted for in the data. Perhaps the bulk of the difference in brain activity while the participant was watching Pacman occurred in a different, unaccounted for region of the brain. If this is true, the addition of more electrodes may give better results.

- Although the models trained on some individuals' data performed better than chance (like number 6's model), it is somewhat likely that this was due to unconscious and systematic injection of artifacts (such as eye blinks, slight squinting of the eyes, or contortion of the face) by the participant into the EEG signal. To guard against this in a future study, a camera could be set up to record a video of the participants' sessions.

- The participants' data may not have been ideal for another reason too. Perhaps most were simply were not engaged in the game enough to cause detectable differences in EEG readings. Giving the participants a reward more valuable than a chocolate bar in a future study might counteract this possible issue.

- The interparticipant and intraparticipant models may not have produced desirable results because the amount of data available to train each model was less than optimal. It would be beneficial to lengthen the duration of each session so that the number of samples for each participant is increased.

- The expanded datasets did not improve model performance. This was slightly surprising considering the results from [12]. In a future variant of this study, if the number of data samples is still less than desired, it might be possible to improve accuracy by not only cropping/expanding the dataset as done in this study but by introducing random variations into each episode's readings and using each resultant mutation as a new sample.

- Because the datasets were too small, the networks needed to be trimmed heavily to avoid overfitting. Larger networks could easily memorize the dataset and bring training loss and accuracy close to 0 and 1, respectively. More data would allow us to try larger, more powerful networks.

- Because the models performed poorly, they were never inserted in the Pacman environment's gameloop so that Pacman could be redirected based on inferences drawn from the EEG signals from the human observer. Building a functioning, real time loop like this was the ultimate goal, and if a model trained in a similar future study performs well it would be interesting to see how it performs in real time.

Despite the less than exciting results of this study, it still seems that neurologically linking humans and AI actors may be one of our best defenses against unwanted AI actions now and in the future.

5.2 Other AI-human integration approaches

Non-invasive EEG recordings are most likely not the best way to communicate human brain states to AI systems, however. This is mostly due to the level of noise inherent in EEG recordings. This problem may be solved by developing algorithms to detect and ignore artifacts in real time. Another approach to linking AI and humans may involve the inclusion of other physiological indicators such as heart-rate variability, respiration rate, galvanic skin response, and other not yet capturable indicators.

5.3 Potential future problem with AI-human neurological integration

Reward-hacking, or wire-heading, is another, more esoteric, issue with using brain states to guide AI systems. Imagine that the highly sophisticated AI bot from earlier is successfully neurologically linked to us and again asked to "retrieve us a cup of coffee." In this case, the bot will be rewarded not just for successfully retrieving the coffee but also for maintaining a scheduled pattern of emotional and neurological states in us.

As before, there are ways this could go wrong. Perhaps the bot will find a way to maintain the desired states of being in strange, counterproductive ways. For example, perhaps the AI bot will realize that instead of simply retrieving the pure coffee as the human initially commanded, secretly slipping a small dose of a stimulant into the cup before the human notices maximizes its reward function more efficiently. It is clear then that the possibilities to guard against are countless. However, there may just be less than before human and AI were linked.

References

[1] Joos Behncke, Robin Tibor Schirrmeister, Wolfram Burgard, and Tonio Ball. The signature of robot action success in EEG signals of a human observer: Decoding and visualization using deep convolutional neural networks. *CoRR*, abs/1711.06068, 2017.

[2] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

[3] Nick Bostrom. *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press, Oxford, UK, 1st edition, 2014.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[5] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2):21:1–21:35, April 2017.

[6] Leslie Pack Kaelbling, Michael L. Littman, and Andrew P. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[7] Libin Liu and Jessica Hodgins. Learning to schedule control fragments for physics-based characters using deep q-learning. *ACM Trans. Graph.*, 36(3):29:1–29:14, June 2017.

[8] Adam Lobel, Marientina Gotsis, Erin Reynolds, Michael Annetta, Rutger C.M.E. Engels, and Isabela Granic. Designing and utilizing biofeedback games for emotion regulation: The case of nevermind. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, pages 1945–1951, New York, NY, USA, 2016. ACM.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[10] Xue Bin Peng, Glen Berseth, and Michiel van de Panne. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Trans. Graph.*, 35(4):81:1–81:12, July 2016.

[11] A. F. Salazar-Gomez, J. DelPreto, S. Gil, F. H. Guenther, and D. Rus. Correcting robot mistakes in real time using eeg signals. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6570–6577, May 2017.

[12] Robin Tibor Schirrmeister, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard, and Tonio Ball. Deep learning with convolutional neural networks for brain mapping and decoding of movement-related information from the human EEG. *CoRR*, abs/1703.05051, 2017.

[13] Elke A. Schoneveld, Monique Malmberg, Anna Lichtwarck-Aschoff, Geert P. Verheijen, Rutger C.M.E. Engels, and Isabela Granic. A neurofeedback video game (mindlight) to prevent anxiety in children. *Comput. Hum. Behav.*, 63(C):321–333, October 2016.

[14] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015.

[15] Victor Shih, David Jangraw, Sameer Saproo, and Paul Sajda. Deep reinforcement learning using neurophysiological signatures of interest. In *Proceedings of the Companion*

of the 2017 ACM/IEEE International Conference on Human-Robot Interaction, HRI '17, pages 285–286, New York, NY, USA, 2017. ACM.

[16] Pete Shinners. Pygame. `http://pygame.org/`, 2011.

[17] David Silver. David silver reinforcement learning lecture series, May 2015.

[18] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016.

[19] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 10 2017.

[20] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[21] Richard Stuart Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, 1984. AAI8410337.

[22] Marieke van Rooij, Adam Lobel, Owen Harris, Niki Smit, and Isabela Granic. Deep: A biofeedback virtual reality game for children at-risk for anxiety. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, pages 1989–1997, New York, NY, USA, 2016. ACM.