Predicting Preference with Sparse Data in Personalized Gamification via Deep Learning

Philip Wilson
Computer Science
Florida Polytechnic University
Lakeland, USA
pwilson8655@floridapoly.edu

Dr. Bradford A. Towle Jr.
Computer Science
Florida Polytechnic University
Lakeland, USA
btowle@floridapoly.edu

Abstract* – Personalized gamification is a practice that is relatively well defined and improves the effectiveness of a gamified system. However, in practical application the improvement is not as significant as expected. The process of personalizing a gamified system is taxing and relatively unfeasible, with far too many aspects to consider to produce an effective result. Artificial intelligence, and neural networks, can step in to alleviate much of the work, but even still results are inconsistent at best. This project seeks to remove this inconsistency by attempting to personalize only one aspect of a gamified system, rather than the entire system as a whole. By attempting the personalization problem in this manner the amount of individual characteristics to consider is reduced dramatically, thus allowing for a neural network to more quickly and accurately determine personalization characteristics and apply them for any given user. Results show that an RNN can detect preference patterns and apply user preferences to a scheduling system. These results were produced with little run time and a more sparse dataset than normally expected for a neural network, which showcases the novel fact that detecting user preference does not require large datasets.

***Keywords* – gamification, personalized, adaptive, dynamic adaptive, neural network**

I. Introduction

Gamification is a field within computer science that is directed toward the modification of system design to mimic those of gaming entertainment systems. It is the process of adding game-like elements to a non-game setting. The purpose of this approach to design is to increase user interaction with the system in question [1] [2] [3]. There is no one single way to approach the design of a gamified system. Despite this lack of structure to the design approach, there are common factors that are considered when adding game-like elements [4] [5]. While the lack of rigidity gives developers freedom, that freedom comes at the expense of producing a universally effective product. Gamification motivates the user by encouraging interaction with the system; however, this benefit is directly proportional to the user's willingness to interact with the system. This issue is what limits the effectiveness of gamification [6] [7]. The main issue stems not from the design, but from the users directly. Everyone has a different psychological profile; they do things for different reasons as they have unique preferences. Modern gamified systems are designed to work moderately well for everyone, these systems are designed to be enjoyable to as many users as possible. This type of design and implementation often only leads to moderate improvement among the users, leaving gamification in a situation where more effort is put into creating the system for only a slight to middling perceived benefit.

This conundrum created the sub-field of personalized gamification. Personalized gamification seeks to increase engagement and effectiveness of gamified systems by designing the system for a specific user. This design shift of gamification emerged because gamified systems are designed with a one size that fits all approach. An un-personalized gamified system is rigid and unchanging. It will function exactly how it was built to do forever, regardless of user input and preference. This is usually not a problem as gamification as a study always shows results that are indicative of an improvement. The problem is that the improvements can sometimes be minimal. Personalized gamification design alleviates this by creating a system that responds to data and allows for a shift in its processes that make it more desirable to the user. Research shows that gamified systems that contain elements that users respond positively to produce better results than those that don't. However, there is a glaring issue when considering the design and development of a personalized system.

The issue with these types of systems is not the data collection, but the ability to use that data meaningfully. This process would be very painstaking as well as economically unfeasible. Each time more data is gathered and new personalization strategies emerge, the design and implementation work would multiply exponentially [8]. Designing a system for one user, or type of users is less of an issue. The real problem is that systems are often designed for billions of users, and the process of personalizing that system for each of the expected users is not feasible. This would require an understanding of all the possible user characteristics and contextual characteristics. This issue is what keeps personalized gamified systems from being more mainstream. [6] [7]

The proposed fix that research considers is adaptive gamification. This is where systems are designed to collect data as they are used and, by using artificial intelligence, respond to that data by modifying the system based on the collected data. A sort of middle ground between a normal system and a personalized one. Adaptive gamified systems

are automated to adjust dynamically as they are being used. These types of systems are more effective than rigid systems, but not as effective as a personalized one. Research was conducted to see if the gap between personalized and adaptive gamification could be widened; instead of using AI to merely adapt the system, if it could be used to personalize them. [9]

This project selects one gamification element to personalize, scheduling priority, and seeks to prove that automation of personalization is possible. Adaptation of a system automatically has been done and has been shown to work well. The issue is that adaptive systems react at a large scale to all the data collected from across its entire user base. Rather than attempting to personalize an entire system, this project attempts to show that a single type of gamification element, in this case task scheduling, can be personalized automatically. The hypothesis is if this process can be applied to one gamification element it can be applied to any of them. If that is true, then all those elements can be personalized and combined to produce a resulting system that is tailored at the user level rather than a user base level. Multiple tests will be run with increasing levels of element complexity. Both human tests and AI generated tests will be conducted to determine if there is a noticeable difference in neural network learning speed and accuracy among data that is procedurally generated versus data that is created by human input, which is expected to be more raw and less uniform. This study differs in a novel way from previous personalized gamification studies in that it focuses on the results of a neural network and its ability to properly personalize a system, rather than the results of a gamification user study.

II. Related Works

A. Justification for Researching Personalized Gamification

Previous research done on gamification is widespread and a well-documented and studied field of computer science. Personalization is a newer field within gamification research, and the application of artificial intelligence is even newer still. The need for continued research for personalized gamification lies in its core problem. There is a very large overhead cost for creating a personalized gamified system. Players crave novelty, they will get bored doing the same challenge day in and out. Developers experience a large amount of pressure attempting to create endless amounts of challenges and adapting existing games based on feedback. [10]. As mentioned before, the common idea presented to fix this issue is automation. This fix however belies another question, that being which part to automate, and how to automate it.

Creating a system that dynamically personalizes while a player, or user, interacts with it is no easy feat, again going back to the core problem of personalization. Even an automated system still needs to be developed such that it can adapt to all possible user characteristics. Personalization when done properly needs to be based on player psychology and aligning player types to specific game elements. A time-consuming process to undertake and one that presents an issue with personalization, as well as research on the concept.

A large motivation for gamification research is enhancing the learning process for students. As a result, a lot of personalized gamification studies are also directed toward student's learning. However, the psychological aspects of learning are imposing and difficult to personalize effectively, and the results are usually lackluster. There was a study done that addresses this issue by focusing on a smaller scale problem. This study produced very good results compared to non-personalized methods which could indicate that many current studies may not be producing realistic results for how effective personalized gamified models can be [11].

Another break from traditional personalization research comes from a project that presented an approach that did not simply personalize a gamified system, but rather a recommender system that could be applied to the game. This study continued to discuss the importance of player psychology and the alignment of a player's personality to specific game elements, and much like the project referenced above, narrowed in on specific elements of the user to focus on, as well as game elements that should be personalized first [12]. It was shown that not every aspect of the system needs to be personalized, only certain aspects such as activities, gameful elements, and persuasive strategies. These studies help to show that personalized gamification may not require as much overhead as people believe and that the scale of what needs to be personalized may be smaller than anticipated.

B. Personalized Versus Adaptive Gamification

The improvement of gamification extends past the personalization approach. As mentioned before, adaptive gamification is another approach that seeks to tailor a gamified system for the user. However, there are some downsides to this approach. The most glaring of them is that the adaptation process is done at the beginning of the design process. This is done by means of surveying the prospective users and determining what their player type is. From that result, gamified systems are then tailored to the player. However, this approach is very static and does not allow for changes as the players use the system[13]. Data is best gathered from use rather than from surveys alone. People can misunderstand questions and give inaccurate results. Rather than simply making an adaptive system, the better approach is to make a dynamically adaptive system, which is one that learns and changes to better suit the user as they interact with it. Another such term for this type of design is also referred to as adaptable [14]. This crossing of terms is one of the reasons that research into this field is so difficult. The same term is often said to describe similar ideas, but does not have one concrete, agreed upon definition. As it stands, adaptive gamification is the style where features are created to fit expected player types. Personalized gamification is seemingly the same as dynamic adaptive gamification, or adaptable gamification. The style in which users can affect the system directly to personalize it themselves. For the purposes of this paper, personalized gamification and dynamically adaptive gamified system will be used interchangeably.

C. AI and Gamification

The dynamic approach to personalization is often done through artificial intelligence. Methods are created to track user progress through the game and compare what is being done to preference tables that are created at the beginning of

the design process. As player scores are updated, dynamic systems adjust to present specific preset elements that are likely to be desirable based on where the user score falls in terms of player type [15]. This approach does still suffer from the static approach that plain adaptive models do. Players are sorted into their player type from the beginning based on preliminary survey results. Once sorted it can be difficult to adjust one's scores enough to reach a presentation style that is truly personalized, especially if the original survey results happened to be incorrect. Other approaches use deep learning to personalize the gamified learning system from the ground up [16]. All users start with the same system, but as the users interact with it, multiple aspects of the system adapt to their preference and cognitive level. The adaptation style is notable as well because only select aspects of the system adjust. This relates back to the notion that research into adaptable systems produces mediocre results because too much is being attempted at once. This study adapts very few features: the presentation style, the ability to skip lessons, and the adding of hints and extra attempts. This study showed great results by reducing time spent using the system and increasing the scores of the students versus typical results for gamified learning systems. This brings up two crucial questions; how much a system needs to change to become personalized and exactly how personalized a system needs to become to produce better results.

The questions above drove this project to be smaller scale than most other prominent research involving dynamically adapting gamification systems. To determine an answer, this project focused on only a single aspect of gamification, task ordering. That a to-do list is considered a gamification element is not common sense to most individuals. Most games, however, have some sort of order applied to them. Whether it be linear through a story mode that has to be completed in order, or as the game progresses players can be given two or more paths to determine their own adventure. In the same way that people order their daily tasks, games give players a task structure as well. This feature seemed a proper element to focus on because people have an innate capacity to order tasks. Each individual has their own method for untangling a messy workload and determine what, for them, is the best and most comfortable way to approach them [17]. There are a variety of ways in which people could choose to order tasks: due date, priority, how long the task will take, and even by cognitive load [18]. Despite the innate wanting to have an ordered list of tasks, people can have a weakness for organizing and managing those to-do lists. New tasks can be added on after the list is made which leads to the need and creation of a separate list. Over time things become cluttered and unmanageable which defeats the purpose. In a previous study, it was determined that an automated and intelligent to-do assistant can help users with the process of breaking down a list of tasks and helping them create and manage their list of tasks [19]. This helps to show that the automation of task ordering is both beneficial and not entirely unfounded. A perfect model simply has yet to be created. Bridging this concept into personalized gamification seemed the most logical approach to take when considering how to further what could be considered a field stuck in a rut. Dynamic adaptive systems should work better, but the results are often just slightly better than normal.

Personalized gamification research suffers from doing too much at once. By focusing on just a single attribute of personalization, it is plausible that a higher degree of accuracy, when concerning personalization levels, can be achieved. By choosing an attribute that is core to human personalities, a simple but effective system could be produced.

D. Using Neural Networks for Gamification

Automating a gamified system has been the common approach to personalizing systems. The breadth of the complexity the automation takes varies. There are many factors to take into consideration. It makes logical sense that a neural network of some kind should be utilized. Projects done previously that directly utilize neural networks solidify this [16]. A neural network is a technique in machine learning that was derived from and built to resemble the human nervous system, which in of itself is a marvel of engineering, biological as it may be. Neural networks are made up of units that are organized into different layers: input, hidden, and output. Each unit in a layer is connected to a different unit within a different layer and each of those connections has a weight. Inputs are multiplied by the weights at each unit and undergo transformations. The output of those transformation functions is then fed into the next layer of units. When there are no more layers left, the solutions of the problem is considered finished [20]. Neural Networks can be applied to a variety of problems such as pattern recognition, classification, clustering, dimensionality reduction, computer vision, natural language processing, regression, predictive analysis, scheduling, and many more. Neural networks can also be broken into five different types of networks. Feedforward neural networks, recurrent neural networks, radial bias function neural network, Kohonen self-organizing neural network, and modular neural network. Each of these have specific advantages over the others and with enough effort can all be used to solve essentially the same problem with varying degrees of success. The nature of the problem at hand with this project leads to the conclusion that pattern recognition and the ability to make predictions on future occurrences would be either necessary or beneficial. Research and past implementations indicate that feedforward and recurrent neural networks are the two strongest candidates for this project.

Feedforward neural networks (FFNN) are a type of neural network where information only flows in one direction, from input to output. Recurrent neural networks (RNN) differ in that the units form a cycle. Output from one layer becomes the input to another allowing for RNNs to have memory of previous states and use that memory to influence the current output [21], [22], [23], [24], [25]. As previously stated, neural networks have many applications to various problems. Past applications of FFNNs include pattern classification, data mapping, forecasting, and have even been used to determine the optimal size that a FFNN should have for specific problems before diminishing returns [26], [27], [28], [29], [30], [31]. Applications of RNNs include sequence prediction, stock market prediction, pattern-shape recognition, speech recognition, language modeling, and scheduling problems [32], [33], [34], [35]. Provided what is believed to be necessary for the completion of this project and based on the applications above, FFNNs and RNNs are the two types of networks that are to be used

for testing. Particularly the ability of RNNs to process sequential data and learn patterns and dependencies. This makes them powerful tools for determining orders of operations.

Despite the difference between FFNNs and RNNs there is a common similarity between these two, and the other types, of neural networks. That being the size of the network. A general rule of thumb for neural networks is to have them be relatively large. This is required to prevent overfitting and reduce generalization on new, unseen data. Depth and width are the common terms for the size of a neural network, depth being the number of layers and width being the number of neurons per layer. Commonly, depth is the more relevant factor. It is better to be narrow and deep than shallow and wide [36]. It is also worth noting that the applications cited above have little to do with task ordering, save for the ability of neural networks to perform scheduling operations.

E. Personalized Gamification and Job-Shop Scheduling

Scheduling as a problem both in computer science and industry is approached as an optimization problem. The job-shop scheduling problem is a quintessential example of this, where there are jobs to be completed, and they have prescribed numbers of operations that can only be done by a dedicated machine. The goal is to minimize the function of job completion times such that no two operations are ever done by the same machine at any given moment [37], [38], [39]. In the past, this problem was solved by human experts alone. The introduction of Ai agents and the implementation of neural network schedulers opens many possibilities for other practical approaches to solve the job-shop problem [40], [41], [42]. The application of neural networks to the solving of the job-shop problem not only performs better, but also increases the scale of problems that can be feasibly solved [43]. Scheduling problems as they have been approached before are aimed at business and manufacturing, minimizing their time and cost to make a profit. Minimization is an optimization problem. This is not the problem that is being addressed in this project. Research has shown that neural networks are powerful tools capable of solving complex problems by learning and acting in a similar manner to how our brains function. The human mind, however, is more than just logic and optimal solutions. Emotion, personal bias, and preferences impact one's decision-making process. The goal of this project is to train a neural network to recognize those preferences and tailor a system to that person. By only focusing on one aspect of preference, task organization, it is believed that the system will be able to learn quickly and accurately. Based on the applications cited above, FFNNs and RNNs make the best candidates to work with. The largest differences between this project and what has been done before are the hyper focus on one user preference/game element and the use of a scheduler that does not seek an optimal solution, but rather a preferred one.

III. METHODOLOGY

To test and prove that machine learning can be used for the purposes of creating a personalized schedule for users' data will be required. Data is created in one of two ways. The first is by means of self-generating data and storing it in a file to be sent to the neural network later. The second is generating the data as it is needed, but in smaller amounts at a time. These two types of data generation are important as one represents how neural networks often see data, in large amounts, and the other mimics a user interacting with the system during their use of it. All the data is formatted in the same way regardless of the size of the pool. Each data entry contains 5 separate metrics that represent factors individuals are likely to consider when ordering tasks. The 5 metrics to be used include: time remaining, task novelty, priority level, task length, and job code. This data represents what will become X when splitting the data into training and testing sets. The Y value is created by taking each point of data and comparing it to each other point in the same set. In the case of 10 tasks there would then be 100 data points that represent each task being compared to all others. The Y value will be either a 0 or 1 depending on if task i comes before or after task j. The criterion for ordering varies, but is always consistent for any specific test. The resulting data is then placed into pairs of inputs with a single output. Once the data is created, it is split into training and testing sets.

	Column	Index	Result
0	[4, 6, 12, 10, 1]	[4, 6, 12, 10, 1]	0
1	[4, 6, 12, 10, 1]	[19, 3, 5, 6, 4]	1
2	[4, 6, 12, 10, 1]	[24, 8, 10, 5, 3]	1
3	[4, 6, 12, 10, 1]	[40, 5, 6, 7, 5]	1
4	[4, 6, 12, 10, 1]	[54, 4, 15, 9, 8]	1
...	...	...	...
1220	[294, 7, 20, 7, 8]	[157, 10, 17, 9, 5]	0
1221	[294, 7, 20, 7, 8]	[175, 4, 18, 4, 1]	0
1222	[294, 7, 20, 7, 8]	[248, 5, 12, 5, 2]	0
1223	[294, 7, 20, 7, 8]	[273, 6, 11, 6, 9]	0
1224	[294, 7, 20, 7, 8]	[294, 7, 20, 7, 8]	0

1225 rows × 3 columns

Fig. 1. Example of input data

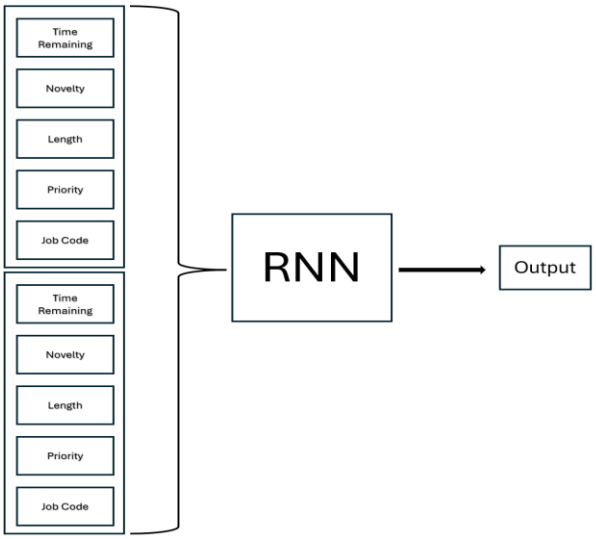

Fig. 2. Example of Data Flow to Output

In addition to requiring data, neural networks also need to be created. The data being sequential in nature indicates that an RNN is the most correct choice. Preliminary testing had FFNN fail to produce acceptable results. This is due to the nature of the data format. FFNNs when considering binary outputs take two inputs and produce a single output. In a normal case this would have functioned well. The issue with the shape of the data being input is that each X has 5 subpoints. This essentially means there are 10 inputs for a single output. It is assumed that this is the reason FFNN was unable to produce an acceptable model. Due to the form of the data essentially containing 10 inputs the RNN is going to be set up with an input size of 10 and will have a single hidden layer with a size of 16. The output size is 1 for the purpose of binary classification. The goal is to place one task either before or after the other by outputting either a 1 or 0. The batch size is set to 32 to strike a balance between efficiency and learning ability. The epoch size will be 100 for the same reasons. The RNN needs to be able to detect and learn the patterns in the training data. The learning rate will be set to .001 to allow for gradual updates to the weights of the RNN.

After a trained model is produced, its training accuracy is calculated. Should the value be sufficient the model is then shown new data. This new data is created exactly the same as training data but always contains only 10 tasks. The newly generated data is sent into an insertion sort algorithm. Within this algorithm, the trained RNN will sort the entire set of 10 tasks by producing a 1 or 0. That value is used to gradually sort the new data. Once finished, the data sorted by the RNN is then checked by hand to determine how many of the tasks were sorted properly. This is done by comparing what the RNN produced against the true order the tasks should have had. Each task in the incorrect position is considered ignored and the validation accuracy is reduced by 10 percent. The resulting score represents the validation accuracy of the RNN.

IV. Testing Procedure

The process of testing the RNN is broken into two parts, A and B. Test A is meant to replicate the scenario in which a user would be interacting with a gamified system that is attempting to personalize as they interact with it. Test B more closely mimics adaptive gamification in which a user would provide details about their preferences beforehand and the system would modify itself from the beginning rather than gradually over time.

The process of performing test A begins with generating data. 10 tasks are created with 5 metrics that are randomized between set ranges. Time remaining ranges from 1 through 365. Priority, novelty, and job code have ranges from 1 through 10. Length has a range from 1 to 20. After the data is properly generated and shaped, the RNN is run through the training process. After 100 epochs the training accuracy of the RNN is determined. If the accuracy is lower than 98%, new data is randomly generated in the exact same manner as before, though the values of the task metrics will be different. This new data is then added to the original to create a larger set for the RNN to use for training. This process simulates what would happen if a user were interacting with this system, adding new preference data for the RNN to work with while retaining the data from the previous interaction.

With the new data added, the RNN is completely retrained using the larger amount of information available to it. This process repeats until the training accuracy reaches 98% or higher, or until 20 iterations have been run. If the RNN does not reach 98% accuracy after 20 loops, the accuracy it currently attained will be used for validation. The validation process is conducted 3 times with a single trained model. This is done to ensure the RNN is not producing correct values by pure happenstance and correctly guessing how tasks should be sorted but genuinely learned the pattern that was present in the data. The 3 values are then averaged and become the final value for that iteration of the current test. Once validation is completed, the entire training process is conducted again 2 more times. This is to ensure that the RNN is consistently capable of learning preference patterns.

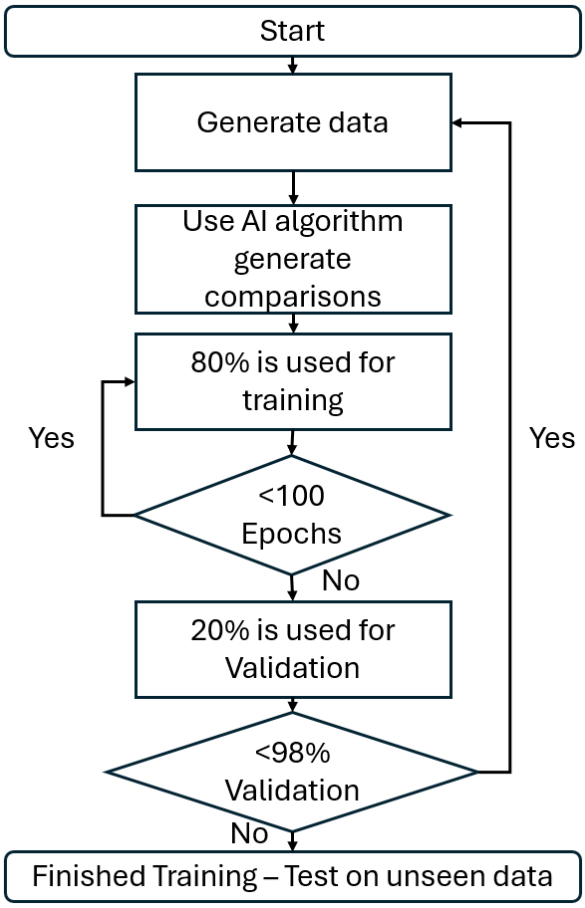

Fig. 3. Test A Flowchart

Test B follows a similar process to test A. The biggest difference is that test B does not use randomly generated data, but rather a stockpile of 35 tasks. Those 35 tasks are ordered using the same logic as those from test A, but the outcome presents the RNN with over 1200 points of comparative data as opposed to 100. This is because a single task is compared with 34 others instead of 9, which gives the RNN far more correlational information to work with. Even as test A loops and the dataset grows larger, the data is self-contained every 10 tasks. Task 1 is never compared to task 14 because they were created and ordered at different times. Test A was designed this way because a user, after getting a new set of tasks, would not go back and consider previous work they had done.

With the data for test B formatted, it is then sent to the RNN for training. Training for test B is not repeated until 98% accuracy is reached, but instead only runs once. The validation process is conducted 3 times and averaged 3 times, just as test A. The last difference is that test B is not run 3 times. It is meant to represent preset data inputs, to show the difference between personalized gamification and adaptive gamification.

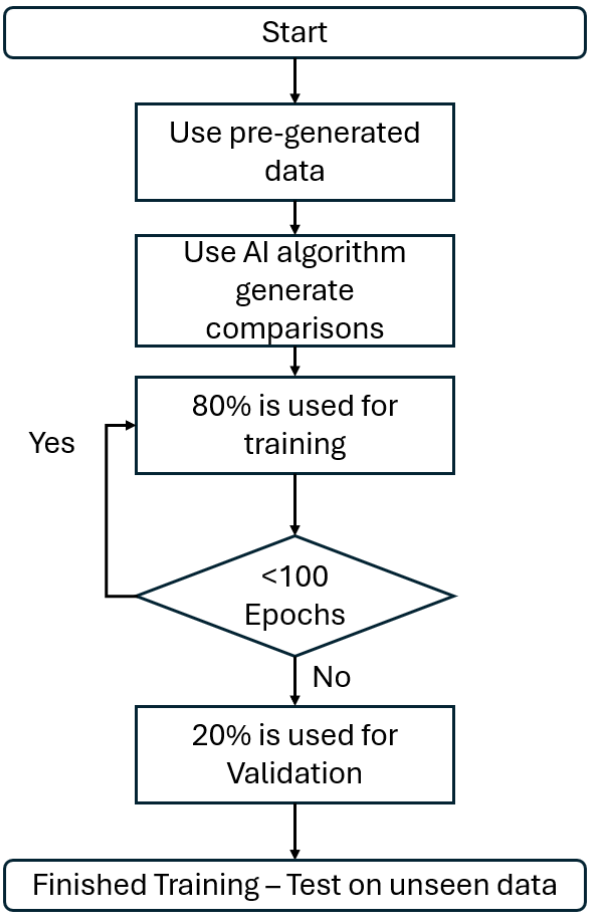

Fig. 4. Test B Flowchart

A. *Test 1*

Test 1 uses the simplest method of ordering logic. It only focuses on one of the 5 metrics, in this case, task length. If the length of task i is less than that of task j, the resulting value for that pair will be 1, otherwise, the value returned is 0. RNNs can often have trouble with patterns that are extremely simple, especially when the dataset is small. For this reason, test 1A, rather than being run 3 times with the same parameters on the training data, it has seven different iterations where the training data varies between each. Each of these iterations made changes to the input data. The first iteration used data that was generated using the prescribed values, serving as a control test.

Iteration 2 had the values of length changed to range between 1-10,000. This was done to determine if performance was low because too many of the values were the same in the length metric. The goal was to reduce the chance of repeat values. Iteration 3 followed the same logic as the one before it but only ranged from 1-100. Iteration 2 performed poorly and the cause could have been due to the values in the length field being too varied from the other 4 metrics. Iteration 4 adjusts the value range for length to 1.00-20.00. This strikes a balance between 2 and 3 where there are thousands of possible values, but they will not be drastically different than the other metric values, which would reduce skewing of the model. Iteration 5 expands on this by reducing the time remaining to 45 from 365. Iteration 6 dropped all metrics except for length while iteration 7 normalized the values of all 5 metrics.

B. *Test 2*

Test 2 adds a second metric for calculating order, that being time remaining. In the case that two tasks share the same length, time remaining is used to determine which should come first amongst the two. Data generated using this logic was iterated upon two times. The third iteration changes the second metric to novelty as opposed to time remaining. This was because time remaining, having such a large range of possible values meant that no two tasks in groups of 10 ever had the same values. It was believed that perhaps the RNN was detecting time remaining as the primary metric for ordering instead of using it as a secondary one.

C. *Test 3*

Test 3 expands on the idea of test 2 and adds a third metric to be considered when ordering. For iteration 1 and 2 that metric was novelty. Iteration 3 switches the position of time remaining and novelty as the second and third metric for ordering. The logic for this swap follows that of test 2. Time remaining is likely never going to have repeat values across 10 tasks. This would mean that the third metric for ordering was never reached. To ensure that the RNN can detect multiple levels of preference, novelty was made to be the second metric. Length having a range of 1 through 20 and novelty being from 1 to 10 means there is a much higher likelihood for there to be repeat values. In the case that there is, time remaining would be useful as a third metric.

D. *Test 4 (Complex Sorting logic)*

The fourth test to be conducted uses a much more complex sorting logic than tests 1, 2, and 3. Relatively speaking they have simple logic for sorting tasks, only using less than or greater than operators to determine order. To test the RNN's ability to learn highly complex patterns, test 4 orders its data in a manner that attempts to confuse the RNN.

Data for test 4 is ordered by first looking at the priority of 2 tasks. If the first task has a priority of 8 or higher and the other does not, then that task comes first. If both tasks have a priority of 8 or higher, the one with the higher priority goes first. Should both tasks have the same priority that is either 8 or higher, or if both tasks have a priority lower than 8, then time remaining is used to determine which should come first. This ordering schematic, while still checking if some values are greater than others, adds extra layers to that comparison, serving to better simulate how an actual person might order their tasks.

Just as with test 1, test 4 has more than 3 iterations and each iteration makes changes to parameters rather than simply repeating for consistency evaluation. The difference between this and test 1 is that rather than adjusting the training data, adjustments were made to the parameters of the RNN itself. The first iteration serves as the control test where no modifications are made. Iteration 2 sets the batch size to

16 rather than 32. Iteration 3 only changes the epoch count, raising it to 200. Iteration 4 combines the changes of iterations 2 and 3 by dropping the batch size to 16 and increasing the epoch count to 200. The final iteration checks to see what happens when the batch size is 2. Test 4 also does not run a test B. Being created to mimic complex human preferences it seemed unnecessary to run a test designed to work like a preference setting rather than a dynamic learning model.

V. Results and Analysis

A. Test 1A

Test 1A produces rather perplexing results(Fig. 3). It was presumed that such a simple pattern would be easily recognizable. Multiple conclusions can be drawn from these results. One such is that the pattern is too simple for such a large input size. 5 metrics for 2 tasks are given. Asking a RNN to automatically draw a conclusion from a large input that only uses a fraction of it may be too difficult. However, iteration 6 dropped the impertinent values to zero, and while that test does have the highest training accuracy it still has low validation(Table 1). This leads to a conclusion that the shape and size of the data is producing too much noise for the RNN to filter out and perhaps requires that the irrelevant columns of data be removed completely. It is clear here that for this preference pattern, there is a large degree of underfitting. This leads to the assumption that the RNN itself may need to be modified to accommodate learning for this test, as the input data was manipulated in many ways to alleviate the accuracy problems. Under most circumstances, validation is to be run 3 times per iteration to determine if the trained model is consistent with its ordering skills. With such low values of accuracy being returned, it was deemed unnecessary to check for consistency. The only two that had validation run 3 times were iterations 1 and 7 as those are the two that likely had the highest chance of performing well, though the results prove otherwise.

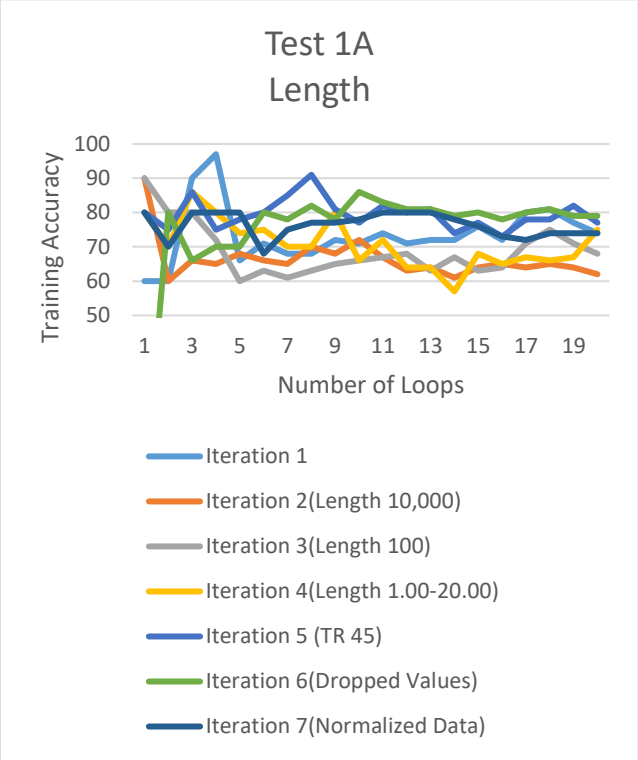

Fig. 5. Test 1A Training Accuracy

TABLE I. TEST 1A VALIDATION ACCURACY

Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Iter. 6	Iter. 7
30%	20%	10%	50%	20%	40%	40%
50%	N/A	N/A	N/A	N/A	N/A	50%
50%	N/A	N/A	N/A	N/A	N/A	40%
AVG: 43%	AVG: 20%	AVG: 10%	AVG: 50%	AVG: 20%	AVG: 40%	AVG: 43%

B. Test 1B

Test 1B performs much better than test 1A for training accuracy (Fig. 3) and validation accuracy(Table 1). Having training accuracy, no lower than 95 across all 3 iterations(Fig. 4) and having a validation accuracy close to those values on average for each of those iterations (Table 2). This is indicative of one of the generic fixes for underfitting, an issue seen from test 1A. That fix being to give the RNN more data to work with. Test A technically contains more comparisons after a few loops. The difference is the amount of correlation between all the data. Test A only compares groups of 10 tasks at a time, meaning the data is more separated. Test B compares 35 tasks at once, creating one large correlational dataset. So, the data is smaller but contains more precision on what specific pattern is present for the RNN to learn.

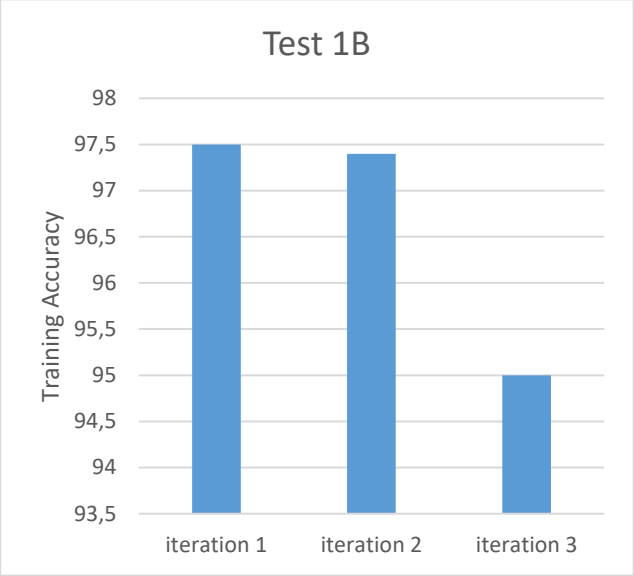

Fig 6. Test 1B Training Accuracy

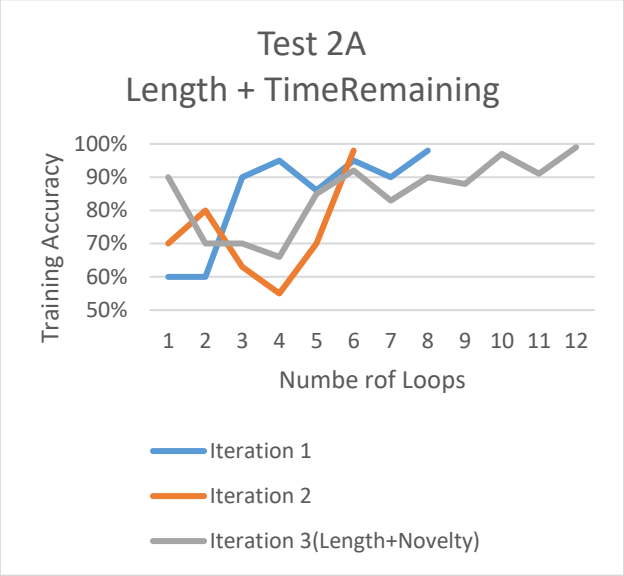

Fig. 7. Test 2A Training Accuracy

TABLE II. TEST 1B VALIDATION ACCURACY

Iter. 1	Iter. 2	Iter. 3
80%	80%	90%
90%	100%	80%
90%	90%	90%
AVG: 87%	AVG: 90%	AVG: 87%

TABLE III. TEST 2A VALIDATION ACCURACY

Iter. 1	Iter. 2	Iter. 3
80%	100%	100%
70%	70%	100%
90%	90%	90%
AVG: 80%	AVG: 87%	AVG: 97%

C. Test 2A

Immediately after increasing the complexity of the pattern, the RNN performs drastically better. At most needing only 12 loops to reach 99% accuracy on training data(Fig. 5). The validation also proves that the RNN learned properly on average across all 3 iterations(Table 3). The RNN only mis-orders one task. Oddly enough when novelty is the second metric the RNN learns better, but slightly slower. This is not entirely surprising as length and novelty have ranges that are capped at 20 and 10 respectively. This would mean a higher chance of repeat values. When two tasks have the same value in the metric being focused on the program returns a zero. Zero usually means task i comes after task j, but in this case it would mean nothing at all. This problem explains why iteration 3 took longer than iterations 1 and 2. 12 loops is not considered to be a terrible amount of time, especially when considering it reached 90% after 7 loops, or when considering it has a nearly perfect validation score, averaging 97% over 3 attempts.

D. Test 2B

Test 2B performs slightly worse than expected. Training accuracy is good across all three iterations(Fig. 6) and when time remaining is the second metric the validation scores are acceptable. When novelty comes second it does not perform so great The validation does worse despite having the best training score(Table 4). Only missing 2 tasks on average is not bad, but considering how well test 2A(Table 3) does it is odd to see test 2B perform poorly. Especially considering the data is larger for test 2B which makes the overfitting issue present all the more baffling. It is curious to note that test 1B(Table 2) performs much better than test 2B when testing validation. While test 1B has slightly lower training scores, the validation scores are much more consistent and don't dip as low as test 2B does on iteration 3.

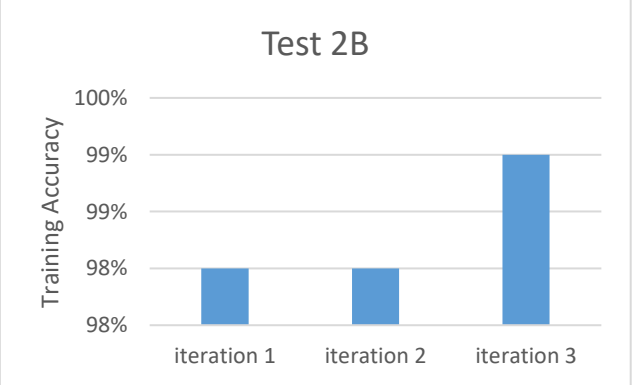

Fig. 8. Test 2B Training Accuracy

TABLE IV. TEST 2B VALIDATION ACCURACY

Iter. 1	Iter. 2	Iter. 3
90%	80%	70%
90%	80%	80%
100%	90%	80%
AVG: 93%	AVG: 83%	AVG: 77%

E. Test 3A

Test 3A does amazingly well. It does take longer than test 2A(Fig. 5) when training(Fig. 7), but the results on unseen data are higher on average(Table 5) than those for test 2A(Table 3). Iteration 3 did take 19 loops to reach the 98% threshold but remained above 90% starting at loop 9. Presumably the results on unseen data would have been similar even if the training broke after 20 loops having not reached 98%.

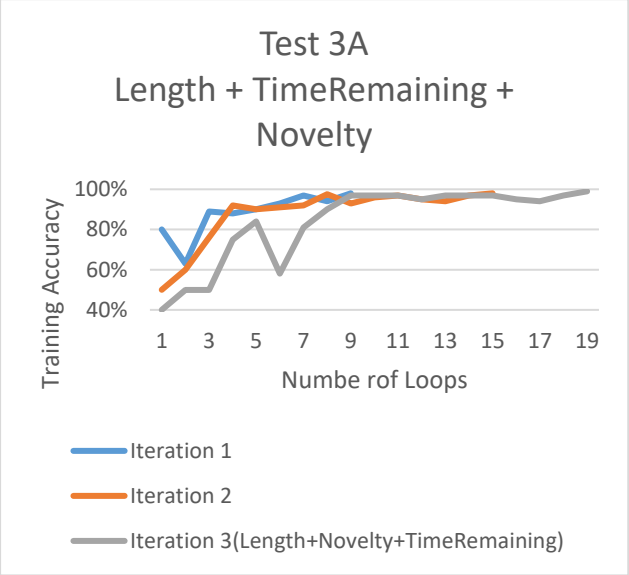

Fig. 9. Test 3A Training Accuracy

TABLE V. TEST 3A VALIDATION ACCURACY

Iter. 1	Iter. 2	Iter. 3
100%	90%	90%
100%	80%	90%
90%	80%	90%
AVG: 97%	AVG: 83%	AVG: 90%

F. Test 3B

Test 3B trained to 99% accuracy all 3 iterations(Fig. 8) and the performance on unseen data averages the best of all versions of test B(Table 7), but not drastically better than test 3A which is what it is meant to be compared directly to(Table 6). It may be faster and more accurate marginally, but preset preferences are not as personalized as ones that are developed over time as users interact with the system.

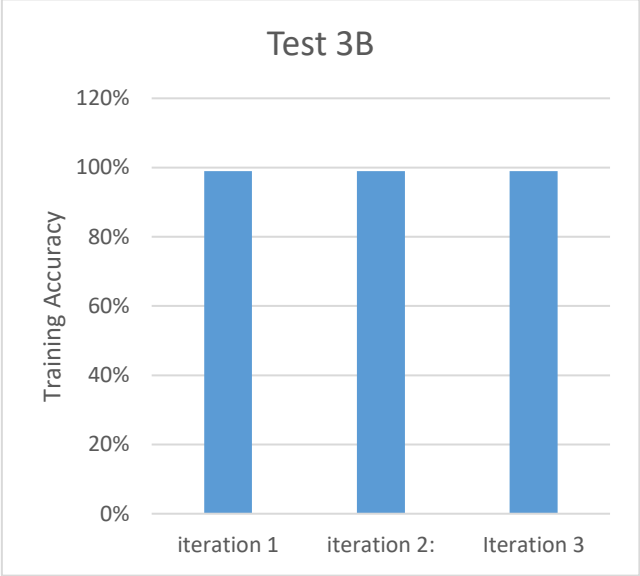

Fig. 10. Test 3B Training Accuracy

TABLE VI. TEST 3B VALIDATION ACCURACY

Iter. 1	Iter. 2	Iter. 3
100%	100%	100%
90%	80%	100%
100%	100%	80%
AVG: 97%	AVG: 93%	AVG: 93%

G. Test 4

Iteration 1 of test 4 caused a crash of the system. The chart shows that the RNN reached 87% accuracy at loop 17 but crashed before it reached 18(Fig. 9). This caused the model for that test to be lost. The pattern showed that the RNN was picking up on the complex logic but was getting stuck. Continuously bouncing between 80% and 90% accuracy. Looking at the results from all iterations it is noticeable that all tests barely break 90% training accuracy. Upon adjusting the batch size, we see a much faster convergence to 98% training accuracy, which is also corroborated by the validation testing(Table 8). Iteration 3 shows that a lower batch size is more important than a higher epoch count. Iteration 5 further supports this. Despite having the lowest training accuracy, it has the second highest validation accuracy, coming second to iteration 4 with the best training and validation accuracies.

Each of these tests, aside from test 1A(Fig. 3, Table 1), present training and validation scores that are very high. This is a very interesting fact to note as the amount of data is relatively small when compared to other datasets used for neural networks. It is a well-known rule of thumb that neural networks perform best when working with very large amounts of data. It helps to prevent the underfitting problem [44]. The datasets for this project are shockingly small, barely breaking even a kilobyte in size. The admirable performance of the RNN with such a sparse amount of data is noteworthy when considering previous iterations of projects concerned with personalized gamification.

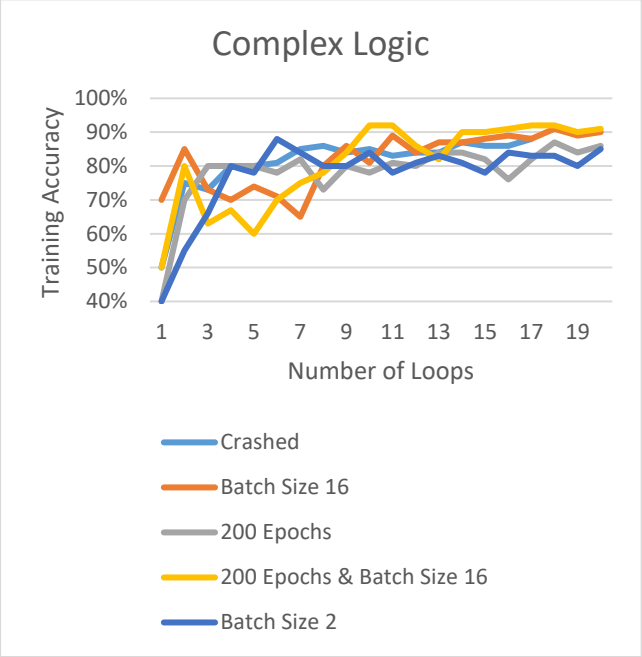

Fig. 11. Test 4 Training Accuracy

TABLE VII. TEST 4 VALIDATION ACCURACY

Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5
N/A	90%	60%	90%	90%
N/A	80%	60%	90%	90%
N/A	70%	70%	90%	80%
AVG : N/A	AVG : 80%	AVG : 63%	AVG : 90%	AVG : 87%

VI. Future Work

The results from this project are indicative of the ability of a RNN to schedule tasks based on learned preferences of a user. This model was achieved with very little data and produces great results. With the RNN's ability to learn scheduling preferences, future work with this model can be expanded to other gamification elements. Those elements should be simple, yet very ingrained into human psychology. User studies should be conducted with this model to get a better understanding both of what the model can learn, and what the model should be asked to learn. Previous personalized gamification studies often seemed to have the issue of trying to personalize too much with too little impact. It seems prudent that future research should focus on finding a balance. "What aspects need to adapt to the user" and "what level of personalization is required to boost engagement levels with the system" are the two questions that have few concrete answers. Determining what is considered successful in terms of improved engagement is also a relevant question to be answered in the future.

In addition to providing better defined parameters for personalized gamification, further tests should be conducted with the model created for this project. Test 1A underperformed drastically. More tests that follow the procedure of test 4 should be conducted, where the parameters of the RNN are adjusted rather than just the input data. In addition, running test 4B is an option to be considered.

Lastly, it would be prudent to determine how the RNN handles a drastic change in user preference. Individuals are usually very set in stone, their preferences unique to them being rather unchanging. However, circumstances may change that force a user to adjust how they approach tasks. It is currently unknown how the RNN presented would handle a sudden change in the data it is being fed and how quickly it could learn this new pattern and forget the old one.

VII. Conclusion

The works of this project resulted in a RNN model that could quickly and accurately determine a scheduling preference and proceed to apply those preferences to unseen task data. This shows that neural networks are potentially capable of learning and applying user preferences to any gamified elements with extreme accuracy. This project also shows that sparse amounts of data is not necessarily a bad thing when regarding neural networks. A comparatively small amount of data was created for this project and the RNN built and used performed near flawlessly with both training and validation scores. This showcases the point that this project was ultimately attempting to prove. Personalized gamification systems are likely to perform faster and more accurately the more targeted the game element is. The theory of personalized gamification is that it should produce far better results than non-personal systems, but in practice does not. This work sought to show that perhaps if the focus was tighter and the system was not trying to personalize every aspect at once, then it would be able to adapt faster.

References

[1] B. Morschheuser, J. Hamari, and A. Maedche, "Cooperation or competition – When do people contribute more? A field experiment on gamification of crowdsourcing," *Int. J. Hum.-Comput. Stud.*, vol. 127, pp. 7–24, Jul. 2019, doi: 10.1016/j.ijhcs.2018.10.001.

[2] A. Rapp, F. Hopfgartner, J. Hamari, C. Linehan, and F. Cena, "Strengthening gamification studies: Current trends and future opportunities of gamification research," *Int. J. Hum.-Comput. Stud.*, vol. 127, pp. 1–6, Jul. 2019, doi: 10.1016/j.ijhcs.2018.11.007.

[3] R. Van Roy and B. Zaman, "Unravelling the ambivalent motivational power of gamification: A basic psychological needs perspective," *Int. J. Hum.-Comput. Stud.*, vol. 127, pp. 38–50, Jul. 2019, doi: 10.1016/j.ijhcs.2018.04.009.

[4] F. Faiella and M. Ricciardi, "Gamification and learning: a review of issues and research," *J. E-Learn. Knowl. Soc.*, vol. 11, no. 3, Sep. 2015, Accessed: Nov. 19, 2023. [Online]. Available: https://www.learntechlib.org/p/151920/

[5] D. Ašeriškis and R. Damaševičius, "Gamification Patterns for Gamification Applications," *Procedia Comput. Sci.*, vol. 39, pp. 83–90, Jan. 2014, doi: 10.1016/j.procs.2014.11.013.

[6] M. Passalacqua, P. D. Sylvain Senecal, M. Fredette, L. Nacke, R. Pellerin, and P.-M. Leger, "Should Gamification be Personalized? A Self-deterministic Approach," *AIS Trans. Hum.-Comput. Interact.*, vol. 13, no. 3, pp. 265–286, Sep. 2021, doi: 10.17705/1thci.00150.

[7] A. Khakpour and R. Colomo-Palacios, "Convergence of Gamification and Machine Learning: A Systematic Literature Review," *Technol. Knowl. Learn.*, vol. 26, no. 3, pp. 597–636, Sep. 2021, doi: 10.1007/s10758-020-09456-4.

[8] A. Knutas, R. van Roy, T. Hynninen, M. Granato, J. Kasurinen, and J. Ikonen, "A process for designing algorithm-based personalized gamification," *Multimed. Tools Appl.*, vol. 78, no. 10, pp. 13593–13612, May 2019, doi: 10.1007/s11042-018-6913-5.

[9] University of Osnabrück *et al.*, "Adaptive and Personalized Gamification Designs: Call for Action and Future Research," *AIS*

Trans. Hum.-Comput. Interact., vol. 13, no. 4, pp. 479–494, Dec. 2021, doi: 10.17705/1thci.00158.

[10] R. Khoshkangini, G. Valetto, A. Marconi, and M. Pistore, "Automatic generation and recommendation of personalized challenges for gamification," *User Model. User-Adapt. Interact.*, vol. 31, no. 1, pp. 1–34, Mar. 2021, doi: 10.1007/s11257-019-09255-2.

[11] E. Nasirzadeh and M. Fathian, "Investigating the effect of gamification elements on bank customers to personalize gamified systems," *Int. J. Hum.-Comput. Stud.*, vol. 143, p. 102469, Nov. 2020, doi: 10.1016/j.ijhcs.2020.102469.

[12] G. F. Tondello, R. Orji, and L. E. Nacke, "Recommender Systems for Personalized Gamification," in *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization*, Bratislava Slovakia: ACM, Jul. 2017, pp. 425–430. doi: 10.1145/3099023.3099114.

[13] I. Rodríguez, A. Puig, and À. Rodríguez, "Towards Adaptive Gamification: A Method Using Dynamic Player Profile and a Case Study," *Appl. Sci.*, vol. 12, no. 1, Art. no. 1, Jan. 2022, doi: 10.3390/app12010486.

[14] F. Rozi, Y. Rosmansyah, and B. Dabarsyah, "A Systematic Literature Review on Adaptive Gamification: Components, Methods, and Frameworks," in *2019 International Conference on Electrical Engineering and Informatics (ICEEI)*, Jul. 2019, pp. 187–190. doi: 10.1109/ICEEI47359.2019.8988857.

[15] S. Suresh Babu and A. Dhakshina Moorthy, "Application of artificial intelligence in adaptation of gamification in education: A literature review," *Comput. Appl. Eng. Educ.*, vol. 32, no. 1, p. e22683, 2024, doi: 10.1002/cae.22683.

[16] W. S. Sayed *et al.*, "AI-based adaptive personalized content presentation and exercises navigation for an effective and engaging E-learning platform," *Multimed. Tools Appl.*, vol. 82, no. 3, pp. 3303–3333, Jan. 2023, doi: 10.1007/s11042-022-13076-8.

[17] V. Bellotti, B. Dalal, N. Good, P. Flynn, D. G. Bobrow, and N. Ducheneaut, "What a to-do: studies of task management towards the design of a personal task list manager," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Vienna Austria: ACM, Apr. 2004, pp. 735–742. doi: 10.1145/985692.985785.

[18] L. R. Fournier, E. Coder, C. Kogan, N. Raghunath, E. Taddese, and D. A. Rosenbaum, "Which task will we choose first? Precrastination and cognitive load in task ordering," *Atten. Percept. Psychophys.*, vol. 81, no. 2, pp. 489–503, Feb. 2019, doi: 10.3758/s13414-018-1633-5.

[19] Y. Gil, V. Ratnakar, T. Chklovski, P. Groth, and D. Vrandecic, "Capturing Common Knowledge about Tasks: Intelligent Assistance for To-Do Lists," *ACM Trans. Interact. Intell. Syst.*, vol. 2, no. 3, pp. 1–35, Sep. 2012, doi: 10.1145/2362394.2362397.

[20] A. Shrestha and A. Mahmood, "Review of Deep Learning Algorithms and Architectures," *IEEE Access*, vol. 7, pp. 53040–53065, 2019, doi: 10.1109/ACCESS.2019.2912200.

[21] D. J. Montana, L. Davis, and M. St, "Training Feedforward Neural Networks Using Genetic Algorithms".

[22] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings, Mar. 2010, pp. 249–256. Accessed: Jan. 31, 2024. [Online]. Available: https://proceedings.mlr.press/v9/glorot10a.html

[23] S. Grossberg, "Recurrent Neural Networks," *Scholarpedia*, vol. 8, no. 2, p. 1888, Feb. 2013, doi: 10.4249/scholarpedia.1888.

[24] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent Advances in Recurrent Neural Networks," Feb. 22, 2018, *arXiv*: arXiv:1801.01078. Accessed: Feb. 05, 2024. [Online]. Available: http://arxiv.org/abs/1801.01078

[25] J. Collins, J. Sohl-Dickstein, and D. Sussillo, "Capacity and Trainability in Recurrent Neural Networks," Mar. 03, 2017, *arXiv*: arXiv:1611.09913. doi: 10.48550/arXiv.1611.09913.

[26] C. Blum and K. Socha, "Training feed-forward neural networks with ant colony optimization: an application to pattern classification," in *Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*, Rio de Janeiro, Brazil: IEEE, 2005, p. 6 pp. doi: 10.1109/ICHIS.2005.104.

[27] R. Kocjančič and J. Zupan, "Application of a Feed-Forward Artificial Neural Network as a Mapping Device," *J. Chem. Inf. Comput. Sci.*, vol. 37, no. 6, pp. 985–989, Nov. 1997, doi: 10.1021/ci970223h.

[28] A. Y. Shamseldin, A. E. Nasr, and K. M. O'Connor, "Comparison of different forms of the Multi-layer Feed-Forward Neural Network method used for river flow forecasting," *Hydrol. Earth Syst. Sci.*, vol. 6, no. 4, pp. 671–684, Aug. 2002, doi: 10.5194/hess-6-671-2002.

[29] G. Bebis and M. Georgiopoulos, "Feed-forward neural networks," *IEEE Potentials*, vol. 13, no. 4, pp. 27–31, Oct. 1994, doi: 10.1109/45.329294.

[30] F. Li, S. Lang, B. Hong, and T. Reggelin, "A two-stage RNN-based deep reinforcement learning approach for solving the parallel machine scheduling problem with due dates and family setups," *J. Intell. Manuf.*, vol. 35, no. 3, pp. 1107–1140, Mar. 2024, doi: 10.1007/s10845-023-02094-4.

[31] S. Liu, C. Zhang, and Y. Chen, "Scheduling Optimization of real-time IOT system based on RNN," in *2020 International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI)*, Dec. 2020, pp. 249–253. doi: 10.1109/ICHCI51889.2020.00061.

[32] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2015. Accessed: Feb. 05, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/hash/e995f98d56967d946471af29d7bf99f1-Abstract.html

[33] K. Kamijo and T. Tanigawa, "Stock price pattern recognition-a recurrent neural network approach," in *1990 IJCNN International Joint Conference on Neural Networks*, San Diego, CA, USA: IEEE, 1990, pp. 215–221 vol.1. doi: 10.1109/IJCNN.1990.137572.

[34] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to Construct Deep Recurrent Neural Networks," Apr. 24, 2014, *arXiv*: arXiv:1312.6026. Accessed: Feb. 05, 2024. [Online]. Available: http://arxiv.org/abs/1312.6026

[35] R. M. Schmidt, "Recurrent Neural Networks (RNNs): A gentle Introduction and Overview," Nov. 23, 2019, *arXiv*: arXiv:1912.05911. doi: 10.48550/arXiv.1912.05911.

[36] R. Eldan and O. Shamir, "The Power of Depth for Feedforward Neural Networks".

[37] A. Arisha, P. Young, and M. E. Baradie, "Job Shop Scheduling Problem: an Overview".

[38] J. Zhang, G. Ding, Y. Zou, S. Qin, and J. Fu, "Review of job shop scheduling research and its new perspectives under Industry 4.0," *J. Intell. Manuf.*, vol. 30, no. 4, pp. 1809–1830, Apr. 2019, doi: 10.1007/s10845-017-1350-2.

[39] D. J. Hoitomt, P. B. Luh, and K. R. Pattipati, "A practical approach to job-shop scheduling problems," *IEEE Trans. Robot. Autom.*, vol. 9, no. 1,pp. 1–13, Feb. 1993, doi: 10.1109/70.210791.

[40] A. Jones, L. C. Rabelo, and A. T. Sharawi, "Survey of Job Shop Scheduling Techniques," in *Wiley Encyclopedia of Electrical and Electronics Engineering*, 1st ed., J. G. Webster, Ed., Wiley, 1999. doi: 10.1002/047134608X.W3352.

[41] A. S. Jain and S. Meeran, "A STATE-OF-THE-ART REVIEW OF JOB-SHOP SCHEDULING TECHNIQUES".

[42] T. M. Willems and J. E. Rooda, "Neural networks for job-shop scheduling," *Control Eng. Pract.*, vol. 2, no. 1, pp. 31–39, Feb. 1994, doi: 10.1016/0967-0661(94)90571-1.

[43] G. R. Weckman, C. V. Ganduri, and D. A. Koonce, "A neural network job-shop scheduler," *J. Intell. Manuf.*, vol. 19, no. 2, pp. 191–201, Apr. 2008, doi: 10.1007/s10845-008-0073-9.

[44] J. G. A. Barbedo, "Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification,"*Comput. Electron. Agric.*, vol. 153, pp. 46–53, Oct. 2018, doi: 10.1016/j.compag.2018.08.013.