Tomorrow's Classrooms: An Exploration of AI's Role in Adaptive Learning

Anuoluwa Akibu

Thesis Advisor: Professor Eric Fouh

Course Coordinator: Professor Norm Badler

ASCS CIS 4980 Senior Capstone Thesis

University of Pennsylvania | School of Engineering and Applied Science

May 6, 2024

ABSTRACT

In higher education, artificially intelligent (AI) learning technology emerged as a transformative pedagogical approach, particularly in adaptive learning systems (ALS). Breakthroughs in AI and machine learning (ML) are helping to enhance intelligent, data-driven algorithms for personalized and optimized learning experiences. However, their implementations have been limited in scale and, as a result, a precise measurement of successful and failing functionality. Thus, we explore the gap between theory and practice in adaptive learning systems. We begin with an overview of adaptive learning and its relationship with AI/ML. Following this, we examine existing academic applications using AI/ML algorithms and their role in adaptive learning. We conclude by discussing the ethics regarding adaptive learning algorithms, potential mitigations, and ideas for bridging the gap.

TABLE OF CONTENTS

I. INTRODUCTION

Adaptive learning describes a responsive teaching method that provides personalized instructional content compatible with the given learners' abilities. This educational approach is a data-driven form of customized learning designed to make real-time analyses and adjustments to students' learning pathways in response to their performance. The foundational framework of ALS revolves around three pivotal components: the learner model, the content personalization techniques, and the continuous, real-time assessments [70]. Each parameter can be specified according to academic features and social, emotional, and psychological content, which are critical factors in student motivation and retention per the learner model requirements. Ultimately, adaptive learning aims to provide a holistic educational experience that meets the learners where they are and assists them in academic success [1, 36].

While adaptive learning can exist both with and without technology, it is becoming evident that intelligent technology has the potential to revolutionize the way we approach teaching and learning in education. Computer-based education blossomed in the 1970s with the increasing accessibility of personal computers and flourished with the introduction of the World Wide Web in the early 1980s. Many contemporary adaptive learning initiatives leverage adaptive learning technologies (ALTs), often supported by quickly advancing AI and ML-base algorithms [2]. The core objectives of the technologies supporting adaptive learning are automation, sequencing, assessment, real-time data collection, and self-organization [1]. The three championing algorithms used to achieve these core requirements of an adaptive system are feedforward neural networks (FFNNs), Bayesian neural networks (BNNs), and Recurrent Neural Networks (RNNs) [3, 22]. By applying these AI-backed structures, adaptive learning platforms can now analyze vast amounts of data on learner behavior and performance, enabling unprecedented levels of personalization.

We will provide an in-depth analysis of adaptive learning within the context of higher education. Beginning with a historical overview, we will examine the traditional educational methodologies and the pedagogical challenges ALSs present. We will then transition into a discussion on the advent of adaptive learning, highlighting its evolution and the pivotal role of AI in shaping its current and future trajectory. The core components and mechanisms that define adaptive learning systems, including personalized content delivery, learner assessment, and competency frameworks, will be explored in detail [1]. Through a critical lens, we will also address the potential pitfalls and ethical considerations associated with the deployment of AI-driven adaptive learning systems, including but not limited to algorithmic bias and data privacy concerns, providing a balanced perspective crucial for informing new and updating old educational practices and policies.

II. A BRIEF HISTORY OF ADAPTIVE LEARNING IN EDUCATION

Recognition of diverse learner needs and the efficacy of adaptive technologies in enhancing student outcomes has been a concept for decades. Typical teaching methods (i.e., standardized lectures, textbooks, examinations, etc.) designed to cater to an average student profile were combined with adaptive techniques such as one-on-one tutoring and individualized learning plans (i.e., AP, English language learner classes, special education, Socratic seminars, etc.). However, implementing strategies to address diverse learner needs faces challenges such as limited resources, accessibility issues, cultural and linguistic diversity, and socioeconomic disparities, all of which require a concerted effort and commitment to overcome. Thus, instructors and institutions have continuously sought more flexible, responsive, and engaging teaching models to address these challenges, leading to the development and adoption of adaptive learning technologies.

2.1 Origins and Early Innovations

The varying scales at which tailored educational experiences are designed and implemented are often a hybrid of macro- and micro-level adaptation approaches.

2.1.1 Macro-Adaptive Systems

Macro-adaptive systems operate broadly, focusing on overarching decisions regarding instructional goals, curriculum content, and delivery systems. The primary objective is to accommodate diversity among learners by offering different alternatives or pathways to meet their needs effectively. Essentially, students are organized into different learner groups based on similar learning styles, goals, and capabilities, and the instruction is tailored to each group's needs and requirements. The key part here is that students are not just placed into diverse, homogeneous groupings but also receive different curricula and resources based on that grouping. For example, higher achieving students may have a curriculum that differs from the one used for students who need more support, or different majors will have different instructional systems [4].

2.1.2 Micro-Adaptive Systems

Micro-adaptive systems operate more granularly, focusing on individual learners' specific learning needs during instruction. The primary objective is to diagnose each learner's strengths, weaknesses, and learning preferences in real time and provide personalized instructional prescriptions to address those needs. The system uses real-time performance data (response time, accuracy, emotional state, etc.) to perform personalized interventions or updates to ongoing instruction, categorized into diagnostic or prescriptive measures. For example, a smart tutoring system operating at the micro-adaptive level would diagnose which students need more help or more challenging activities and prescribe guided content or additional practice questions, respectively [4].

2.1.3 Early Innovations of Macro-Adaptive Systems: The Keller Plan

The earliest applications of ALSs trace back to the foundational principles of macro-adaptive learning systems, primarily manifested through modifications to curriculum or pedagogical

methods rather than technology. In the latter half of the twentieth century, personalized systems of instruction (PSIs) gained significant popularity after psychologist Fred S. Keller developed the Keller Plan in the 1960s. The Keller Plan is a mastery-based approach to teaching that emphasizes individualized instruction, self-paced learning, and frequent formative assessment. It outlines five foundational elements of PSI: 1) subject mastery, (2) proctors, (3) self-pacing, (4) written materials (i.e., textbooks, written assignments, etc.), and (5) motivational lectures [5]:

- **Mastery:** In a standard PSI class, course content is broken down into smaller parts. Mastery of one part is a prerequisite for progression to the next, without penalization for incorrect answers, emphasizing a thorough understanding of each topic.
- **Proctors:** Proctors serve as teaching assistants for the class to provide feedback and other student support services.
- **Self-Pacing:** PSI courses were designed to be unbounded by traditional academic timelines (i.e., semesters, trimesters, etc.) to allow students enough time to master the course material at their own pace.
- **Written Material:** Engaging with instructional content outside the classroom is a cornerstone of PSI. Students are expected to interact with detailed written materials, such as textbooks and assignments, to reinforce their learning independently.
- **Motivational Lectures:** Since students are expected to learn outside of the classroom on their own time, classroom lectures are meant to encourage students in their learning, and educators are meant to provide any assistance that makes the learning process easier rather than teach.

The Keller Plan is often used in higher education settings but can be adapted for various educational levels and subjects. While its popularity waned in subsequent years due to new, state-of-the-art technologies, the Keller Plan's impact on self-paced learning remains evident in the continued research of student-centered instructional methods and the principles of mastery-oriented education. Other examples of macro-adaptive applications inspired by the Keller Plan include the Individually Guided Education (IGE) and Individually Prescribed Instructional System (IPI) developed in the mid-to-late 1970s, which also served as comprehensive models for future macro-adaptive solutions [5].

2.1.4 Early Innovations of Micro-Adaptive Systems: Teaching Machines

While these historical models for adaptive learning were powerful tools, they required tools to be implemented at scale in the classroom. Born out of this need were the roots of micro-adaptive learning, which trace back to the 1920s, when early innovators like Sidney L. Pressey and B. F. Skinner laid the groundwork for its development. The two American psychologists explored the principles of individualized instruction and behaviorism through teaching machines.

Pressey is often credited with inventing the first teaching machine in the 1920s. His teaching machines were mechanized devices equipped with instructional content, such as text, images,

or audio recordings, along with mechanisms for sequentially presenting questions or problems and providing immediate feedback on users' responses. If a user provided the correct answer, the machine would move them to the next question. If a user provided an incorrect answer, the machine recorded it and remained on the question until the learner chose the correct answer. This design allowed students to work at their own pace and learn from their mistakes.

Skinner expanded upon Pressey's design, creating a machine that focused more on the learning process than simply serving as a self-testing device. Skinner's machine emphasized carefully curated programming material, as he believed that the content and the order of its presentation were essential to learning complex behavior effectively. The goal of Skinner's machine was to function as a personal tutor for the student, offering gradual learning, immediate reinforcement, engaging content, and real-time assessment of student understanding for educators. However, these teaching machines were often analog devices operated by simple mechanisms like buttons or levers, limiting their adaptability to all types of programming material (e.g., verbal, written, or kinesthetic). Additionally, the frame-by-frame presentation of material could be time-consuming and unsuitable for all learner styles or content, potentially impacting student motivation. Despite these limitations, teaching machines served as precursors to today's intelligent tutoring systems and contributed to the evolution of psychological philosophies and practices in academia [6].

2.1.5 The Prelude of AI in Educational Settings

Educational methodologies and their corresponding instructional materials continue to evolve, incorporating psychological theories and philosophies in a standardized, quality-driven manner. In pursuit of adaptive agents, or intelligent systems capable of dynamically adjusting to individual learner needs, several AI movements emerged in the late 1970s. These movements often occurred once funding opportunities increased and reinforced the development of more sophisticated, adaptive learning tools. For instance, between 1980-1987, AI research shifted towards creating expert systems and domain-specific applications, diverging from the earlier reliance on general-purpose search mechanisms or "weak methods." Adopting robust, domain-specific knowledge enabled the design of systems proficient in conducting intricate reasoning within specialized areas, thereby enhancing the performance and efficiency of AI applications. Among the noteworthy advancements during this era were Rodney Brooks' contributions, including the development of Cog, iRobot, and Roomba and the introduction of Gammonoid, the first software to win a world championship in Backgammon. These technological strides would not have been possible without the substantial improvements in computer hardware, which promoted the transition of computers from bulky, inaccessible units to personal and approachable devices. This evolution mirrored a broader societal shift towards individualism, characterized by a growing demand for personalized technology solutions that resonated with users' unique preferences and requirements. In essence, the AI movements of the twentieth century significantly propelled the intersection of AI with educational methodologies, ushering in a new age of adaptive learning tools, systems, and philosophies requiring more interdisciplinary research than ever before to craft these technologies [7, 23].

III. THE ADAPTIVE LEARNING FRAMEWORK

An actual ALS integrates three essential components to enhance the learning experience. First, it features a **learner model** that dynamically (and accurately) profiles each user by tracking and analyzing their performance, preferences, and interactions. Secondly, it delivers **customized educational content** that fits each learner's specific requirements based on their learner model. Lastly, **real-time feedback** is provided to learners as they engage with the material, allowing learners to adjust their study plans and improve efficiency in real-time [1, 70].

3.1 Learner Model

- **Metadata:** Metadata such as social, academic, mental, and emotional identifiers must be incorporated into the learner's profile to personalize the content they receive.
- **Prerequisite Knowledge and Prior Knowledge Qualifiers:** The ALS must set a baseline to determine what the student already knows and still needs to know. Without a baseline, the ALS will be unable to properly engage with the learner content that is suitable for the learner. The baseline is often set using what the ALS gathers about the following after the initial knowledge assessment.
  - **Competencies/Sub-Competencies:** Core competencies refer to the knowledge needed for proficiency in a field or discipline. In ALSs, core competencies should be composed of smaller sub-competencies that students gradually master to learn the core competency.
  - **Skill Standards Libraries:** Skill standard libraries provide ALSs with a structured, pedagogical framework for identifying, organizing, and categorizing the essential skills, knowledge, and abilities a student needs for mastery.

3.2 Content Personalization

- **Content Interoperability:** Content should be adaptable, flexible, and dynamic, updating to match the student's current learning abilities, course material, and any new information gathered about the learner. This ensures that the content remains relevant and engaging for each student.
- **Social Interaction:** The system correlates learners through collaborative content based on their attributes (such as past grades, behaviors, or current mental health state). This communal aspect enhances learning through shared resources and interactions.

3.3 Assessment

- **Diagnostic Classification Modeling:** An ALS must be able to diagnose what the learner needs, identifying areas of strength and weakness to guide personalized instruction and support.
- **Normed vs. Criterion-Referenced Assessments:** Assessments measure students' performance relative to others or the knowledge they should know. Normed-referenced

assessments compare students to a predetermined "norm group" of their peers with similar characteristics (i.e., age, previous test taker, learning style, etc.). In contrast, criterion-referenced assessments evaluate performance against standard criteria or results. Assessment should incorporate predictive psychometric design to determine the right learning pathway (i.e., content sequence) for each student based on their answers.

- **Zone of Proximal Development:** The system should consider the Zone of Proximal Development, which refers to the difference between what a learner can do on their own and what they can achieve with support from a more knowledgeable individual, typically a teacher, tutor, or peer. This helps determine when a student needs more or less content scaffolding.

- **Self-Assessment:** Self-assessment allows students to compare what they believe they have learned to what the system has gathered about their knowledge and understanding. This promotes metacognition and self-awareness, empowering students to take ownership of their learning process.

3.4 Assessing an Adaptive Learning System

When evaluating the effectiveness of an adaptive learning system, it is imperative to assess its capability to dynamically tailor learning experiences, provide accurate feedback, and foster learner engagement. These core requirements collectively contribute to what constitutes a genuine adaptive learning system. First and foremost, personalization stands as the foundation of adaptive learning, epitomizing its ability to discern individual students' unique needs, preferences, and learning styles. A good ALS employs sophisticated algorithms to construct and maintain comprehensive learner profiles, incorporating factors such as prior knowledge, learning pace, and preferred modalities to tailor learning pathways accordingly. Moreover, an effective ALS should seamlessly adapt to various types of content without compromising the integrity of the adaptive process. For example, an ALS that offers diverse content formats, including text, multimedia, simulations, and interactive exercises, gives students more onus in how they engage with the course content. Secondly, feedback mechanisms facilitate continuous adaptation and improvement within adaptive learning systems. Feedback helps ALSs understand how to positively influence the third evaluation criterion: learner engagement. Learner engagement is a crucial determinant of an adaptive learning system's effectiveness, impacting motivation, retention, and comprehension. ALSs that sustain student motivation and enthusiasm for learning, enhancing retention and comprehension over extended periods, are considered satisfactory. In summary, a "good" adaptive learning system excels in personalization, feedback mechanisms, and learner engagement, thereby fostering optimal learning experiences and outcomes for students across diverse educational settings [1, 70]. The following section will explore the main AI methods employed in adaptive learning systems to achieve these objectives.

IV. COMMON AI TECHNIQUES USED IN ADAPTIVE LEARNING

This section delves into the primary AI-based learning methods pivotal in tailoring educational experiences, emphasizing the versatility and depth of ML technologies. We begin by examining a precursor to artificial neural networks (ANNs), Decision Trees (DTs), and the challenges associated with this traditional AI method. We then introduce the fundamental components and architectural blueprints of ANNs, along with the principles of nonlinearity that drive their learning capabilities. Further, we expand our discussion to include BNNs and RNNs, each representing unique models within machine learning. These methodologies illustrate the broad spectrum of ML techniques employed to make informed learning path predictions (i.e., skipping, including, or revisiting lessons) given observed data such as engagement level and past assessment lengths.

4.1 A Traditional AI Algorithm: Decision Trees

DTs in adaptive learning originated from ML and data mining methodologies, initially developed for classification and regression tasks. They are supervised learning algorithms, and their adaptation to educational contexts provides a hierarchical approach to decision-making based on learner attributes and learning content features. The primary goal of DTs in machine learning is to create a predictive model that can accurately classify or predict outcomes based on a set of input features.

In classification tasks, where the target variable is categorical, DTs assign data points to predefined classes based on their features. For instance, a classification decision tree could categorize students into distinct learning styles—visual, verbal, auditory, kinesthetic, or a combination—by examining engagement patterns and performance across various content types like text, videos, and hands-on activities. This approach enables tailoring teaching methods and materials to better align with each student's learning preferences. Conversely, regression tasks involve a continuous target variable representing a numerical value. A regression decision tree might predict students' future test scores by leveraging historical data, including past performance, study duration, and attendance. This predictive capability allows for the customized tailoring of interventions to enhance individual student outcomes, demonstrating the versatile application of DTs in addressing both categorical and continuous data scenarios within educational settings.

The architecture of DTs in adaptive learning comprises nodes representing attributes or features, branches denoting the details of the decision, and leaf nodes indicating outcomes, as illustrated in Figure 1.

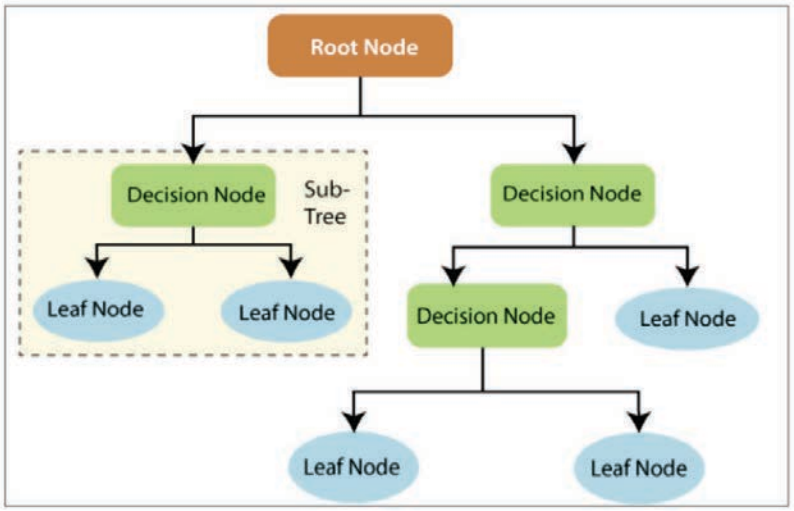

Figure 1: An illustration of the decision tree structure and its components: the root node, decision node, and leaf node [16].

Starting from the top, the root node acts as the starting point, containing the source set of input features that inform the decision-making process. As we traverse the tree, we find internal or decision nodes, each labeled with a feature from the source set. These nodes signify the criteria based on which decisions are made, guiding the path taken through the tree. At the culmination of each decision path lie the leaf nodes, or terminal nodes, representing a distinct, homogeneous class of the target feature, essentially the possible outcomes derived from the source set. These leaf nodes signify the final classification decisions made by the tree. Connecting these components are the edges, or branches, of the tree. Each branch delineates a decision point that leads to a subsequent decision node for further analysis based on additional input features or to a leaf node, marking the conclusion of the decision-making path with a definitive classification outcome. This structured approach enables the decision tree to systematically break down complex decision-making processes into more straightforward, manageable parts, ultimately leading to precise classification or regression outcomes [16, 17].

4.1.1 Limitations of Decision Trees

Despite their ease of implementation, DTs often encounter several limitations affecting their performance and applicability. DTs are prone to overfitting, mainly when dealing with complex, noisy, or high-dimensional data [18, 19, 20]. Additionally, DTs exhibit brittleness, meaning they can drastically change structure with slight variations in training data, leading to challenges in generalization across different datasets [50, 51]. The inherent sensitivity to noise and the inability to capture complex nonlinear relationships without significant complexity further complicate their use in practical scenarios. This section explores these limitations, detailing how they impact the effectiveness of DTs and model performance relative to their more novel counterparts.

Firstly, complex DTs tend to overfit and thus do not generalize well to new data, unlike their more advanced counterparts, such as ANNs. As the sample size increases, neural networks

perform better than DTs. This scalability is crucial in practical applications where large datasets are standard. Neural networks can handle high dimensional data better by using it to fine-tune their parameters, enhancing their predictive accuracy. In contrast, DTs may suffer from increased overfitting without careful pruning and complexity control [18, 19, 20]. For instance, in Kim [2008], the performance of ANNs improved relative to DTs as the sample size increased. With five independent variables (including categorical variables), the Root Mean Squared Error (RMSE) for ANNs decreased more significantly compared to DTs as the sample size went from 100 to 10,000 [51].

Secondly, their tendency towards overfitting makes them highly sensitive to variations in the data and ultimately brittle. The depth of a decision tree is a primary contributor to its variance. A deeper tree with more branching can capture more details of the training data. While this might improve accuracy on the training set, it can lead to overfitting, where the model captures noise, assuming it is a significant pattern. The model's variance increases as the tree's depth increases because it becomes more sensitive to fluctuations in the training data [18, 19, 20]. Additionally, DTs aim to create leaf nodes that are as pure as possible. In pursuit of purity, a tree might create splits that are too specific and do not reflect genuine relationships between the nodes. This pursuit can increase the tree's complexity, leading to higher variance [50, 51]. Unlike DTs, which make all decisions based on a single pass through the nodes from root to leaf, neural networks process data through layers where each layer captures different levels of abstraction. This layered approach helps control the complexity of the model, as lower layers do not directly make final decisions but contribute to a more distributed learning process. Furthermore, neural networks optimize their parameters through backpropagation, a concept discussed later. This continuous and iterative refinement helps ensure that neural networks find a balance between accuracy and generalization, unlike DTs that optimize for local purity or accuracy at each node independently, often at the expense of overall performance [10].

Lastly, DTs often cannot handle data with uncertain or missing values effectively. In many real-world scenarios, the class labels or attributes may have a certain level of uncertainty that needs to be modeled. This is, again, where neural networks step in [50]. Neural networks can be adapted and designed to handle various data and tasks. This versatility is achieved through changes to the amount and type of layers, connections, statistical inferences, and so on, resulting in different architectures, such as the BNN, capable of modeling uncertainty [13].

Overall, DTs, though a simple and intuitive AI approach, possess significant drawbacks in dynamic and uncertain environments, both of which describe adaptive learning systems. With their sophisticated structure and learning capabilities, neural networks offer a stronger and more versatile solution for the complex and large-scale data modeling challenges that adaptive learning systems bring.

4.2 Artificial Neural Networks

ANNs are machine learning models distinguished by their capacity to assimilate, encode, and apply knowledge to execute complex tasks. Their ability to mirror the dynamic learning process inherent in human cognition is central to our exploration of adaptive learning

systems.

4.2.1 Components of an Artificial Neural Network

The building blocks of a neural network are the nodes (also known as processing units, perceptrons, etc.), which can be organized into three layers: the input layer, the hidden layer(s), and the output layer. As depicted in Figures 2 and 3, the input layer receives the raw data and transports it to the succeeding layer, and nothing is transformed here. The hidden layer processes the data from the input nodes by applying nonlinear functions called activation functions. The output layer receives the processed data and outputs a prediction. These layers are interconnected through weighted, directed edges. The flow of information in a neural network is facilitated by the components visualized in Figure 2 [8, 24, 25, 26].

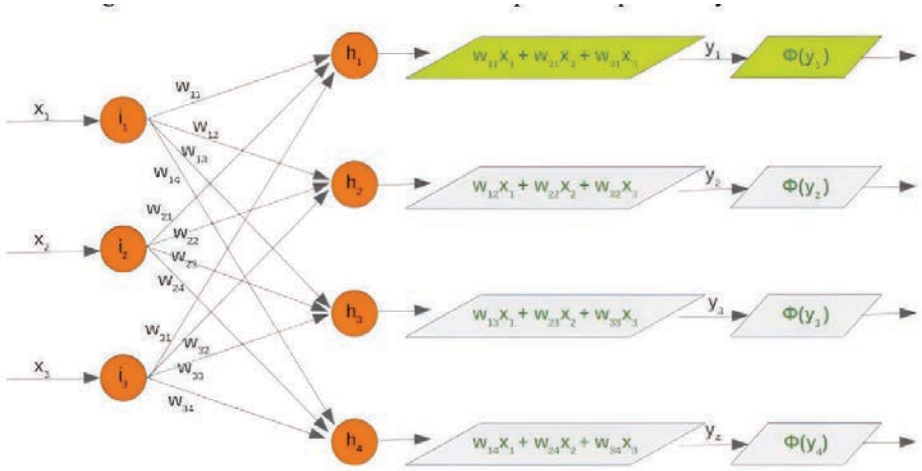

Figure 2: The first half of a neural network, specifically a feedforward (acyclic) neural network. This section depicts the connections between the input layer and hidden layer leading to the outputs of the hidden layer [8].

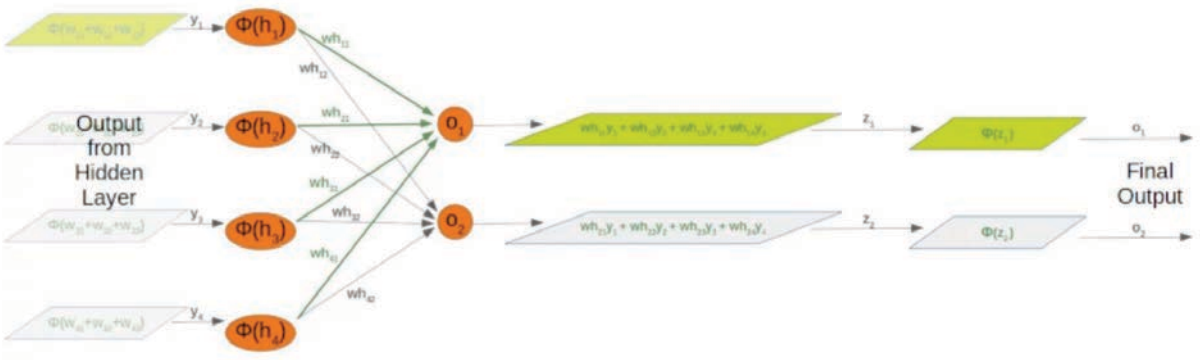

Figure 3: A continuation of the neural network shown in Figure 2. It illustrates the connections between the hidden and output layer leading to the final outputs of the network [8].

- **Input Vector, x:** The raw data input to the neural network, where each element corresponds to a feature or attribute of the data.

- **Weights, w & Biases, b:** The learnable parameters of the network that describe its behavior and performance.

- **Weight Matrices, W_{ih} & W_{ho}:** The matrices that determine the strength of influence of inputs on outputs. W_{ih} describes the connections between the input and hidden layer while W_{ho} describes the connections between the hidden and output layer.

- **Hidden Vector, h:** An internal vector containing the information learned by the network. h is computed through a dot product operation between the input vector and the weight matrix, followed by activation through an activation function

- **Output Vector, o:** The neural network's final output. o is typically processed through an activation function like softmax to produce a probability distribution.

4.2.2 The Training Process: How the Artificial Neural Network Learns

Before training begins, the appropriate construction of these neural network components must be selected based on the problem at hand. In a neural network, knowledge, or patterns and information the network has learned, is stored within the parameters of the network, such as the weights and biases. Determining the number of layers, types of layers and connections between them, number of nodes in each layer, types of activation functions, parameters, and so on can significantly impact how these models acquire, represent, and utilize knowledge to perform tasks effectively [10].

With this understanding of neural network architecture and knowledge acquisition in mind, let's delve into the specifics of the training process. As our running example, we will use the three-layer feedforward network, pictured in Figure 2. During training, the network iteratively adjusts its parameters to minimize the difference between its predictions and the actual outputs, effectively refining its understanding of the underlying patterns in the data. This process involves several key steps:

1. **Initialization:** Initially, the network parameters, including weights and biases, are randomly initialized. This sets the starting point for the training process.

2. **Backpropagation:** Backpropagation is a commonly used algorithm for learning or training the parameters. It consists of two phases: the forward phase and the backward phase. During the forward phase (forward propagation), input data is fed into the hidden layer. The weight matrix W_{ih} governs the connections between the input and hidden layers. To compute the hidden layer's outputs, the input vector x undergoes a dot product operation with W_{ih}. A bias term b may also be added to each neuron's activation to allow for shifting along the activation function's axis [8, 10]. The combined result of the dot product operation and bias term forms an intermediate vector y, the argument to an activation function as shown below

$$
\begin{aligned}
y &= w_1x_1 + w_2x_2 + ... + w_nx_n + b \\
&= \sum_{i=1}^{n} w_ix_i + b \text{ [10]}
\end{aligned} \tag{1}
$$

The activation function is applied element-wise to the y, producing the final output vector h of the hidden layer:

$$
h = \phi(\sum_{i=1}^{n} w_ix_i + b) \text{ [8]} \tag{2}
$$

Activation functions play a crucial role in learning by introducing nonlinearity to the network's computations. Nonlinearity allows neural networks to capture complex patterns and relationships in the data (i.e., recognizing learner speed, determining comprehension level, grouping learners) that are not immediately apparent or are too complex for conventional, linear methods. The activation function determines both the range and the interpretation of the output value, enabling the network to transform input features into higher-dimensional representations and enhancing its capacity to learn and represent knowledge effectively [9]. For neural networks, this activation function is either Rectified Linear Unit (ReLU)

$$
f(x) = max(0, x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \text{ [9]} \tag{3}
$$

or sigmoid,

$$
\sigma(x) = \frac{1}{1 + e^{-x}} \text{ [52].} \tag{4}
$$

Similarly, the connections between the hidden layer and the output layer are represented by a weight matrix W_{ho}. The output vector o is calculated in a similar fashion to h, but recognizing that the typical output of a neural network is a probability distribution, an activation function like softmax,

$$
\sigma(\overrightarrow{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ [11]} \tag{5}
$$

is applied instead to ensure it represents probabilities.

If the predicted output of the neural network is not similar to the expected output, then the backward phase, or backpropagation, ensues. The primary objective of this phase is to minimize a cost function or loss function, which measures how well the model performs on a given task. The computation of loss functions differs based on the availability of labeled data, which is determined by the learning algorithm applied. Supervised and unsupervised learning are two fundamental types of algorithms in machine learning that determine how the weights, or strength of connections between the perceptrons, are modified throughout backpropagation [10].

4.2.3 The Supervised Learning Approach

In supervised learning, algorithms are trained on labeled data, where the correct outputs are provided, guiding the adjustment of parameters based on the error between predicted and actual outputs. An iteration of training under supervised learning begins with comparing the predictions generated by the network to the exact target values using a loss function, typically the mean squared error (MSE) method. After the loss has been computed, the next step is to update the network's weights to minimize this loss. The algorithm performs a gradient descent, which iteratively updates the model's parameters in the direction that reduces the loss. This direction is determined by calculating the derivative of the loss function J with respect to each model parameter. The derivative indicates how the J changes concerning changes in the model parameters. The update rule for gradient descent specifies how to adjust the model's parameters based on the derivative of the loss function. It typically involves subtracting a fraction of the derivative (scaled by a learning rate, α) from the current weight value, w, to get the weight of the next time step, w_{t+1}, as shown in Equation 6.

$$w_{t+1} = w_t - \alpha \frac{\partial J}{\partial w_t} \text{ [12]} \tag{6}$$

Here, α is the learning rate, which is a positive scalar determining the size of the step, and $\Delta J(\theta)$ is the gradient of the objective loss function J concerning the parameters θ [12].

The learning rate is a crucial parameter of the gradient descent as it determines the length of the training process. A high rate may expedite the network's convergence on an output but risks overshooting, while a low rate ensures stability at the cost of longer training times. Typically, $\alpha = 0.01$ is used, but it can also be calculated to be iteration dependent rather than a simple constant using either exponential decay,

$$\alpha_t = \alpha_0 e^{-kt} \text{ [12]} \tag{7}$$

inverse decay,

$$\alpha_t = \frac{\alpha_0}{1 + kt} \text{ [12]} \tag{8}$$

or potential decay

$$\alpha_t = k^t \alpha_0 \text{ [12]}. \tag{9}$$

Once the weights are updated, the loss function is computed again using the updated parameters and the current training data. This iterative process continues until a stopping criterion is met, such as reaching a certain number of iterations, attaining convergence, or observing no significant improvement in the loss function [12].

Supervised learning encompasses two primary types of tasks: classification and regression, distinguished by the nature of the target variable. In classification tasks, the objective is to categorize input data into predefined classes or labels. This involves mapping input features to discrete output categories. For instance, consider a scenario where a neural network is tasked with predicting a student's major based on the classes they have taken. Here, the network's goal is to assign each student to one of the predetermined majors correctly.

Conversely, regression tasks involve predicting continuous numerical values as the output. In this context, the network learns to map input features to a continuous target variable. For example, in a dataset containing students' grades from previous semesters, the task might involve predicting a student's GPA for an upcoming semester. Unlike classification, where the output is a discrete label representing a class, regression outputs a continuous value, such as a GPA score. These distinctions guide the design of supervised learning algorithms and the interpretation of their outputs, catering to different types of predictive modeling tasks based on the nature of the target variable [10].

4.2.4 Challenges and Limitations of Neural Networks

Navigating the landscape of neural networks, we encounter various challenges and limitations, including concerns regarding overfitting, vanishing gradients, computational complexity, and scalability, which collectively impact their efficacy and applicability in diverse contexts. Overfitting occurs when a model, likely one with too many parameters relative to the number of observations, learns the noise in the training data instead of the underlying distribution. Additionally, the issue of vanishing gradients is prominent in deep neural networks. When this happens, the gradient of the loss function with respect to the weights may become increasingly small as it is propagated back through the network. The computational complexity can be roughly estimated as $O(nm)$ for a fully connected layer where each node in the first layer is connected to each node of the subsequent layer. n is the number of nodes in the first layer, and m is the number of nodes in the second. Lastly, scalability concerns arise from this significant computational need. That said, the challenge is not only computational but also statistical, as models must maintain or improve performance without succumbing to issues like overfitting [10]. As these issues are addressed, the potential for neural networks to revolutionize more domains remains vast, from automating complex tasks to enabling breakthroughs in predictive analytics. Alternative architectures like RNNs and Bayesian Networks are introduced to address some of these issues.

4.3 Recurrent Neural Networks

Recurrent Neural Networks are designed for sequential data, allowing previous outputs to be used as inputs while having hidden states, thus helpful in creating adaptive models that better understand and predict based on evolving learner patterns. Unlike in conventional FFNNs, information can be passed into and processed by itself, allowing it to consider the current and past inputs as depicted:

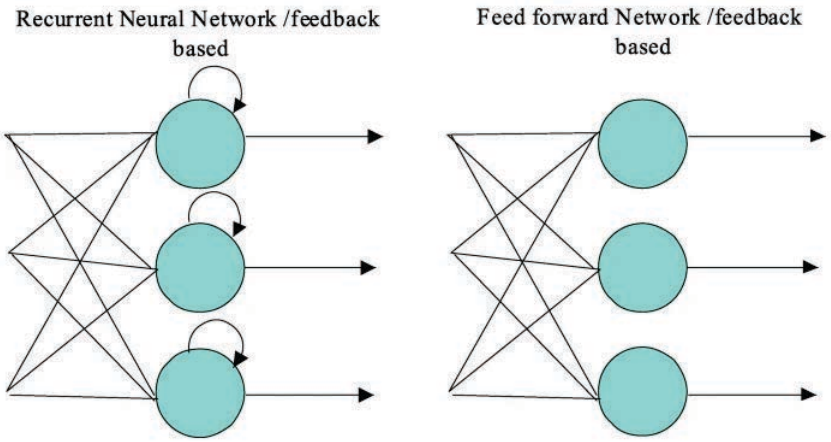

Figure 4: The recurrent neural network and a feedforward neural network [54].

4.3.1 Architecture of a Recurrent Neural Network

The mathematical notation, pictured in Figure 5, describing the RNN architecture contains similar components as FFNNs in addition to a weight matrix $W_{hh} \in \mathbb{R}^{hxh}$ to account for the loops.

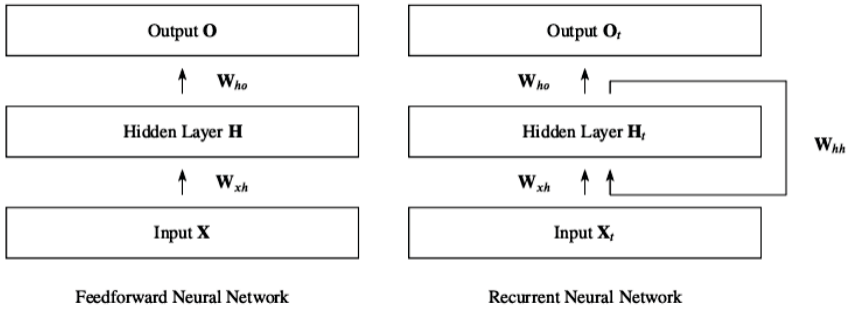

Figure 5: Comparison between the mathematical notations of RNNs and FFNNs [53]

The slight differences in notation seen moving forward are because the core of an RNN's operation lies in its ability to retain information across time steps, t, as seen in the state update equation below

$$H_t = \phi_h(X_t W - ih + H_{t-1} W_h h + b_h) \text{ [53]} \tag{10}$$

where H_t is the hidden state at time t, X_t is the input at time t, W_{ih} and W_{hh} are weight matrices for the input to hidden and hidden to hidden states respectively, b_h is the bias, and ϕ_h is the activation function typically sigmoid or hyperbolic tangent function (tanh),

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \text{ [53].} \tag{11}$$

Tanh is often used due to its output range of [1, 1], which centers the activation outputs around zero, enhancing the stability and efficiency of learning. The symmetric nature of tanh around zero provides balanced gradients, reducing bias in updates and improving convergence during training [52].

The output of an RNN at time t, O_t, is

$$O_t = \phi_o(H_t W_{ho} + b_o) \text{ [53]} \tag{12}$$

where O_t is the ouput at time t, W_{ho} is the weight matrix from the hidden state to, b_0 is the output bias, and ϕ_h is the output activation function [53].

4.3.2 The Training Process: How a Recurrent Neural Network Learns

Training RNNs involves adjusting weights to minimize error over sequences, which is achieved through Backpropagation Through Time (BPTT). BPTT simply performs backpropagation on an unrolled RNN visualized below [53].

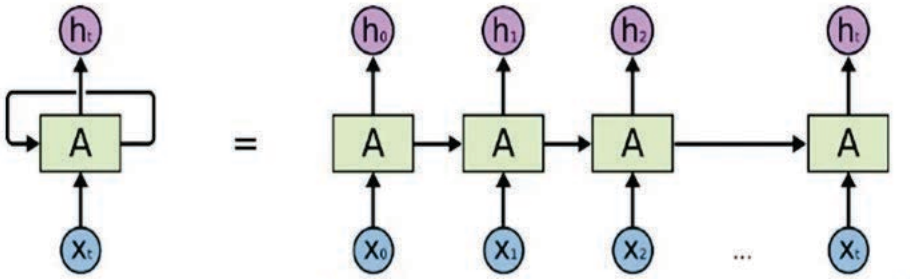

Figure 6: An unrolled RNN expanded across each time step representing the sequential dependencies underpinning their design [54].

For BPTT, the gradients of the loss function concerning the weights are computed by unfolding the network through time and applying the chain rule. For instance, the loss function is given by Equation 13.

$$\mathcal{L}(O, Y) = \sum_{t=1}^{T} \ell_t(O_t, Y_t) \text{ [53]} \tag{13}$$

The gradients of the loss function with respect to the weights are computed by unfolding the network through time and applying the chain rule [54]. The gradient with respect to W_{ho} is computed by Equation 14.

$$\frac{\partial \mathcal{L}}{\partial W_{ho}} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial O_t} \cdot \frac{\partial O_t}{\partial \phi_o} \cdot \frac{\partial \phi_o}{\partial W_{ho}} = \sum_{t=1}^{T} \frac{\partial \ell_t}{\partial O_t} \cdot \frac{\partial O_t}{\partial \phi_o} \cdot H_t \text{ [53]} \tag{14}$$

Here, ℓ_t is the loss at time t, and $\mathcal{L}$ is the cumulative loss over all time steps T [53].

As with standard neural networks, RNNs also experience issues with vanishing or exploding gradients. This problem occurs because RNNs process data sequentially, using matrix multiplication to pass information from one time step to the next. If the values in this multiplication are small (less than 1), the gradients get smaller as they are passed back through each time step. Eventually, these gradients can become so small that they effectively "vanish," meaning they don't add much to the learning process. Conversely, if these values are large (greater than 1), the gradients can grow too large or explode, leading to wildly fluctuating network weights and unstable training [53]. Specialized structures like

Long Short-Term Memory (LSTM) units and Gated Recurrent Units (GRU) were developed to address the issue of vanishing gradients. RNNs using either LSTM or GRU have proven to be more effective than traditional RNNs for various tasks, making them a popular choice for models that need to recall information over long sequences [53, 54, 55].

4.3.3 Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs)

LSTMs are composed of three gates (input gate I_t, forget gate F_t, and output gate O_t) depicted in Figure 7. In contrast, GRUs are a simplification of the LSTM model using only two gates (the update gate z_t and the forget gate r_t) depicted in Figure 8 [53, 54, 55].

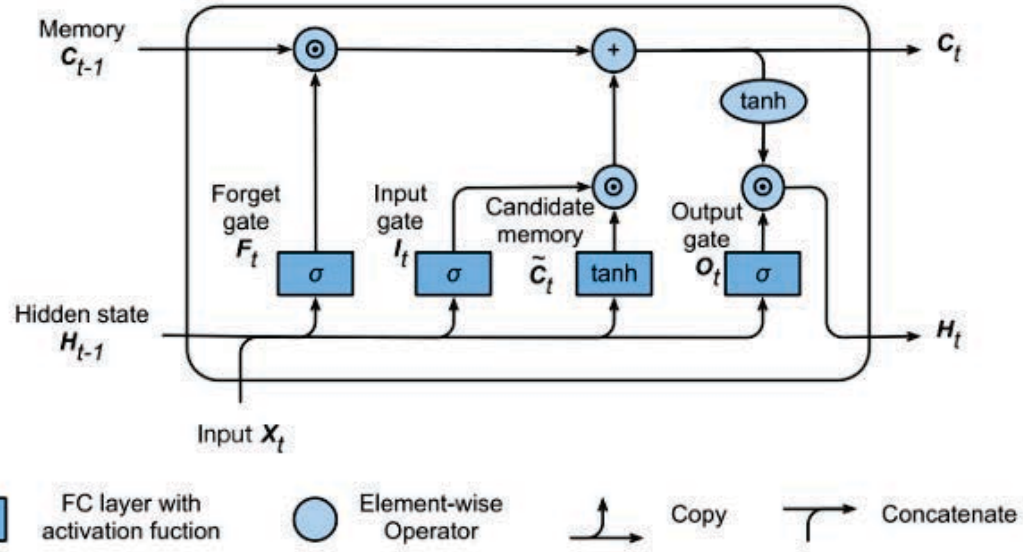

Figure 7: An LSTM cell processing data sequentially while maintaining its hidden state over time [53].

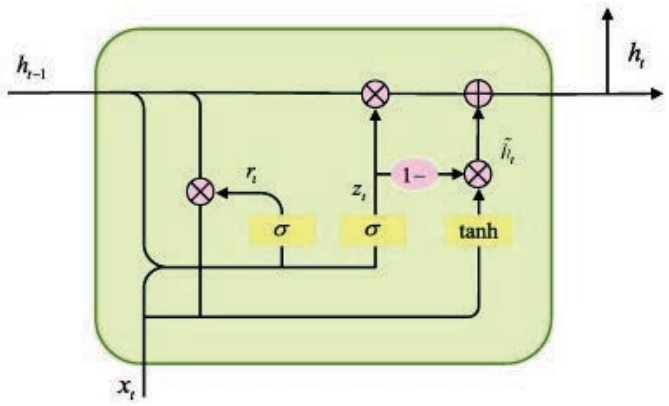

Figure 8: A GRU cell processing data sequentially while maintaining its hidden state over time [55].

Information in both models is kept in structures called gate cells, and the gates determine what is done with the information. I_t and z_t read inputs into cells, O_t reads the contents of cells, and F_t and r_t forget, or delete, the contents of cells. The gates essentially determine how much past information is kept, how much new information is added, and how much of the current state is considered in the output. The equations describing I_t, F_t, O_t, z_t, and r_t are shown in Equations 15 through 19, respectively [53, 55].

$$I_t = \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i) \text{ [53]} \tag{15}$$

$$O_t = \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o) \text{ [53]} \tag{16}$$

$$F_t = \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f) \text{ [53]} \tag{17}$$

$$z_t = \sigma(W x[x_t, h_{t-1}] + b_z) \text{ [55]} \tag{18}$$

$$r_t = \sigma(W r[x_t, h_{t-1}] + b_r) \text{ [55]} \tag{19}$$

RNNs using either units learn similarly to those without them. Firstly, a candidate memory cell (proposed update to the previous memory cell) is computed using tanh, representing new information the LSTM might add to its internal memory. Equation 20 depicts the candidate memory cell for LSTMs, while Equation 21 depicts the candidate memory cell for GRUs.

$$\tilde{C}_t = tanh(X_t W_x c + H_{t-1} W_{hc} + b_c) \text{ [53]} \tag{20}$$

$$\tilde{h}_t = tanh(W_h[x_i, r_i \odot h_{t-1}] + b_h) \text{ [55]} \tag{21}$$

The actual memory cell update is a combination of old memory content, C_{t-1} modulated by F_t (to determine which past features should be forgotten), and C_t is modulated by I_t (to determine the extent to with the new information should be incorporated). Equation 22 depicts the actual memory cell for LSTMs, while Equation 23 depicts the actual memory cell for GRU [53, 55].

$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t \text{ [53]} \tag{22}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \text{ [55]} \tag{23}$$

Finally, the hidden state of the LSTM for the current timestep is computed, as shown in Equations 22 (substituting C_t for h_t for GRU), by filtering the memory cell's content through O_t which modulates the C_t's influence on the hidden state at time t [53].

$$H_t = O_t \odot tanh(C_t) \text{ [53]} \tag{24}$$

Ultimately, each component and operation within an LSTM and GRU is designed to control the flow and influence of information throughout the sequence processing, making it robust

for tasks involving long or complex dependencies between events in the data.

4.4 Bayesian Neural Networks

Traditional neural networks, while powerful tools, often encounter hurdles like overfitting and overconfidence, particularly in the context of conventional deep learning methods. Enter BNNs, a promising solution to mitigate overfitting and overconfidence issues through training "uncertainty-aware" neural networks. In this section, we show how BNNs spearhead the paradigm shift towards uncertainty quantification and modeling, particularly crucial in academia, where accurately capturing uncertain parameters such as a learner's knowledge state and learning processes is critical [27].

BNNs are skilled at describing the uncertainty of the network because of their probabilistic treatment of neural network parameters. In doing this, BNNs can consider a range of possible models (weights, biases, or activations) and their associated probabilities, enabling uncertainty expressions in their predictions. The fusion of stochastic modeling and neural network architecture, as illustrated in Figure 9, is underpinned by Bayesian inference.

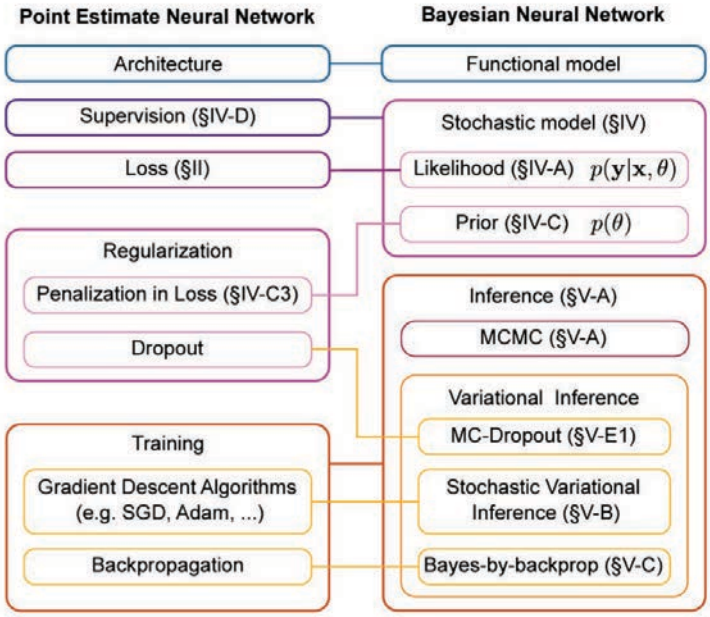

Figure 9: A schematic matching concept used in deep learning for point-estimate neural networks with their counterparts in a BNN [13].

Bayesian inference hinges upon the Bayes Theorem named after statistician and philosopher, Thomas Bayes. The theorem in Equation 25 below describes two ideas: 1) probability is a measure of "belief," and 2) prior beliefs influence posterior beliefs. Here, "belief" refers to the probability distribution that represents the network's current knowledge or uncertainty about the parameters (such as weights and biases).

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)} = \frac{P(D,H)}{\int_H P(D,H')dH'} \text{ [13].} \tag{25}$$

The parameters of the theorem are defined as follows:

- **Hypothesis Space, H:** A set encompassing all possible values that the model's parameters, such as weights and biases, can assume. It represents the neural network's beliefs or hypotheses about the observed data D.
- **Observed Data, D:** A set of inputs, x_i, and corresponding labels y $\{(x_1, y_1), ..., (x_n, y_n)\}$ used to update the network's beliefs about H.
- **Likelihood,** $P(D|H)$: A probability distribution quantifying the likelihood of observing D given a particular H. It encodes the network's aleatoric uncertainty or inherent noise in the data.
- **Prior,** $P(H)$: The probability of H occurring. It is determined before any data has been observed and thus serves as the initial belief about the hypothesis space.
- **Evidence,** $P(D)$: Also known as the normalization constant or marginal likelihood, $P(D)$ is the probability that the model will generate D regardless of H. It ensures that the posterior distribution, which reflects updated beliefs, integrates into one.
- **Posterior Probability,** $P(H|D)$: The posterior probability describes the probability that H (old belief) was true given D. It quantifies the updated beliefs about the hypothesis space based on new evidence, encoding the model's epistemic uncertainty.

4.4.1 Bayes Theorem Applied to Neural Networks

When applied to neural networks, H is often called $\theta = (w, b)$. We will be using θ moving forward. To build a BNN, the following elements are required: a stochastic model, a functional model, and a training dataset.

A. Functional Model: Choosing a functional model, or neural network structure (i.e., ANN, CNN, RNN, etc.), is the first step as it determines the physical construction of the network (i.e., how many layers, which parameters will be stochastic, etc.) and what the output of the network should be. The mathematical derivation of the functional model is

$$y = \Phi_0(x) + \epsilon \text{ [15]} \tag{26}$$

where y is the output the model intends to predict, $\Phi_0(x)$ is the function modeled by the neural network, and ϵ is the variable that denotes random noise. Since the activation function is only an approximation, ϵ captures the aleatoric uncertainty of the model [13, 15].

B. Stochastic Model: We then begin to establish the priors, $P(\theta)$, and $P(D|\theta)$, which provide our initial convictions on the model's parametrization and predictive accuracy. Before observing any data, our belief about θ is encapsulated in $P(\theta)$. Each unique set of values that θ can take represents a different instance of the hypothesis. For instance, if θ consists of weights w_1 and w_2, then $w_1 = 0.2$ and $w_2 = 0.4$ represents one instance while $w_1 = 0.1$ and $w_2 = 0.3$ represents another. This distribution reflects our initial confidence in the different parameter values based on previous data, expert knowledge, or any other information available before the current analysis. We calculate the $P(D|H)$ when observing the data. A higher likelihood under a particular set of parameters suggests that the network with those parameters is more consistent with the observed data [13, 15].

C. Training Data: The network needs a training set of input features, x, and target variables, y, pairs to feed into the network. With these elements, we can then build the Bayesian posterior, which reflects our updated understanding of the parameters after considering the evidence provided by the data. Bayes' theorem shows us how to calculate the posterior probability. However, the integral required to compute the evidence, $P(D)$,

$$P(\theta|D) = \int_{\theta} P(D, \theta')d\theta' \text{ [13]} \tag{27}$$

is often challenging and computationally expensive. BNNs can have thousands of weights and biases, resulting in a high-dimensional parameter space where each dimension corresponds to one of these variables. Markov Chain Monte Carlo (MCMC) and variational inference are two generally used alternatives for addressing these issues.

4.4.2 Markov Chain Monte Carlo (MCMC)

MCMC describes a family of methods that explore some state space, Ω, so that, over time, the samples collected eventually converge to the stationary distribution, π, of interest. For BNNs, the state space would be θ, where a state is some set of values for the weights and biases at that point in training, and the stationary distribution $P(\theta|D)$. Sampling occurs from some number, T, of steps, beginning at an arbitrary initial state in Ω. MCMC methods progress from the initial to the final state, π, through Markov chains, which are built on the principle that the probability of transitioning to a subsequent state depends exclusively on the current state, not on the sequence of transitions that preceded it. The final state is a finite collection of samples that were found to approximate the target distribution [14] best.

A Markov process is said to have a unique π if the following two conditions are true:

1. **A π exists for the state space in question:** A common way to ensure this is through the principle of detailed balance, which states for every pair of states, the probability of moving from state θ to state θ' is balanced by the probability of moving from state θ' to state θ Mathematically, this is expressed as

$$\pi(\theta)P(\theta'|\theta) = \pi(\theta')P(\theta|\theta') \text{ [14]} \tag{28}$$

   where P is a transition matrix whose entries represent the probability of transitioning from state θ to state θ' [14].

2. **The process is ergodic:** Ergodicity means that the Markov chain will eventually reach π from any initial state if given enough time. Once the system has reached π, mathematically expressed as $\pi(\theta)P = \pi(\theta)$, it will remain in π as the Markov chain evolves because, at this point, the chain is producing samples representative of the target distribution [14].

Of the many MCMCs, the Metropolis-Hastings algorithm, described by Algorithm 1 below, is notably the most applicable to BNNs [13].

Draw $\boldsymbol{\theta}_0 \sim$ *Initial probability distribution*;
while $n = 0$ **to** N **do**
  Draw $\boldsymbol{\theta}' \sim Q(\boldsymbol{\theta}'|\boldsymbol{\theta}_n)$;
  $p = \min\left(1, \frac{Q(\boldsymbol{\theta}'|\boldsymbol{\theta}_n)}{Q(\boldsymbol{\theta}_n|\boldsymbol{\theta}')} \frac{f(\boldsymbol{\theta}')}{f(\boldsymbol{\theta}_n)}\right)$;
  Draw $k \sim$ Bernoulli(p);
  if k **then**
    $\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}'$;
    $n = n + 1$;
  end if
end while

Algorithm 1: The steps for the Metropolis-Hasting algorithm [13].

The appeal of the Metropolis-Hastings algorithm lies in its ability to sample from a distribution without knowing the precise form of the target probability distribution $P(x)$. It operates effectively with a function $f(x)$ proportional to $P(x)$. The algorithm goes as follows:

1. **Initialization**
   (a) Select an initial state θ from the target distribution $P(\theta)$ to be sampled from
   (b) Select a proposal distribution $Q(\theta'|\theta_n)$ where θ' is the new suggested state and θ_n is the prior state. Q, often a Gaussian distribution centered at θ_n, is used to propose candidate states θ' for the Markov chain to transition to.
2. **Iteration:** Each step n involves the following
   (a) **Proposal:** A new candidate θ' is chosen from Q
   (b) **Acceptance Probability:** The acceptance probability, or ratio, p for transitioning to θ', is calculated
   (c) **Acceptance Decision:** A uniform random number $k \in [0, 1]$ is generated and checked against p.
   $$\theta_{n+1} = \begin{cases} \theta', & \text{if } k \leq \ p(\theta', \theta_n) \\ \theta_n, & \text{otherwise} \end{cases} \quad [13] \tag{29}$$
   Note that if θ' is more probable than θ_n, the algorithm always accepts the transition to θ'. This occurs when θ' exists in a region of $P(\theta)$ with higher density than θ_n, which results in the ratio $\frac{P(\theta')}{P(\theta)}$ being greater than 1. When $p = 1$ is checked against k the condition $k \leq p$ is consistently met.
   (d) **Convergence and Burn-in:** After enough iterations, the samples generated by the algorithm are deemed to approximate the target distribution $P(\theta)$ [13].

4.4.3 Variational Inference

Though MCMC algorithms are indispensable when addressing posterior distributions of BNNs, they are limited in their ability to scale. When dealing with MCMC methods in

general, there is a burn-in time, which describes the initial phase of sampling where the chain requires time to attain a steady state that properly represents the target distribution. Though samples from this phase are often discarded, it can be challenging to determine the appropriate length of the burn-in period, and generating unused computations is computationally expensive. Additionally, storing all of the sustained samples of high dimensional parameter space can be very resource-intensive [13, 14].

This is where variational inference comes into play. Instead of sampling from the posterior, variational methods approximate the posterior using a parametrized distribution $q_\phi(H)$, called the variational distribution or approximate posterior. During training, the parameters ϕ, analogous to θ, undergo optimization to refine $q_\phi(H)$, aiming for maximal alignment with the exact posterior distribution $P(H|D)$. Maximizing the closeness between the approximate and true posteriors if often done through minimizing Kullback-Liebler (KL) divergence. KL-divergence measures the dissimilarity between two distributions and is calculated using Equation 30.

$$D_{KL}(Q\|P) = \int_H q_\phi(H') \log\left(\frac{q_\phi(H')}{P(H'|D}\right) dH' \text{ [13]} \tag{30}$$

Minimizing the KL-divergence, however, is difficult because it still requires that we compute the posterior. Thus, variational inference often employs an alternative approach that focuses on maximizing the Evidence Lower Bound (ELBO) derived from a rearrangement of terms in the KL divergence equation, leading to the following formula:

$$\int_H q_\phi(H') \log\left(\frac{q_\phi(H')}{P(H'|D}\right) dH' = \log(P(D)) - D_{KL}(q_\phi\|P) \text{ [13].} \tag{31}$$

In this equation, $\log(P(D))$ is constant with respect to ϕ as it only depends on the prior. Therefore, the optimization process is the adjustment of ϕ to increase the ELBO, leveraging the fact that doing so indirectly minimizes the KL divergence between $q_\phi(H)$ and the true posterior. To do this, Stochastic Variational Inference (SVI) is a type of variational inference that utilizes stochastic gradients to incrementally update ϕ in a direction that increases the ELBO. Due to this incremental update process, SVI can process and handle data that cannot be stored in memory entirely, making it scalable and computationally efficient [13].

4.4.4 Challenges and Limitations

Despite their advantages, Bayesian Networks face significant challenges that hinder their widespread application. Overparameterization complicates their structure and undermines computational efficiency, often obscuring authentic causal relationships and impeding the effective utilization of insights. Additionally, BNNs struggle with unsupervised algorithms and managing sparse or noisy data, common in real-world settings. To address these issues, advanced methodologies are needed to handle incomplete datasets and adapt to non-standard models, though this increases complexity, consuming more time and resources. A specific challenge arises with autocorrelation within the MCMC methods. Since each sample in the chain depends on its predecessor, samples can become highly similar or autocorrelated, leading to a biased representation of the probability distribution, which requires subsampling techniques to obtain a more accurate and representative set of samples for inference.

However, subsampling itself introduces the need for careful consideration regarding the selection of samples and the potential for information loss, further adding to the complexity of employing Bayesian Networks effectively [15].

In spite of these hurdles, Bayesian networks are a robust framework for probabilistic learning amidst uncertainty. Enhancing their handling of sparse data, reducing overparameterization, and personalizing learning models are crucial for improving their efficacy. Leveraging the probabilistic nature of Bayesian inference, BNNs use both MCMC and SVI for efficient training, maintaining scalability, and meeting diverse learning needs. This integration of stochastic modeling and neural networks enables the creation of adaptive, personalized learning environments, potentially transforming education by making learning more tailored, effective, and inclusive.

V. APPLICATIONS OF AI IN ADAPTIVE LEARNING SYSTEMS

This section will delve into various implementations of AI, focusing on systems like Assessment and LEarning in Knowledge Spaces (ALEKS) and Bayesian Intelligent Tutoring System for Computer Programming (BITS), among others. Exploring these tools will not only highlight their technical intricacies but also discuss the practical benefits and challenges posed by the integration of AI into adaptive learning environments.

5.1 ALEKS

Before the incorporation of AI, the ALEKS learner model only leveraged Knowledge Space Theory (KST) to provide a tailored educational experience for students from grades 3-12 and in higher education [59]. This cognitive framework allows ALEKS to construct a detailed map of a student's comprehension and mastery over discrete concepts, defined as 'items.' In practice, KST enables systems like ALEKS to identify which 'items' students know and which they are ready to learn based on their current knowledge state or set of items they have mastered. For example, in the context of learning Python, one part of the knowledge structure could involve mastering basic Python syntax. Within this knowledge state, specific items might include tasks such as writing and executing a Python script requiring the student to know how to use the "print()" function and understand how to enclose strings [60]. The ALEKS system, backed by billions of past student interactions with the software, continually updates these states through regular Progress Knowledge Checks [28, 61]. During these checks, ALEKS categorizes each assessment item into three categories based on the student's responses: in-state, items that the student is likely to know based on their answers; out-of-state, items that the student is likely not to know; and uncertain, items ALEKS was unable to classify as known or unknown [56, 57, 58]. Tools like the ALEKS Pie Report, as shown in Figure 10, work to visualize the results of these checks, showing tangible progress and helping time the introduction of further practice or new questions, ensuring that learning interventions are both timely and contextually relevant [28].

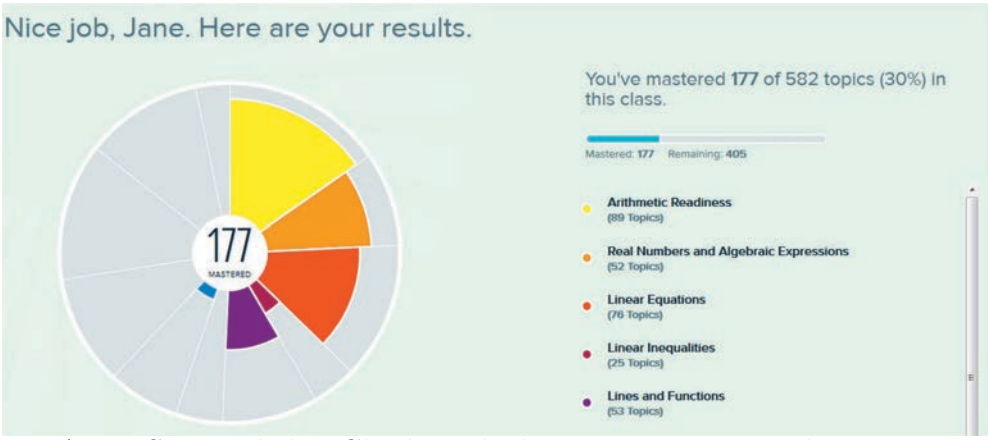

Figure 10: ALEKS Knowledge Checks, which contain no more than 30 questions, help determine which subjects a learner has mastered and identify areas needing further work. The results from these Knowledge Checks are represented in the ALEKS Pie Chart within, showing the learner both their mastered topics and those requiring more practice [28, 61].

However, while these methods helped ALEKS be effective at "know[ing] what each student is ready to learn," feedback revealed that students were experiencing "assessment fatigue" due to frequent progress and placement assessments [56, 59]. They expressed a preference for shorter testing, which would afford them more time to learn new material [59]. Researchers at ALEKS then decided to figure out a way to shorten the tests. In their 2019 study, Matayoshi et al. [2019a] analyzed over 3.3 million questions from progress tests across 10 different math and chemistry courses, identifying key areas for improvement in ALEKS's learning retention model. They found that uncertain items were more frequently retained and that there was a negative correlation between the length of the tests and the retention of learned items, as illustrated in Figures 11 and 12 respectively. In response, ALEKS worked to improve their KST-powered assessment algorithm by incorporating the classification capabilities of RNN models. RNNs are well-suited for modeling time-dependent data, as they can retain past inputs in their hidden layers, enabling them to make informed predictions based on observed patterns within a sequence. The incorporation of RNNs into their system enhanced ALEKS' capacity to both predict which items were more likely to be retained and stop the test once a classification had been reached, reducing the need for frequent testing [56].

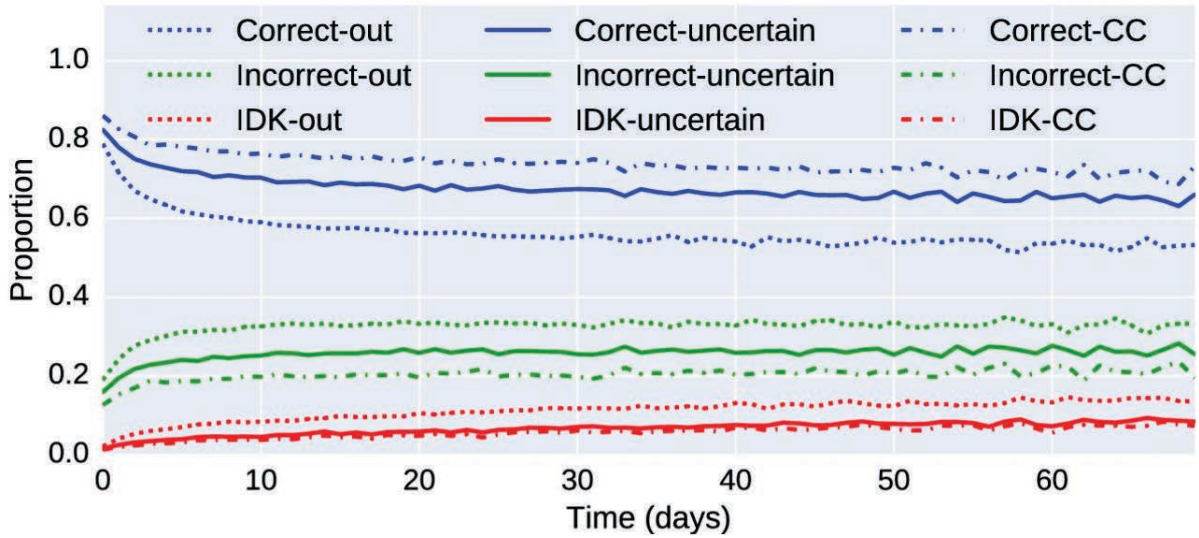

Figure 11: This graph displays how items classified during initial ALEKS assessments as "out-of-state," "uncertain," or "CC" (learning sequence where students successfully answered the first two questions of an uncertain item correctly) are retained by students over time. The three colored lines represent different response types: blue for correct, green for incorrect, and red for "I don't know" responses [56].

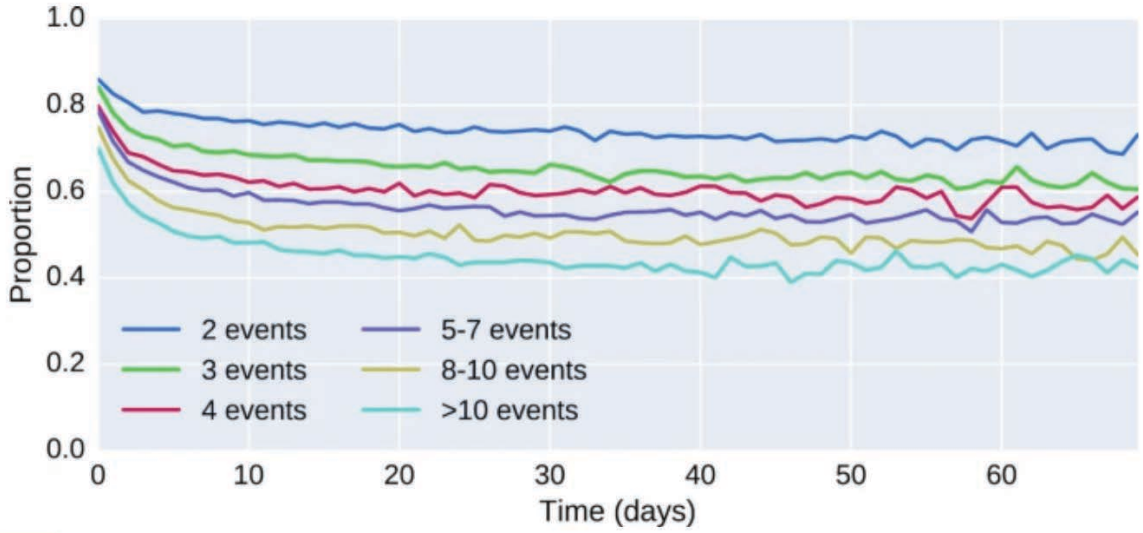

Figure 12: The proportion of correct test responses decreases as the number of events, or questions, in a learning sequence increases indicating that longer learning sequences, initially thought to provide more practice, do not necessarily lead to better retention of the material [56].

Matayoshi et al. [2019a] developed a neural network classifier model to predict whether a student would correctly answer questions about recently learned items. The model incorporated several key features: a categorical variable representing ALEKS courses, which had ten distinct values corresponding to different classes; a categorical variable accounting for a diverse range of 2,190 distinct items that students might encounter; a continuous variable ranging from 0 to 1 representing the initial score of their Initial Knowledge Check; a discrete variable spanning from 0 to 399 measuring the time in days since each item was first learned; and lastly, a sequence of categorical variables, each with three values representing a correct answer, an incorrect answer, or the action of reading an explanation. The learning sequences, inherently sequential as they track actions over time, were then processed by the hidden layers of the RNN, for which two types of hidden processing units were evaluated: GRUs and LSTMs. To optimize the neural network and prevent overfitting, the researchers used the following techniques: batch normalization by adjusting and scaling activations to stabilize the network, early stopping to prevent overfitting, and randomly dropping processing units (and their connections) during training also to prevent overfitting. The researchers determined that the most effective configuration of their RNN model consisted of four layers of LSTM units. The output from these LSTM layers was then concatenated with the other features to form the input for a Multi-Layer Perceptron structured with an initial hidden layer containing 800 units, followed by two hidden layers, each with 400 units with each unit using a ReLU as the activation function, which contributed to the model performing better than the forgetting curve. Overall, as illustrated in Figure 13, the neural network was more precise, meaning the ratio of items correctly identified as retained to all items predicted as retained was higher, and capable of recalling or classifying more items in one of the three sequence categories. Since the neural network model combined sequential data from students' learning activities with other static (non-sequential) variables to enhance predictions, it pro-

vided more insight into which variables significantly affect retention beyond what traditional one-dimensional forgetting curve models could. With this information, ALEKS can better classify the uncertain items thereby minimizing time spent on recently learned material [56].

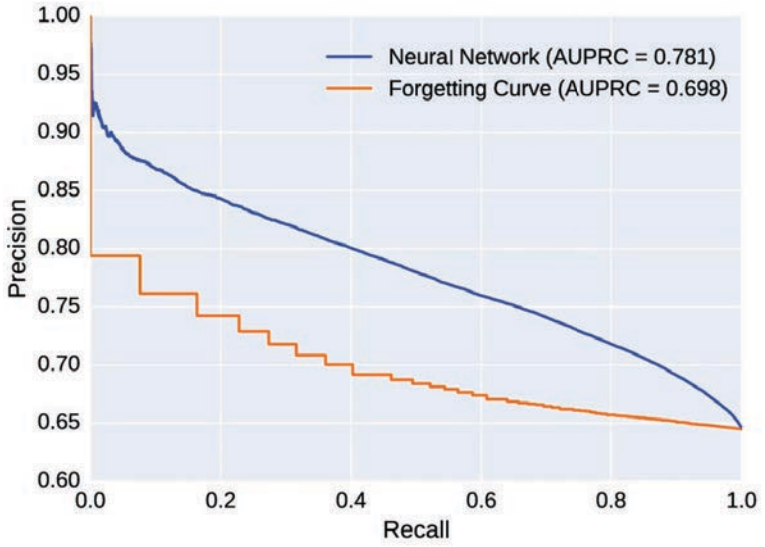

Figure 13: A precision-recall curve demonstrating the predictive accuracy of the neural networks. As recall increases, precision tends to decrease, indicating that while the model identifies a larger number of items as retained, it does so with decreasing accuracy.

Another method they discovered for streamlining assessments was an intelligent stopping algorithm. Matayoshi et al. experimented with this approach using the ALEKS Placement, Preparation, and Learning (ALEKS PPL) assessment, which is administered to students in post-secondary mathematics courses to ascertain their placement in one of six math courses. The ALEKS PPL assessment categorizes each of the 314 items it tests for as either in-state, out-of-state, or uncertain [57]. The original stopping algorithm, prior to the additions of AI, terminated when either ALEKS classified all items or the preset limit of 29 questions had been answered [29]. Seeing as course placements did not require students to answer all questions, Matayoshi et al. [2019b] proposed the current stopping algorithm described in Algorithm 2 and an updated one in 2021 described in Algorithm 3 for the RNN models [57, 58].

Inputs:
α, stopping threshold probability
$P(k \mid \mathbf{x}_n)$, predicted probability of class k, $k = 1, \ldots, 6$, after question n
$K_n = \arg\max_{k=1,\ldots,6} P(k \mid \mathbf{x}_n)$; i.e., the most likely class after question n
C_n, the current recommended course placement after question n

Iterations:
for $n = 10$ to 29 **do**
  Compute K_n and C_n using information from questions 1 to n
  if n == 29 or $\left(P(K_n \mid \mathbf{x}_n) > \alpha$ and $K_n == C_n\right)$ **then**
    Stop the assessment
  end if
end for

Output:
C_n, the (predicted) course placement recommendation

Algorithm 2: The 2019 stopping algorithm [57].

Inputs:
α, stopping threshold probability
$\mathbf{x}_n$, the input features of the classification model after question n
$P(k \mid \mathbf{x}_n)$, predicted probability of class k, $k = 1, \ldots, 6$, after question n
$K_n = \arg\max_{k=1,\ldots,6} P(k \mid \mathbf{x}_n)$; i.e., the most likely class after question n
C_{29}, the recommended course placement after question 29 (based on computing the student's percentage score and applying the cut scores in Table 1)

Iterations:
for $n = 5$ to 29 **do**
  if $n == 29$ **then**
    Return C_{29}
  else if $P(K_n \mid \mathbf{x}_n) > \alpha$ **then**
    Stop the assessment and return K_n
  end if
end for

Output:
The predicted course placement recommendation

Algorithm 3: The 2021 stopping algorithm [58].

Both algorithms employ a vector, x_n, consisting of 942 independent variables, which represent all possible combinations of three item states across 314 assessment items. x_n classifies each item's state after the nth question. The variable k indicates one of the six potential course placements, and $P(k|x_n)$ is the probability of the student being placed in course k after answering question n. Subsequently, both algorithms compute the classifier's predicted class label, K_n, alongside the recommended course placement based on the student's current percentage score, C_n. The assessment process halts when the confidence K_n surpasses a preset threshold, or the 29th question is reached. The methodology for deriving course placement from these variables has evolved. Unlike the earlier algorithm by Matayoshi et al. [2019b], the 2021 stopping algorithm no longer required $C_n = K_n$. Initially, this alignment aimed to boost performance by integrating predictions from both sources, but it sometimes

led to unnecessarily prolonged assessments. Moreover, the 2021 version initiates after five questions, a reduction from the previous 10-question minimum. This change addresses instances where prolonged assessments were unnecessary, as the classifier frequently provided high probability estimates well before the 10th question [57, 58].

The 2021 stopping algorithm significantly enhanced the efficiency and accuracy of the course placement assessments, as demonstrated by several key metrics. On the test set of 45,470 students, the GRU model, leveraging this algorithm, showed marked improvements over the logistic regression model used for baseline comparison. Specifically, when completing all 29 questions, students took an average of 93.6 minutes, with a typical (median) time of 82.5 minutes. However, after applying a stopping algorithm, the researchers saw the average test-taking time drop to 70.1 minutes and the typical time fall to just under an hour at 59.7 minutes. Furthermore, for the 3,483 students who took over three hours to complete the assessment, their average duration was reduced dramatically from 221.1 minutes to 149.8 minutes, as shown by the stopped assessments distribution's left skew in Figure 14. Consequently, the number of students taking more than three hours to complete the assessment dropped to 1,299, representing a reduction by a factor of approximately 2.7. These improvements underscore both the algorithms and the artificial intelligence behind its role in optimizing the assessment process, making it a valuable tool for educational institutions aiming to streamline course placements while ensuring precise student evaluations [58].

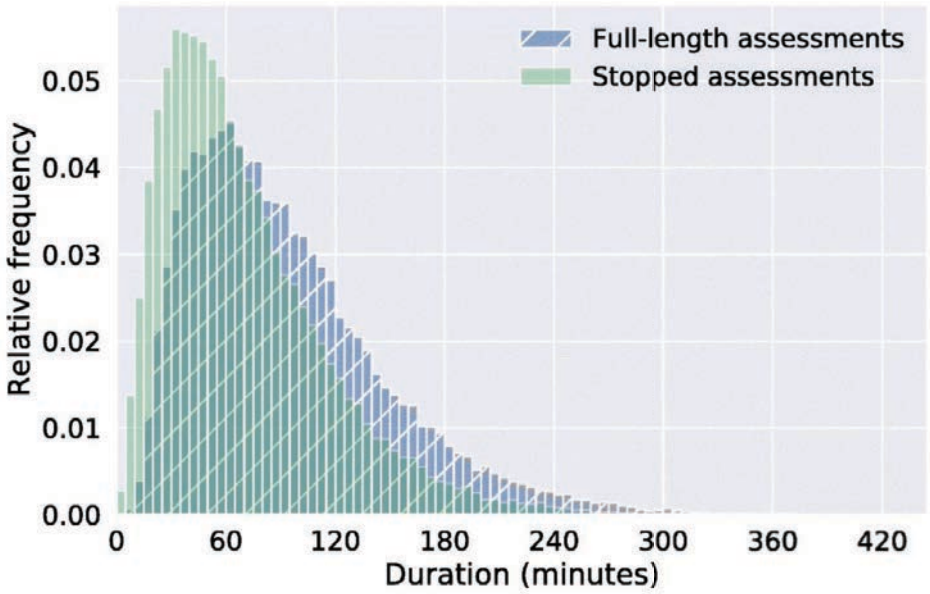

Figure 14: A histogram depicting the distribution of durations for assessments employing the 2021 stopping algorithm utilizing the combined features GRU model with a threshold of $\alpha = 0.99$, and for full-length assessments encompassing all 29 questions [58].

5.2 BITS

BITS is an Intelligent Tutoring System (ITS) designed to tutor students in introductory $C++$ [62]. ITS are advanced computer programs designed to simulate personalized teaching by adapting to the individual learner's needs. They generally consist of four interconnected

modules simulating the effectiveness of one-on-one tutoring at scale: Expert module, User Interface module, Student module, and Tutor module [32]. Each module is captured in Figure 15, and they work together to accomplish BITS' main goal: help students navigate online learning environments through appropriate recommendations of learning goals and the generation of learning sequences to achieve them [62].

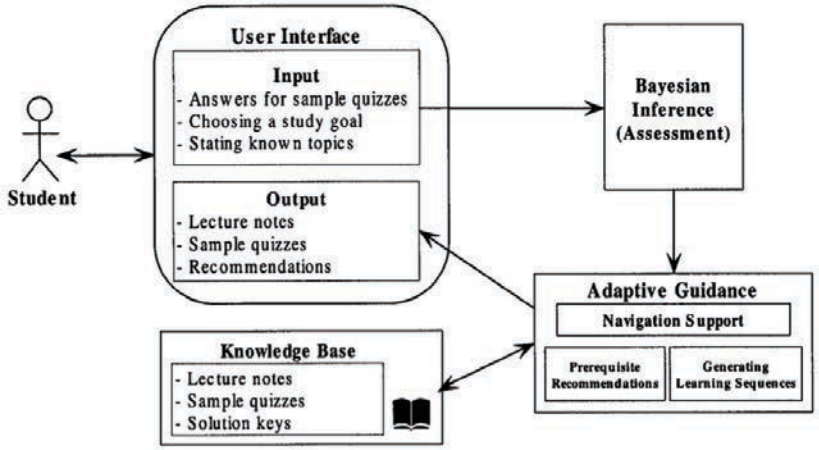

Figure 15: The architecture of BITS.

The Expert module, known as the Knowledge Base in BITS, serves as the core repository of what the system knows about the subjects it teaches. Students interact with the Knowledge module through BITS' User Interface module, depicted in Figure 16 below.

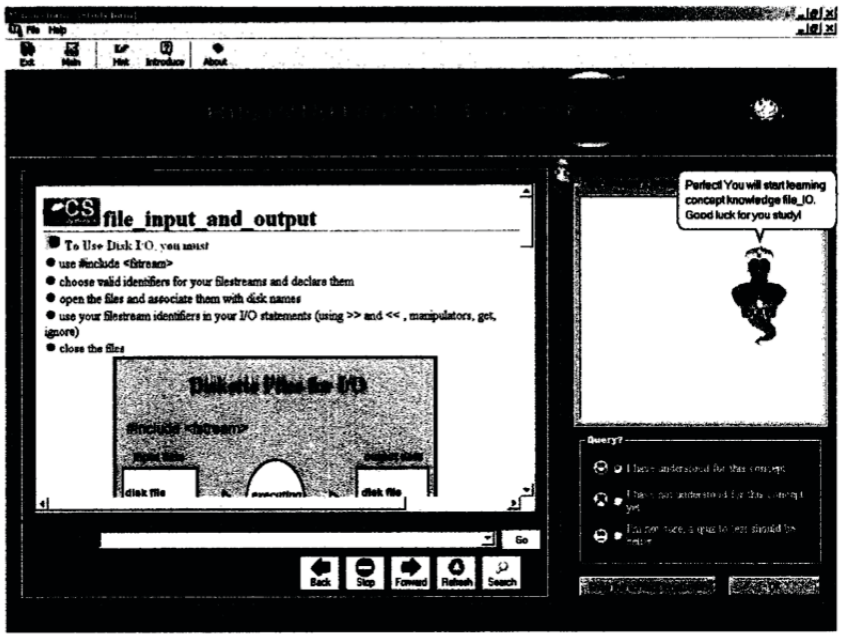

Figure 16: The UI of BITS presenting lecture notes on the File I/O concept while also prompting the user to indicate their understanding of this concept (note "original in color") [62].

BITS' front-end design displays the problem with an accompanying figure, the question to

solve, and a place to answer (either through selection or filling in the blanks). Then the learner model, or Student module, is developed from the user's interaction with the UI. Students have the option to follow the prerequisite path recommended by BITS or select a path from the generated learning sequences. This latter option enables students to tailor their learning experience by choosing topics of interest carefully curated into a sequence accommodating their learning style. The most important module backing BITS' ability to help students navigate online educational resources is the Tutor module, where the human-like tutor provides feedback to the learner and monitors how it impacts their course of study. BITS formulates its Student and Tutor modules using BNNs [62].

5.2.1 Implementation of Bayesian Neural Networks in BITS

Bayesian Networks are often utilized in adaptive tutoring systems for their ability to model the uncertainty in learners' knowledge states and dynamically adapt instructional strategies based on probabilistic inference. Furthermore, a Bayesian Network can model prerequisites as it is a DAG, as shown in Figure 17, which depicts the BNN for the For Loop construct in $C++$ [30, 62].

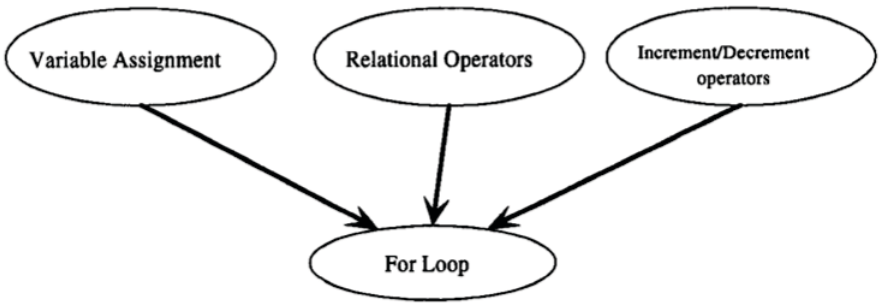

Figure 17: Model of the prerequisites to understand the For Loops [62].

Each node in the network is associated with conditional probability distributions (CPDs) that describe the likelihood of a student mastering a concept given their mastery of prerequisite concepts. As students interact with the system, BITS updates its beliefs about their knowledge state, approximated by calculating the following posterior distribution [62].

$$p(a_i = known | P_i = known) = \frac{p(a_i = known, P_i = known)}{p(P_i = known)} \text{ [62]} \tag{32}$$

, where a_i is the system's initial hypothesis, θ, about their knowledge of concept a_i, and P_i is the system's already observed data, D, of the student's mastery of the concept's prerequisites. Using this, BITS can calculate all of the CPDs for a given concept, such as for loops, shown in Figure 18, to determine what the student knows and doesn't know [13, 62].

Variable Assignment	Relational Operators	Incre/Decrement Operators	For Loop	p(F|V,R,I)
known	*known*	*known*	*known*	0.75
known	*known*	*known*	*not known*	0.25
known	*known*	*not known*	*known*	0.39
known	*known*	*not known*	*not known*	0.61
known	*not known*	*known*	*known*	0.50
known	*not known*	*known*	*not known*	0.50
known	*not known*	*not known*	*known*	0.22
known	*not known*	*not known*	*not known*	0.78
not known	*known*	*known*	*known*	0.50
not known	*known*	*known*	*not known*	0.50
not known	*known*	*not known*	*known*	0.29
not known	*known*	*not known*	*not known*	0.71
not known	*not known*	*known*	*known*	0.40
not known	*not known*	*known*	*not known*	0.60
not known	*not known*	*not known*	*known*	0.15
not known	*not known*	*not known*	*not known*	0.85

Figure 18: The CPD associated with all possible known, unknown combinations of the for loop's prerequisites [62].

These probabilities help determine the Student module consisting of the most likely current state of the student's understanding. If $p(a_i = known|P_i = known) \geq 0.70$ (an arbitrarily chosen threshold), the a_i is considered *already known*. If $p(a_i = known|P_i = known) < 0.70$ and all P_i are known, the a_i is considered *ready to learn*. Lastly, if at least one concept in P_i is unknown, a_i is considered *not ready to learn known*. Based on this, the BITS' Tutor, or Adaptive Guidance, module decides the next educational steps by refining its prerequisite recommendations and learning sequence generation. Students can study the next best topic BITS recommends or select a learning sequence path. This latter option enables students to tailor their learning experience by choosing topics of interest that are thoughtfully curated into sequences that accommodate their learning style [62].

In essence, the BNN within BITS supports handling uncertainty and variability in student learning paths and enhances the system's ability to provide personalized education efficiently. This intelligent system significantly improves the adeptness of web-based tutoring environments.

5.3 Other Applications

Several other adaptive tools stand out for their innovative use of artificial intelligence to enhance learning experiences. These include systems like ChatGPT, OATutor, Jill Watson, and various virtual assistants, leveraging different aspects of AI technology, including natural

language processing (NLP), data mining, and predictive analytics.

5.3.1 ChatGPT

ChatGPT, developed by OpenAI, is built on the Generative Pre-trained Transformer (GPT) architecture, a neural network designed for natural language processing (NLP) tasks. It enhances user interaction by interpreting and generating human-like text, enabling dynamic, context-aware conversations. This AI tool is handy in educational settings, where it provides real-time explanations, tutoring, and personalized feedback, making learning more accessible and tailored to individual needs. Through its ability to respond to diverse educational scenarios, ChatGPT supports various functions such as automated grading and adaptive curriculum development, enhancing engagement and academic outcomes [66].

5.3.2 OATutor

OATutor, developed at UC Berkeley, is an open-source ITS that supports the learning sciences research community. It provides students and educators with one-on-one tutoring, primarily in mathematics, that understands and responds to their learning needs in real-time. It does this using Bayesian Knowledge Tracing (BKT), a widely implemented model that operates on the principles of Bayesian inference [33]. Recently, generative AI tools such as ChatGPT have been added to OATutor's functionality to quicken the time it takes to generate hints for learners and enhance the hints provided. Additionally, the open-source nature of generative AI was particularly attractive to OATutor. By integrating ChatGPT into their platform, instructors could have greater control over the content delivered to students. Their familiarity with their students would allow them to tailor inputs into the system, ensuring that the content is more aligned and relevant to their specific student populations [31, 32, 60, 67].

5.3.3 Jill Watson

Jill Watson is an AI-powered teaching assistant created by Professor Goel at the Georgia Institute of Technology. Like ChatGPT, Jill employs advanced NLP capabilities to answer students' questions based on the course syllabus and other educational materials, effectively reducing teachers' time on administrative tasks. By handling routine queries, Jill Watson enables educators to focus more on personalized teaching and less on repetitive duties. Since its debut in 2016, it has been fulfilling its function to the fullest, having been deployed in approximately 17 different classes, ranging from undergraduate to graduate levels, online and in-person [68, 69].

In the end, AI in education is not just about automating tasks but about enhancing and personalizing the learning experience to meet the unique needs of every student. Integrating AI-driven tools like ChatGPT, OATutor, and Jill Watson into educational environments represents a significant advancement in adaptive learning. However, it is crucial to approach the deployment of these AI technologies with caution, which the following section discusses in further detail.

VI. ETHICAL FRONTIERS IN AI-DRIVEN ADAPTIVE LEARNING

AI-driven systems can now analyze vast amounts of data in real time, tailoring educational pathways to suit individual learners' strengths, weaknesses, and preferences. However, this technological leap has sparked significant ethical concerns within academic circles. Issues such as algorithmic bias, data privacy, and an over-reliance on AI tools have initiated crucial conversations regarding the future of educational practices and policies. This section will delve into the ethical implications of AI in adaptive learning, exploring how these technologies can both enhance and challenge traditional educational frameworks. By balancing the transformative potential of AI with moral considerations, educators aim to ensure that technology serves learners' best interests while upholding principles of equity, privacy, and pedagogical integrity. As AI continues to revolutionize adaptive learning systems, providing personalized educational experiences and insights into student performance, it is imperative to address potential ethical challenges. Ultimately, the focus lies not on whether AI should be utilized in classrooms but rather on how to establish structures that prevent its misuse.

6.1 Bias

In AI-powered adaptive learning, the issue of bias is particularly critical because it can influence both the design and the outcome of learning algorithms. Harini Suresh and John Guttag of MIT identified six distinct categories of bias in machine learning, each stemming from various sources. Historical Bias results from past inequalities or societal prejudices being considered in the training data, perpetuating biased outcomes in the algorithm's predictions. This bias is notably prevalent in education, particularly in Educational Data Mining (EDM) [39]. In their study, Paquette et al. [2020] found that nearly half of the papers analyzing demographic data used such attributes as predictive features in models without including them in testing or validation [40]. Measurement Bias occurs when the data collection process systematically favors certain ideologies, structures, or groups and excludes others. Aggregation Bias arises when data from different sources or contexts are combined in a way that ignores variations among individual or group needs. Representation Bias occurs when the features used to represent data fail to capture the full diversity of the population [39]. For instance, Anderson et al. [2019] studied the fairness of machine learning models predicting student graduation rates. They noted that the small sample size of only 44 indigenous learners limited meaningful predictions for dropout rates among this demographic [40, 41]. Evaluation Bias is present when the metrics used to evaluate performance do not accurately represent the user groups upon which the algorithm operates. Finally, Deployment Bias refers to discrepancies that arise when an algorithm is implemented in a different setting than the one it was originally designed for. An example of this would be deploying a student engagement monitoring tool originally meant to help educators identify struggling students and provide targeted support, but school administrators now use it to evaluate teacher effectiveness based solely on student engagement metrics. Without addressing these biases, new implementations of educational practices and policies risk being based on data that misrepresents reality and leads to educational interventions that are not only ineffective but also unjust [37, 39].

6.1.1 Addressing Bias

Mitigating bias in machine learning is complex but not impossible. Implementing a combination of diverse data sets, transparent methodologies, and regular ethical audits can ensure that AI-driven tools function equitably across all student demographics. This begins with gathering data from a broad spectrum of sources, encompassing various educational institutions, geographic regions, and socioeconomic backgrounds, to ensure a rich diversity that reflects the vast array of student experiences. The researchers and engineers behind these technologies should collaborate with educators, sociologists, and cultural experts to ensure the data's relevance and inclusivity, providing essential context that prevents skewed perspectives. Furthermore, an ongoing data evaluation is necessary to maintain representation and address emerging societal changes or educational shifts [43, 48]. This may require targeted efforts to collect data or forming partnerships with organizations that advocate for these groups (such as Diversity.AI, an organization leading research in combating discrimination by humans and AI and providing open access datasets of representative data for developers), thereby enriching the data pool with a wider range of perspectives [49]; incorporating feedback loops allows for the continuous refinement of data collection and AI training processes, as feedback from users and stakeholders can highlight unforeseen biases and areas for improvement; and lastly, training for faculty on when to trust or question the outputs given by the adaptive learning systems. This should not be a one-time task but a continuous process that ensures the AI's outputs are consistently fair and unbiased. By adhering to these practices, AI in education can be harnessed as a powerful tool for personalized learning without compromising fairness or ethical standards [34, 43, 48].

6.2 Data Privacy and Security

Data privacy in the academic sector is becoming a critical issue, mainly due to the various problems of disregarding the fair information practice principles: notice, consent, choice, and transparency. First and foremost, efficient adaptive learning systems require the collection and analysis of vast amounts of personal data (i.e., age, gender, school location, learning style, emotional state, past test scores, etc.) from learners, which raises concerns about how this data is stored, who has access to it, and how it is used beyond the educational context. A notable example illustrating these concerns is the case of InBloom. InBloom, originally a data aggregator initiative, encountered substantial resistance and ultimately shut down due to widespread privacy concerns, lack of parental consent, and issues surrounding access to sensitive student information. It collected data across 400 optional fields, which included sensitive details such as disability status and social security numbers—details that were often collected without full user awareness. The controversy, highlighted by a lawsuit from concerned parents and subsequent legislative actions, led to the withdrawal of several states and the shutdown of InBloom in 2014 [42]. Secondly, maintaining confidentiality poses significant challenges with the enhancement of tools for aggregating large datasets. Although personally identifiable information may be removed from these datasets, the risk of re-identification persists. Furthermore, the increased tracking of past and present actions to predict future behaviors exposes new realms of personal data, all in the pursuit of enhanced understanding and productivity. The wrongful disclosure of student information can undermine the beneficial impacts of adaptive learning platforms on social and academic development. It

deprives the users of these educational technologies of control over their learning path because characteristics about themselves, which they did not consent to be included in their student profiles, are utilized to generate probabilistic predictions about the opportunities and resources they can or cannot access. Lastly, these concerns are compounded by artificial systems and algorithms that are often opaque, creating a "black-box" scenario where data processing and decision-making mechanisms are hidden from view. As seen with InBloom, the users and their parents were unaware of the extensive tracking being done. This lack of transparency inhibits the ability of stakeholders—students, parents, and educational authorities—to audit and understand how personalized learning decisions are being made. Such conditions prevent stakeholders from holding institutions accountable and hinder efforts to build trust and acceptance of digital assessment tools. As algorithms play a more significant role in shaping educational outcomes, the call for transparency, accountability, and ethical considerations in their deployment becomes more urgent [38, 42].

6.2.1 Addressing Data Privacy and Security

Data protection in educational technology requires a multifaceted approach that includes technical, organizational, and legal defense. Each defense operates on two fronts: protecting the user and securing the technology. Technology safeguards must combine transparency and privacy-preserving algorithms so that users are aware of and can consent to how and why their data is being collected and that developers are held accountable for how they extract, process, store, and distribute sensitive data [64]. The privacy-preserving techniques often used are *k-anonymity*, which ensures that each record in a dataset is indistinguishable from at least $k-1$, *L-diversity*, which addresses k-anonymity's limitations by ensuring that each "equivalence class" (a group of records indistinguishable from each other) contains at least l distinct values for sensitive attributes, and *T-closeness*, which addresses L-diversity's shortcomings by ensuring that the distribution of a sensitive attribute in any group of records closely resembles the distribution of that attribute in the entire dataset, reducing the likelihood that these attributes can be guessed or deanonymized [47, 63, 64].

Organizations must implement strict access controls to student data and establish robust auditing procedures to evaluate the impact of educational technologies, ensuring that risks are minimized. They should regularly update these protocols and any associated technical defense systems to adapt to new security challenges and maintain compliance with evolving privacy regulations. Additionally, regular training sessions for educators and updates on data protection best practices and relevant legal changes are essential for those utilizing educational technologies [47].

Finally, on the legal side, a thorough understanding of relevant laws such as the Family Educational Rights and Privacy Act (FERPA) and the Children's Online Privacy Protection Act (COPPA) is essential. FERPA is a federal law in the United States that restricts disclosing personally identifiable information from student records without explicit consent from the parent or eligible student. At the same time, COPPA is another U.S. federal law requiring websites and online services to obtain parental consent before collecting, using, or disclosing personal information from children under 13. Outlining these (and similar) rights, most notably in terms of data use, needs to be precise. Participating in regular training

sessions and updates on the best practices in data protection and any legal changes that might affect their use of educational technologies needs to occur. By collaboratively taking these comprehensive measures, developers and educators can ensure a secure educational environment [42, 64].

6.3 Overdependence on Technology

Concerns regarding the overdependence on AI in academic spaces are twofold, affecting educators and students. From an educator's standpoint, overreliance on AI can lead to compliance challenges, especially when adhering to educational standards and regulations. The outputs of AI systems can significantly shape how insights from AI systems are presented and interpreted. On the one hand, some adaptive learning systems may offer advanced learner modeling, yet they carry inherited biases, potentially leading to unjust treatment or outcomes for specific groups of students. On the other hand, other tools may generate less informative observations [45, 46]. A case in point involves BlackBoard Learn, a predictive analytics tool used in an organic chemistry class at California State University, which estimated that 80 percent of the students were likely not to complete the semester. While statistically significant, this information was not news to Professor Fernandes, who was left without actionable guidance on what steps to take next [35, 44]. In both situations, the AI behind the learning system cannot grasp the full context of a student's situation, such as personal challenges or cultural backgrounds. Relying heavily on AI for profiling and decision-making can diminish the role of educators in understanding and supporting students. This might lead to a one-size-fits-all approach, where subtle nuances in individual student needs and contexts are overlooked, contradicting the intended purpose of these systems. Educators need to be aware of the benefits *and* the limitations to prevent complacency with artificial suggestions that do not align with their professional judgments [45, 46].

From a student standpoint, one primary concern revolves around reduced learner agency, which describes the capacity for individuals to take ownership of their learning journey. With AI assuming a predominant role in decision-making processes, educators worry that avenues for creativity, scientific discovery, and reflection may be quashed. Sophisticated technologies such as natural language processing especially call into question the authenticity and originality of student work, particularly in tasks involving writing assignments and scientific research. For instance, if students use AI to generate homework answers, opportunities to set goals and reflect on their learning progress are missed. It also becomes challenging for educators to assess their students' analytical and critical thinking abilities. Moreover, the use of AI in educational settings prompts discussions about the role of human instructors in guiding students' interactions with these technologies. Furthermore, while AI can provide valuable support in personalized learning experiences and feedback mechanisms, it minimizes essential human elements in education, such as teacher-student interaction and peer collaboration, that help develop students' social and behavioral skills [45].

6.3.1 Addressing Overdependence

The strategy to tackle overreliance on AI in education spans various fronts, from educational initiatives to regulatory measures. While the degree of restriction on AI usage is left to the discretion of educators, we recommended that if restrictions are imposed, they should be

accompanied by digital literacy programs. These initiatives can help students view AI as a supportive tool rather than a crutch. Central to this approach is incorporating educational initiatives that enhance understanding and effective use of AI technologies while emphasizing ethical considerations. By promoting a curriculum that underscores the importance of human oversight and ethical use of technology, students are better equipped to recognize AI's limitations and navigate its complexities responsibly [45, 46]. Research findings, such as those by Vasconcelos et al. [2023], shed light on the nuanced nature of AI reliance in decision-making processes. These insights emphasize the significance of adequate explanations in mitigating overreliance alongside considerations of cognitive costs and contextual utility [65]. Integrating AI into project-based learning environments offers students hands-on experience, demystifying its complexities, fostering high effort, and rewarding collaborative engagement. Measures such as plagiarism checkers and requiring students to cite AI usage can be implemented to reinforce accountability and responsible AI usage further. These steps promote transparency and underscore the importance of ethical considerations in AI utilization. By adopting a comprehensive approach that combines educational, ethical, and practical strategies, educators can empower students to navigate the evolving landscape of AI with confidence and responsibility [45, 46, 65].

VII. CONCLUSION

Exploring the landscape of adaptive learning, particularly when hand-in-hand with AI, revealed a profound evolution in educational methodologies. From the early days marked by the pioneering efforts of macro and micro-adaptive systems like the Keller Plan and teaching machines to the contemporary period dominated by advanced AI algorithms, the trajectory underscores a relentless pursuit of diverse learning experiences. The adaptive learning framework outlines essential elements such as learner modeling, content personalization, and instantaneous feedback, illustrating the holistic approach necessary for the success of adaptive systems. However, it is imperative to acknowledge the drawbacks of older AI algorithms like Decision Trees, which paved the way for more dynamic solutions such as ANNs, RNNs, and BNNs. In doing so, we understand what to keep from old technologies and what to leave behind, much like an LSTM or GRU. We also discussed several implementations of AI in ALSs, as exemplified by platforms like ALEKS and BITS, showcasing the tangible impact of these technologies in revolutionizing education. Yet, alongside these advancements come ethical considerations that demand careful navigation and proactive mitigation strategies. In navigating these challenges lies the promise of a future where adaptive learning powered by AI enhances educational outcomes and promotes inclusivity, equity, and ethical practice. It is a future where technology catalyzes empowering learners, educators, and academic institutions alike to thrive in an ever-changing environment of knowledge sharing. Ultimately, the intertwining of AI and adaptive learning represents more than just technological innovation; it embodies a paradigm shift, parallel to the 1970s AI Movement, in how we consume, deliver, and engage with education. As we stride into this future, it is imperative to remain vigilant, continuously assessing, refining, and reshaping our approaches to ensure that the transformative potential of AI in adaptive learning is realized in its fullest, most beneficial form for all stakeholders involved.

References

[1] L. Pugliese. 2016. Adaptive Learning Systems: Surviving the Storm. Retrieved from https://er.educause.edu/articles/2016/10/adaptive-learning-systems-surviving-the-storm.

[2] Wikipedia contributors, 2024. Educational Technology. Retrieved from https://en.wikipedia.org/wiki/Educational_technology.

[3] T. Kabudi, I. Pappas, and D. H. Olsen. 2021. AI-enabled adaptive learning systems: A systematic mapping of the literature. *Computers and Education: Artificial Intelligence*, 2, p. 100017. https://doi.org/10.1016/j.caeai.2021.100017

[4] Felix Moedritscher, Victor Garcia-Barrios, and Christian Guetl. 2004. The Past, the Present and the Future of Adaptive E-Learning: An Approach within the Scope of the Research Project AdeLE. In *Proceedings of the international conference on interactive computer aided learning*.

[5] Heidi L. Eyre. 2007. Keller's Personalized System of Instruction: Was it a Fleeting Fancy or is there a Revival on the Horizon? *The Behavior Analyst Today,* 8(3), p.317. https://doi.org/10.1037/h0100623

[6] K. A. R.A Nuri and Nese Sevim. 2013. Adaptive learning systems: Beyond teaching machines. *Contemporary Educational Technology*, 4(2), 108-120.

[7] Holloway. A Brief History of AI. (2022). Retrieved from https://www.holloway.com/g/making-things-think/sections/a-brief-history-of-ai

[8] Bernd Klein. 2023. Neural Networks, Structure, Weights and Matrices. (June 2023). Retrieved from https://python-course.eu/machine-learning/neural-networks-structure-weights-and-matrices.php

[9] Wikipedia contributors. 2024. Rectifier (neural networks). In *Wikipedia, The Free Encyclopedia.* Retrieved from https://en.wikipedia.org/w/index.php?title=Rectifier_(neural_networks)&oldid=1199800058

[10] Melody Y. Yang. 2003. Neural Networks. In *Encyclopedia of Information Systems,* 303-315, https://doi.org/10.1016/B0-12-227240-4/00121-0

[11] Hunter Philips. A Simple Introduction to Softmax. (May 2023). Retrieved from https://medium.com/@hunter-j-phillips/a-simple-introduction-to-softmax-287712d69bac

[12] Ariel Lu. 2019. Understanding Neural Networks: The Secret Lies in Our Brains. Retrieved from https://medium.com/analytics-vidhya/understanding-neural-networks-the-secret-lies-in-our-brains-5d8dc907ab34

[13] Vikram Mullachery, Aniruddh Khera, & Amira Husain. Bayesian Neural Networks. Retrieved March 29, 2024 from https://arxiv.org/pdf/1801.07710.pdf

[14] Mark Jerrum and Alistair Sinclair. 1996. The Markov chain Monte Carlo method: an approach to approximate counting and integration. *Approximation algorithms for NP-hard problems*, 482-520.

[15] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. 2022. Hands-on Bayesian neural networks—A tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2), 29-48.

[16] Bahzad Charbuty and Adnan Abdulazeez. 2021. Classification based on decision tree algorithms for machine learning. *Journal of Applied Science and Technology Trends,* 2(01), 20-28.

[17] Sonia Singh and Priyanka Gupta. 2014. Comparative study ID3, cart and C4. 5 decision tree algorithm: a survey. *International Journal of Advanced Information Science and Technology*, 27(27), 97-103.

[18] IBM. IBM SPSS Modeler V18.0.0 documentation. 2021. Retrieved from https://www.ibm.com/docs/en/spss-modeler/18.0.0?topic=networks-basics-neural

[19] Tony Thomas, Athira P. Vijayaraghavan, Sabu Emmanuel, Tony Thomas, Athira P. Vijayaraghavan, and Sabu Emmanuel. 2020. Applications of decision trees. In *Machine learning approaches in cyber security analytics,* 157-184, https://doi.org/10.1007/978-981-15-1706-8_9

[20] Sotiris B. Kotsiantis. 2013. Decision trees: a recent overview. *Artificial Intelligence Review* 39, 261-283.

[21] Dragos D. Margineantu and Thomas G. Dietterich. 2003. Improved class probability estimates from decision tree models. In *Nonlinear Estimation and Classification*, 173-188,

[22] Khalid Colchester, Hani Hagras, Daniyal Alghazzawi, and Ghadah Aldabbagh. 2017. A survey of artificial intelligence techniques employed for adaptive educational systems within e-learning platforms. *Journal of Artificial Intelligence and Soft Computing Research* 7, 1, 47-64 https://doi.org/10.1515/jaiscr-2017-0004

[23] Shreeharsh Kelkar. 2022. Between AI and learning science: the evolution and commercialization of intelligent tutoring systems. *IEEE Annals of the History of Computing*, 44(1), 20-30, https://doi.org/10.1109/MAHC.2022.3143816.

[24] Nataliia Valko and Viacheslav Osadchyi. 2020. Education individualization by means of artificial neural networks. *The International Conference on Sustainable Futures: Environmental, Technological, Social and Economic Matters,* Article 10021, 6 pages, https://doi.org/10.1051/e3sconf/202016610021

[25] Norsham Idris, Norazah Yusof and Puteh Saad. 2009. Adaptive course sequencing for personalization of learning path using neural network. *Int. J. Advance. Soft Comput. Appl*, 1(1), 49-61.

[26] Erwin T. Lau, Licheng Sun, and Quiang Yang. (2019) Modelling, prediction and classification of student academic performance using artificial neural networks. *SN Appl. Sci.* 1, 982 (2019), https://doi.org/10.1007/s42452-019-0884-7

[27] Rohit B. Kaliwal and Santosh Deshpande. (2021). Design of intelligent e-learning assessment framework using bayesian belief network. *J Eng Educ Transform*, 34, 651-658.

[28] McGraw Education. All About Knowledge Checks. Retrieved from https://www.aleks.com/resources/ALEKS_Knowledge_Checks_Overview_for_Students.pdf

[29] Anthony F. Botelho, Ryan S. Baker, and Neil T. Heffernan. 2017. Improving sensor-free affect detection using deep learning. In *Artificial Intelligence in Education: 18th International Conference, AIED 2017, Wuhan, China, June 28–July 1, 2017, Proceedings 18*, 40–51, https://doi.org/10.1007/978-3-319-61425-0_4

[30] R. Subha, N. Gayathri, S. Sasireka, R. Sathiyabanu, B. Santhiyaa, and B. Varshini. 2024. Intelligent Tutoring Systems using Long Short-Term Memory Networks and Bayesian Knowledge Tracing. In *2024 5th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI)*, 24-29, https://doi.org/10.1109/ICMCSI61536.2024.00010.

[31] UC Berkeley. Leveraging AI to improve adaptive tutoring systems. Retrieved from https://bse.berkeley.edu/leveraging-ai-improve-adaptive-tutoring-systems

[32] Anagha Koustubh Joshi, Thakare, A. P., & Thorat, S. B. (2015). An Intelligent Tutoring System using Bayesian Network-A Review. International Journal of Electronics, Communication and Soft Computing Science & Engineering (IJECSCSE), 4, 143.

[33] Hana Fakhira Almarzuki1, Khyrina Airin Fariza Abu Samah, Siti Khatijah Nor Abdul Rahim, Shafaf Ibrahim, and Lala Septem Riza. 2024. Enhancement of Prediction Model for Students' Performance in Intelligent Tutoring System. *Journal of Artificial Intelligence and Technology,* https://doi.org/10.37965/jait.2024.0319.

[34] Mark Krawitz, Jonathan Law, and Sacha Litman. 2018. How higher-education institutions can transform themselves using advanced analytics. McKinsey and Company. Retrieved from https://www.mckinsey.com/industries/education/our-insights/how-higher-education-institutions-can-transform-themselves-using-advanced-analytics

[35] Andy Viano. 2023. How Learning Analytics Impacts Higher Education. EdTech. Retrieved from https://edtechmagazine.com/higher/article/2023/06/learning-analytics-impact-higher-education-perfcon

[36] Shuai Wang, Claire Christensen, Wei Cui, Richard Tong, Louise Yarnall, Linda Shear, and Mingyu Feng. 2023. When adaptive learning is effective learning: comparison of an adaptive learning system to teacher-led instruction. Interactive Learning Environments, 31(2), 793–803. https://doi.org/10.1080/10494820.2020.1808794.

[37] IBM. 2023. Shedding light on AI bias with real world examples. Retrieved from https://www.ibm.com/blog/shedding-light-on-ai-bias-with-real-world-examples/.

[38] The University of Kansas. 2024. Helping students understand the biases in generative AI. Retrieved from https://cte.ku.edu/addressing-bias-ai.

[39] Pinkesh Patel. 2021. Addressing different types of bias in AI. Retrieved from https://medium.com/unpackai/addressing-different-types-of-bias-in-ai-e4b3f06f7c18.

[40] R. S. Baker and A. Hawn. Algorithmic Bias in Education. *International Journal of Artificial Intelligence in Education* 32, 1052–1092 (2022). https://doi.org/10.1007/s40593-021-00285-9.

[41] Henry Anderson, Afshan Boodhwani, and Ryan Baker. 2019. Assessing the Fairness of Graduation Predictions, In *12th International Conference on Educational Data Mining*, 2019.

[42] P. M. Regan and J. Jesse. 2019. Ethical challenges of edtech, big data and personalized learning: twenty-first century student sorting and tracking. *Ethics and Information Technology* 21, 167–179 (2019). https://doi.org/10.1007/s10676-018-9492-2.

[43] O'Connor, S., Liu, H. 2023. Gender bias perpetuation and mitigation in AI technologies: challenges and opportunities. *AI & Society*, https://doi.org/10.1007/s00146-023-01675-4

[44] Dian Schaffhuaser. 2018. The Rocky Road of Using Data to Drive Student Success. Retrieved from https://campustechnology.com/Articles/2018/07/26/The-Rocky-Road-of-Using-Data-to-Drive-Student-Success.aspx?Page=1.

[45] Eman A. Alasadi and Carlos R Baiz. 2023. Generative AI in education and research: Opportunities, concerns, and solutions. *Journal of Chemical Education,* 100(8), 2965-2971, https://doi.org/10.1021/acs.jchemed.3c00323.

[46] Mitchel Resnik. 2024. Generative AI and Creative Learning: Concerns, Opportunities, and Choices. Retrieved from https://mit-genai.pubpub.org/pub/gj6eod3e/release/2.

[47] Reid Blackman and Beena Ammanath. 2022. Building Transparency into AI Projects. Retrieved from https://hbr.org/2022/06/building-transparency-into-ai-projects.

[48] Maryam Roshanaei. 2024. Towards best practices for mitigating artificial intelligence implicit bias in shaping diversity, inclusion and equity in higher education. *Education and Information Technologies*, https://doi.org/10.1007/s10639-024-12605-2

[49] Diversity.AI. 2024. Problem Definition. Retrieved from https://diversity.ai/#problemdefinition

[50] Samuel Drews, Awa Albarghouthi, and Loris D'Antoni. 2020. Proving data-poisoning robustness in decision trees. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, 1083-1097.

[51] Yong Soo Kim. 2008. Comparison of the decision tree, artificial neural network, and linear regression methods based on the number and types of independent variables and sample size. *Expert Systems with Applications* 34, 2, 1227-1234. https://doi.org/10.1016/j.eswa.2006.12.017

[52] Pragati Baheti. 2021. Activation Functions in Neural Networks [12 Types & Use Cases]. Retrieved from https://www.v7labs.com/blog/neural-networks-activation-functions.

[53] Robin M. Schmidt. 2019. Recurrent neural networks (RNNs): A gentle introduction and overview.

[54] G. R. Kanagachidambaresan, A. Ruwali, D. Banerjee, and K. B. Prakash. 2021. Recurrent Neural Network. *Programming with TensorFlow*, 53-61, https://doi.org/10.1007/978-3-030-57077-4_7

[55] Zhaoyang Niu, Guoqiang Zhong, Guohua Yue, Li-Na Wang, Hui Yu, Xiao Ling, and Junyu Dong. 2023. Recurrent attention unit: A new gated recurrent unit for long-term memory of important parts in sequential data. *Neurocomputing* 517, 1-9, https://doi.org/10.1016/j.neucom.2022.10.050.

[56] J. Matayoshi, E. Cosyn, and H. Uzun. 2019a. Deep (Un)Learning: Using Neural Networks to Model Retention and Forgetting in an Adaptive Learning System. In *Artificial Intelligence in Education* 11625, 258-269, https://doi.org/10.1007/978-3-030-23204-7_22.

[57] J. Matayoshi, E. Cosyn, and H. Uzun. 2019b. Using Recurrent Neural Networks to Build a Stopping Algorithm for an Adaptive Assessment. In *Artificial Intelligence in Education* 11626, 179 - 184, https://doi.org/10.1007/978-3-030-23207-8_34.

[58] J. Matayoshi, E. Cosyn, and H. Uzun. 2021 Are We There Yet? Evaluating the Effectiveness of a Recurrent Neural Network-Based Stopping Algorithm for an Adaptive Assessment. *International Journal of Artificial Intelligence in Education* 31, 304–336, https://doi.org/10.1007/s40593-021-00240-8.

[59] McGraw Hill. 2024. About ALEKS. Retrieved from https://www.aleks.com/about_aleks.

[60] Christina Stahl and Cord Hockemeyer. 2022. Knowledge Space Theory. Retrieved from https://cran.r-project.org/web/packages/kst/vignettes/kst.pdf

[61] ALEKS. 2013. Instructor's Manual for Chemistry. Retrieved from https://www.aleks.com/manual/pdf/educators-ichem.pdf.

[62] Shan Hua. 2004. *BITS: A Bayesian intelligent tutoring system for computer programming.* Master's Thesis. University of Regina, Regina, Saskatchewan.

[63] Priyank Jain, Manasi Gyanchandani, and Nilay Khare. 2016. Big data privacy: a technological perspective and review. *Journal of Big Data* 3, 25, https://doi.org/10.1186/s40537-016-0059-y.

[64] Joel R. Reidenberg and Florian Schaub. 2018. Achieving big data privacy in education. *Theory and Research in Education* 16, no. 3 (2018): 263-279, https://doi.org/10.1177/1477878518805308

[65] Helena Vasconcelos, Matthew Jörke, Madeleine Grunde-McLaughlin, Tobias Gerstenberg, Michael S. Bernstein, and Ranjay Krishna. 2023. Explanations can reduce overreliance on ai systems during decision-making. In *Proceedings of the ACM on Human-Computer Interaction* 7, no. CSCW1, 1-38, https://doi.org/10.1145/3579605.

[66] Jan Kocoń, Igor Cichecki, Oliwier Kaszyca, Mateusz Kochanek, Dominika Szydło, Joanna Baran, and Julita Bielaniewicz. 2023. ChatGPT: Jack of all trades, master of none. *Information Fusion* 99, https://doi.org/10.1016/j.inffus.2023.101861.

[67] Zachary A. Pardos, Matthew Tang, Ioannis Anastasopoulos, Shreya K. Sheel, and Ethan Zhang. 2023. Oatutor: An open-source adaptive tutoring system and curated content library for learning sciences research. In *Proceedings of the 2023 chi conference on human factors in computing systems*, 1-17,2023.

[68] Ashok K. Goel and Lalith Polepeddi. 2018. Jill Watson: A virtual teaching assistant for online education." In *Learning engineering for online education,* 120-143, https://doi.org/10.4324/9781351186193

[69] Online Education. 2024. AI-Powered Adaptive Learning: A Conversation with the Inventor of Jill Watson. Retrieved from https://www.onlineeducation.com/features/ai-teaching-assistant-jill-watson

[70] Nisha S. Raj and V. G. Renumol. 2018. Architecture of an adaptive personalized learning environment (APLE) for content recommendation." In *Proceedings of the 2nd international conference on digital technology in education*, 17-22.