Hindawi
Complexity
Volume 2019, Article ID 1306039, 22 pages
https://doi.org/10.1155/2019/1306039

WILEY | Hindawi

Review Article

A Systematic Review of Deep Learning Approaches to Educational Data Mining

Antonio Hernández-Blanco,[1] **Boris Herrera-Flores**,[2] **David Tomás**,[3] **and Borja Navarro-Colorado**[3]

[1]*Technical University of the North, Ecuador*
[2]*Central University of Ecuador, Ecuador*
[3]*University of Alicante, Spain*

Correspondence should be addressed to Antonio Hernández-Blanco; antoniojhb@gmail.com

Received 28 December 2018; Revised 28 March 2019; Accepted 10 April 2019; Published 12 May 2019

Academic Editor: Roberto Natella

Educational Data Mining (EDM) is a research field that focuses on the application of data mining, machine learning, and statistical methods to detect patterns in large collections of educational data. Different machine learning techniques have been applied in this field over the years, but it has been recently that Deep Learning has gained increasing attention in the educational domain. Deep Learning is a machine learning method based on neural network architectures with multiple layers of processing units, which has been successfully applied to a broad set of problems in the areas of image recognition and natural language processing. This paper surveys the research carried out in Deep Learning techniques applied to EDM, from its origins to the present day. The main goals of this study are to identify the EDM tasks that have benefited from Deep Learning and those that are pending to be explored, to describe the main datasets used, to provide an overview of the key concepts, main architectures, and configurations of Deep Learning and its applications to EDM, and to discuss current state-of-the-art and future directions on this area of research.

1. Introduction

The research field of *Educational Data Mining* (EDM) focuses on the application of techniques and methods of data mining in educational environments. EDM is concerned with developing, researching, and applying assumption-based decisions and impulsive reasoning to detect patterns in large collections of educational data that would otherwise be impossible to analyze[1].

EDM leverages e-learning platforms such as *Learning Management Systems* (LMS), *Intelligent Tutoring Systems* (ITS), and, in the last years, Massive Open Online Courses (MOOC), to obtain rich and multimodal information from student's learning activities in educational settings. For instance, these platforms record when the students access a learning object, how many times they accessed it, whether the answer provided to an exercise is correct or not, or the amount of time spent reading a text or watching a video.

All this information can be analyzed to address different educational issues, such as generating recommendations, developing adaptative systems, and providing automatic grading for the students' assignments. Different machine learning techniques have been applied over time to analyze this data, but it has been in recent years that the use of Deep Learning techniques has emerged in the field of EDM.

The topic of *Deep Learning* (DL) has gained increasing attention in the industry and research areas in the last decade, revolutionizing the field of machine learning by obtaining state-of-the-art results in perception tasks such as image and speech recognition [2]. Major companies such as Google, Facebook, Microsoft, Amazon, and Apple are heavily investing in the development of software and hardware innovations in this field, trying to leverage DL potential in the production of smart products.

DL is based on neural network architectures with multiple layers of processing units that apply linear and nonlinear

transformations to the input data. These a rchitectures can be applied to all type of data: image, audio, text, numerical, or some combination of them. Many research fields have benefited from applying these technologies, and EDM is not an exception.

In the last few years there has been a proliferation of research in the EDM field using DL architectures. This article presents a review on the literature of DL techniques applied to EDM, from its first appearance in 2025 to the present day. The primary contributions of this article are as follows:

(i) Summarize the main EDM tasks and classify the existing works that have applied DL on each of these tasks.

(ii) Identify the tasks that have gained major attention and those that are still unexplored.

(iii) Describe and categorize the main public and private datasets employed to train and test DL models in EDM tasks.

(iv) Introduce key DL concepts and technologies, describing the techniques and configurations most widely used in EDM and its specific tasks.

(v) Discuss future directions for research in DL applied to EDM based on the information gathered in this study.

The rest of this article is organized as follows: Section 2 presents and compares previous surveys in the field of EDM; Section 3 describes the process carried out to retrieve the papers reviewed in this study, including a quantitative analysis of the papers gathered; Section 4 describes the main tasks in EDM, identifies the existing literature in each task, and describes the main datasets employed in the field; Section 5 presents the key concepts of DL, the main architectures, configurations, and frameworks, summarizing the characteristics (in terms of DL technologies) of the work done in EDM; Section 6 presents a discussion about the information compiled during this review work; finally, conclusions are presented in Section 7.

2. Review of Previous Surveys

The application of data mining techniques to educational environments has been an active research field in the last few decades, gaining much popularity in recent times thanks to the availability of online datasets and learning systems. Different surveys have been published about EDM so far, and this section summarizes these works and presents the key differences between the current proposal and the previous reviews in this field.

The first EDM survey identified in the literature was developed in 2007 by Romero and Ventura [3], which was further improved in 2010 [4] and 2013 [5]. In the later, the authors analyzed more than 300 studies carried out before 2010, identifying eleven categories or tasks in EDM: analysis and visualization of data, providing feedback for supporting instructors, recommendations for students, predicting student's performance, student modeling detecting undesirable student behaviors, grouping students, social network analysis, developing concept maps, constructing coursewares, and planning and scheduling. The survey presented methods and techniques employed in the EDM field in each of these categories.

In 2009, a new EDM survey was presented by Baker and Yacef [6]. This study discussed trends and shifts in research conducted by this community, comparing its current state with the early years of EDM. In this case, the authors identified four applications/tasks in this field: improving student models, improving domain models, studying the pedagogical support provided by learning software, and scientific research into learning and learners. The most-cited papers in EDM between 1995 and 2005 were listed, discussing their influence on the EDM community.

Peña-Ayala proposed in 2014 a thorough survey by applying data mining techniques to more than 240 papers in EDM [7]. The execution of statistical and clustering processes identified a set of educational functionalities, a pattern of EDM approaches, and two patterns of value-instances to depict EDM approaches based on descriptive and predictive models. Unlike previous literature reviews, this work mainly focused on computational techniques rather than EDM applications.

More recently, two new studies have been added to this list of surveys. The first one was carried out by Bakhshinategh et al. in 2018 [8]. This work studied various tasks and applications existing in the field of EDM and categorized them based on their purposes. Based on the eleven categories proposed by [4], they suggested a hierarchy of thirteen categories grouped into five main tasks: Student Modeling, Decision Support Systems, Adaptive Systems, Evaluation, and Scientific Inquiry. In Section 4.1, this taxonomy of tasks is used as the basis to classify the current studies in DL for EDM.

Finally, the most recent review devoted to EDM has been developed by Aldowah et al. [9] in 2019. This study constrained the research to works applied in the context of higher education. The analysis presented was based on four dimensions: computer supported learning analytics, computer supported predictive analytics, computer supported behavioral analytics, and computer supported visualization. Based on the results of previous studies, the authors found that specific EDM techniques could offer the best means of solving certain learning problems, offering student-focused strategies and tools for educational institutions.

In these review papers there are two aspects that have not been studied in a systematic way, and that the present work intends to analyze: the existing datasets and the use of DL techniques in EDM. Firstly, in order to empirically compare different approaches, it is necessary to know the underlying datasets employed in the experiments. In this paper, a section is devoted to review and summarize these resources (see Section 4.2). Secondly, although previous proposals have taken into account (shallow) neural networks approaches in the literature, none of them is specifically focused on DL techniques. In this paper, Section 5 provides an introduction to the foundations of DL (main architectures, training process, hyperparameters, and frameworks), characterizing these techniques in the EDM domain and relating them to the papers reviewed.

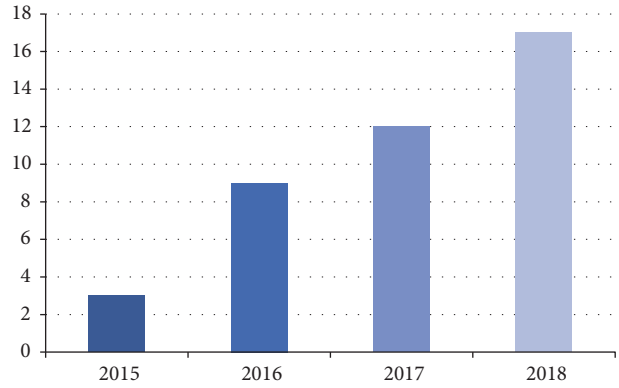

Figure 1: Number of papers published per year.

3. Methodology

This section describes the methodology followed to carry out this review and the process of gathering, analyzing and extracting the existing works on DL applications to EDM.

In order to perform a systematic review, the following scientific repositories were accessed: ACM Digital Library (https://dl.acm.org/), Google Scholar (https://scholar.google.es/), and IEEE Xplore (https://ieeexplore.ieee.org/). These sources were queried with the following search string: `"deep learning" AND "educational data mining"`. As a result, a large set of papers was retrieved and a manual review process was applied to filter out duplicates and papers on unrelated topics. The bibliography cited in the papers that initially passed the filter was also reviewed. This allowed expanding the number of relevant papers retrieved. The final set contained 41 papers. Figure 1 summarizes the number of publications per year. The first papers applying DL to EDM were published just four years ago, in 2015, and there is clearly an increase in the number of publications over the years until 2018.

Table 1 summarizes the number of papers published in each publication venue. Most of them have been published in conferences (80%). The *International Conference on Educational Data Mining* accumulates the maximum number of publications (considering the last three editions), with a total of 16. Not surprisingly, this is the congress of reference in the EDM field.

Finally, Figure 2 shows a choropleth map of the world showing the density of researchers per country involved in the area of DL applied to EDM, based on their affiliation. Authors are weighted by the number of contributors to the paper. For instance, in a paper with n authors, each one will contribute to their country with a weight of $1/n$. The map shows that United States is the more active country in this field, followed (at a great distance) by India, Canada and China. Other countries where researchers have contributed to this field are New Zealand, Singapore, Japan, Argentina, Australia, and Serbia.

4. Educational Data Mining

The first part of this section shows taxonomy of the tasks addressed by EDM systems. The works reviewed are briefly described and classified using this taxonomy in order to differentiate the tasks that have been faced by DL approaches from those that are still unexplored. The second part of the section describes the main datasets used in the field, also grouped by the task addressed.

4.1. Tasks. In the last years, different surveys have focus in different aspects of EDM systems. A recent study is described in [8]. An interesting aspect of this work is the development of a novel taxonomy of tasks in EDM. This taxonomy is used in this section as the basis to classify the papers gathered in the field of DL applied to EDM. The taxonomy comprises thirteen tasks:

(i) Predicting student performance: the objective is to estimate a value or variable describing the students' performance or the achievement of learning outcomes.

(ii) Detecting undesirable student behaviors: the focus here is on detecting undesirable student behavior, such as low motivation, erroneous actions, cheating, or dropping out.

(iii) Profiling and grouping students: the purpose is to profile students based on different variables, such as knowledge background, or to use this information to group students for various purposes.

(iv) Social network analysis: the aim is to obtain a model of students in the form of a graph, showing different possible relationships among them.

(v) Providing reports: the purpose is to find and highlight the information related to course activities which may be of use to educators and administrators, providing them with feedback.

(vi) Creating alerts for stakeholders: the objective is to predict student characteristics and detect unwanted behavior, serving as an online tool for informing stakeholders or creating alerts in real time.

(vii) Planning and scheduling: the aim is to help stakeholders in the task of planning and scheduling.

(viii) Creating courseware: the purpose is to help educators to automatically create and development course materials using students' usage information.

(ix) Developing concept maps: the objective is to develop concept maps of various aspects to help educators define the process of education.

(x) Generating recommendation: the objective is to make recommendations to any stakeholders, although the main focus is usually on helping students.

(xi) Adaptive systems: this task is related to the use of intelligent systems in computer based learning, where the system has to adapt to the user's behavior.

Table 1: Number of papers over publication venue.

Type	Publication venue	Number
Conference	International Conference on Educational Data Mining (2016, 2017, 2018)	16
	Third ACM Conference on Learning @ Scale (2016, 2017)	5
	Artificial Intelligence in Education	2
	IEEE International Conference on Data Mining Workshop (ICDMW 2015)	1
	International Symposium on Educational Technology (ISET)	1
	Seventh International Learning Analytics and Knowledge Conference	1
	Annual Conference on Neural Information Processing Systems (NIPS)	1
	Conference on Empirical Methods in Natural Language Processing (2016)	1
	26th Conference on User Modeling, Adaptation and Personalization	1
	2nd International Conference on Crowd Science and Engineering	1
	Neural Information Processing Systems, Workshop on Machine Learning for Education	1
	2nd International Conference on Innovation in Artificial Intelligence	1
	20th ACM International Conference on Multimodal Interaction	1
Journal	CoRR	4
	International Journal of Applied Engineering Research	1
	Journal of Educational Data Mining	1
	Journal of Engineering and Applied Sciences	1
	Journal of Educational Computing Research	1

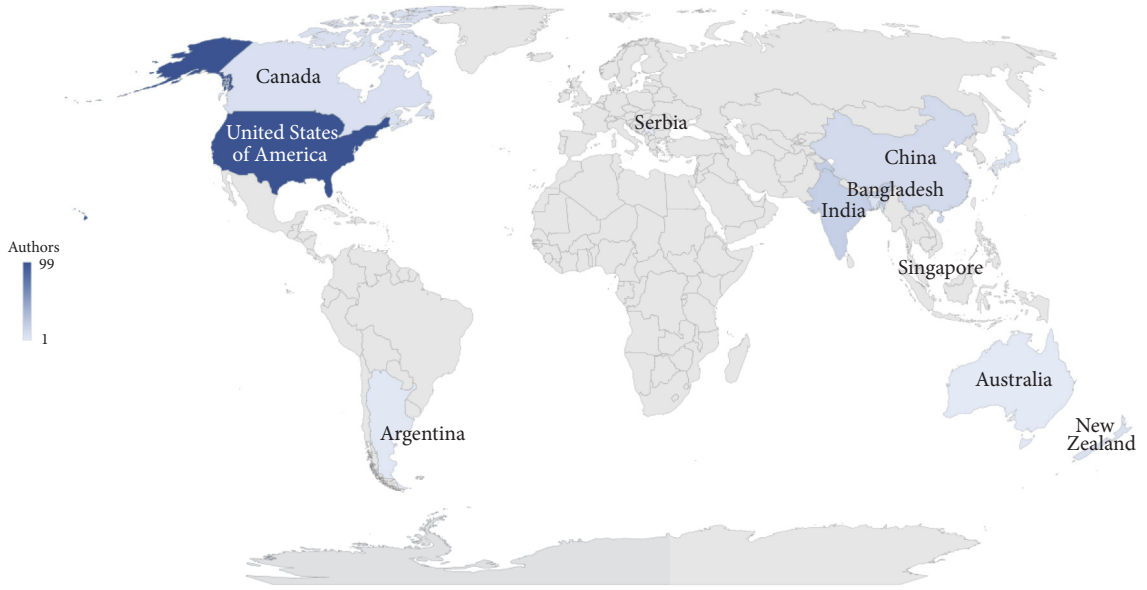

Figure 2: Choropleth map showing the density of researchers per country in the papers reviewed based on their affiliation.

(xii) Evaluation: the goal is to provide an automatic evaluation tool to help educators.

(xiii) Scientific inquiry: mostly targeted on researchers as the end users, but developed or tested theories can be used afterwards in other applications with different stakeholders.

All the works analyzed in this review fall into four of these thirteen categories: predicting student performance, detecting undesirable student behavior, generation recommendations, and evaluation. The other nine categories remain empty. Table 2 summarizes these four tasks in EDM (first column), the references to the works in the field (second column), the datasets employed (third column), and the types of datasets (fourth column). This last column specifies if the dataset has been created specifically for the experiments carried out ("Specific") or if it is a general dataset used in other works ("General"). The following subsections present

Table 2: Summary of EDM tasks, approaches, datasets, and types of datasets. "Specific" means that the dataset has been created for a specific study, and "General" means that it has been used in different publications.

Task	Reference	Dataset	Type
Predicting student performance, achievement of learning outcomes or characteristics	Lin and Chi, 2017 [11]	ITS Pyrenees	Specific
	Zhang et al., 2017 [49]	ASSISment and OLI datasets	General
	Kim et al., 2018 [26]	Udacity	Specific
	Lalwani and Agrawal, 2017 [14]	Funtoot dataset	Specific
	Okubo et al., 2017 [24]	Information Science Course dataset	Specific
	Guo et al., 2015 [23]	High schools dataset	Specific
	Sharada et al., 2018 [22]	ASSIStment 2018	General
	Wang et al., 2017 [12]	Code course dataset	Specific
	Tang et al., 2016 [21]	Kaggle Automated Essay Scoring	General
	Bendangnuksung and P., 2018 [20]	Kaggle Students' Academic Performance dataset	General
	Mao et al., 2018 [15]	ITS Pyrenees and ITS Cordillera	Specific
	Wilson et al., 2016 [50]	ASSISTment 2009-2010, KDD Cup 2010 and ITS Knewton	General
	Wilson et al., 2016 [16]	ASSISTment 2009-2010 dataset, KDD Cup 2010 dataset and ITS Knewton	General
	Khajah et al., 2016 [17]	Assistment 2009-2010 dataset, virtual student dataset, and data from Spanish and Engineering courses	General and Specific
	Xiong et al., 2016 [18]	ASSISTments 2009-2010 dataset	General
	Wang et al., 2017 [53]	KDD Cup 2015 dataset	General
	Kim et al., 2018 [27]	Udacity	Specific
	Montero et al., 2018 [13]	ASSISTment 2009-2010 dataset, KDD Cup 2010 dataset and ITS Woot Math	General and specific
	Piech et al., 2015 [10]	Virtual student dataset and Assistments 2009-2010 dataset	General
	Singh et al. 2018 [54]	Kaggle Automated Essay Scoring	General
	Alam et al., 2018 [25]	Kaggle Students' Academic Performance dataset	Specific
	Yeung and Yeung, 2018 [19]	ASSISTment 2009, ASSISTment 2015, ASSISTment Challenge, Statics2011, Simulated-5	Specific
Detecting undesirable student behaviors	Aung et al., 2018 [36]	YouTube videos of school classrooms	Specific
	Sharma et al., 2016 [34]	StyleX dataset (multimedia)	Specific
	Teruel and Alemany, 2018 [29]	ASSISTment 2009-2010 dataset and KDD Cup 2015	General
	Fei and Yeung, 2015 [28]	-	-
	Whitehill et al., 2017 [31]	HarvardX MOOCs	General
	Wang et al., 2017 [30]	Code course dataset	Specific
	Min et al., 2016 [33]	Game-based virtual learning environment Crystal Island	Specific
	Tato et al., 2017 [37]	French corpus	Specific
	Yang et al., 2018 [35]	Videos collected in unconstrained environments	Specific
	Xing and Du, 2018 [32]	Canvas project management MOOC	Specific
Generating recommendations	Wong, 2018 [39]	Student transcript records	Specific
	Abhinav et al., 2018 [38]	Learner's profile data	Specific
Evaluation	Akram et al., 2018 [44]	problem-solving dataset from game-based learning environment	Specific
	Zhang et al., 2016 [42]	Short answers from ITS Cordillera	Specific
	Taghipour and Ng, 2016 [41]	Kaggle Automated Essay Scoring	General
	Zhao et al., 2017 [40]	ASSISTment 2009-2010 and Kaggle Automated Essay Scoring	General
	Alvarado et al., 2018 [43]	Short-answer question dataset from biology course	Specific
	Choi et al., 2017 [45]	PODS dataset	Specific
	Sales et al., 2018 [46]	2015 ASSISTments Skill Builder Data	General

each task and the works related in more detail. The details about the DL implementation on each paper are described in Section 5.

4.1.1. Predicting Student Performance. One of the challenges that has gained more attention in this area is *knowledge tracing*. In this subtask the goal is to predict student's future performance based on their past activity. Piech et al. [10] were the first to introduce DL techniques to address this task, largely outperforming previous approaches based on traditional machine learning techniques. These remarkable achievement leads to other researchers to question the validity of the results. A series of works were published afterwards that were for [11–13] or against [14–19] the claims in this paper. The studies that disagree with Piech et al. tried to replicate the results of the experiments and compare them with traditional machine learning techniques in a more fair scenario, arguing that the differences between DL and previous models were not so evident. Also in the task of knowledge tracing, but away from the controversy initiated by Piech et al., the work in [20] proposed also a DL classifier to predict whether students will fail or pass an assignment.

The work by [21] leveraged a DL model to explore two different contexts within the educational domain: writing samples from students and clickstream activity within a MOOC. The use of a single model and architecture highlighted the flexibility and broad applicability of DL to large, sequential student data.

The work by [22] applied DL to a dataset obtained from a web based mathematics tutor to model student knowledge retention, i.e., the ability of the students to retain the acquired knowledge. The proposal significantly outperformed the baseline method proposed. This approach was later employed to personalize retention tests.

In [23], the authors presented a DL classifier for predicting students' performance, which took advantage of a relatively large real world students' dataset of unlabeled data. The system automatically learned multiple levels of representation and the experimental results showed the effectiveness of the method. In this line, [24] proposed a method for predicting final grades of students applying DL to the log data stored in an educational system. The log data represented the learning activities of students who used the LMS, the e-portfolio system, and the e-book system. The results showed that DL outperformed the traditional machine learning baseline proposed. Reference [25] proposed a model to categorize students into high, medium and low, to determine their learning capabilities and help them to improve their study techniques. A DL model was implemented to provide predictions based on the top features identified. Finally, [26, 27] recast the student performance prediction problem as a sequential event prediction problem and proposed a DL algorithm, called GritNet. The results showed that their proposal outperformed the baseline chosen, obtaining substantially gain in the few weeks when accurate predictions are most challenging.

4.1.2. Detecting Undesirable Student Behaviors. The works focused in the task of detecting undesirable students' behavior have faced three different subtasks: *predicting dropping out in MOOC platforms, addressing the problem of students engagement in their learning, and evaluating social functions.*

In the subtask of dropout prediction in MOOCs, [28] treated this task from a sequence labeling perspective, applying temporal models to solve the problem. Using DL techniques, they obtained significantly better performance than traditional machine learning methods for all three definitions of dropout: participation in the final week, last week of engagement, and participation in the next week. References [29, 30] defined dropout as a binary classification problem. Reference [30] combined different DL architectures in a bottom-up manner, selecting three attributes from the dataset as an input. The results showed that the proposed model could achieve comparable performance to approaches relying on feature engineering performed by experts. Reference [29] optimized a joint embedding function to represent both students and course elements into a single shared space. The results indicated that coembeddings were able to capture the latent causes involved in dropout, outperforming other disjoint and not embedded representations. Reference [31] questioned the fact that dropout prediction focuses on exploring different feature representations and classification architectures, comparing the accuracy of a standard dropout prediction architecture with clickstream features, classified by logistic regression, across a variety of different training settings in order to better understand the trade-off between accuracy and practical deployability of the classifier. Finally, [32] focused on personalize student intervention to compute the dropout probability of individual students each week. A DL model was used to build dropout models and further produce individual student dropout probabilities. Instructors could use this information to personalize and prioritize intervention for academically at-risk students. The results supported the benefits of DL for prediction and personalized intervention design on a MOOC course data.

Regarding the study of how engaged are students in their learning, in [33] the students were observed through a live feed that included the student's facial video, the student's gaze superimposed in real time over a video capture of the screen, and the student's voice as recorded through a headset microphone. To these end, a DL-based dialogue act classifier that utilizes these three data sources was implemented. Empirical results suggested that DL models that utilize game trace logs and facial action units achieved the highest predictive accuracy. In [34] the assumption was that if educational videos are not engaging, then students tend to lose interest in the course content. The authors combined audio and visual information to predict the liveliness in a video using DL. The results demonstrated significant improvement compared to traditional state-of-the-art methods. The work by [35] was focused on the movements of gaze and pose to determine the engagement intensity while watching online educational courses videos. The authors developed a DL framework that accepted multiple input features (statistical characteristics, facial descriptors, and action features) and evaluated how different modalities performed using this framework. Experimental results demonstrated the effectiveness of the method proposed. Another work addressing students engagement

was developed by [36]. They identified disengaged or distressed students, helping teachers to better recognize whether they are paying attention to the right thing or the right student in the classroom. A DL-based prototype system was developed for automated eye gaze following, which estimated for each person in the classroom where they were looking at. The proposed method could estimate the gaze target location of each person in the image with accuracy substantially better than chance and higher than other traditional baseline methods.

Finally, the work by [37] proposed a DL model to evaluate sociomoral reasoning maturity, a key social ability necessary for adaptive social functioning. This model was used in a serious game to evaluate students, outperforming traditional machine learning approaches in this context.

4.1.3. Generating Recommendation. There are two works addressing the recommendation of learning items to assist students. Both studies focused on generating personalized searches based on their preferences and curriculum planning.

Reference [38] proposed a hybrid recommendation system (called *LeCoRe*) that recommended learning opportunities to students based on their (implicit or explicit) preferences, allowing connecting them by similar interests on the platform. LeCoRe combined both content-based and collaborative filtering techniques in its phases. The learner training step applied traditional collaborative filtering algorithms and content-based DL algorithms separately. The authors concluded that the proposed framework was able to successfully model the learner's preferences. In another work, [39] focused on the less investigated problem of curriculum planning for students, providing a novel approach to this domain based on two components: a DL approach to sequential recommendations and a recommender to provide a personalized pathway to completion using sequence, constraint, and contextual parameters.

4.1.4. Evaluation. Different approaches have faced the challenge of providing evaluation tools to help teachers in the grading process. These approaches can be broadly classified in two subtasks: *automated essay scoring* (AES) and *automatic short answer grading* (ASAG).

AES systems are used to evaluate and score written student essays based on a given prompt. Reference [40] proposed a DL-based automated grading model. For each possible score in the rubric, student responses graded with the same score were collected and used as the grading criteria. The DL model learned to predict a score by computing the relevance between the students response and the grading criteria collected. In [41] the authors followed a DL approach to identify the best feature representation to learn the relation between an essay and its assigned score. Results showed an improvement with respect to other approaches requiring feature engineering.

ASAG systems automatically classify students answers as correct or not, based on a previous set of correct answers. Reference [42] studied answer-based, questions, and student models features, both individually and combined, integrating them in different machine learning models. DL obtained the best performance in their experiments. In [43], the authors compared several features for the classification of short open-ended answers, such as n-gram models, entity mentions and entity embeddings. The authors obtained inconclusive results regarding the benefits of using embeddings with respect to traditional n-grams.

Other specific subtasks related to evaluation are also faced in the DL for EDM literature. Reference [44] introduced a temporal analytics framework for stealth assessment that analyzed students' problem-solving strategies in a game-based learning environment. The authors used a DL model on a dataset of problem-solving behaviors, outperforming baseline approaches with respect to stealth assessment predictive accuracy. Reference [45] explored how a DL-based text analysis tool could help assess how students think about different moral aspects. The model was not compared in this case with traditional machine learning approaches. Finally, [46] proposed a DL method to help estimate whether students achieved skill mastery in a set of experiments using A/B tests. This proposal was not compared with traditional machine learning methods.

4.2. Datasets. All these EDM related tasks need different types of educational datasets, both for training and for evaluating the machine learning systems. Some of these datasets are related to how students learn (for example, the success of students developing different types of exercises) and others to how student interact with digital learning platforms (e.g., clickstream or eye-tracking data in MOOCs). This section presents an overview of the main datasets used for EDM in the reviewed papers, as well as other datasets developed for specific studies. These datasets will be related to the tasks identified in the previous section. This information is summarized in the last two columns of Table 2.

4.2.1. Predicting Student Performance. In order to predict student performance it is necessary a dataset of exercises with answers gathered from real students during a period of time. This is exactly the aim of ASSISTment (https://sites.google.com/site/assistmentsdata/) [47, 48]. This dataset is used in many papers to predict student performance [10, 13, 16, 18, 19, 22, 29, 46, 49, 50]. It is composed of a series of mathematics exercises offered to middle-school students through the ASSISTment platform (https://www.assistments.org/), including information such as assignment and user identification, whether the answer is correct on the first attempt or not (a binary flag indicating if the student completed the exercise correctly), the number of student attempts on a problem, answer type, etc. (The full list of features is available here: https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data.) The platform is currently up and running, with new and updated datasets released occasionally (see https://sites.google.com/site/assistmentsdata/home and https://sites.google.com/view/edm-longitudinal-workshop/home).

This dataset is often used jointly with others. For instance, [10] combined ASSISTments 2009-2010 with another two datasets: a sample of anonymized student usage interactions on Khan Academy (https://www.khanacademy.org/) (1.4 million exercises completed by 47,495 students across 69 different exercises) and a dataset of 2,000 virtual students performing the same sequence of 50 exercises drawn from 5 skills. Reference [13] also combined ASSISTments 2009-2010 dataset, in this case with KDD Cup 2010, and with a dataset collected by the Woot Math system (https://www.wootmath.com/). The KDD Cup 2010 dataset comes from an EDM Challenge in 2010 (http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp) and comprises 100 skills from 574 students. These data was extracted from the Cognitive Algebra Tutor system during 2005 and 2006 [51]. The dataset collected by the Woot Math system, a startup that develops adaptive learning environments for mathematics, consists of exercises and the correctness or not of the answers (binary outcome). Reference [18] used also ASSISTments 2009-2010, together with ASSISTments 2014-2015 and KDD Cup 2010. References [16, 50] combined also these datasets and used in addition data collected from the Knewton adaptive learning platform (https://www.knewton.com/). The work by [49] also combined ASSISTments 2009-2010, in this case with the OLI Engineering Statics dataset (https://pslcdatashop.web.cmu.edu/Project?id=48), which included college-level engineering statics. Reference [17] presented a large dataset combining different resources: the ASSISTments 2009-2010 dataset, a synthetic dataset developed by [10], a dataset of 578,726 trials from 182 middle-school students practicing Spanish exercises (translations and simple skills such as verb conjugation), and a dataset from a college-level engineering statics course comprising 189,297 trials of 1,223 exercises from 333 students [52] (https://pslcdatashop.web.cmu.edu/).

Besides this popular dataset, there are others that have been compiled for specific analysis or experiments. All of them are extracted from educational platforms or Intelligent Tutoring Systems (ITS). Regarding educational platforms, [26, 27] compiled several datasets with information about 30,000 students in Udacity (https://www.udacity.com). This data represents users taking a specific action such as watching a video, reading a text page, taking a quiz, or receiving a grade on a project at a certain time stamp. Another work that leverages educational platforms is [20], which used the Students' Academic Performance Dataset from Kaggle (https://www.kaggle.com/aljarah/xAPI-Edu-Data). This dataset consists of 500 students records collected from a learning management system (Kalboard 360) with 16 different features such as gender, nationality, place of birth, topic, visited resource, discussion group, parent answering survey, parent satisfaction, and student absent days. This resource was also used by [25].

In addition to educational platforms, different works used ITS to collect their datasets. Such is the case of [11]. They extracted information from a ITS called Pyrenees. In this case, the dataset contained information about the degree of success of 524 students answering several tests about probability. All the students received the same 12 training problems in the same order. Pyrenees was also used in [15] (68,740 data points from 475 students) together with other dataset collected from a natural language physics ITS, named Cordillera, that teaches students introductory college physics (44,323 data points from 169 students). Another ITS used in these works is Funtoot (https://www.funtoot.com/). Reference [14] used this system to develop a dataset that comprised information about knowledge tracing in online courses, such as the scope of the question (e.g., subject, topic and complexity), start time, total attempts allowed based on the student's performance, time taken, and attempts taken.

Finally, other studies used their own platforms to gather the data. Reference [23] collected real world data from 100 junior high schools. This data was a multilevel representation of student related information: demographic data (e.g., gender, age, health status, and family status), past studies, school assessment data (e.g., school type and school ranking), study data (e.g., middle-term exam, final-term exam, and average), and personal data (e.g., personality, attention and psychology related data). Reference [24] presented a specific dataset for predicting final grades of students, including information about reports, quiz answers, and logbooks of lectures of 108 students attending an Information Science course.

To sum up, either in isolation or in combination with others, the main dataset used for predicting student performance is 2009-2010 ASSITments. Other popular datasets are KDD Cup 2010 and the datasets available at DataShop repository. The rest are specific datasets used in individual studies, which extract data (mainly exercises with real answers) from educational platforms or ITS such as Khan Academy, Woot Math, Udacity, Knewton, Funtoot, and Cordillera.

4.2.2. Detecting Undesirable Student Behaviors. As shown in the previous section, the most salient task for detecting undesirable student behaviors is the study of student dropout in MOOC platforms. There is a set of general purpose datasets that have been developed to address this task.

The main dataset is the KDD Cup 2015 competition (https://biendata.com/competition/kddcup2015/). The challenge proposed in this competition was to predict student dropout on XuetangX, one of the largest MOOC platforms in China. The dataset contains, among others, information about which student enrolls in which course and activity records of the students from 39 courses. Unfortunately, it seems that the data is no longer available. This dataset was used in [29, 53]. The largest dataset for the analysis of student dropout was presented in [31]. This corpus comprises 40 MOOCs from HarvardX with information about number of registered participants and number of participants who certified. It includes additional information such as clickstream data about answers to quiz questions, play/pause/rewind events on lecture videos, and reading and writing to the discussion form. Reference [32] presented a specific dataset for student dropout analysis created from a project management MOOC course hosted by Canvas. It included information about clicks (pages, sources visited, etc.), data from the discussion forum, and quiz scores for every student.

References [12, 30] used a corpus of programming exercises (http://code.org/research) that contains 1,263,360 code submissions about multiple concepts such as loops, if-else statements and nested statements. It is important to note that this dataset is focused on the knowledge of the student (exercises and answers) rather than their behavior in the MOOC platform.

Besides these datasets focused on student dropout, other works have developed datasets for more specific tasks in the context of detecting undesirable student behavior. Related to multimodal interactions, [33] developed a dataset of students interactions within a game-based virtual learning environment called Crystal Island. Both game actions and parallel sensor data were captured to collect cognitive and affective features. This dataset includes information about student interactions in the virtual environment, but not about the student's body of knowledge. Reference [34] also developed a multimedia corpus for the analysis of liveliness of educational videos. The dataset comprises 450 one-minute video snippets featuring 50 different instructors, 10 major topics in engineering, and various accents of spoken English, all of them annotated for liveliness by multiple annotators. Reference [35] presented also a multimedia dataset for engagement prediction. It includes more than 200 videos of 5 minutes long approximately, about 78 subjects (25 female and 53 male) that have been collected in unconstrained environments including office, hotels, and open ground.

In order to detect PODS (privilege, oppression, diversity, and social justice) issues in learning environments, [45] created a domain-specific corpus of short written responses from students on PODS topic in a School of Social Work. From this corpus, authors extracted a specific PODS vocabulary. Finally, for the specific analysis of sociomoral reasoning maturity, [37] developed a corpus of 691 texts in French manually coded by experts, stating the level of maturity in a range from 5 (highest) to 1 (lowest).

4.2.3. Generating Recommendations. Two datasets from the papers reviewed fall in the category of generating recommendation sequences for learning. The first one was described in [38] and presents a dataset of learner's profile information and the courses they have enrolled or completed. The dataset consists of 5,000 unique learners and 49,202 unique course contents, resulting in a total of 2,140,476 enrollments. The second dataset addresses the curriculum planning problem. Reference [39] developed a corpus with 10 years of university student transcript records including 2.1 million transcript results, 30 degrees, 14 majors, 400 courses and 72,000 graduation records. In their research, the authors used a subset containing only undergraduate Engineering and IT students information.

4.2.4. Evaluation. As mentioned in Section 4.1.4, the task of evaluation comprises two main subtasks: automated essay scoring and automatic short answer grading. The essay scoring subtask requires real essays, written by students and graded by teachers, in order to develop systems that are able to score text essays automatically. For this purpose, the Kaggle platform has been used to obtain datasets for automated essay scoring. In fact, there were a specific competition for this task called ASAP (https://www.kaggle.com/c/asap-aes) whose dataset has been used in different works [21, 40, 54]. It consists of essays written in English by students (from Grade 7 to Grade 10), including a score for each one. The essays length is between 150 and 550 words. Reference [21] combined the Kaggle ASAP dataset with clickstream data from a BerkeleyX MOOC from Spring 2013. This is an interesting dataset since it combines content-based resources that show student knowledge with data about student behavior in an online educational platform.

The subtask of automatic short answer grading requires datasets of questions and answers from real students. Reference [42] gathered a corpus from the ITS Cordillera (already mentioned above as a resource for predicting students performance). This dataset includes 16,228 short answers selected from a total of 27,868 dialogues about physics. 61.66% of the corpus is labeled as "correct" while the rest is labeled as "incorrect". Reference [43] presented a corpus of short answer question responses from students, but in this case the topic of the course was human biology. Specifically, the authors used six questions in which students were expected to explain or describe the knowledge obtained during the course in their own words. The answers were manually evaluated by experts with labels like "correct", "incorrect", "incomplete", or "don't-know", among others. Finally, [44] presented a dataset of 244 middle-school students' problem-solving behaviors collected from interactions within a game-based learning environment. The topic of these problems was computational thinking.

5. Deep Learning

DL is undoubtedly the most trending research area in the field of artificial intelligence nowadays. DL is a subfield of machine learning that uses neural network architectures to model high-level abstractions in data. These architectures consist of multiple layers with processing units (*neurons*) that apply linear and nonlinear transformations to the input data. Different DL architectures have been developed and successfully applied to different supervised and unsupervised tasks in the broad fields of natural language processing and computer vision [55].

DL algorithms learn multiple levels of data representations, where higher-level features are derived from lower level features to form a hierarchy. For instance, in an image classification task, the DL model can take pixel values in the input layer and assign labels to the objects in the image in the output layer. Between these layers there are a set of transformation (*hidden*) layers that construct successive higher-order features that are less sensitive to conditions such as lighting and the position of the objects.

The "deep" in DL refers to the multiple transformation layers and levels of representation that lie between the network inputs and hiddens. There is no de facto standard in the number of layers that makes a neural network "deep", but

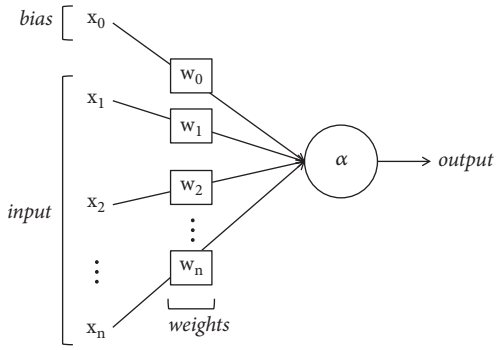

Figure 3: Simple artificial neuron.

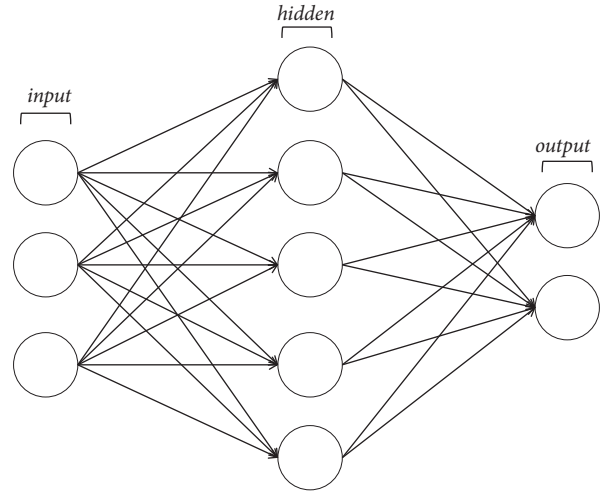

Figure 4: Basic structure of a neural network. Each circular node represents a neuron. Arrows represent connections from the output of one neuron to the input of another.

most research in the field considers that there must be more than two intermediate transformation layers [56].

Many concepts of DL were developed thirty years ago, and some of them long before. However, the most important achievements of DL have taken place in the last ten years. Although there are many factors to explain the raise of DL, it is agreed that the two main causes are the availability of massive amounts of data and the advances in computing power thanks to the use of *Graphic Processing Units* (GPU). In the first case, *big data* facilitates DL algorithms to generalize well. In the second case, GPUs allow massive parallel computing to train bigger and deeper models. Another key factor in the development of DL has been the emergence of software frameworks like TensorFlow, Theano, Keras, and PyTorch, which have allowed researches to focus in the structure of the models rather than in low-level implementation details (see Section 5.5).

Another reason for DL success is that it avoids the need for the feature engineering process. In traditional machine learning, *feature engineering* is the process of selecting the most representative features necessary for the algorithms to work, discarding noninformative attributes. This process is difficult and time-consuming since the correct choice of features is fundamental to the performance of the system [57]. DL performs *feature learning* to automatically discover the representations needed for the task at hand [58].

The following sections describe the foundations of neural networks, training process, main architectures, hyperparameter tuning, and frameworks for developing DL models. Besides providing a general introduction, all these topics will be characterized within the EDM domain, relating them to the papers reviewed.

5.1. Neural Networks. *Neural networks* are computational models based on large sets of simple artificial neurons that try to mimic the behavior observed in the axons of the neurons in human brains. Each node in the network is a neuron, which is the basic processing unit of a neural network.

The form of a simple neuron is depicted in Figure 3. The components of the neuron are input data (x_1, x_2, ..., x_n), which can be the output of another neuron in the network; *bias* (x_0), a constant value that is added to the input of the activation function of the neuron; the weights of each input (w_0, w_1, w_2, ..., w_n), identifying the relevance of the neurons in the model; and the output produced (α). The output of the neuron is computed following this equation:

$$\alpha = f\left(\sum_{i=0}^{n} w_i \cdot x_i\right), \tag{1}$$

where f is the *activation function* of the neuron. This function provides flexibility to neural networks, allowing to estimate complex nonlinear relations in the data and providing a normalization effect on the neuron output (e.g., bounding the resulting value between 0 and 1). The most widely used activation functions are *sigmoid*, *tanh* (hyperbolic tangent), and *ReLU* (Rectified Linear Unit). Each neuron is connected to many others and the links between them can increment or inhibit the activation state of the adjacent neurons.

Figure 4 shows the basic structure of a neural network. The first layer is the *input layer*, which is used to provide input data or features to the network. The *output layer* provides the predictions of the model. Depending on the problem, the activation function used in this layer differs: for binary classification, where output values are either 0 or 1, the sigmoid function is used; for multiclass classification, *softmax* (a generalization of sigmoid to multiple classes) apply; for a regression problem where there are no predefined categories, a linear function can be used.

The ReLU activation function is commonly used in hidden layers. The hidden layers can compute complex functions by cascading simpler functions. The type of hidden layers defines the different neural network architectures, such as CNN, RNN, or LSTM (see Section 5.3). The number of hidden layers determines the depth of the network. In general, networks with more hidden layers can learn more complex functions. In DL architectures, usually dozens or even hundreds of hidden layers are used, which can automatically learn as the model is trained with data.

5.2. Training Process. Any machine learning algorithm tries to assign inputs (e.g., an image) to target outputs (e.g., the "cat" label) by observing many input and output examples. As mentioned before, DL does this mapping between inputs and objective outputs (i.e., what the network is expected to produce) using artificial neural networks composed of a large number of layers forming a hierarchy.

The network learns something simple in the initial layer of the hierarchy and then sends this information to the next layer. This layer then takes this simple information, combines it with something more complex, and sends it to a third layer. This process continues, each layer building something more complex from the input received from the previous layer. The specification of what each layer is doing to the input received is stored in the weights of the layer. In order for the network to learn, it is necessary to find the weights of each layer that provides the best mapping between the input examples and the corresponding objective outputs.

Training the neural network means finding the right parameters setting (weights) for each processing unit in the network. The problem is that DL networks may potentially have millions of these parameters and finding the correct values for all of them can be a really difficult task. For example, VGG16 [59], a popular neural network architecture applied to image classification, has 138 million parameters. Initially, the weights of each neuron can be assigned randomly, or follow some initialization strategy, including unsupervised pretraining [60].

In order to control the quality of the output of the neural network, it is necessary to measure how close is the obtained output from the expected output. This task is carried out by the *loss function* of the network. This function takes the predictions of the model and the objective values and calculates how far the predicted outputs are from the objective values. The result of this function indicates how well is working the model for the specified examples. A common loss function is the *Mean Squared Error* (MSE), which measures the average of squared errors made by the neural network over all the input instances.

The goal of the training process is to find the weights that minimize the loss function. The error calculated by this function is fed back through the network, usually by means of *backpropagation*. This information is used to adjust the weights of each connection in the network in order to reduce the error. This process can be carried out by applying a general method for nonlinear optimization called *gradient descent*, in which the network computes the derivative of the loss function with respect to the weights, changing them such that the error decreases. The amount by which the weights are changed is determined by a parameter called *learning rate* (see Section 5.4).

After a number of training cycles (known as *epochs*) repeating this process, the model will usually converge to a state where the error is small and the network is considered to have learned the target function.

5.3. Architectures. Depending on the type of input (images, text, audio, etc.) there are different neural network architectures that are better suited to process that information. The number of architectures and algorithms that are used in DL is wide and varied. In this section, the most popular architectures, their common tasks, and their use in EDM will be described. Table 3 summarizes the works in EDM studied in this article (first column), the architectures implemented (second column), the baseline methods employed (third column), the evaluation measures used to compare DL approaches and baseline methods (fourth), and the performance achieved by DL methods in that comparison (fifth).

The architectures include MLP (Multilayer Perceptron), LSTM (Long Short-Term Memory), WE (Word Embeddings), CNN (Convolutional Neural Networks) and variants (VGG16 and AlexNet), FNN (Feedforward Neural Networks), RNN (Recurrent Neural Networks), autoencoder, BLSTM (Bidirectional LSTM), and MN (Memory Networks).

The baseline methods are SVD (Singular Value Decomposition), Slope One, K-NN (K-Nearest Neighbors), Majority class, RF (Random Forest), SVM (Support Vector Machine), N-grams, Random guess, LinReg (Linear Regression), DT (Decision Tree), NB (Naïve Bayes), LogReg (Logistic Regression), HMM (Hidden Markov Model), IOHMM (Input Output HMM), BKT (Bayesian Knowledge Tracing), IBKT (Intervention BKT), PFA (Principal Factor Analysis), Majority voting, CRF (Computational Random Fields), LSA (Latent Semantic Analysis), LDA (Latent Dirichlet Allocation), SVR (Support Vector Regression), BLRR (Bayesian Linear Ridge Regression), AdaBoost, GTB (Gradient Tree Boosting), GNB (Gaussian Naïve Bayes), IRT (Item Response Theory), TIRT (Temporal IRT), and HIRT (Hierarchical IRT).

Finally, evaluation measures include MAE (Mean Absolute Error), RMSE (Root Mean Square Error), Accuracy, Precision, Recall, F-measure, AUC (Area Under the Curve), Krippendorff's alpha, Log Loss (Logarithmic Loss), R^2, Gini, MPCE (Mean per Class Error), and QWK (Quadratic Weighted Kappa). The last column of this table indicates whether, in the experiments carried out in the paper, the DL approach outperformed baseline methods (">"), underperformed ("<"), or obtained similar results, with higher performance in some of the evaluations and lower performance in others ("="). The symbol "–" represents approaches that do not compare DL with traditional machine learning techniques. Instead, they present comparisons of different DL architectures [19, 29, 35, 45, 50], comparisons of different hyperparameters for the same DL architecture [31, 46], or proposals not evaluated yet [39].

5.3.1. Feedforward Neural Networks. FNNs represent the first generation of neural networks. Nodes in these networks do not form cycles, i.e., the information propagates always forward in a single direction, from the input nodes to the output nodes [61]. The main representatives of this type of networks are *perceptron* and *Multilayer Perceptron* (MLP).

Perceptrons are the simplest kind of neural network [62]. They consist of a single layer of output nodes, where inputs are sent directly to the output via a series of weights. Each node calculates the sum of the products of the weights and the inputs. If the result is above a threshold, the neuron activates; otherwise it takes the deactivated value. Single-layer

Table 3: Deep Learning approaches in the EDM field: architectures employed, baseline methods, and evaluation measures. The column Performance indicates whether the approaches outperformed baseline methods (>), underperformed (<), or obtained similar results (=).

Reference	Architecture	Baseline	Evaluation	Performance
Abhinav et al., 2018 [38]	MLP	SVD, Slope One, K-NN	MAE, RMSE	>
Akram et al., 2018 [44]	LSTM	Majority class, RF, SVM	Accuracy, Precision, Recall, F1	>
Alam et al., 2018 [25]	MLP	DT, RF, SVM, KNN	Accuracy	>
Alvarado et al., 2018 [43]	WE	N-grams	Precision, Recall, F-measure	=
Aung et al., 2018 [36]	CNN, VGG16, AlexNet	Random guess, LinReg	AUC	>
Bendangnuksung and P., 2018 [20]	FNN	DT, NB, MLP	Accuracy	>
Choi et al., 2017 [45]	WE	-	Krippendorff's alpha	–
Fei and Yeung, 2015 [28]	RNN, LSTM	SVM, LogReg, IOHMM	AUC	>
Guo et al., 2015 [23]	Autoencoder	NB, SVM, MLP	Accuracy	>
Khajah et al., 2016 [17]	LSTM	BKT	AUC	=
Kim et al., 2018 [27]	BLSTM	LogReg	AUC	>
Kim et al., 2018 [26]	BLSTM	LogReg	AUC	>
Lalwani and Agrawal, 2017 [14]	LSTM	PFA, BKT	AUC	=
Lin and Chi, 2017 [11]	RNN, LSTM	Majority voting, BKT	Accuracy, Precision, Recall, F-measure	>
Mao et al., 2018 [15]	LSTM	BKT, IBKT	RMSE, Accuracy, Recall, F-measure, AUC	=
Min et al., 2016 [33]	LSTM	CRF	Accuracy	=
Montero et al., 2018 [13]	LSTM	BKT	AUC	>
Okubo et al., 2017 [24]	LSTM	LinReg	Accuracy	>
Piech et al., 2015 [10]	RNN, LSTM	BKT	AUC	>
Sales et al., 2018 [46]	LSTM	-	-	–
Sharada et al., 2018 [22]	MLP	RF, LogReg	Log Loss, RMSE, R^2, AUC, Gini, MPCE	<
Sharma et al., 2016 [34]	CNN, AlexNet, VGG16, LSTM	SVM, HMM	Accuracy	>
Taghipour and Ng, 2016 [41]	LSTM	SVR, BLRR	QWK	>
Tang et al., 2016 [21]	LSTM	Majority class	Accuracy	=
Tato et al., 2017 [37]	CNN	SVM, NB, LSA, LDA, MLP	Accuracy, F-measure	>
Teruel and Alemany, 2018 [29]	LSTM	LSTM	AUC, RMSE, R^2	–
Wang et al., 2017 [30]	CNN, RNN	SVM, LogReg, DT, AdaBoost, GTB, RF, GNB	Precision, Recall, F-measure, AUC	=
Wang et al., 2017 [12]	LSTM	LogReg	Accuracy	>
Wang et al., 2017 [53]	LSTM	LogReg	Recall, Precission, F-measure	=
Whitehill et al., 2017 [31]	FNN	-	AUC	–
Wilson et al., 2016 [50]	RNN	IRT, TIRT, HIRT	Accuracy, AUC	<

Table 3: Continued.

Reference	Architecture	Baseline	Evaluation	Performance
Wilson et al., 2016 [16]	LSTM	-	-	–
Wong, 2018 [39]	LSTM	-	-	–
Xiong et al., 2016 [18]	LSTM	PFA, BKT	AUC, R^2	>
Xing and Du, 2018 [32]	RNN	DT, KNN, SVM	Accuracy, AUC	>
Yang et al., 2018 [35]	LSTM	-	MSE	–
Yeung and Yeung, 2018 [19]	LSTM	-	AUC	–
Zhang et al., 2016 [42]	DBN	NB, LogReg, DT, Perceptron, SVM	Accuracy, AUC, Precission, Recall, F-measure	>
Zhang et al., 2017 [49]	LSTM, Autoencoder	LSTM	AUC, R^2	>
Zhao et al., 2017 [40]	MN	FNN, SVR, BLRR, LSTM	QWK	>

perceptrons are only capable of learning linearly separable patterns. Networks without hidden layers are quiet limited in the patterns they can learn, and introducing more layers of linear units does not overcome this limitation. It is therefore necessary to introduce multiple layers of nonlinear hidden units. MLP consists of multiple layers of neurons, where each neuron in one layer has directed connections to the neurons of the following layer. In many applications, the sigmoid function is used as the activation function in these neurons.

FNNs are applicable to many areas where classical machine learning techniques have been applied, although major success have been achieved in computer vision [63] and speech recognition applications [64]. FNNs are primarily used for supervised learning tasks where the input data is neither sequential nor time-dependent, offering good results when the number of layers, neurons and training data is large enough. One of the main problems of this architecture is the possibility of ending up in a local minima of the loss function, getting a suboptimal solution to the problem at hand.

In the area of EDM, FNNs have been used for predicting students performance [20, 22] and for recommending learning opportunities to students based on their preferences [38].

Another type of FNN is *autoencoders* [65]. This architecture is similar to MLP, but in this case the output layer has the same number of neurons as the input layer. The goal is to reconstruct its own inputs instead of predicting a target value. This is an example of unsupervised learning, since no labeled data is required. Autoencoders (and its variants *stacked*, *sparse* and *denoising*) are typically used to learn compact representations of data [66]. Another application of this architecture is pretraining a deep network: a stacked autoencoder is trained in an unsupervised way and weights are obtained. Then this weight can be used for the deep network (with the same configuration in terms of hidden layers, number of neuros per layer, etc.) as a better choice rather than using randomly initialized weights [67]. Focusing in EDM, the work by [23] used a sparse autoencoder in the task of predicting students performance. They pretrained hidden layers of features using an unsupervised sparse autoencoder from unlabeled data, and then used supervised training to fine-tune the parameters of the network.

5.3.2. Convolutional Neural Networks. CNNs are multilayer neural networks particularly useful in image-processing applications [68]. In this architecture, the first layers recognize simple features in images (e.g., edges) and the last layers combine these initial features into higher-level abstractions (e.g., recognizing faces). CNNs are similar to FNNs in different aspects: they are composed of neurons where bias and weights have to be learned, each neuron has some inputs, performs a dot product, and applies an activation function, and there is a loss function in the last (fully connected) layer that measures the difference between the predicted and the expected value.

In general, a CNN is formed by an structure that contains three different types of layers: a convolutional layer that extracts features from the input (usually an image); a reduction (*pooling*) layer, which reduces the dimensionality of the extracted features through down-sampling while retaining the most important information (usually *max pooling* is applied [69]); and a fully connected classification layer, which provides the final result at the end of the network. The use of deep layers of convolution, pooling and classification, has facilitated the emergence of new applications of CNN. In addition to image processing [70], this type of networks has been applied to video recognition [71], game playing [72], and different natural language processing tasks [73].

The main advantage of CNNs is their accuracy in pattern recognition tasks, such as image recognition, requiring considerably fewer parameters than FNNs. On the negative side, they have disadvantages such as the high computation cost, the need for large amounts of training data, and the work required to properly initialize the network according to the problem addressed.

In the field of EDM, CNNs have been used in detecting undesirable student behaviors using VGG16 [59] and AlexNet [70] architectures for video analysis [36], using also VGG16 and AlexNet architectures for audio and video analysis [34],

performing text classification [37], and predicting student dropout [30].

5.3.3. Recurrent Neural Networks. A distinctive feature of FNNs is that they do not provide persistence mechanisms. RNNs address this problem by implementing a feedback loop that allows for information to persist [74]. Instead of completely feedforward connections, RNNs may have connections that feed back previous or the same layer. This feedback allows RNNs to keep a memory of past inputs.

RNNs can be thought as networks with multiple copies of themselves, in which each one passes a message to its successor. This structure makes them convenient for dealing with sequences and lists, and thus one of their common uses is modeling text. RNNs have been successfully applied to a variety of problems such as speech recognition [75], language modeling [76], and machine translation [77]. One of the main disadvantages of RNNs is the issue of vanishing gradients, where the magnitude of the gradients (values used to update the neural network weights) gets exponentially smaller (vanish) as the network back propagates, resulting in a very slow learning of the weights in the lower layers of the RNN. This makes the training process difficult in several ways: this architecture cannot be stacked into very deep models and cannot keep track of long-term dependencies. Another issue of RNNs is that they require a high performance hardware to train and run the models.

In the context of EDM, this type of networks has been used in the task of anticipate students dropout [28, 30, 32], and in the task of predicting students performance for learning gain predictions [11] and proficiency estimation [50].

There are different RNN architectures (see LSTM in the next section). The key difference is the feedback mechanisms within the network, which can manifest in a hidden layer, in the output layer or in a combination of them. RNNs can be trained with standard backpropagation or by using a variant called backpropagation through time (BPTT) [78].

5.3.4. Long Short-Term Memory Networks. LSTMs are a special type of RNN that has grown in popularity in recent years [79]. This architecture introduces the concept of memory cell, which allows to learn dependencies in the long term. The memory cell retains its value for a period of time as a function of its inputs and contains three gates that control information flow into and out of the cell: the *input gate* defines when new information can flow into the memory; the *forget gate* controls when the information stored is forgotten, allowing the cell to store new data; the *output gate* decides when the information stored in the cell is used in the output.

Each gate in the memory cell is also controlled by weights. The training algorithm (e.g., BPTT) optimizes these weights based on the resulting network output error. Recently, a simplification of LSTM called *Gated Recurrent Unit* (GRU) has been introduced [80]. This recurrent unit has fewer parameters than LSTMs, since it has two gates instead of three, lacking an output gate.

As a type of recurrent network, LSTMs are especially suitable for problems dealing with sequences. Several tasks can be added to the list of tasks previously mentioned for RNNs: text generation [81], question answering [82] and action recognition in video sequences [83], among others. In conjunction with CNNs, LSTMs have been used to produce image [84] and video [85] captioning: the CNN implements the image/video processing whereas the LSTM converts CNN output into natural language. One of the main advantages of LSTMs, compared to RNNs, is the extension of the memory that allows this architecture to remember their inputs over a long period of time. Unlike LSTMs, a RNN may leave out important information from the beginning while trying to process a paragraph of text to do predictions. LSTMs also overcome the issue of the vanishing gradient described above for RNNs. Finally, compared to this architecture, LSTMs reduce the amount of training data required to build the models.

In the set of works studied in this article, LSTM has been the most widely used architecture. In fact, it has been applied to all the EDM tasks covered by DL approaches: predicting students performance [21, 24, 53]; detecting undesirable student behaviors by predicting students dropout [28], predicting dialogue acts [33], modeling student behavior in learning platforms [29], and predicting engagement intensity [35]; generating recommendations [39]; and evaluation by doing stealth assessment [44], improving casual estimates from A/B tests [46], and automating essay scoring [41].

As already mentioned in Section 4.1.1, there is a controversy between a set of studies, falling in the task of predicting students performance, which have focused on knowledge tracing, i.e., modeling the knowledge of students as they interact with coursework. The controversy arose after the publication of *Deep Knowledge Tracing* (DKT) [10], an LSTM-based model which significantly outperformed previous approaches that used BKT and PFA. A series of works were published afterwards that were for [11–13, 19] or against [14–18] the claims in this paper. All these studies used the LSTM implementation of DKT, although some of them introduced their own variants.

5.3.5. Other Architectures. Apart from the architectures already described, other network structures have been employed in the literature reviewed on DL applied to EDM. One of these architectures is *Deep Belief Networks* (DBN), used in the task of evaluation [42]. This type of neural network has been used for image recognition, information retrieval and natural language understanding, among other tasks. The DBN is a multilayer network where each pair of connected layers is a *Restricted Boltzmann Machine* (RBM) [86]. Training in DBN occurs in two steps: unsupervised pretraining and subsequent supervised fine-tuning. In the unsupervised phase, each RBM is trained to reconstruct its input using the previous hidden layer output [87].

Memory networks (MN) have also been used in the task of evaluation [40]. MN are a new class of models designed to address the problem of learning long-term dependencies in sequential data, including a long-term memory component that can be read and written to provide an explicit memory representation for each token in the sequence [88].

Finally, *bidirectional LSTM* (BLSTM) is employed in the work developed for the task of predicting student performance [26, 27]. The difference with conventional LSTMs is that these networks only preserve information from the past, whereas BLSTMs run inputs in two ways: one from past to future and other from future to past, preserving information from the future in the backward run [89].

5.4. Hyperparameters Tuning. DL models include *hyperparameters*, which are variables set before optimizing the parameters (weights and bias) of the models. Hyperparameters can be set by hand, selected by a search algorithm (e.g., grid search or random search), or optimized applying a model-based method [90].

This section describes hyperparameters typically found when building neural networks. They have been classified in two types: those related to the training process and those related to the model itself. Although not all the studies analyzed in this article provide details about the hyperparameters used, references are provided when available.

5.4.1. Training. The hyperparameters described here that affect the training process are learning rate, batch size, momentum, weight update, and stopping criteria.

Learning Rate. The *learning rate* controls how much the weights of the network are adjusted with respect to the loss gradient. The lower the value is, the slower the algorithm traverses the downward slope. This helps to avoid missing local minima, but on the downside it takes a long time to converge and arrive at the best accuracy of the model.

The learning rate employed in the works studied ranges from a minimum of 0.0001 [34, 36] to a maximum of 0.1 [31], with other values such as 0.00025 [23] and 0.01 [19, 29, 35, 41].

Batch Size. The *batch size* defines the number of training instances that are propagated through the neural network. For instance, a set of 1000 training samples could be split in 10 batches of 100 samples. Using a batch size lower than the number of all samples has some benefits, such as requiring less memory (the network is trained using fewer samples in each propagation) and training faster (weights are updated after each propagation). The disadvantage of using a batch instead of all samples is that the smaller the batch size, the less accurate the estimate of the gradient.

Batch sizes used in the works reviewed include 10 [31, 38], 32 [19, 27, 33, 41], 48 [25], 100 [10, 11, 18], 500 [37], and 512 [23].

Momentum. *Momentum* is a popular extension of backpropagation that helps to prevent the network from falling into local minima. This technique adds a fraction of the previous weight update to the current weight. When the gradient keeps pointing in the same direction, this increases the size of the steps taken towards the minimum. When the gradient keeps changing direction, momentum will smooth out the variations.

Only three papers in EDM explicitly stated the use of momentum, all of them with a value of 0.9 [23, 35, 36].

Weight Update. DL models usually employ *stochastic gradient descent* (SGD) in the training phase. Although this is an easy to implement approach, it is difficult to tune and parallelize, making it challenging to debug and scale up DL networks. There are more sophisticated optimization methods such as *limited memory Broyden–Fletcher–Goldfarb–Shanno* (L-BFGS) and *conjugate gradient* (CG) that can speed up the process of training DL algorithms [91].

Most of the papers reviewed used SGD in the training phase [10, 18–20, 22, 27, 31–33, 36, 40, 41, 49, 50]. Other works used *Adam* [25, 38], an efficient gradient descent algorithm [92]. Finally, as an alternative to backpropagation in the training process, some studies used BPTT to train RNNs [28, 29, 34].

Stopping Criteria. There are different ways to determine the number of epochs employed to train the algorithms. If training and validation errors are high, the system is probably underfitting (it can neither model the training data nor generalize to new data), and the number of epochs can be increased. *Early stopping* is a form or regularization used to avoid overfitting. It updates the network so as to make it better fit the training data with each iteration, improving also the model performance on the validation dataset. At a certain point, improving the model fit to the training data increases generalization errors. Early stopping rules provide a guide to identify how many iterations can be run before overfitting.

Most of the works studied established a fixed number of epochs to train the algorithms: 22 [22], 50 [20, 38, 41, 49], 60 [35], 100 [11], 150 [21], and 250 [37]. In [25] the authors employed 50,000 epochs, but considering a very limited number of input features. Reference [36] used a validation set for early stopping, whereas [33] defined a strategy consisting in stopping the training if there is no improvement in the last 15 epochs (with a maximum of 100 epochs).

5.4.2. Model. The hyperparameters listed here, related to the model architecture, are depth and width of the network, initial weights, and dropout.

Depth and Width. These hyperparameters refer to the number of hidden layers (depth) and the number of hidden units (width) in the network. There is no analytical approach to setting these two parameters and choosing the best configuration for a task is sometimes a matter of trial and error. Whereas shallow neural networks (with a single hidden layer) can in theory approximate any function (according to the *universal approximation theorem* [93]) many empirical results in different tasks and domains demonstrate that adding more hidden layers improves the performance of the network. A possible explanation for this phenomenon is that the number of units in a shallow network grows exponentially with task complexity, requiring much more neurons than a deep network to achieve the same performance [2].

Since these are two key elements of a network architecture, most of the papers reviewed provide information about the depth and width of their implementation. Regarding the number of layers, most of the implementation ranges from

1 to 6 layers: 1 hidden layer [10, 13, 14, 17–19, 24, 32, 49, 50, 53], 2 hidden layers [11, 15, 20, 21, 34, 44], 3 hidden layers [22], 4 hidden layers [23, 26, 27, 37, 40, 41], 5 hidden layers [25, 31], and 6 hidden layers [30, 38]. In [35] the authors set 2 hidden layers for each modality feature (e.g., eye gaze and head pose), adding up to 8 hidden layers. The work by [36] defined 16 (since it employs the VGG16 architecture). Reference [33] implemented an LSTM with 64 layers (obtaining better results than with 32 layers). Finally, [29] experimented with different configurations of layers: 20, 50, 100, and 200.

With respect to the number of units per hidden layer, the most common value in the papers reviewed is 200 [10, 11, 14, 15, 17–19, 49], followed by 100 [22, 40, 50], 64 [33, 35], 128 [21, 27], and 256 [26, 34]. Other configurations include 5 [31], 15 [44], 20 [28], 40 [37], and 300 [20]. Some works tested different ranges of width values in their implementation: 10 to 200 [13], 50 to 300 [41], and 64 to 512 [36].

Initial Weights. The initial values assigned to the weights of the network play an important role in finding the global minima of the cost function in a deep neural network [94]. One way to do this initialization is assigning random values, although this method can potentially lead to two issues: vanishing gradient (the weight update is minor and the optimization of the loss function is slow) and exploding gradient (oscillating around the minima). There are more sophisticated approaches such as using unsupervised stacked RBMs to choose these weights.

The most common initialization procedure in the papers reviewed is to randomly select the initial weights: Gaussian distribution with zero mean and small variance [19], uniform weights in the range $[-0.1, 0.1]$ [20, 28, 44], and uniform weights in the range $[-0.05, 0.05]$ [13]. A sparse autoencoder was used for pretraining in [23]. Transfer learning [95] was used in [36] to initialize CNNs with weights pretrained on ImageNet. Finally, [31] used *Net2Net*, a technique to accelerate transfer learning from a previous network to a new one [96].

Dropout. *Dropout* is a regularization technique used in neural networks to prevent overfitting. The core of this approach is to randomly select neurons that will be ignored ("dropped out") during the training process. Their contribution to the activation of neurons in the next layer is temporally removed on the forward pass and weight updates are not applied to the neuron on the backward pass [97]. As neurons are randomly dropped out during training, other neurons have to handle the representation required to make predictions for the missing units. The result is that the neural network is less sensitive to specific weights of neurons achieving better generalization.

Some of the works studied reported dropout values in their network configurations. The most repeated values are 0.2 [11, 27, 34] and 0.5 [19, 23, 41], followed by 0.3 [29, 36]. Other values reported are 0.25 [50], 0.4 [49], 0.6 [13], and 0.7 [33]. There are three works that used dropout in their networks but did not reported the specific value of this hyperparameter [10, 18, 38].

5.5. Frameworks. Nowadays there are a large set of frameworks available for fast prototyping DL models that can efficiently run in parallel taking advantage of GPU infrastructures. In this way, researchers can focus on the architecture of the model and overlook low-level details. This section introduces the frameworks used in the DL for EDM literature, including some additional popular frameworks that have not yet been used in this domain. Note that not all the papers reviewed provide implementation details.

Keras (https://keras.io/) is the most popular framework in the articles reviewed. It was used in their implementation by [11, 14, 17, 25, 31, 38, 41, 44]. Keras provides a Python interface to facilitate the rapid prototyping of different deep neural networks, such as CNNs and RNNs, which can be executed on top of other more complex frameworks such as TensorFlow and Theano (see below). The code produced using Keras runs seamlessly on both CPUs and GPUs.

TensorFlow (https://www.tensorflow.org/) is the second most popular framework in this list. It is available for both desktop and mobile applications, and supports developing DL models using languages such as Python, C++ and R. The framework includes TensorBoard, a tool to visualize data modeling and network performance. It is supported by Google and by a large community of developers that provide numerous documentation, tutorials and guides. The works in EDM using this framework are [13, 18–20, 25, 29, 49].

Third in the list is Theano (http://deeplearning.net/software/theano/). It was the most widely used library for DL before the arrival of other competitors such as Tensorflow, Caffe, and PyTorch. It is a low-level library supporting both CPU and GPU computation. After launching the release of version 1.0, it was announced that the development and support for this tool would be discontinued. The works by [23, 30, 50] used this framework.

Caffe (http://caffe.berkeleyvision.org/) is a library written in C++ that includes a Python interface. It is specialized in the development of CNNs for image-processing tasks. One of the biggest benefits of using this library is the ability to access out-of-the-box pretrained networks from the Caffe Model Zoo (http://caffe.berkeleyvision.org/model_zoo.html). It was used by [36] for automatic eye gaze following in the classroom.

Torch (http://torch.ch/) is a relatively old machine learning library, since it was first released fifteen years ago. The primary programming language is Lua, although there is an implementation in C. It contains both DL and other traditional machine learning algorithms, supporting CUDA for parallel computation. It was used by [10, 12] to develop their DL models using Lua for the task of knowledge tracing. There is an open-source machine learning library for Python based on Torch, called PyTorch (https://pytorch.org/), which has gained increasing attention from the DL community since its release in 2016. This library was used in the work by [35].

Other relevant frameworks for DL, not used in any of the presented works, are Caffe2 (https://caffe2.ai/), Deeplearning4j (https://deeplearning4j.org/), MXNet (urlhttps://mxnet.apache.org/), Microsoft Cognitive Toolkit (https://www.microsoft.com/en-us/cognitive-toolkit/), and Chainer (https://chainer.org/).

Some works described in this article use *word embeddings* to reduce the dimensionality of the input space. Word embeddings are used in the area of natural language processing to map words (or phrases) to vectors of real numbers. This mapping can be done using neural network approaches [98]. They aim to identify semantic similarities between words based on their cooccurrence with other words in large samples of texts. The frameworks chosen for this task in the EDM field are word2vec [29, 45] and Glove (https://nlp.stanford.edu/projects/glove/) [40, 43]. Other popular frameworks to work with word embeddings are fastText (https://fasttext.cc/), although none of the works described here used it in their implementation.

6. Discussion

The first question to analyze in this section is the current status of EDM tasks with respect to the use of DL models. Based on the taxonomy of EDM applications defined by [8], the papers reviewed on the present study were categorized according to the problem addressed. This classification revealed that only 4 of the 13 tasks defined in that taxonomy have been faced using DL approaches: predicting students performance, detecting undesirable student behaviors, generating recommendations, and automatic evaluation. The other 9 tasks remain as an opportunity for researchers in the field to explore the application of DL techniques.

In the task of predicting student performance, a large sample of the papers analyzed were devoted to compare the performance of BKT (probabilistic) and DKT (deep learning) models, resulting in an interesting discussion between traditional and deep learning approaches (see Section 5.3.4). While DKT usually obtained better performance, BKT offered better interpretation of its predictions. Since DL is a very active research topic, it is expected that advances in DL will provide in a future theoretical understanding and interpretability of the models generated, and these findings will benefit all the fields where DL is applied, including EDM.

The prediction of dropping out in MOOC platforms is the subtask that has gained more attention in detecting undesirable student behaviors. Most of these studies focused on predicting student's dropout at a given point in time. These studies performed video analysis to identify the loss of interest in the contents of the course, extracting features such as the student's gaze. Including multimodal features to train DL models, such as behavioral traits (e.g., asking for help in the classroom or cheating in tests), could benefit future approaches to this task.

The third task studied, generating recommendations, was the target of two papers that focused on generating personalized searches based on the students' preferences and curriculum planning. An open challenge for future research is the recommendation of learning resources in an informal setting. The problem in this case would be the impossibility to manually structure the large amount of data that comes from sources such as expert communities and educational blogs.

Finally, in the evaluation task different frameworks were built to help teachers in the grading process, primarily focused on automatic essay scoring and short answer grading. The use of game-based environments and A/B testing has demonstrated its benefits as an automatic evaluation tools, and either would be an interesting line of research for future works.

The second relevant aspect of this work is the study of existing datasets used by DL models in educational contexts. As shown in Section 4.2, several datasets have been developed for predicting student performance and student behaviors in online platforms. Although only some of them are available (e.g., ASSISTment and KDD cup 2010 for predicting student performance, and KDD Cup 2015 for predicting student dropout), there are many online learning platforms, ITSs and MOOCs that can provide large amounts of information to train EDM systems.

Based on the literature reviewed, it seems necessary to develop specific datasets for two tasks: educational recommender systems based on data mining and automatic essay scoring. The main problem for the first task is that there is not a single "correct" sequence of learning items to recommend to a student, and this recommendation largely depends on the background knowledge, abilities, and goals of the learner. For this reason, it is necessary not only datasets with coherent sequences of learning (such as the sequences that can be found in a MOOC), but also to know which sequences are appropriate for each student profile. The second task, automatic essay scoring, is a hard challenge that requires a deep linguistic analysis to achieve automatic evaluations of texts. Although there are successful machine learning based natural language processing tools, automatic essay scoring requires a fine and deep semantic analysis in order to identify the topic of the essay, the main idea, arguments for and against, and, in general, the reasoning process carried out by the student. Unfortunately, there is no dataset available today that comprises this type of complex linguistic information that would benefit DL approaches in this task.

Finally, the last point studied in this review is the different DL models and configurations used in the EDM literature. Regarding DL architectures, LSTMs have been the most used approach, both in terms of frequency of use (59% of the papers used it) and variety of tasks covered, since it was applied in the four EDM tasks addressed by the works analyzed. In principle, this could be considered a good starting point to develop a system in any of the tasks covered. In the case of other architectures, Vanilla RNNs were used for predicting students performance and detecting undesirable student behaviors, FNNs were constrained to predicting students performance and CNNs to detect undesirable student behaviors. Other proposals considered the use of MLP, DBN, MN, and autoencoders.

The main hyperparameters of DL models were also reviewed in the previous section. Given the empirical nature of the development process of DL models, there is no one-size-fits-all solution to set the best configuration for a specific architecture, and the hyperparameters chosen will depend on the input data available and the task at hand. Among those analyzed, learning rate, batch size, and the stopping criteria (number of epochs) are considered to be critical to model performance. In theory, larger batch sizes imply

more stable gradients, facilitating higher learning rates. A larger batch sizes is also more computationally efficient, as the number of samples processed in each iteration increases. Nevertheless, a general advice with deep neural networks is to take many small steps (smaller batch sizes and learning rates) instead of fewer larger ones, although this is a design trade-off that requires experimentation. The third hyperparameter mentioned, the number of epochs, must also be properly adjusted to avoid the problem of overfitting. Another aspect to take into account is the size of the network. Adding more layers (depth) and neurons (width) can lead to more powerful models, but these architectures are also easier to overfit. A model with a large number of parameters requires also a large number of samples to achieve generalization. In this respect, more training data means almost always better DL models.

Manually choosing these hyperparameters is time-consuming and error-prone. As the models change, previous choices may no longer be the best ones. To avoid this drawback, there are a number of techniques to automatically pick the best hyperparameters (such as grid search). The summary provided in Section 5.4 can give a hint of the starting point and suitable ranges of values for these hyperparameters in the development of new architectures. In this regard, the most commonly used configuration values were: 0.0001 and 0.01 learning rate; 32 and 100 batch size; 0.9 momentum; SGD weighting update; 50 epochs stopping criteria; 1 or 2 hidden layers depth; 100 or 200 hidden units per layer width; random weight initialization; and 0.2 dropout.

It should be noted that the limited number of hidden layers in most of these works, with 79% of the implementation using 5 or less hidden layers. Indeed, according to [56], 54% of the works reviewed could be considered "shallow" neural networks, since they only include 1 or 2 hidden layers in their architectures. This suggests that there is room for applying more complex and deep architectures in the field of EDM. Popular techniques and architectures, such as transfer learning, reinforcement learning, Generative Adversarial Networks, and unified frameworks, are almost unexplored in the EDM domain.

With respect to the performance of DL techniques in these works, leaving aside the papers that do not offer a comparison between DL and traditional machine learning techniques, 67% of the works reported that DL outperformed the existing baselines, 27% showed inconclusive results (DL performed better only in some of the experiments), and only 6% reported a lower performance of DL techniques. These figures are not surprising given the successful results of DL techniques in many different domains. Nevertheless, these results are not exempt from controversy. Out of the field of EDM, there are detractors who claim that the inner mechanisms of the DL models generated are so complex that researchers often cannot explain why a model produces a particular output from a set of inputs. This controversy has also arisen in EDM, with the aforementioned arguments for and against DKT and BKT.

Taking into account the current DL techniques applied to EDM, there are many open paths to explore new approaches to this field, such as the use or transfer learning for initialization of the neural networks (only used in [36]), the use of reinforcement learning [99], a promising learning technique that reduces the need for training data, and the application of architectures such as MN, DBN, and *generative adversarial networks* (GAN), in tasks where language or image generation are required [100].

7. Conclusions

This study has reviewed the emergence of DL applications to EDM, a trend that started in 2015 with 3 papers published, increasing its presence every year so far with 17 papers published in 2018. After a systematic search, 41 works were retrieved in this area. It is worth mentioning the presence of these approaches in relevant EDM forums such as the annual International Conference in Educational Data Mining, with 7 papers published in the last edition (for a total of 16 in the last three years).

Based on the taxonomy of EDM applications defined by [8], only 4 of the 13 tasks proposed in that study have been addressed by DL techniques. This reveals that there are many open opportunities for the use of DL in unexplored EDM tasks, moreover taking into account the promising results obtained by these models in the works reviewed (67% of them reported that DL outperformed the "traditional" machine learning baselines in all their experiments).

The study carried out also included a revision of the main datasets used in the EDM tasks covered. As in other research areas, some of them are publicly available for the scientific community, which allows for reproducibility of the experiments, whereas others were developed *ad hoc* for specific studies. In the EDM field, an additional problem that exists to make the datasets freely available is the existence of sensitive information concerning (underage) students. This problem could be overcome with proper anonymization of the data.

A thorough study of DL techniques were also provided in this work, starting with an introduction to the field, an analysis of the types of DL architectures used in every task, a review of the most common hyperparameter configurations, and a list of the existing frameworks to help in the development of DL models. Since defining a DL architecture relays mostly in an empirical process, the information provided in this study can serve as a basis for starting future developments of DL applications in EDM.

Given the increasing adoption of DL techniques in EDM, this work can provide a valuable reference and a starting point for researches in both DL and EDM fields that want to leverage the potential of these techniques in the educational domain.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] C. Romero, S. Ventura, M. Pechenizkiy, and R. Baker, *Handbook of educational data mining*, CRC Press, 2010.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, Mass, USA, 2016.

[3] C. Romero and S. Ventura, "Educational data mining: a survey from 1995 to 2005," *Expert Systems with Applications*, vol. 33, no. 1, pp. 135–146, 2007.

[4] C. Romero and S. Ventura, "Educational data mining: A review of the state of the art," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, no. 6, pp. 601–618, 2010.

[5] C. Romero and S. Ventura, "Data mining in education," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 3, no. 1, pp. 12–27, 2013.

[6] R. S. Baker and Y. Yacef, "The state of educational data mining in 2009: A review and future visions," *JEDM-Journal of Educational Data Mining*, vol. 1, no. 1, pp. 3–17, 2009.

[7] A. Peña-Ayala, "Educational data mining: A survey and a data mining-based analysis of recent works," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1432–1462, 2014.

[8] B. Bakhshinategh, O. R. Zaiane, S. ElAtia, and D. Ipperciel, "Educational data mining applications and tasks: A survey of the last 10 years," *Education and Information Technologies*, vol. 23, no. 1, pp. 537–553, 2018.

[9] H. Aldowah, H. Al-Samarraie, and W. M. Fauzy, "Educational data mining and learning analytics for 21st century higher education: A review and synthesis," *Telematics and Informatics*, vol. 37, pp. 13–49, 2019.

[10] C. Piech, J. Bassen, J. Huang et al., "Deep knowledge tracing," in *Annual Conference on Neural Information Processing Systems (NIPS)*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., pp. 505–513, Curran Associates, Inc., 2015.

[11] C. Lin and M. Chi, "A comparisons of bkt, rnn and lstm for learning gain prediction," in *Artificial Intelligence in Education*, vol. 10331 of *Lecture Notes in Computer Science*, pp. 536–539, Springer International Publishing, 2017.

[12] L. Wang, A. Sy, L. Liu, and C. Piech, "Deep Knowledge Tracing On Programming Exercises," in *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale, (L@S '17)*, pp. 201–204, ACM, New York, NY, USA, April 2017.

[13] S. Montero, A. Arora, S. Kelly, B. Milne, and M. Mozer, "Does deep knowledge tracing model interactions among skills?" in *Proceedings of the 11th International Conference on Educational Data Mining*, 2018.

[14] A. Lalwani and S. Agrawal, "Few hundred parameters outperform few hundred thousand?" in *Proceedings of the 10th International Conference on Educational Data Mining*, 2017.

[15] Y. Mao, C. Lin, and M. Chi, "Deep learning vs. bayesian knowledge tracing: Student models for interventions," *Journal of Educational Data Mining*, vol. 10, no. 2, pp. 28–54, 2018.

[16] K. H. Wilson, X. Xiong, M. Khajah et al., "Estimating student proficiency: Deep learning is not the panacea," in *Neural Information Processing Systems, Workshop on Machine Learning for Education*, pp. 1–8, 2016.

[17] M. Khajah, R. V. Lindsey, and M. Mozer, "How deep is knowledge tracing?" in *Proceedings of the 9th International Conference on Educational Data Mining*, 2016.

[18] X. Xiong, S. Zhao, E. V. Inwegen, and J. Beck, "Going deeper with deep knowledge tracing," in *Proceedings of the 9th International Conference on Educational Data Mining*, pp. 545–550, 2016.

[19] C. Yeung and D. Yeung, "Addressing two problems in deep knowledge tracing via prediction-consistent regularization," 2018, https://arxiv.org/abs/1806.02180.

[20] P. P. Bendangnuksung, "Students' performance prediction using deep neural network," *International Journal of Applied Engineering Research*, vol. 13, no. 2, pp. 1171–1176, 2018.

[21] S. Tang, J. C. Peterson, and Z. A. Pardos, "Deep neural networks and how they apply to sequential education data," in *Proceedings of the Third ACM Conference on Learning @ Scale (L@S '16)*, pp. 321–324, ACM, New York, NY, USA, April 2016.

[22] N. Sharada, M. Shashi, and X. Xiong, "Modeling student knowledge retention using deep learning and random forests," *Journal of Engineering and Applied Sciences*, vol. 13, no. 6, pp. 1347–1353, 2018.

[23] B. Guo, R. Zhang, G. Xu, C. Shi, and L. Yang, "Predicting students performance in educational data mining," in *Proceedings of the International Symposium on Educational Technology, (ISET '15)*, pp. 125–128, China, July 2015.

[24] F. Okubo, T. Yamashita, A. Shimada, and H. Ogata, "A neural network approach for students' performance prediction," in *Proceedings of the the Seventh International Learning Analytics & Knowledge Conference (LAK '17)*, pp. 598-599, ACM., New York, NY, USA, March 2017.

[25] M. M. Alam, M. K. Islam, K. Mohiuddin, M. S. Kaonain, A. K. Das, and M. H. Ali, "A reduced feature based neural network approach to classify the category of students," in *Proceedings of the 2nd International Conference on Innovation in Artificial Intelligence, (ICIAI '18)*, pp. 28–32, China, March 2018.

[26] B. Kim, E. Vizitei, and V. Ganapathi, "Gritnet 2: Real-time student performance prediction with domain adaptation," 2018, https://arxiv.org/abs/1809.06686.

[27] B. Kim, E. Vizitei, and V. Ganapathi, "Gritnet: Student performance prediction with deep learning," 2018, https://arxiv.org/abs/1804.07405.

[28] M. Fei and D.-Y. Yeung, "Temporal models for predicting student dropout in massive open online courses," in *Proceedings of the IEEE International Conference on Data Mining Workshop (ICDMW '15)*, pp. 256–263, IEEE Computer Society, Washington, DC., USA, November 2015.

[29] M. Teruel and L. A. Alemany, "Co-embeddings for student modeling in virtual learning environments," in *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization, (UMAP '18)*, pp. 73–80, ACM, New York, NY, USA, July 2018.

[30] W. Wang, H. Yu, and C. Miao, "Deep model for dropout prediction in MOOCs," in *Proceedings of the 2Nd International Conference on Crowd Science and Engineering, (ICCSE'17)*, pp. 26–32, New York, NY, USA, July 2017.

[31] J. Whitehill, K. Mohan, D. Seaton, Y. Rosen, and D. Tingley, "Delving deeper into MOOC student dropout prediction," 2017, https://arxiv.org/abs/1702.06404.

[32] W. Xing and D. Du, "Dropout Prediction in MOOCs: Using Deep Learning for Personalized Intervention," *Journal of Educational Computing Research*, pp. 1–24, 2018.

[33] W. Min, J. B. Wiggins, L. Pezzullo et al., "Predicting dialogue acts for intelligent virtual agents with multimodal student interaction data," in *Proceedings of the 9th International Conference on Educational Data Mining*, pp. 454–459, 2016.

[34] A. Sharma, A. Biswas, A. Gandhi, S. Patil, and O. Deshmukh, "LIVELINET: A multimodal deep recurrent neural network to predict liveliness in educational videos," in *Proceedings of the 9th International Conference on Educational Data Mining*, pp. 215–222, 2016.

[35] J. Yang, K. Wang, X. Peng, and Y. Qiao, "Deep recurrent multi-instance learning with spatio-temporal features for engagement

intensity prediction," in *Proceedings of the 20th ACM International Conference on Multimodal Interaction, (ICMI '18)*, pp. 594–598, Boulder, CO, USA, October 2018.

[36] A. M. Aung, A. Ramakrishnan, and J. Whitehill, "Who are they looking at? automatic eye gaze following for classroom observation video analysis," in *Proceedings of the 11th International Conference on Educational Data Mining*, pp. 166–170, Xi'an, China, May 2018.

[37] A. A. N. Tato, R. Nkambou, and A. Dufresne, "Convolutional neural network for automatic detection of sociomoral reasoning level," in *Proceedings of the 10th International Conference on Educational Data Mining*, 2017.

[38] K. Abhinav, V. Subramanian, A. Dubey, P. Bhat, and A. D. Venkat, "Lecore: A framework for modeling learner's preference," in *Proceedings of the 11th International Conference on Educational Data Mining*, 2018.

[39] C. Wong, "Sequence based course recommender for personalized curriculum planning," in *Artificial Intelligence in Education*, vol. 10948 of *Lecture Notes in Computer Science*, pp. 531–534, Springer International Publishing, 2018.

[40] S. Zhao, Y. Zhang, X. Xiong, A. Botelho, and N. Heffernan, "A memory-augmented neural model for automated grading," in *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, pp. 189–192, ACM, Cambridge, Massachusetts, USA, April 2017.

[41] K. Taghipour and H. T. Ng, "A Neural Approach to Automated Essay Scoring," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1882–1891, Association for Computational Linguistics, Austin, Texas, November 2016.

[42] Y. Zhang, R. Shah, and M. Chi, "Deep learning + student modeling + clustering: a recipe for effective automatic short answer grading," in *Proceedings of the 9th International Conference on Educational Data Mining*, pp. 562–567, 2016.

[43] J. G. Alvarado, H. A. Ghavidel, A. Zouaq, J. Jovanovic, and J. McDonald, "A comparison of features for the automatic labeling of student answers to open-ended questions," in *Proceedings of the 11th International Conference on Educational Data Mining*, 2018.

[44] B. Akram, W. Min, E. N. Wiebe, B. W. Mott, K. Boyer, and J. C. Lester, "Improving stealth assessment in game-based learning with lstm-based analytics," in *Proceedings of the 11th International Conference on Educational Data Mining*, 2018.

[45] H. Choi, Z. Wang, C. Brooks, K. Collins-Thompson, B. G. Reed, and D. Fitch, "Social work in the classroom? A tool to evaluate topical relevance in student writing," in *Proceedings of the 10th International Conference on Educational Data Mining*, 2017.

[46] A. Sales, A. Botelho, T. Patikorn, and N. T. Heffernan, "Using big data to sharpen design-based inference in A/B tests," in *Proceedings of the 11th International Conference on Educational Data Mining*, 2018.

[47] M. Feng, N. Heffernan, and K. Koedinger, "Addressing the assessment challenge in an online system that tutors as it assesses," *User Modeling and User-Adapted Interaction: The Journal of Personalization Research*, vol. 19, no. 3, pp. 243–266, 2009.

[48] N. T. Heffernan and C. L. Heffernan, "The ASSISTments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching," *International Journal of Artificial Intelligence in Education*, vol. 24, no. 4, pp. 470–497, 2014.

[49] L. Zhang, X. Xiong, S. Zhao, A. Botelho, and N. T. Heffernan, "Incorporating rich features into deep knowledge tracing," in *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, pp. 169–172, ACM, New York, NY, USA, April 2017.

[50] K. H. Wilson, Y. Karklin, B. Han, and C. Ekanadham, "Back to the basics: Bayesian extensions of IRT outperform neural networks for proficiency estimation," 2016, https://arxiv.org/abs/1604.02336.

[51] J. Stamper, A. Niculescu-Mizil, S. Ritter, G. J. Gordon, and K. R. Koedinger, "Algebra I 2005-2006. Challenge data set from KDD Cup 2010 Educational Data Mining Challenge," 2010, http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp.

[52] K. Koedinger, R. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper, "A data repository for the EDM community: The PSLC DataShop," in *Handbook of Educational Data Mining*, C. Romero, S. Ventura, M. Pechenizkiy, and R. Baker, Eds., Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, pp. 43–55, CRC Press, 2010.

[53] L. Wang, A. Sy, L. Liu, and C. Piech, "Learning to represent student knowledge on programming exercises using deep learning," in *Proceedings of the 10th International Conference on Educational Data Mining*, pp. 201–204, Cambridge, Mass, USA, April 2017.

[54] H. Singh, S. K. Saini, R. Chaudhry, and P. Dogga, "Modeling hint-taking behavior and knowledge state of students with multi-task learning," in *Proceedings of the 11th International Conference on Educational Data Mining*, 2018.

[55] W. G. Hatcher and W. Yu, "A survey of deep learning: platforms, applications and emerging research trends," *IEEE Access*, vol. 6, pp. 24411–24432, 2018.

[56] J. Schmidhuber, "Deep learning in neural networks: an overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[57] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[58] G. Zhong, L. Wang, X. Ling, and J. Dong, "An overview on data representation learning: From traditional feature learning to recent deep learning," *The Journal of Finance and Data Science*, vol. 2, no. 4, pp. 265–278, 2016.

[59] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, https://arxiv.org/abs/1409.1556.

[60] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade*, vol. 7700 of *Lecture Notes in Computer Science*, pp. 437–478, Springer, Berlin, Germany, 2nd edition, 2012.

[61] T. L. Fine, *Feedforward Neural Network Methodology*, Springer, Berlin, Germany, 1999.

[62] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[63] R. Beale and T. Jackson, *Neural Computing - An Introduction*, CRC Press, Boca Raton, Fla, USA, 1990.

[64] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Perseus Publishing, 1991.

[65] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, and D.-R. Liou, "Autoencoder for words," *Neurocomputing*, vol. 139, pp. 84–96, 2014.

[66] S. Chandar, S. Lauly, H. Larochelle et al., "An autoencoder approach to learning bilingual word representations," in *Advances in Neural formation Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., pp. 1853–1861, Curran Associates, Inc., 2014.

[67] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[68] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[69] J. Nagi, F. Ducatelle, G. A. Di Caro et al., "Max-pooling convolutional neural networks for vision-based hand gesture recognition," in *Proceedings of the IEEE International Conference on Signal and Image Processing Applications (ICSIPA'11)*, pp. 342–347, Malaysia, November 2011.

[70] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 60, pp. 1097–1105, Curran Associates, Inc., 2012.

[71] S. Ji, W. Xu, M. Yang, and K. Yu, "3D Convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[72] D. Silver, A. Huang, C. J. Maddison et al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[73] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[74] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Acadamy of Sciences of the United States of America*, vol. 79, no. 8, pp. 2554–2558, 1982.

[75] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '13)*, pp. 6645–6649, May 2013.

[76] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pp. 655–665, Association for Computational Linguistics, June 2014.

[77] K. Cho, B. Van Merriënboer, C. Gulcehre et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing, (EMNLP '14)*, pp. 1724–1734, Qatar, October 2014.

[78] M. C. Mozer, "A focused backpropagation algorithm for temporal pattern recognition," in *Backpropagation*, vol. 3, pp. 137–169, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1995.

[79] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[80] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, vol. 2014, pp. 103–111, Association for Computational Linguistics, Doha, Qatar, October 2014.

[81] A. Graves, "Generating sequences with recurrent neural networks," 2013, https://arxiv.org/abs/1308.0850.

[82] D. Wang and E. Nyberg, "A long short-term memory model for answer sentence selection in question answering," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pp. 707–712, Association for Computational Linguistics, China, July 2015.

[83] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, "Action recognition in video sequences using deep bi-directional lstm with cnn features," *IEEE Access*, vol. 6, pp. 1155–1166, 2018.

[84] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille, "Deep captioning with multimodal recurrent neural networks (m-rnn)," 2014, https://arxiv.org/abs/1412.6632.

[85] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, "Sequence to sequence - Video to text," in *Proceedings of the 15th IEEE International Conference on Computer Vision, ICCV 2015*, pp. 4534–4542, December 2015.

[86] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, pp. 194–281, MIT Press, Cambridge, MA, USA, 1986.

[87] G. E. Hinton, "Deep belief networks," *Scholarpedia*, vol. 4, no. 5, article 5947, 2009.

[88] A. Bordes, S. Chopra, and J. Weston, "Memory networks," 2014, https://arxiv.org/abs/1410.3916.

[89] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[90] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, pp. 2546–2554, USA, 2011.

[91] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*, pp. 265–272, Omnipress, USA, 2011.

[92] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.

[93] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[94] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning, ICML'13*, pp. 1139–1147, 2013.

[95] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[96] T. Chen, I. J. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," 2015, https://arxiv.org/abs/1511.05641.

[97] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[98] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS'13)*, pp. 3111–3119, Curran Associates Inc., USA, 2013.

[99] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[100] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial nets," in *Advances in Neural formation Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., pp. 2672–2680, Curran Associates, Inc., 2014.