Modeling and Experimental Design for MOOC Dropout Prediction: A Replication Perspective

Josh Gardner
Paul G. Allen School of Computer Science & Engineering
University of Washington
jpgard@cs.washington.edu

Yuming Yang
Department of Statistics
The University of Michigan
yangym@umich.edu

Ryan S. Baker
Graduate School of Education
The University of Pennsylvania
rybaker@upenn.edu

Christopher Brooks
School of Information
The University of Michigan
brooksch@umich.edu

ABSTRACT

Replication of machine learning experiments can be a useful tool to evaluate how both *modeling* and *experimental design* contribute to experimental results; however, existing replication efforts focus almost entirely on modeling alone. In this work, we conduct a three-part replication case study of a state-of-the-art LSTM dropout prediction model. In our first experiment, we replicate the original authors' methodology as precisely as possible in collaboration with the original authors. In a second experiment, we demonstrate that this initial experiment likely overestimates the generalization performance of the proposed model due to the design of its validation. In a third experiment, we attempt to achieve the previously-reported performance in a more difficult, but more relevant, hold-out set design by exploring a large space of model regularization configurations. We demonstrate that we can reduce overfitting and improve generalization performance of the model, but cannot achieve the previously-reported level of performance. This work demonstrates the importance of replication of predictive modeling experiments in education, and demonstrates how experimental design and modeling decisions can impact the extent to which model performance generalizes beyond the initial training data.

1. INTRODUCTION

The repeated verification of scientific findings is central to the construction of robust scientific knowledge, particularly in a fast-growing field such as educational data mining. This can take the form of (a) reproduction (reproducibility), using the original methods applied to the original data to reproduce the original results, and (b) replication (replicability), applying the original methods to *new* data to assess the robustness and generalizability of the original findings. Since reproducibility is a necessary condition for replicability (an experimental procedure cannot be applied to new data if the procedure cannot be reproduced), achieving replicability requires solving the problem of reproducibility.

In this work, we discuss the reproducibility crisis in machine learning, noting specific challenges faced by applied researchers in the learning sciences, particularly in the subfields of educational data mining and learning analytics. We argue that existing frameworks for reproducible machine learning such as open code-sharing platforms and public code notebooks are valuable steps, but are insufficient to fully address the challenges both within our subfield of interest and the broader machine learning community. In particular, we argue that code-sharing does not address the breadth of challenges – experimental, methodological, and data – we face as practitioners, as Section 3 details. Instead, we propose a paradigm of *end-to-end* reproducibility for machine learning: fully reproducing (or replicating) the pipeline from raw data to model evaluation. End-to-end reproducibility is possible with current freely-available computing technologies, namely containerization.

Using an open-source platform for conducting reproducible end-to-end machine learning experiments on large-scale educational data, the MOOC Replication Framework (MORF), we conduct a three-stage replication experiment in Section 4, which is the primary contribution of this work.[1] Our case study evaluates both the experimental design, by comparing different train-test regimes, as well as the modeling, by replicating the original results and attempting to extend them via modern neural network regularization methods. We present practical recommendations based on our results in Section 5. We describe additional benefits, beyond reproducibility, afforded by MORF, and replication in machine learning more broadly, in Section 6, concluding in Section 7.

[1]The code to fully replicate these experiments, including their execution environments and software dependencies within a Docker environment, is available at https://github.com/educational-technology-collective/dl-replication/.

2. PRIOR WORK

2.1 The Reproducibility Crisis in Machine Learning

Much has been written about the reproducibility crisis in science, particularly in fields which conduct human subjects research such as social psychology. Recent empirical evidence has shown that issues with reproducibility are also widespread in the field of machine learning. A survey of 400 research papers from leading artificial intelligence venues shows that none of the works surveyed document all aspects necessary to fully reproduce the work; only 20-30% of the factors evaluated were adequately reported in the works surveyed [19]. A replication study of deep reinforcement learning algorithms [21] show that the variance inherent to statistical algorithms, the use of different hyperparameter settings, and even different random number generation seeds contribute to a lack of reproducibility in machine learning research and have a direct impact on whether experimental results and baseline model implementations replicate. A survey of 30 machine learning studies in text mining identified poor reproducibility due to lack of access to data, software environment, randomization control, and implementation methods [28]. None of the 30 works surveyed provided source code, and only one of 16 applicable studies provided an executable to be used to reproduce the experiments.

These reproducibility issues are partly attributable to culture and convention. A survey of authors published in the Journal of Machine Learning Research found that roughly one third intentionally did not make their implementations available, for reasons including a lack of professional incentives, a reluctance to publish messy code, and the convention that doing so is optional [33]. [30] observes that only five of 125 published articles in the journal *Biostatistics* have passed the (voluntary) reproducibility review since its inception two years prior, yet considers this effort "successful" compared to the reproducibility of previous work.

As big data and machine learning permeate disciplines, this crisis in replication has also affected other fields of study, including the learning sciences. This is especially relevant in cases of very large datasets where the majority of learning is computer-mediated, such as in Massive Open Online Courses (MOOCs). For example, [16] showed that a large-scale replication of machine learning models led to substantially different conclusions about the optimal modeling techniques for MOOC dropout, with several findings replicating significantly in the *opposite* direction of the original study (which was conducted on only a single MOOC). In an attempted replication of the "deep knowledge tracing" method originally introduced in [31], the results showed that much simpler methods could achieve equivalent performance, and that the performance gains demonstrated in the original work were at least partially due to data leakage [24]. Somewhat more favorably, in [2], the authors find that 12 of 15 experimental findings in MOOCs replicated using a production-rule framework, but an additional two findings replicated significantly in the *opposite* direction.

2.2 Existing Tools for Reproducible Machine Learning

An exhaustive survey of tools and platforms to support reproducible machine learning research is beyond the scope of this paper. However, we include a brief survey of tools most relevant to reproducible machine learning for predictive analytics in education.

OpenML [35] is "an open, organized, online ecosystem for machine learning" that allows users to create data science pipelines to address specific "tasks", such as classification and clustering. The OpenAI Gym is an open-source interface for developing and comparing reinforcement learning algorithms [6]. Its wide use for both teaching and research serve as an example of how a subfield can create and adopt shared tools that meet researchers' needs while enhancing reproducibility. Recently, several publishing platforms dedicated to reproducible computational research have also formed, such as ReScience [2], CodaLab [3], and WholeTail [5]. These platforms unify code, data, computation, and presentation in a single location. CodaLab and WholeTail also use Docker containerization to ensure reproducibility.

Each of these platforms is an important step toward reproducible machine learning research, and many of them address key barriers. However, these tools are insufficient for many types of machine learning tasks, including supervised learning with large-scale behavioral data from MOOCs. In particular, none of these platforms supports replication where the underlying data sets are privacy-restricted and cannot be publicly shared. In some cases, such as WholeTail, the platform scope is explicitly limited to public (non-restricted) datasets [5]. However, in educational data, many of the types of unanonymizable data that are privacy-restricted are also necessary for analysis (such as the text of discussion forum postings, IP addresses, or student names). Such restrictions are also likely to drive away machine learning researchers from working with this data, as gaining access to unprocessed raw educational data can be difficult or impossible without close collaborators and strong institutional support. Even with institutional support, government regulations such as the Family Educational Rights and Privacy Act (FERPA) may restrict or complicate data sharing.

3. THE MOOC REPLICATION FRAMEWORK

The replication crisis is the result of a confluence of forces which must be collectively addressed in order to achieve end-to-end reproducibility. Prior work has identified three groups of challenges: experimental, methodological, and data challenges [17]. No existing solution discussed in Section 2.2 currently addresses all three barriers. In this section, we outline three key barriers to reproducibility, and describe how these barriers are addressed by the MOOC Replication Framework (MORF), the research tool used to conduct this experiment.

MORF itself is a Python toolkit, accompanied by a platform-as-a-service (the "MORF Platform"), which collectively address the challenges faced by researchers studying large-scale online learning data noted above [17].[4]

[2]http://rescience.github.io/about/
[3]http://codalab.org/
[4]The MORF website, which includes documentation

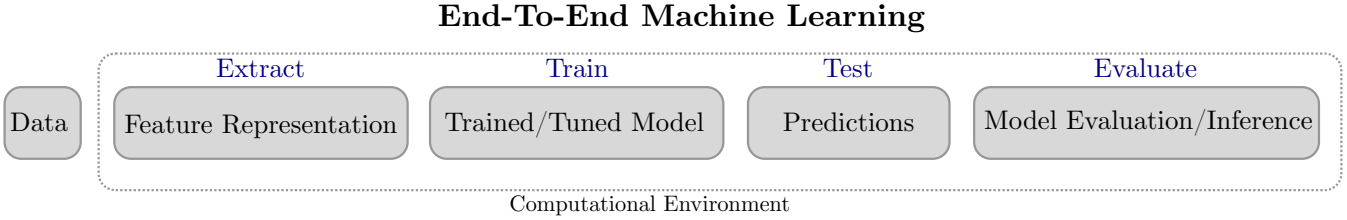

Figure 1: End-to-end reproducibility requires addressing data, technical, and methodological issues with reproducibility. Replication of the computational environment, in particular, is key to replicating this complete pipeline from raw data to results. Each of the four stages of the supervised learning pipeline executed inside the computational environment are encapsulated in MORF's execution model (see Section 3).

Users submit jobs to the MORF Platform using short, 4-5 line "controller" scripts which guide the execution of each stage of the end-to-end supervised learning pipeline (extract, train, test, and evaluate) shown in Figure 1. The use of controller scripts is a common approach to conducting reproducible computational research [25]. MORF's combination of containerization and controller scripts allow the user to manage low-level experimental details (operating system and software dependencies, feature engineering methods, and statistical modeling) by constructing a Docker container which is submitted to MORF for execution. MORF manages high-level implementation details (parallelization, data wrangling, caching of results) by "bringing the computation to the data." This prevents the download of sensitive raw data (currently, this includes the complete raw data exports from over 270 MOOCs offered by two institutions [17]). The containers used to execute each job are persisted in MORF's public Docker Cloud repository, and the configuration file and controller scripts are persisted in Zenodo and assigned a unique Digital Object Identifier (DOI). This yields a reproducible end-to-end pipeline that is flexible, easy to use, and computationally efficient.

MORF eases the computational expense of conducting such research at scale by providing nearly an order of magnitude greater computational infrastructure than any of the platforms discussed in Section 2.2, and out-of-the-box parallelization to utilize it. See [17] for a more thorough comparison to other platforms.

3.1 Experimental Reproducibility via Containerization

Experimental challenges with reproducibility relate to reproducing the exact experimental protocol [17]. It has been noted that code-sharing alone is insufficient to guarantee reproducibility in computational research. For example, [10] showed that the published code accompanying 20% of their large sample of 613 published computer systems papers failed to build or run, and in total, it was not possible to verify or reproduce 75.1% of studies surveyed using the artifacts provided in publication.

Even when code is available, other technical issues can prevent reproducibility in research workflows [25]. These include *code rot*, in which code becomes non-functional or its functionality changes as the underlying dependencies change over time (for example, an update to a data processing library which breaks backwards compatibility, or a modified implementation of an algorithm which changes experimental results), as well as *dependency hell*, in which configuring the software dependencies necessary to install or run code prevents successful execution [3]. This complex web of interdependencies is rarely described or documented in published machine learning and computational science work [19, 28], despite over two decades of evidence that it is a necessary condition for reproducing computational results [7].

MORF uses containerization to support end-to-end reproducibility (Figure 1). The Docker containers submitted to MORF fully encapsulate the code, software dependencies, and execution environment of an end-to-end machine learning experiment in a single file, ensuring end-to-end reproducibility and enabling sharing of the containerized experiment. Docker containers were developed to resolve many of the experimental reproducibility challenges described above in software development contexts [27], and are frequently used in industrial software applications, computational modeling, and computer systems research [3, 9, 23]. A major advantage of containerization over simple code-sharing is that containers fully reproduce the entire execution environment of the experiment, including code, software dependencies, and operating system libraries. Docker containers are more lightweight than a full virtual machine, but achieve the same level of reproducibility [27, 23]. Building Docker containers requires only a single Dockerfile (akin to a makefile) which contains instructions for building the environment. This imposes minimal additional burden on researchers relative to configuring, programming, and executing an experiment, but achieves a considerable increase in reproducibility. While other existing machine learning research platforms sometimes utilize Docker "under the hood," this limits users' ability to fully leverage containerization by configuring or sharing these environments. We are not aware of any platform which allows users to build and submit Docker images directly for execution as MORF does.

As part of MORF, we are assembling an open-source library of pre-built Docker containers to replicate experiments conducted on MORF to serve as shared baseline implementations. These containers can be loaded with a single line of code, allowing the research community to replicate, fork, interrogate, modify, and extend the results presented here[5].

and short tutorials, is at https://educational-technology-collective.github.io/morf/

[5]The experiment presented below can be loaded by running

3.2 Methodological Reproducibility via Platform Architecture

Existing work on reproducibility largely focuses on strictly technical challenges, but as our experiment in Section 4 shows, methodological issues are at least as important. *Methodological challenges* to reproducibility reflect the methods of the study, such as its procedure for model tuning or statistical evaluation. Poor methodological decisions can lead to a lack of inferential reproducibility [19]. We see such issues in e.g. the use of biased model evaluation procedures [8, 36]; improperly-calibrated statistical tests for classifier comparison [12]; large-scale hypothesis testing where thousands of hypotheses or models are tested at once, such as in massive unreported searches of the hyperparameter space without statistical evaluation or appropriate corrections, or "random seed hacking," wherein the random number generator itself is systematically searched in order to make a target model's performance appear best or a baseline model worse [21].

MORF is designed to provide sensible default methodological procedures for many machine learning tasks, such as model evaluation, in practical terms nudging researchers to make sound choices. For example, MORF avoids the use of cross-validation for model evaluation: The prediction tasks to which most MOOC models aspire are prediction of *future* student performance (i.e., in an ongoing course where the true labels – such as whether a student will drop out – are unknown at the time of prediction). As such, using cross-validation within a MOOC session, when the outcome of interest is accuracy on a *future* MOOC session, provides an unrealistic and potentially misleading estimate of model performance. Prior work has demonstrated that within-session cross-validation in the MOOC domain can produce overly favorable estimates of classification performance on a future (unseen) course or future session from the same course [37, 4]. Adopting more effective model evaluation techniques by default requires no additional work for MORF users, and ensures that work produced on the MORF platform follows effective model evaluation procedures. MORF's large data repository also prevents users from having to utilize only a single dataset for both training and testing; with many iterations of many unique MOOCs available, users can have considerable training data available while also conducting effectively-designed experiments with ample and representative test data.

3.3 Data Reproducibility via Execute-Against Access

Data reproducibility concerns the availability of data itself. In many domains, making raw data available is more an issue of convention than a true barrier to reproducibility. However, in the case of educational data mining, data are often governed by strict privacy regulations which protect the privacy of student education records. Similar restrictions affect many other fields, from the health sciences to computational nuclear physics [25]. As a result, researchers are often legally prohibited from making their data available. Efforts such as [26] and [20] have attempted to address this problem in education by only releasing non-identifiable data, but many analyses require the original, unprocessed data for a full replication. Indeed, restricted data sharing is one of the main factors (in our experience) hindering generalizability analysis in educational data mining: investigators are generally limited to one or two courses worth of data (e.g. the courses they instruct or specific publicly available courses), and models are often overfit to these datasets.

MORF achieves data reproducibility while also meeting data privacy restrictions by providing strictly "execute-against" access to underlying data [17]. Most MOOCs are generated by a small number of platforms (e.g. Coursera, edX), and all courses from a given platform use publicly-documented data schemas, e.g. [11]. Thus, users can develop experiments using their own data from a given platform – or even the public documentation – and then submit these experiments for MORF to execute against *any* other course from that platform. This enables MORF to currently provide an interface to over 270 unique sessions of more than 70 unique courses offered by two different institutions on the Coursera platform, and to execute containerized experiments against this data in a secure, sandboxed environment by utilizing the shared public schema of the datasets [11]. These shared public data schemas also ensure that existing experiments in MORF can be replicated against new data (from the same MOOC platform) as it becomes available.

4. REPLICATION EXPERIMENT: NEURAL MOOC DROPOUT MODELS

In the remainder of this paper, we conduct an in-depth exploration of previously-published MOOC dropout prediction models using MORF. Neural models have demonstrated the capacity to achieve state-of-the-art performance on a wide variety of modeling and prediction tasks, from language modeling to computer vision. Their application to MOOC research has also demonstrated initial promising results [13, 29] due to their ability to model complex functional relationships between student behavior data and learning outcomes, but such research has been limited. In this section, we use MORF to replicate a comparison conducted in [13], which compares several machine learning algorithms using a set of seven activity features (e.g. number of lecture videos viewed, quizzes attempted, and discussion forum posts for each student) over each week in a MOOC in order to predict a binary dropout label indicating whether a user showed activity in the final week of a course.

This study is an ideal candidate for replication because it has been highly cited in the field, but compares six models over five weeks of a MOOC (effectively testing $\frac{6 \cdot 5 \cdot 5}{2} = 75$ pairwise hypotheses) using cross-validation on only a single dataset. This testing of many hypotheses/comparisons, with only a single observation for each, can lead to poor methodological reproducibility and provides no information about the variability of the estimates, relative to their magnitude [14]. Particularly because this experiment was concerned with empirical performance (in order to inform future "early warning" dropout prediction systems), obtaining an accurate estimate of models' expected performance on *future* course sessions across a large, representative dataset can provide insight into the generalizability of these findings. The use of within-session cross-validation to draw conclusions about future generalization performance remains common in MOOC prediction research [15].

`docker pull themorf/morf-public:fy2015-replication` in the terminal of any computer with Docker installed.

We present the results of our replication in Section 4.1, which matches the authors' original comparisons and their experimental design. The original work evaluated three different definitions of MOOC dropout; for tractability within this paper, we strictly replicate "Definition 1" of dropout from [13], which is the most commonly-used definition of dropout in MOOC research [16]. In Section 4.2, we present the results of a second experiment, which compares the same models using a different experimental setup (predicting on future course sessions) in order to demonstrate that the original experimental design overestimates the generalization performance of these models, likely due to overfitting. Our experimental analysis quantifies this overestimation and provides evidence using statistical tests. Finally, in Section 4.3, we use a comprehensive hyperparameter tuning and model evaluation procedure to show that we can improve the generalization performance of the LSTM model using batch normalization, but that the addition of two other neural network regularization methods does not further improve either the LSTM or RNN model. We also present detailed information on a variety of parameterizations examined in order to inform future work.

4.1 Experiment 1: Full Replication With Original Design

In our first experiment, we replicated the original experiment using the original design – estimating model performance via cross-validation within a single course – across 45 unique MOOCs using MORF, in consultation with the original authors. Results are shown in Figure 2.

The original work concluded that a Long Short-Term Memory (LSTM) neural network model " beats the ... other proposed methods by a large margin" [13], pp.1. – a result which matches the advances that neural models have made in other domains. Our results in Experiment 1 show, however, that (1) LSTM is actually one of the lowest-performing models, with the lowest average performance of any model tested in weeks 4 and 5; (2) in most cases, the 95% confidence intervals for algorithm performance overlap, and so we cannot conclude that there is a difference in performance between any but the very best and worst models evaluated, even without applying corrections to adjust for the use of multiple comparisons; and (3) observed performance of all models is lower than in [13], particularly in later weeks of the course.

We hypothesize that the relatively poor performance of LSTM may may be due to overfitting on the dataset used in the original experiment in [13]. Particularly when using cross-validation for model selection on a single dataset with a highly flexible model such as LSTM, the experimental design of the original work was quite susceptible to overfitting. Overfitting seems particularly likely because no procedure for selecting hyperparameters was reported in [13], and some key hyperparameter settings for the LSTM model (e.g. batch size) were not reported at all. These hyperparameters were not available even after correspondence with the authors, who did not record them and no longer had the original code available (which itself points to the need for reliable long-term reproducibility solutions such as MORF). The need for detailed hyperparameter reporting in reproducible research has been noted previously [21].

(2) shows the advantage of using MORF's large data repository, which allows us to observe variability in each algorithm's performance across many MOOCs to form confidence intervals for algorithm performance. Experiment 1 suggests that while differences in average performance may exist, these are too small to be interpreted as genuine and not spurious – particularly in light of the results shown in Figure 3, which shows that the differences due to cross-validation bias are larger than the observed differences between algorithms in most comparisons. We note that the out-of-fold prediction error in each cross-validation iteration could have been used to provide an estimate of the variability in model performance when applied to new data in [13]; however, this was not provided in the original work. In any case, having more datasets available makes the estimation of this variability more reliable than would have been possible with only one course.

Finally, the generally lower observed performance for all models may may also be due to overfitting, and particularly due to the experimental design. Experiment 1 uses an identical design to [13]. However, [13] uses only a single course (in comparison to the 45 courses used in Experiment 1), which would have permitted tuning many of the hyperparameters germane to neural models (e.g. learning rate, activation functions, number of training epochs, batch size, number of hidden layers and units) specifically to optimize performance on this individual course. In contrast, when using the large, diverse set of courses in MORF, over-tuning such hyperparameters to an individual course would be disadvantageous (and extremely difficult, due to the diversity of course sizes and learner populations represented in MORF [17]). When fitting multiple courses at once (as in MORF), the incentive is instead to find a set of hyperparameters which generalizes to many different types of courses.

The results of this experiment demonstrate the importance of using large, diverse datasets for machine learning experiments, and of performing multiple experimental replications. Additionally, these results demonstrate that simpler models – such as RNN, radial SVM, and logistic regression – may achieve equivalent or better performance than LSTM for MOOC dropout prediction. Our results, which are contrary to the findings of the original study, also suggest that further replication is necessary to identify the most effective algorithms for MOOC dropout prediction, as we perform no hyperparameter tuning in Experiment 1 and only replicate the original models and features.

4.2 Experiment 2: Replication With Improved Experimental Design

The original aim of the dropout prediction model in [13] was to achieve accurate prediction on *future* courses. Prior work has shown considerable differences in prediction results depending on the prediction and transfer architectures used, particularly when predicting on the same couse session used for model training instead of a future iteration of the same course [37, 4]. In the case of the original experiment, the cross-validation design was due to necessity: data from only a single MOOC was available to the original authors, and the prediction on out-of-fold data within the same session was used. However, MORF makes *all* sessions of each course available, with 45 courses having at least two ses-

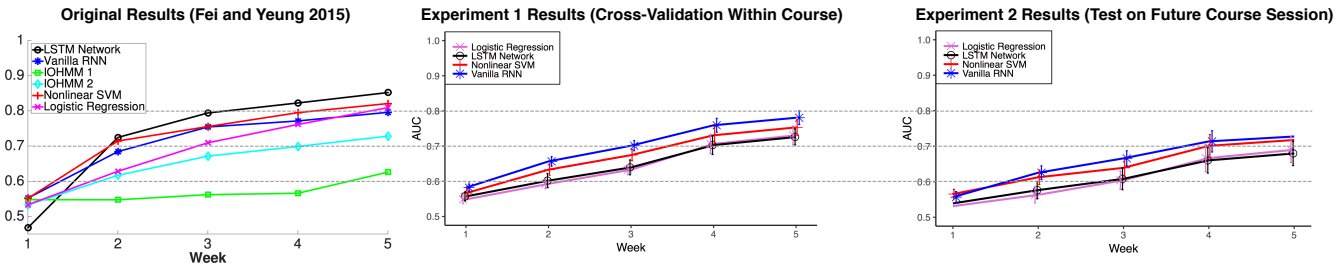

Figure 2: Original results from [13] (left) and replication results using the MOOC Replication Framework evaluated using a held-out future course session (center) and cross-validation (right). 95% confidence intervals shown. IOHMM not replicated due to lack of an open-source implementation which supported prediction.

sions. Therefore, in this section, we conduct an experiment which uses the same modeling methods as the original experiment, but do so using a true holdout architecture, where each model is trained on the first $n-1$ sessions of a MOOC, and is tested on the final session. We refer to this experiment as Experiment 2. Note that only the *design* of the prediction experiment is changed from Experiment 1.

The results of Experiment 2 are shown in the right panel of Figure 2. The contrast between the center (cross-validation) and right (holdout) panels demonstrates the optimistic bias which can be introduced by evaluating generalization performance via within-session cross-validation without the use of an independent hold-out session [36]. This matches previous results demonstrating that the bias of performance estimates when models are optimized over cross-validation folds can often exceed the difference between learning algorithms [8]. These results are further demonstrated by Figure 3, which shows a persistent positive bias for model evaluation performed by cross-validation versus the "true" performance on a future course session. A two-sided Wilcoxon signed-rank test of a null hypothesis of equivalence between the holdout and cross-validated experimental results, where each model-week combination was treated as an observation, was rejected with $p < 2.2 \times 10^{-16}$.

A notable result from Figure 3 is that the observed $2\hat{\sigma}$ upper-bound on the bias due to cross-validation is around begins at roughly 0.035 AUC units when predicting after a single week, and increases to over 0.08 AUC points in later weeks. This difference due to design is as large as the difference between the highest-performing model and the two next-highest performing models in every week of the original experiment (see left panel of Figure 2). This shows that biases introduced by experimental design can entirely overshadow experimental effects, and should be a particularly strong call to action for the EDM community.

The findings of Experiment 2 are threefold. First, it demonstrates that using within-course cross-validation to estimate generalization performance on future course sessions introduces a significant positive bias. This should serve both as a call for machine learning researchers to dedicate additional attention to the experimental design of machine learning experiments, as well as a call to practitioners to rigorously evaluate models prior to their deployment for prediction. Second, we quantify this bias, showing that in practice the bias roughly falls in the 95% CI $[0, 0.05]$ – where the upper-bound is larger than the difference between learning algorithms in the original work. Third, these results provide further evidence that the original results presented in [13] may have been overfit to the data in that work, and that the generalization performance of such a model may be lower than what was suggested by the initial results from [13].

4.3 Experiment 3: Improving Model Generalization Performance via Regularization

In the previous two experiments, we presented results which called into question some findings of the initial study in [13], that neural models can significantly improve MOOC dropout prediction. Our observations largely centered on the problematic potential for overfitting and optimistic bias due to experimental design in [13]. In this section, we show that the consequences of overfitting can be at least partially ameliorated by using modern regularization methods for neural models. This section is intended to explore (a) whether we can provide further evidence that the original models were overfit to the data in [13], and (b) whether we can improve the models' generalization performance, and still achieve state-of-the-art dropout prediction performance with these models, through the use of regularization to reduce overfitting. In particular, we explore the recurrent neural network ("vanilla RNN") and Long Short-Term Memory (LSTM) models from the original study, but introduce different architectures and explore various configurations of three regularization methods, none of which were in the original work. We show that we can approach, but not match, the performance reported in [13], even when predicting on future course sessions, by applying and tuning these regularizations to the RNN and LSTM models.

Regularization is a modification made to a learning algorithm that is intended to reduce its generalization error but not its training error [18]. We explore three types of regularization in this experiment, which we describe below.

Dropout [34] randomly drops neural network units (along with their connections) during each training iteration with probability p. We test models with $p = \{0, 0.2\}$

Batch normalization [22] normalizes each feature in every minibatch of data during training, which stabilizes model training by reducing covariate shift (large changes in parameter updates due to differences in the distribution across fea-

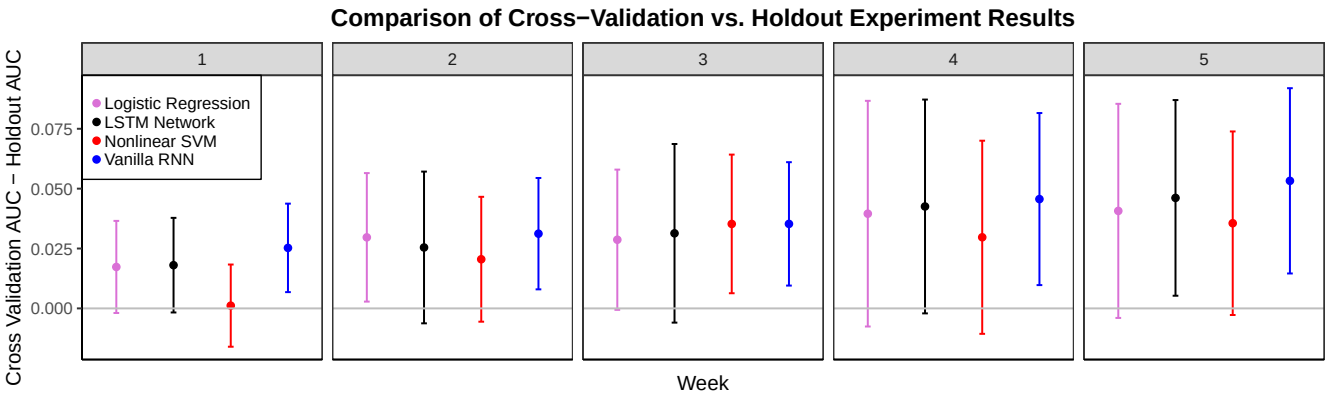

Figure 3: Comparison of AUC estimates from identical experiments using a holdout vs. within-course cross-validation evaluation architecture. These results show a persistent positive bias when within-course cross-validation is used to estimate predictive performance on a future, held-out session of a MOOC.

tures) and preventing small changes to the parameters from amplifying into larger and suboptimal changes in activations in gradients. We test models with and without batch normalization for each model.

L2 Regularization perhaps the most common regularization method in modern deep learning research [18], L2 regularization adds a penalty term to the loss function based on the L2 norm of the network weights with the parameter λ controlling the level of penalization (higher λ yields greater regularization). We test $\lambda = \{0, 0.01, 0.001\}$ for each model.

The study under replication was published in 2015, and since then, research on the regularization of neural models has advanced considerably. Dropout was only originally proposed in 2014, and batch normalization in 2015, so these techniques were still quite new at the time of publication of [13] and implementations were not widely available as part of standard neural network software, as they are now. As a result, the original publication quite reasonably did not explore these novel methods, despite their potential to improve their results in practice.

Experiment 3 evaluates dropout prediction after four weeks of the course (this is the final time point shown in Figure 2). Evaluating the full range of all weekly prediction tasks was beyond the scope of this experiment, as even the experiment here required testing 72 different hyperparameter configurations (36 LSTM and 36 RNN models) across 45 courses, resulting in a total of 3,240 total models trained. We choose the week 4 prediction task because, after four weeks, there would be maximal data for models to learn from – and also, potentially, for models to overfit to. Week 4 prediction therefore provides the best opportunity to separate models with strong generalization performance from models which overfit.

Results from Experiment 3 are shown in Figure 4. We statistically evaluate the model comparisons, and visualize the results, using the Bayesian method of [14]. This method uses a hierarchical Bayesian model to evaluate all pairwise comparisons of a set of k models across N datasets by accounting for the correlation in performance across models and datasets. While the original comparison also accounts for fold-level correlation when applying the testing procedure to the results of a cross-validation procedure, here we only have a single estimate for each dataset, and this simply adds slightly more uncertainty to our estimation, which will tend to make the model more conservative (less likely to make a decision). The procedure estimates three probabilities for each pair of classifiers X and Y: $\mathbb{P}(X > Y), \mathbb{P}(ROPE), \mathbb{P}(X < Y)$, where ROPE indicates that the difference in performance between the models is within a "region of practical equivalence", which in this experiment is set to $ROPE = 0.01$. We use a decision threshold of 0.9, which means that the procedure makes a "decision" (indicated by a colored entry in the windowpane plot of Figure 4) only when the posterior probability of one of the events is greater than or equal to 0.9; otherwise, the procedure makes no decision (indicated by a blank white entry in Figure 4).

The results in Figure 4 demonstrate that batch normalization considerably improves the LSTM model – all LSTM models with batch normalization performed better, on average, than any LSTM model without batch normalization. The hierarchical Bayesian model used to compare the modeling results across the 45 course in MORF indicated that almost all of the batch-normalized LSTM models were within the "region of practical equivalence", or ROPE, indicating a high posterior probability that these models achieve practically equivalent predictive performance (in this experiment, a ROPE of 0.01 was used, meaning that a decision of ROPE indicates a confident decision that models' test AUC differed by less than 0.01). Tuning of the other hyperparameters (L2 regularization λ, and dropout probability p) had little effect on observed model performance, with almost all hyperparameter configurations being practically equivalent within a fixed batch normalization group. These results suggest that the LSTM model, which had considerably more parameters, was likely overfitting to the data – and that the different distributions across the different input features (which is countered by batch normalization) were a strong factor contributing to this overfitting. The LSTM model with the highest average performance, with $p = 0.0, h = 10, \lambda = 0.0$ and batch normalization, achieved a mean AUC of 0.726 on the MORF dataset – only 0.002 less than the best RNN

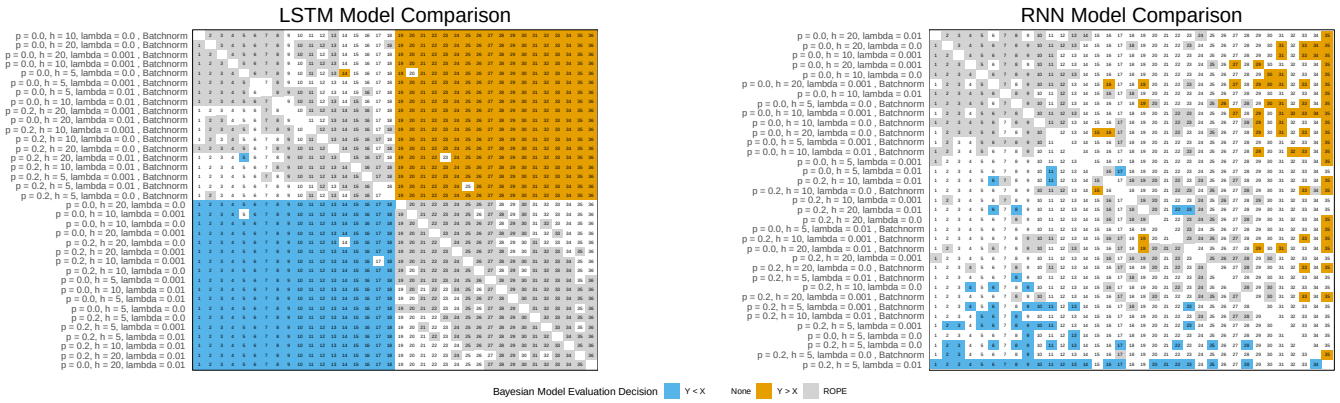

Figure 4: Windowpane plots for LSTM model tuning experiment (left) and RNN model tuning experiment (right). Batch normalization considerably improves the LSTM model. Other regularization methods (L2 regularization, dropout) show little effect. The RNN shows stable performance with a variety of configurations.

model in Experiment 3.

Figure 4 shows a more complicated picture with respect to the RNN model. Generally, the results show that the RNN performance is more robust to the hyperparameters: the RNN achieves practically equivalent performance with a range of settings. For example, models with both 10 and 20 hidden units, and with every λ value considered, achieve performance practically equivalent to the highest-performing model. The exception to this robustness is dropout – every RNN in the "family" of best models (those with performance equivalent to the highest-performing model , as in [14]) had no dropout, or $p = 0.0$. The RNN model with the highest average performance, with $p = 0.0, h = 20, \lambda = 0.01$, achieved a mean AUC of 0.728.

Collectively, these results support the hypothesis that the LSTM model as parameterized in the original experiment tends to overfit on standard-sized MOOC datasets, while providing evidence that the RNN model is less prone to overfitting. The results also show that the LSTM model can be improved, with performance to match the best RNN model, through the use of regularization. For reference, the LSTM model in the original experiment (with a single layer and 20 hidden units) has 2,261 trainable parameters, while the RNN model has 581 tunable parameters. Regularization is less likely to impact a properly-parameterized model when sufficient data is provided, while its impact can be much more evident when a model has too many free parameters for the available data, as we see in the contrast between the LSTM and RNN results of Figure 4 and the lack of effect on RNN performance (mean AUC for the original RNN show in in Experiment 2 was 0.727 at week 5).

5. IMPLICATIONS FOR PRACTICE

5.1 Experimental Design for Predictive Modeling in EDM

The current work demonstrates that experimental design can have important implications for the conclusions generated from a machine learning experiment. Our results show that both researchers and practitioners should clearly identify the hypotheses to be tested by a machine learning experiment, or the goals of a deployed model, and then utilize experimental designs which allow for inference about the types of prediction scenarios to be encountered in the task of interest. Cross-validation can be useful in evaluating a specific dataset, e.g. by fitting an explanatory model where the interpretation of learned parameters is used to understand the dataset. Cross-validation may also be useful when the available data is limited. However, in cases where prediction on a future course or generalization to *new* data are of interest, using data from a future course or session will provide more reliable estimates of model performance. For further discussion of the design of machine learning experiments, see [1].

5.2 Hyperparameter Tuning for Neural Models in EDM

The current work provides evidence regarding the importance of effective model tuning and regularization in educational data mining. In practice, educational models need to operate effectively in a wide variety of scenarios. Predictive models in EDM are often required to obtain reasonable performance even as the dataset size, target population, and even data attributes change across course populations, institutions, or platforms, making model robustness a key consideration.

This experiment adds to an existing body of evidence (e.g. [14]) that selecting an appropriate statistical model can affect predictive performance much more than hyperparameter selection (when reasonable hyperparameters are selected). The introduction of regularization methods can improve models which are overfit, but has little impact on those which are not overfit. However, introducing regularization as an additional experimental factor may be particularly challenging as it introduces an additional dimension of model tuning and can considerably increase the computational cost of experiments.

Our finding that batch normalization, in particular, imrpoved the generalization performance of the LSTM model suggests that normalization of counting-based features – which can be highly skewed and show very different distributions

for different types of actions – is an important tool for use in educational models.

6. BEYOND VERIFICATION: ADDITIONAL ADVANTAGES OF REPLICATION IN MACHINE LEARNING

Much prior work on reproducibility has focused on verification – ensuring that published results are true and can be reproduced. However, end-to-end reproducible machine learning frameworks, such as MORF, provide benefits beyond mere verification, including:

"Gold standard" benchmarking: open replication platforms allow for the comparison of results which were previously not comparable, having been conducted on different data. The use of such benchmarking datasets has contributed to the rapid advance of fields such as computer vision (e.g. MNIST, IMAGENET), natural language processing (Penn Tree Bank, Brown corpus), and computational neuroscience (openFMRI). These datasets have been particularly impactful in fields where it is difficult or expensive to collect, label, or share data (as is the case with MOOC data, due to legal restrictions on sharing and access). This can help advance the "state of the art" by providing a common performance reference which is currently missing in MOOC research.

Shared baseline implementations: We noted above that variability in so-called "baseline" or reference implementations of prior work has contributed to concerns about reproducibility in the field [21]. By providing fully-executable versions of existing experiments, MORF ameliorates these issues, allowing for all future work to properly compare to the exact previous implementation of a baseline method.

Forkability: containerization produces a ready-made executable which fully encompasses the code and execution environment of an experiment. These can be readily shared and "forked" much in the same way code is "forked" from a git repository. This allows MOOC researchers to build off of others' work by modifying part or all of an end-to-end pipeline (for example, by experimenting with different statistical algorithms but using the same feature set as a previous experiment) within the same software ecosystem.

Generalizability analysis: Each successive replication of an experiment provides information about its generalizability. Evaluating the generalizability of experiments has been a challenge in MOOC research to date, where studies conducted on restricted and often homogeneous datasets are common [15]. When containerized end-to-end implementations are available, replicating these analyses on new data – even data which are not publicly available but share the schema of the original data – becomes as straightforward as running the containerized experiment against new data.

Sensitivity Analysis: This technique, used widely in Bayesian analysis, evaluates how changes to the underlying assumptions or hyperparameters affect model fit and performance. Such an evaluation can provide useful information about a model's robustness and potential to generalize to new data. Without being able to fully reproduce a model on the original data, sensitivity analyses are not possible. In MORF, such analyses can be conducted by simply forking and modifying the containerized version of the original experiment, then re-executing it against the same data. These analyses can also include so-called *ablation analyses*, wherein individual components are removed from a model to observe their contribution to the results, as well as *slicing analyses*, where fine-grained analysis of performance across different subgroups (e.g. demographic groups) is explored [32].

Full Pipeline Evaluation: Each stage of an end-to-end machine learning experiment (feature extraction, algorithm selection, model training, model evaluation) can be done in many different ways. Each stage also affects the others (for example, some algorithms might perform best with large feature spaces; others might perform poorly with many correlated features). However, current research usually evaluates only one or two components of this pipeline (e.g. training several algorithms and tuning their hyperparameters on a fixed feature set). Not only are the remaining stages often described in poor detail or not at all [19]; such work also leaves future researchers unable to evaluate the synergy between different aspects of the end-to-end pipeline in a published experiment (for example, exploring whether an algorithm's performance improves with a different feature set). MORF fully encapsulates this end-to-end pipeline for a given experiment and it makes it available for modification to any other researcher.

Meta-Analysis: While meta-analyses are common in fields with robust empirical research bases, such analyses have been less common in the field of machine learning, which has an emphasis on novelty. The open availability of executable machine learning experiments affords detailed meta-analyses by providing complete results of all modeling stages for meta-analysis.

7. CONCLUSION

Further attention and analysis should be dedicated to replication in the field of educational data mining, and specifically to MOOC dropout prediction. This work proposes a paradigm of end-to-end reproducibility using the MOOC Replication Framework. Our case study in replication using MORF demonstrates the insights that can be gained from replication studies in EDM, and the importance of factors related to experimental design.

8. ACKNOWLEDGEMENTS

An earlier version of this work was presented at the Workshop on Reproducibility in Machine Learning at the 2018 International Conference on Machine Learning. We would like to thank the workshop organizers and participants for useful feedback on this work. We would also like to acknowledge the helpful assistance of the original authors of [13] in conducting this replication.

9. REFERENCES

[1] E. Alpaydin. *Introduction to Machine Learning*. MIT Press, Aug. 2014.

[2] J. M. L. Andres, R. S. Baker, G. Siemens, D. Gašević, and S. Crossley. Studying MOOC completion at scale using the MOOC replication framework. In *Proc. LAK*, pages 71–78, New York, Mar. 2018. ACM.

[3] C. Boettiger. An introduction to docker for reproducible research. *Oper. Syst. Rev.*, 49(1):71–79, Jan. 2015.

[4] S. Boyer and K. Veeramachaneni. Transfer learning for predictive models in massive open online courses. In *Proc. AIED*, pages 54–63. Springer, Cham, June 2015.

[5] A. Brinckman et al. Computing environments for reproducibility: Capturing the "whole tale". *Future Gener. Comput. Syst.*, Feb. 2018.

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI gym. *arXiv preprint*, June 2016.

[7] J. B. Buckheit and D. L. Donoho. WaveLab and reproducible research. In A. Antoniadis and G. Oppenheim, editors, *Wavelets and Statistics*, pages 55–81. Springer New York, New York, NY, 1995.

[8] G. C. Cawley and N. L. C. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *J. Mach. Learn. Res.*, 11(Jul):2079–2107, 2010.

[9] J. Cito, V. Ferme, and H. C. Gall. Using docker containers to improve reproducibility in software and web engineering research. In *Web Engineering*, Lecture Notes in Computer Science, pages 609–612. Springer, Cham, June 2016.

[10] C. Collberg, T. Proebsting, G. Moraila, A. Shankaran, Z. Shi, and A. M. Warren. Measuring reproducibility in computer systems research. Technical report, Univ. Arizona Dept. of Comp. Sci., 2014.

[11] Coursera. *Coursera Data Export Procedures*. Coursera, June 2013.

[12] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput.*, 10(7):1895–1923, Sept. 1998.

[13] M. Fei and D. Y. Yeung. Temporal models for predicting student dropout in massive open online courses. In *Proc. ICDMW*, pages 256–263, 2015.

[14] J. Gardner and C. Brooks. Evaluating predictive models of student success: Closing the methodological gap. *Journal of Learning Analytics*, 5(2):105–125, 2018.

[15] J. Gardner and C. Brooks. Student success prediction in MOOCs. *User Modeling and User-Adapted Interaction*, 2018.

[16] J. Gardner, C. Brooks, J. M. Andres, and R. Baker. Replicating MOOC predictive models at scale. In *Proc. ACM Learning@Scale*, 2018.

[17] J. Gardner, C. Brooks, J. M. Andres, and R. S. Baker. MORF: A framework for predictive modeling and replication at scale with Privacy-Restricted MOOC data. In *IEEE Intl. Conf. on Big Data*, pages 3235–3244, Dec. 2018.

[18] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press.

[19] O. E. Gundersen and S. Kjensmo. State of the art: Reproducibility in artificial intelligence. In *Proc. AAAI*, 2017.

[20] M. A. HarvardX. HarvardX-MITx Person-Course academic year 2013 De-Identified dataset, version 2.0, May 2014.

[21] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *Proc. AAAI*, pages 3207–3214, 2018.

[22] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. Feb. 2015.

[23] D. M. Jacobsen and R. S. Canon. Contain this, unleashing docker for hpc. *Proc. Cray User Group*, 2015.

[24] M. Khajah, R. V. Lindsey, and M. C. Mozer. How deep is knowledge tracing? In *Proc. EDM*, pages 94–101, 2016.

[25] J. Kitzes, D. Turek, and F. Deniz. *The Practice of Reproducible Research: Case Studies and Lessons from the Data-Intensive Sciences*. Univ of California Press, Oct. 2017.

[26] K. R. Koedinger, R. S. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper. A data repository for the EDM community: The PSLC DataShop. *Handbook of educational data mining*, 43, 2010.

[27] D. Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239), Mar. 2014.

[28] B. K. Olorisade, P. Brereton, and P. Andras. Reproducibility in machine Learning-Based studies: An example of text mining. In *ICML Reproducibility in ML Workshop*, 2017.

[29] Z. A. Pardos, S. Tang, D. Davis, and C. V. Le. Enabling Real-Time adaptivity in MOOCs with a personalized Next-Step recommendation framework. In *Proc. ACM Learning@Scale*, pages 23–32, 2017.

[30] R. D. Peng. Reproducible research in computational science. *Science*, 334(6060):1226–1227, Dec. 2011.

[31] C. Piech et al. Deep knowledge tracing. In *NIPS 28*, pages 505–513, 2015.

[32] D. Sculley, J. Snoek, A. Wiltschko, and A. Rahimi. Winner's curse? on pace, progress, and empirical rigor. In *ICLR 2018 Workshops*, Feb. 2018.

[33] S. Sonnenburg et al. The need for open source software in machine learning. *J. Mach. Learn. Res.*, 8(Oct):2443–2466, 2007.

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

[35] J. N. van Rijn et al. OpenML: A collaborative science platform. In *Machine Learning and Knowledge Discovery in Databases*, pages 645–649. Springer Berlin Heidelberg, 2013.

[36] S. Varma and R. Simon. Bias in error estimation when using cross-validation for model selection. *BMC Bioinformatics*, 7:91, Feb. 2006.

[37] J. Whitehill, K. Mohan, D. Seaton, Y. Rosen, and D. Tingley. Delving deeper into MOOC student dropout prediction. *arXiv preprint*, Feb. 2017.