nature protocols

PROTOCOL

https://doi.org/10.1038/s41596-021-00513-5

Check for updates

Use of deep learning to develop continuous-risk models for adverse event prediction from electronic health records

Nenad Tomašev[1]✉, Natalie Harris[2], Sebastien Baur[2], Anne Mottram[1], Xavier Glorot[1], Jack W. Rae[1,3], Michal Zielinski[1], Harry Askham[1], Andre Saraiva[1], Valerio Magliulo[2], Clemens Meyer[1], Suman Ravuri[1], Ivan Protsyuk[2], Alistair Connell[2], Cían O. Hughes[2], Alan Karthikesalingam[2], Julien Cornebise[1,12], Hugh Montgomery[4], Geraint Rees[5], Chris Laing[6], Clifton R. Baker[7], Thomas F. Osborne[8,9], Ruth Reeves[7], Demis Hassabis[1], Dominic King[2], Mustafa Suleyman[1], Trevor Back[1], Christopher Nielson[7,10,13], Martin G. Seneviratne[2,13]✉, Joseph R. Ledsam[1,2,11,13]✉ and Shakir Mohamed[1,13]

Early prediction of patient outcomes is important for targeting preventive care. This protocol describes a practical workflow for developing deep-learning risk models that can predict various clinical and operational outcomes from structured electronic health record (EHR) data. The protocol comprises five main stages: formal problem definition, data pre-processing, architecture selection, calibration and uncertainty, and generalizability evaluation. We have applied the workflow to four endpoints (acute kidney injury, mortality, length of stay and 30-day hospital readmission). The workflow can enable continuous (e.g., triggered every 6 h) and static (e.g., triggered at 24 h after admission) predictions. We also provide an open-source codebase that illustrates some key principles in EHR modeling. This protocol can be used by interdisciplinary teams with programming and clinical expertise to build deep-learning prediction models with alternate data sources and prediction tasks.

Introduction

Early prediction of patient outcomes can support data-driven health care, helping to target preventive interventions and guide resource allocation. Rule-based scores are common in clinical practice for risk stratification, for example, the National Early Warning Score for acute deterioration or the LACE score for hospital readmission[1,2]. These scores are typically not personalized to the patient (the same thresholds and coefficients are used population wide), rarely use temporal or trend information, have a relatively limited input feature space and have been associated with high false-positive rates and consequent alert fatigue[3].

Running machine-learning (ML) models on continuously streamed electronic health record (EHR) data could help to tackle some of the above limitations. ML models have been developed to predict a wide array of clinical and operational outcomes, to risk-stratify patients and inform treatment decisions. Use cases have ranged from mortality[4] and length of stay (LoS) prediction[5] to more targeted algorithms for sepsis[6], shock[7] or delirium[8]. Recent studies have increasingly used deep-learning approaches[9–12]. To date, few of these models have been validated prospectively and adopted in routine practice[13]. However deployment studies are now being undertaken[14]. To support more widespread clinical implementation, it is critical that upstream data science pipelines are robust and well specified.

In this protocol, we outline a practical workflow for developing deep-learning models with structured EHR data (excluding free-text clinical notes), with a focus on continuous risk models for

[1]DeepMind, London, UK. [2]Google Health, London, UK. [3]CoMPLEX, Computer Science, University College London, London, UK. [4]Institute for Human Health and Performance, University College London, London, UK. [5]Institute of Cognitive Neuroscience, University College London, London, UK. [6]University College London Hospitals, London, UK. [7]Department of Veterans Affairs, Washington, DC, USA. [8]VA Palo Alto Healthcare System, Palo Alto, CA, USA. [9]Stanford University School of Medicine, Stanford, CA, USA. [10]University of Nevada School of Medicine, Las Vegas, NV, USA. [11]Google, Tokyo, Japan. [12]Present address: University College London, London, UK. [13]These authors contributed equally: Christopher Nielson, Martin G. Seneviratne, Joseph R. Ledsam, Shakir Mohamed. ✉e-mail: nenadt@google.com; martsen@google.com; jledsam@google.com

inpatient endpoints. We developed this workflow while establishing an acute kidney injury (AKI) prediction model, introduced in Tomasev et al.[15]. The prediction of AKI is included here as the primary exemplar. However, the protocol can be applied to a wide range of clinical or operational use cases.

Overview of the protocol

The protocol involves five broad stages: (a) formal problem definition, (b) data pre-processing, (c) architecture selection, (d) calibration and uncertainty estimation and (e) model generalizability evaluation. The components of the protocol with greatest novelty include multitasking with physiological auxiliaries (Step 6), separate training and evaluation masks (Step 16), architecture ablation (Step 22), calibration and uncertainty estimation (Steps 26–28), clinically motivated operating points (Step 29) and temporal/regional generalizability evaluations (Steps 32 and 33). The protocol is described with reference to the AKI use case but is intended to be applicable to a range of prediction targets. In 'Anticipated results', we present performance metrics on three additional endpoints: mortality, LoS and readmission. These endpoints were chosen for this methodological work because they could be reliably timestamped in the current dataset, and numerous ML benchmarks were available in the literature. We also present results for alternate temporal configurations including different triggering schemes (i.e., static predictions versus continuous or regularly triggered predictions) and different lookahead windows. The protocol could feasibly be generalized to other targets over acute and chronic timescales, such as sepsis, intensive care unit transfer or chronic disease progression.

It should be noted that the protocol allows only for associative modeling between input features and outcome targets. Although not addressed here, causal inference (i.e., whether specific features directly cause a particular outcome) using observational data is an active area of research that stands to assist in knowledge discovery, robustness and fairness[16].

The protocol is accompanied by an open-source codebase and synthetic dataset illustrating the pre-processing stages (available at https://github.com/google/ehr-predictions). Although customization is required to use this protocol on new datasets and tasks, we believe this is a useful high-level framework that showcases some of the nuances of supervised learning with EHRs.

Expertise needed to implement the protocol

Successful application of the protocol requires interdisciplinary collaboration. Some steps call for significant technical expertise in deep learning and should be executed by researchers with statistical and ML skills, whereas others require clinical or operational knowledge and should be executed by clinicians or informaticians. Prospective evaluation calls for experts in trial design and outcomes evaluation. As a guide, Steps 1–6, 8, 15, 16, 24, 29 and 30 are likely to be clinician led, whereas Steps 7, 9–14, 17–23, 25–28 and 31–33 are likely to be engineer led. In Box 1, we provide a glossary of some of the terms used in this protocol to facilitate understanding across disciplines.

Dataset requirements and limitations

Implementation of this workflow requires access to an EHR dataset. Here, we use an EHR dataset from the US Department of Veterans Affairs (VA) prepared for our previous work on AKI prediction[15]. The dataset consisted of de-identified longitudinal data on 703,782 adult patients across 172 inpatient and 1,062 outpatient sites collected using the Vista EHR system and associated databases. Inclusion criteria were patients aged between 18 and 90 years admitted for secondary care to medical or surgical services from the beginning of October 2011 to the end of September 2015. To protect patient privacy, sites with fewer than 250 admissions during the 5-year time period were excluded; 4 of the 1,243 health care facilities from which the VA is composed were excluded on the basis of this criterion. All exclusion criteria were established before beginning the work. The actual dataset size required for each specific application is dependent on many factors, including the patient cohort, input feature space present and the endpoint of interest; for further details on power calculations for clinical prediction models, we refer the reader to Riley et al.[17].

Because of the patient population of the VA, the dataset consisted of 93.6% male subjects. The dataset included laboratory tests, vital signs, medications, admissions, transfers, outpatient visits, diagnoses as International Classification of Diseases, Ninth Revision (ICD-9) codes and procedures as Current Procedural Terminology (CPT) codes.

Use of this retrospective dataset received Tennessee Valley Healthcare System Institutional Review Board approval from the VA. All patient data were de-identified in accordance with the Health

Box 1 | Glossary

Ablation	The process of removing components of the model architecture and retraining to quantify the performance attributable to that component
AUPRC	For a classification model, the area under the precision versus recall curve. Scalar value between 0 and 1, with higher values representing superior performance. Useful in situations of class imbalance (between positive and negative endpoint labels)
AUROC	For a classification model, the area under the true positive versus false positive rates. Scalar value between 0 and 1. It can be interpreted as the probability of giving a higher risk to a positive sample than a negative sample
Auxiliary target	A label that is related to the primary outcome. It is used as an extra training signal and is designed to improve the prediction performance of the model on the primary target
Baseline model	A simpler model that is used to compare the performance of the deep model, typically, a logistic regression or tree-based model
Calibration split	The subset of patients used for measuring and correcting the calibration of a trained model. A well-calibrated model is one in which the predicted risk matches the incidence of the outcome (e.g., 40% of patients with a 0.40 risk of AKI should develop an AKI)
Classification	Supervised learning problem where the goal is to assign a discrete label to the input (e.g., here we set up adverse event prediction as a binary classification)
Encoder (neural network)	A neural network whose purpose is to map the original input to a Euclidean space in a compact way. It can be seen as a form of compression
Feature set	All the input information that is fed to the model during training and evaluation
Generalizability	The ability of the model to preserve performance on datasets different from the training data (e.g., from other sites (cross-site generalizability) or other time periods (temporal generalizability))
Hyperparameter	A parameter that is used for controlling the learning process (as opposed to learned during that process) (e.g., the learning rate of the stochastic gradient descent optimizer, the size of the mini-batches and the number of layers of the neural network)
Imputation	The process of replacing missing values with plausible values. In our work, we use 0 imputation (i.e., all missing values are set to 0), but more complex schemes can be used
Inference	The process of triggering a model prediction (e.g., the risk of an adverse event over the next *x* hours for a given patient)
Label leakage	When information about the outcome label has been inappropriately included in the input feature set (e.g., because of inaccurate timestamping of a certain feature that would not have been available to the clinician until later)
Lookahead horizon	The clinically informed time window that is used for making continuous predictions. This is used to define positive and negative labels at each point in time
Numerical feature	A feature that takes continuous values (e.g., vital signs and laboratory measurements)
Operating point	The risk threshold that is applied to the output of the classification model (a value between 0 and 1) to define what a positive prediction is. If the predicted risk is higher than that threshold, the binary outcome is set to be 1, and 0 otherwise
PR curve	For a classification model outputting a continuous value between 0 and 1 (the risk or probability), the PR curve is a graphical plot that displays the precision (positive predictive value) versus the recall (sensitivity). Each point on the curve corresponds to a different threshold for the predicted risk
Prediction head	The output of the last layer of a neural network model. There may contain one or more prediction heads
Presence feature	A feature that takes binary values. In addition to encoding categorical features, presence features can be used to encode missingness of numerical features
Recurrent neural network	A neural network that is tailored for making predictions on temporal/sequential data
Regression	Supervised learning problem where the goal is to assign a continuous value to the input (e.g., predicting the value of a specific laboratory test)
ROC curve	For a classification model outputting a continuous value between 0 and 1 (the risk or probability), the ROC curve is a graphical plot that displays the true positive versus false positive rates. Each point on the curve corresponds to a different threshold for the predicted risk
Test split	The subset of patients used for reporting the final performance of the model. This is disjoint from all other splits
Training split	The subset of patients used for learning the model's weights by optimizing the loss function
Triggering frequency	How often a model 'fires' (i.e., triggers inference) for a given patient (e.g., hourly)
Validation split	The subset of patients used for evaluating the performance of the model during training, for the purpose of comparing architectures and selecting hyperparameters

Insurance Portability and Accountability Act. Additional precautions were taken to safeguard patient privacy: free text notes and rare diagnoses were excluded, many feature names were obfuscated (i.e., the feature value was preserved, but the name was replaced with a random string), continuous variables were randomly jittered up to a maximum value of 10% of the population standard deviation and all patient records were time-jittered, respecting relative temporal relationships for individual patients.

Our dataset has limitations. The selection of auxiliary tasks and the input features for baseline models were constrained by the panel of features available in the source EHR. Our model performance was comparable with literature benchmarks; however, these models should be considered as prototypes for demonstrating methods rather than as clinical-grade ML systems suitable for deployment.

Although the dataset was drawn from a diverse range of health facilities and geographies within the VA network, spanning >5 years, it was limited by a lack of gender diversity because of the predominantly male patient population. Furthermore, the protocol does not explicitly deal with fairness evaluation. Principles of ML fairness should infuse the entire protocol from problem conception and dataset choice to evaluation strategy and deployment. We refer the reader to more detailed discussions of fairness considerations[18] and guidelines on transparent reporting[19].

Our workflow assumes that the raw EHR data have been extracted in tabular format from a research data warehouse. Although we provide exemplar data at different stages of pre-processing, this is not intended as a canonical data representation. Standard EHR data models (e.g., Fast Healthcare Interoperability Resources (FHIR) or the Observational Medical Outcomes Partnership (OMOP) data model) can be used to construct input feature vectors or embedded as a vector representation as per Rajkomar et al.[20].

We did not use the unstructured text present in the EHR because of privacy concerns. This is an important part of the information content present in EHR data and may improve model performance for certain clinical tasks. Researchers wishing to embed clinical notes in their models are recommended to consult the following references and consider pretraining approaches[21–23].

Comparison with other protocols

Several guidelines for developing ML models with healthcare data have recently been published[24–28]. These cover high-level principles such as problem selection, fairness, bias, model surveillance and outcome evaluation across various data modalities. Our protocol details a practical methodology for developing deep-learning models tailored to longitudinal EHR data, with guidance around data pre-processing, architecture selection, hyperparameter sweeps, post-processing and evaluation.

In addition, there are initiatives to create standardized reporting guidelines for ML studies using clinical data, including the TRIPOD-ML[29] and STARD-AI frameworks[30], which should be referenced early in study design. It is also worth noting the recently released guidelines for reporting on clinical trials of ML systems, including CONSORT-AI[31] and SPIRIT-AI[32].

There is a wealth of recent work using deep learning for predictive modeling with EHRs[10,20,33–38]. Although direct comparison of model performance against these prior works is challenging because different datasets are used and experimental setups for the same task can differ (in terms of endpoint definition, lookahead window, triggering frequency, etc.), in 'Anticipated results' we endeavor to compare our results for mortality, LoS and readmission with previous ML benchmarks.

Materials

Training dataset

A structured EHR data set was required for this work **! CAUTION** It is important to ensure that ethical approval has been obtained before using EHR data, including consideration of implied or explicit patient consents. The example results shown here were derived from an EHR dataset from the VA. We obtained permission to use this for these studies from the Tennessee Valley Healthcare System Institutional Review Board.

Hardware

The protocol and accompanying codebase can be executed on standard computational hardware. There are minimal memory requirements for data storage and pre-processing that depend on the size of the dataset. Model training can be run on central processing units or graphics processing units; having access to greater computational resources would allow faster execution and wider exploration of hyperparameters.

Software

- Data processing framework: Apache Beam (https://beam.apache.org/)
- Plotting library: Matplotlib (https://matplotlib.org/)[39]

- Scientific computing library: Numpy (https://www.numpy.org/)
- Scientific computing library: Scipy (https://www.scipy.org/)
- Plotting library: Seaborn (https://seaborn.pydata.org/)
- Machine learning library: Scikit-learn (https://scikit-learn.org/)[40]
- Machine learning library: Sonnet (https://github.com/deepmind/sonnet/)
- Machine learning framework: TensorFlow (https://github.com/tensorflow/tensorflow/)
- Machine learning library: XGBoost (https://github.com/tensorflow/tensorflow/)

Procedure

Formal problem definition

1 *Evaluate the feasibility of a clinical use case*. Ensure that the selected clinical use case is feasible by checking that the following basic conditions are met: (i) there is a computable definition of the target outcome or sufficiently large manually labeled training dataset, (ii) there is a predictive signal in routinely collected structured EHR data (consultation with specialist clinicians may help identify such signals) and (iii) there is sufficient temporal resolution in the EHR for the prediction to be actionable (e.g., weekly aggregate data are not appropriate for a high-frequency prediction that is most useful the day before the outcome). Clinician and patient engagement, through steering groups and/or structured interviews, can help refine the research question and map out real-world clinical pathways[41]. This ensures that the outputs of the model can be integrated into existing clinical workflows. It is also important to identify who the likely end users will be and thus the most appropriate potential channels for model output (e.g., via interruptive versus non-interruptive alerts[42]).

2 *Define outcome labels*. Identify an appropriate ground truth outcome label for supervised learning. One approach is to use e-alerting criteria, such as the National Health Service AKI e-alert[43] or the St John Sepsis Agent[44], which use data available only up to the time of the trigger. Another is to use data from the entire admission to timestamp outcomes of interest, such as using downstream outcomes to identify the most severe cases of AKI or sepsis. Outcomes may also be defined on the basis of clinician actions (e.g., sepsis definitions based on the collection of a blood culture or antibiotic prescriptions); however, note that this may introduce biases and encourage the model to predict outcomes only on those patients who have historically been investigated/treated. The gold standard method to define outcome labels is manual chart review, which may be used in conjunction with one or more of the above methods to validate the labeling approach. One major pitfall is label leakage, where explicit indications of the outcome label are present at an earlier timepoint. Some studies suggest enforcing a gap time between the outcome label and the prediction trigger to reduce this risk[45]. In some cases, the width of the timesteps may have the effect of introducing a time buffer (although the gap is variable); e.g., in Tomasev et al.[15], all entries in the same 6-h bucket as the outcome label were excluded from the model input.

3 *Assess dataset quality*. Produce a formal dataset specification with descriptive statistics about each data element, including outliers and degree of missingness. Assess the distributions of vital signs and laboratory tests and compare against known physiological ranges. Harmonize admission records on the basis of LoS to concatenate overlapping admissions. Assess and report on the demographic diversity of the dataset. Consider using mitigation strategies to address data imbalance (e.g., oversampling/undersampling). Compile a random sample of the data for manual assessment by clinical experts, with particular focus on the fidelity of the outcome label.

4 *Define inclusion and exclusion criteria*. In preparation for future prospective deployment, define inclusion/exclusion criteria on the basis of baseline criteria available from the point where model inference begins, rather than retrospective criteria such as percent missingness. Consider factors that are dependent on the patient, such as demographics, as well as environmental factors such as clinical setting or ward.

5 *Define time formulation*. Define the trigger time(s) and lookahead window(s) for prediction tasks (Fig. 1). The trigger time refers to when, during an admission, inference will be performed—this may be a single static prediction (e.g., at 24 h after admission), continuous predictions (e.g., triggered on an hourly basis) or dynamic predictions triggered when some criteria are satisfied (e.g., when vital signs are out of range). The lookahead window is the time interval after the trigger time in which the endpoints are defined. For AKI prediction, we trialed lookahead windows ranging from 6 to 72 h in 6-h increments. The trigger time and lookahead window should be guided by domain knowledge about when early clinical markers may manifest and the window

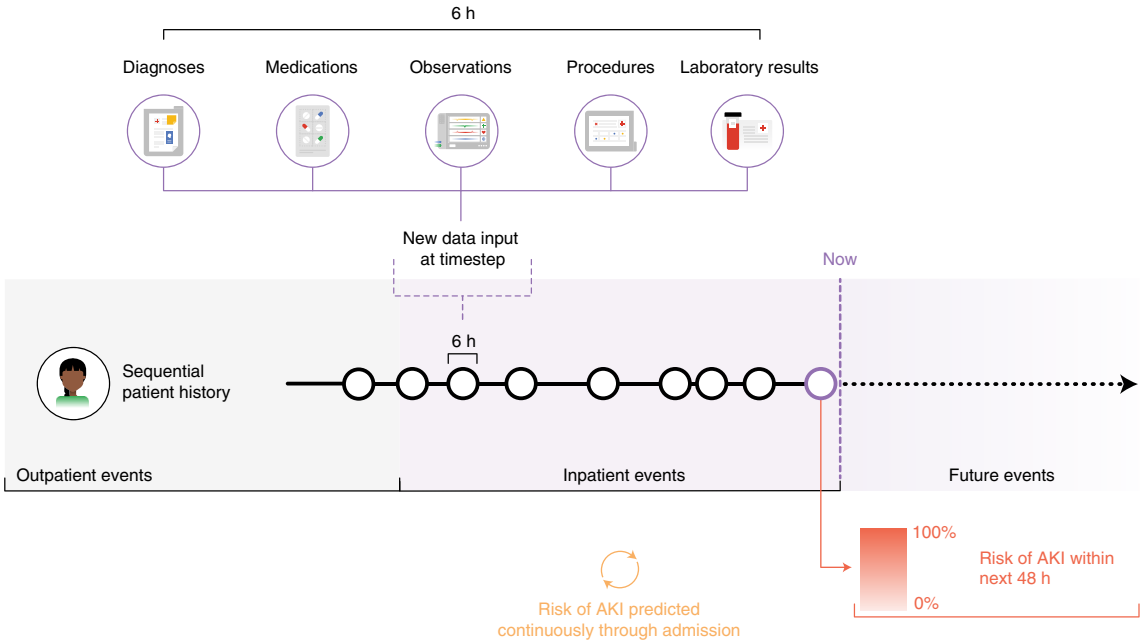

Fig. 1 | Sequential risk prediction from EHR data. Rolling predictions using structured EHR data, here illustrating the prediction of AKI within 48 h at a 6-hourly triggering frequency.

within which a prediction is clinically actionable. The interventions for AKI include medication review, fluid management, septic workup, etc., all of which may be effective in the 48 h before AKI onset[46]. For mortality prediction, longer lookaheads (30 or 90 d) can be considered, which may be appropriate for sub-acute decisions about the limits of care and when to refer for palliative services[38].

6 *Identify auxiliary prediction targets*. Auxiliary prediction targets can help to improve model performance on the primary prediction task, because concurrently learning multiple clinically related endpoints may lead to a better internal data representation. The choice of auxiliary tasks and how to optimize across the losses (also known as multitasking) are topics of active research[33,47,48]. At a high level, the goal is to identify physiological observations directly related to the primary clinical endpoint. In the case of AKI, we used the maximum values (across the same set of lookahead windows as for the primary AKI endpoint) of seven relevant laboratory tests associated with renal function: creatinine, urea nitrogen, sodium, potassium, chloride, calcium and phosphate. The maximum value was chosen because increased levels of most of these electrolytes are commonly associated with AKI. The error derived from regressing these auxiliary targets is combined with the loss associated with the main training objective at each timestep to update the weights of the neural network. Auxiliary prediction tasks can also be used to model competing risks, as well as regularize the model and help with the explainability of predictions. The panel of auxiliary tasks can be refined during training on the basis of the improvement in performance on the validation set.

Data pre-processing

7 *Create data splits*. Partition the dataset by randomly allocating records into the following splits: training, validation, calibration and test. It is common practice to allocate all records for each individual patient to separate data splits, in order to test for generalizability across unseen patients and to avoid information leakage. Alternative options include splitting the sequences of events across time and ordering the splits chronologically. This helps assess the generalizability of the models with respect to the non-stationarity of hospital processes and potential changes in clinical practice across time. The minimum size of each split needs to be sufficient to derive valid statistical conclusions and should be based on an appropriate power calculation. Assigning 80% of the data to the training split, 5% to the validation split, 5% to the calibration split and 10% to the test split is a reasonable choice for sufficiently powered datasets and was used to develop the model presented in Tomasev et al.[15]. This enables just the training set to be used for model development (Steps 8–25),

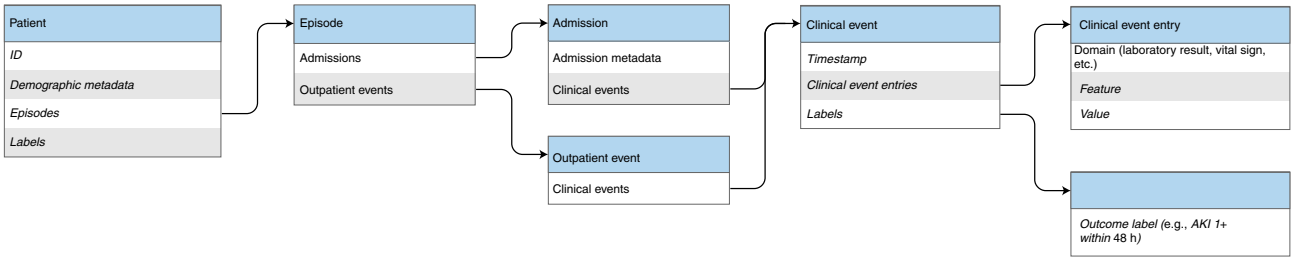

Fig. 2 | Sequential representation. The sequential representation consists of a sequence of clinical event entries grouped into clinical events for each time bucket, which in turn are grouped by episode (admissions versus outpatient events). Outcome labels are a derived data field based on the triggering frequency and prediction window.

with the test set only used for analysis when the final model parameters are fixed. A calibration split is especially critical if the model is going to be used as a risk score or for continuous alerting (see Step 26).

8 *Feature engineering*. Seek input from clinicians and informaticians familiar with the source data to identify which features to use. Appropriate rationales for eliminating features include poor data quality and features overly specific to a particular site, although the research decisions on which features can be removed will be specific to every project. Identify a set of manually defined features that may hold predictive value for the outcome of interest (e.g., clinically relevant ratios such as the ratio of blood urea nitrogen to serum creatinine). Manually engineered features are most important for the baseline models against which the deep models will be evaluated. The feature engineering pipeline should be defined using the training set only: estimating feature statistics from the entire dataset risks leaking information from the test set, and because the choice of features may influence model performance, it is important that the test set does not influence this decision.

9 *Generate a sequential representation of patient data*. First, define the length of the discrete time window (timestep size). We set this to 6 h in Tomasev et al.[15]; however, this can be made shorter or longer (e.g., minutes or days) depending on the granularity of the data and triggering frequency of the prediction. Note that the timestep size must be less than or equal to the triggering frequency. There is a trade-off to be made in selecting the timestep size: short timesteps risk being empty because of irregular sampling of EHR entries; however, longer timesteps may lead to loss of temporal information because the ordering of events is not preserved within each step. Repeated values for a given feature within a time bucket must be aggregated—typically using the mean or median, but other aggregation functions are valid. For entries where the timestamp is unknown, use a surrogate bucket; for example, many EHR events may be associated with a day but no specific timestamp and thus can be grouped into a surrogate bucket. This bucket is assigned to the end of that day to prevent leakage of future information in previous buckets. For empty timesteps during intervals where inference must be regularly triggered (e.g., during an inpatient admission), include an empty set. Finally, concatenate the entire patient record into a sequential representation running from the first to the last available data point, organized into distinct clinical events within inpatient and outpatient episodes (Fig. 2). Labels for the primary outcome and auxiliary targets should be appended at each timestep.

10 *Clean EHR timestamps*. Check whether entries have a discrepancy between the EHR timestamp and the true availability of the data to clinicians. In the raw dataset, for example, diagnosis codes were uniformly timestamped to be available at the beginning of admission even though the actual diagnosis was often assigned at a different time (typically at the time of discharge). Although this granularity was not available in our research dataset, it might be possible to delineate several important timestamps for each entry; for example, a laboratory test might have timestamps for when the order was placed, when the sample was collected and when the result was visible in the EHR. It may be worth encoding each of these timesteps as distinct events or only using the lattermost timestamp. Where there is ambiguity around timestamps, move the relevant entries to the end of the relevant episode to avoid information leakage.

11 *Aggregate historical features*. Compute historical aggregates for a set of important features to include in the input for baseline (non-recurrent) models. Which historical features to include and the duration of past medical history (lookback window) will depend on the primary outcome. Define a set of statistical functions for feature aggregation (e.g., count, mean, median, standard deviation, minimum, maximum and average difference between subsequent measurements). These

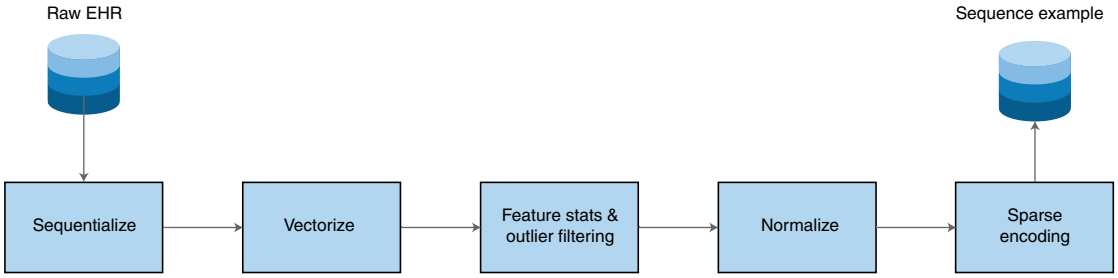

Fig. 3 | Pre-processing workflow. Conversion of raw EHR data into a sequential representation, followed by vectorization, normalization and sparse encoding.

must be treated as distinct features leading to a dimensionality increase. For non-numerical features, record a binary flag for whether they were present in the lookback window. For the AKI model, we used 48 h for acute trends, and chronic trends were captured by considering 6-month, 1-year or 5-year aggregates.

12 *Vectorize the event sequence*. Vectorization refers to transforming the sequential data representation into a feature vector appropriate for model input at each timestep. Because there can be valuable information in the pattern of missing data, a useful strategy for continuous features is to explicitly encode binary indicator variables (presence features) to enable the model to distinguish between a missing value in that timestep and a numerical value of zero[49]. Although zero imputation in conjunction with presence features was used in Tomasev et al.[15], there are numerous imputation strategies available for missing data, including carry-forward, mean/median and physiological reference imputation, as well as more advanced methods that have shown promise in EHR time series, including multivariate imputation by chained equations[50], Gaussian processes[51] and generative adversarial networks[52]. Continuous features may also be associated with additional indicator variables denoting whether the value is high/normal/low based on local reference ranges. Represent categorical features as presence features using one-hot encoding. The total number of variables in this work was 620,000, of which ~165,000 were used in the final vectorized representation; 4% were numerical, and the remainder were presence features.

13 *Cap outlier numerical values*. For every input feature, cap values at the the 1st and 99th percentiles on the basis of the training split (or appropriate maximum/minimum bounds guided by clinician input). This is important because data entry errors can be present, which result in physiologically implausible data being present in the data set. An example of an extreme outlier is an age >150 years.

14 *Normalization and sparse encoding*. To improve convergence speed, normalize the capped numerical input features to unit range or standardize to unit variance. Both approaches yielded similar results in Tomasev et al.[15], and we would recommend experimenting with both methods. Sparse encoding allows for a more efficient data representation of the sparse EHR feature space where only the explicitly non-zero values are represented. The sparse tensor consists of separate dense tensors denoting indices, feature values and the original dense shape. This can then be converted to the required sequence example format for model input (Fig. 3).

15 *Select performance metrics*. Define a set of relevant metrics for the primary use case as well as the auxiliary prediction targets. Select (i) development metrics to be used for architecture selection and (ii) final evaluation metrics to report. For classification tasks, use both the area under the precision-recall curve (AUPRC) and the area under the receiver operating characteristic curve (AUROC) in model development. Although the AUROC is ubiquitous and is thus important for comparison with prior work, AUPRC is better suited to class imbalances[53] and we recommend using AUPRC for model selection. Early prediction histograms are also valuable for fixed-window prediction tasks to demonstrate the latency between prediction and outcome (see Fig. 3 in Tomasev et al.[15]). Time-to-event or survival modeling is an alternate approach that may be used for continuous prediction tasks, for which there are emerging deep-learning formulations[54]. For all evaluation metrics in continuous predictions tasks, there is an important distinction between timestep-level metrics and outcome-level metrics. For example, the timestep precision and recall can be calculated by averaging the performance across all timesteps for which the model is being evaluated; however, we can also calculate an outcome-level recall by examining what percentage of the outcomes (e.g., AKI episodes) have at least one correct prediction within a 48-h window preceding onset. Depending on

the clinical scenario, it may be acceptable to introduce tolerance in evaluation (e.g., accepting a positive prediction 48–60 h before AKI onset as a true positive (as opposed to a false positive under a strict 48-h lookahead)). However, the metrics reported in 'Anticipated results' were computed without a tolerance buffer, which provides a more conservative estimate of model performance.

16 *Interval censoring*. Define interval censoring masks for both model training and evaluation. A mask refers to a sequence of binary flags overlaid on the event sequence, which indicates whether a timestep is included in training or evaluation. For example, in the AKI prediction use case, patients undergoing dialysis were excluded from both training and testing splits using a number of procedure codes to define the mask. Importantly, training and evaluation masks may be different. For example, intervals where the patient had AKI were included in the training procedure because there are still valuable physiological relationships between this timestep and future creatinine values; however, they were excluded from evaluation because these timesteps would not be alerted on in practice. Adapting the training and evaluation masks can enable versatile experimental setups (e.g., predicting inpatient mortality only at points where the patient triggers a National Early Warning Score alert). After evaluation masks have been defined, it is possible to compute the outcome prevalence at a patient level (i.e., percentage of patients with AKI) and at a timestep level (i.e., percentage of timesteps with a positive label for AKI within 48 h). This class distribution should be reported alongside a model to contextualize the performance metrics.

Model architecture selection

17 *Train baseline models*. Train a panel of baseline models, such as logistic regression or XGBoost[55]. The choice of baselines should be motivated by prior literature benchmarks. For these models, a subset of clinically relevant and manually engineered features (Step 8) may be selected. Interrogating the coefficients and feature importances of baseline models can assist in identifying label leakage and guide redefinition of the outcome label if required. Confidence intervals for the performance metrics of baseline models can be calculated using a percentile bootstrap estimator[56].

18 *Feature embedding*. For each timestep, transform the sparse input tensors into a lower- dimensional continuous representation (i.e., embedding) that can be used as an input to the deep recurrent architecture. Multiply the sparse tensor by a lookup-embedding matrix that is randomly initialized. When multiple features are present at a given timestep, aggregate the lookup embeddings (we summed embeddings, but alternate aggregation functions can be used). Pass this to a multilayer perceptron (MLP)-embedding module with residual connections and L1 regularization to reduce overfitting. Sweep over a range of embedding sizes (we used a two-layer embedding module with size 400). In Tomasev et al.[15], there were separate embedding modules for different types of input features (numerical versus presence), and the outputs were then concatenated. Autoencoders (AEs) or variational AEs may be trialed, as these have shown promise in learning richer patient representations for predictive modeling[57].

19 *Trial multiple deep architectures*. Implement a range of recurrent (and optionally convolutional) frameworks that receive the feature embeddings and feed into the primary and auxiliary output heads. Make the frameworks configurable with respect to recurrent cell types and their parameters, as well as different types of convolutional kernels. The following are some of the recurrent neural network (RNN) cells that can be trialed: long short-term memory (LSTM)[58], update gate RNN[59], intersection RNN[59], simple recurrent unit[60,61], gated recurrent unit (GRU)[62], neural Turing machine (NTM)[63], memory-augmented neural network[64], differentiable neural computer (DNC)[65] and relational memory core[66]. Where there are multiple training heads (e.g., the primary output and the auxiliary tasks at various lookahead horizons), weights may be shared through the deep model, culminating in logistic layers specific to each task. Consider adding a cumulative distribution function to these logistic layers to encourage monotonicity in prediction outputs across overlapping lookahead horizons (e.g., the risk of AKI within 48 h should be greater than or equal to the risk within 24 h). A comparison of deep architectures and baseline models is shown in Supplementary Table 4 in Tomasev et al.[15]. The architecture used here (Fig. 4) is a three-layer LSTM with highway connections[67] followed by linear layers for the primary outcome (AKI) and auxiliary heads (laboratory test regressions).

20 *Set up the model optimizer*. By comparing the predicted output and the ground truth labels, compute a scalar loss value for each timestep. Next, compute scalar losses for each auxiliary task. In Tomasev et al.[15], the cross-entropy loss function (Bernoulli log-likelihood) was used for binary outcome prediction, and L1/L2 losses were used for the auxiliary laboratory test regressions.

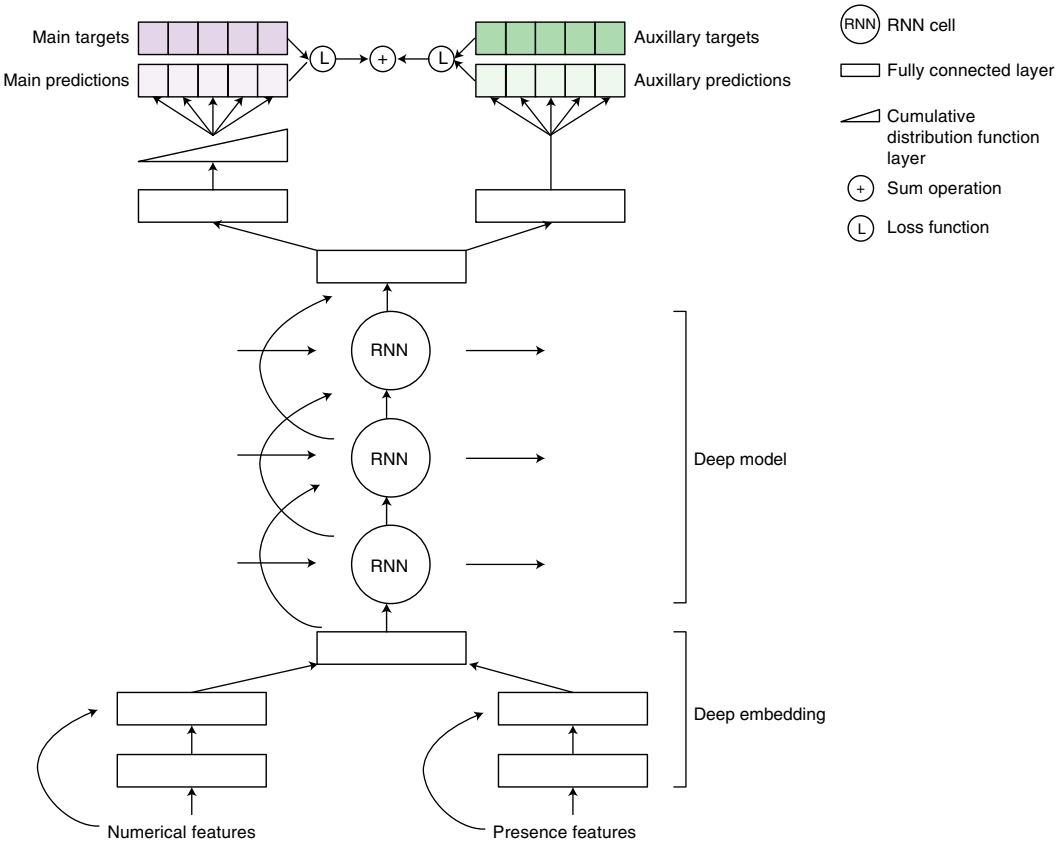

Fig. 4 | Deep recurrent architecture. Numerical and presence features are embedded in parallel, feeding into a multitask deep recurrent highway network architecture, with shared weights until final logistic layers for the primary versus auxiliary targets. The network was trained end-to-end including the embedding modules. Image reproduced from ref. [15].

Optionally, re-weight the loss to account for skew in the target distribution. Define a composite loss as a weighted sum of primary and auxiliary losses, plus regularization losses from the embedding and model. Use the computed loss alongside a mini-batch optimization algorithm (e.g., stochastic gradient descent, Adam or RMSProp) to iteratively adapt the weights of the neural network. Train using the training data split until convergence. Select the optimal learning regime (learning rate and decay) on the basis of hyperparameter sweeps (see Step 21).

21 *Run an iterative sequence of hyperparameter sweeps*. Define hyperparameter sweeps on the basis of domain knowledge and previous literature. As an example, see Table 1 for the ranges of hyperparameters that we tested. Initially perform hyperparameter sweeps without auxiliary tasks to find a performant set of hyperparameters for the main task; then, fine-tune the hyperparameters from that starting point, while executing sweeps to optimize the weight of the auxiliary loss. If target performance is not reached, revisit and expand earlier steps of data pre-processing and architecture selection. In each hyperparameter sweep, formulate a hypothesis for which architecture changes are likely to lead to performance improvements on the basis of ML expertise. For example, overfitting is a significant risk, especially when the feature space is sparse and high dimensional. Mitigating strategies include reducing the RNN cell size, increasing regularization, introducing drop-out or increasing the auxiliary loss weight. For each of the hyperparameter combinations, train a model on the training split and evaluate on the validation split (see Step 15), optimizing for both AUPRC and AUROC. Select the best performing configuration at the end of this process as the final model architecture at this stage.

22 *Perform ablation studies*. The purpose of ablation is to minimize model complexity while preserving performance. Take the final model architecture from the previous step and define a set of components to remove (i.e., ablate) to assess their individual contributions. For example, this can include the number of stacked layers in the deep model, additional feature types like the historical aggregates, regularization techniques and auxiliary prediction tasks. For each ablation experiment, train a new model on the training set. Next, evaluate each of the ablated models on the validation

Table 1 | Hyperparameter sweeps used for the AKI model

Hyperparameter	Values considered
RNN cell type	LSTM, gated recurrent unit, update gate RNN, simple recurrent unit, intersection RNN, memory-augmented neural network, NTM, DNC, relational memory core
RNN cell size	100, 150, 200, 250, 300, 400, 500
RNN num. layers	1, 2, 3
Embedding num. layers	1, 2, 3
Embedding dimension per feature type	200, 250, 300, 400, 500
Embedding combination	Concatenate, sum
Embedding architecture type	MLP, AE, variational AE
Embedding reconstruction loss weight	1×10^{-2}, 1×10^{-3}, 1×10^{-4}
Embedding reconstruction sampling ratio	1, 2, 5, 10
Optimize directly for AUPRC	On, off
Highway connections	On, off
Residual embedding connections	On, off
Input dropout	0, 0.1, 0.2, 0.3
Output dropout	0, 0.1, 0.2, 0.3
Embedding dropout	0, 0.1, 0.2, 0.3
Variational dropout	0, 0.1, 0.2, 0.3
Input regularization type	None, L1, L2
Input regularization term weight	1×10^{-3}, 1×10^{-4}, 1×10^{-5}
BPTT window	32, 64, 128, 256, 512
Embedding activation functions	Tanh, ReLU, Leaky ReLu, Swish, ELU, SELU, ELiSH, Hard ELiSH, Sigmoid, Hard Sigmoid
Auxiliary task loss weight	0, 0.1, 0.5, 1, 5, 10
Learning rate	1×10^{-2}, 1×10^{-3}, 1×10^{-4}, 1×10^{-5}
Learning rate decay scheduling	On, off
Learning rate decay num. steps	6,000; 8,000; 12,000; 15,000; 20,000
Learning rate decay base	0.7, 0.8, 0.85, 0.9, 0.95
Batch size	32, 64, 128, 256, 512
NTM/DNC memory capacity	64, 128, 256
NTM/DNC memory word size	16, 32, 64
NTM/DNC memory num. reads	6, 10
NTM/DNC memory num. writes	1, 2, 3

BPTT, backpropagation through time; num., number.

set. To test for statistical significance, train a collection of ablation models and calculate confidence intervals on the average performance using bootstrapping (see Step 28). If any of the ablated models perform at least as well as the more complex final model, modify the architecture accordingly to favor the simpler version. Repeat this process until the model can no longer be simplified without significant loss of performance. Ablation results for AKI can be seen in Supplementary Tables 10 and 11 of Tomasev et al.[15].

23 *Compute feature saliency*. Estimating the contribution of individual features can aid in understanding what the model is learning, and can also be useful for quality assurance to protect against label leakage and spurious correlations. In Tomasev et al.[15] and 'Anticipated results' here, we performed occlusion analysis that estimates a feature's contribution by evaluating the change in predicted risk when that feature is individually occluded[68]. The occlusion process is similar to replacing a feature with a baseline value[69]. A feature was occluded by setting both its numerical value and associated presence feature to 0; that is, we define the baseline as an absent feature. Feature attribution can then be evaluated by averaging the deviation in predicted risk under occlusion over an interval for a single patient (local saliency) or over all timesteps for the entire

cohort (global saliency). Please note that feature saliency techniques were not tailored for multivariate heterogeneous time series and we advise caution in their use.

24 *Failure case analysis*. Compute timestep-level and outcome-level metrics for all subjects in the validation set. Compile a set of representative success and failure cases (see Supplementary Material in Tomasev et al.[15] for example plots). Cases should be evaluated on the basis of discriminative performance (was the ground-truth label accurate?), as well as actionability (could this alert have affected the clinical trajectory of this patient?). Detailed case review can be targeted toward certain clinical subgroups or cohorts where model performance is poor, in an attempt to mitigate against hidden stratification in model performance (where performance varies in clinically meaningful ways between patient subsets)[70].

25 *Define the final model architecture*. Define the resulting model architecture as final and do not revisit any of the previous steps at this point. Use the fixed set of parameters corresponding to this model to compute the predictions for all timesteps in all patients for each data split.

Risk calibration and uncertainty

26 *Calibration*. A well-calibrated risk model is one in which the predicted risk matches the incidence of the outcome (i.e., 40% of patients with a 0.40 risk of AKI in 48 h should develop an AKI in that time frame). This is critical if a model is to be deployed as a clinical risk stratification tool. Deep-learning models with softmax/sigmoid output trained with cross-entropy loss are prone to miscalibration. Recalibration is often necessary to ensure that consistent probabilistic interpretations of the model predictions can be made[71]. Use the previously defined calibration set to align the predicted values with the underlying probability of the adverse event occurring at a given timestep. One approach is to fit an isotonic regression model on the predictions against the target variable[72]. Assess the quality of the calibration by comparing uncalibrated predictions to recalibrated ones using Brier score and reliability plots[73] (see Extended Data Fig. 3 in Tomasev et al.[15]).

27 *Estimate uncertainty of individual predictions*. To quantify the uncertainty of model predictions (i.e., prediction-level uncertainty), train an ensemble of multiple models with a fixed set of hyperparameters but different random initial seeds[74,75]. To get the uncertainty ranges for each prediction, take the set of predictions from all models and trim the distribution tails depending on the desired level of confidence. Note that alternative uncertainty estimation methods have been explored in the literature, including Monte Carlo dropout[76] and Bayesian neural networks[77], the latter of which can enable efficient patient-level uncertainty estimation via a single model.

28 *Estimate performance uncertainty*. To gauge uncertainty on a trained model's performance (i.e., performance uncertainty), calculate confidence intervals of performance metrics (AUROC and AUPRC) using bootstrapping. First, sample patients from a single split with replacement (200 resamples is a reasonable choice to calculate 95% confidence intervals). Next, compute the pivot bootstrap estimator using resampled values[56]. Uncertainty estimates should be computed on the validation split during model development and on the test split for final performance metrics.

29 *Clinically motivated operating points*. Performance metrics are dependent on the choice of an operating point (OP). Choose multiple OPs on the basis of the PR curve of the final model and report the performance under each[78]. In consultation with clinical experts, evaluate which operating points are most clinically significant on the basis of the validation set metrics (e.g., for AKI an OP of two false positives for one true positive was chosen as being acceptable to assist a nephrology consult team in screening an inpatient population). Results for multiple OPs are shown in Fig. 2 and Extended Data Table 4 in Tomasev et al.[15]. Note that because OPs are set on the validation split, they may not lie exactly on the precision recall (PR) curve computed from the test split.

Model generalizability evaluation

30 *Analyze model performance across subpopulations*. Define a set of clinical subpopulations relevant to the outcome of interest. This could include demographic and clinical characteristics. For the AKI prediction study, our subgroups included patients with chronic kidney injury, diabetes and medical/surgical admissions. Report the performance, including confidence intervals on each subpopulation. In particular, consider the performance and consequent resource allocation across protected groups (i.e., subpopulations vulnerable to health disparities) as part of a broader ML fairness evaluation[18].

31 *Quantify the expected daily alert rate*. Chronologically align all the patient time series from the test set. For each day in the longitudinal test set, compute the percentage of inpatients where the model: (i) produced a true-positive alert, (ii) produced a false-positive alert without having provided a

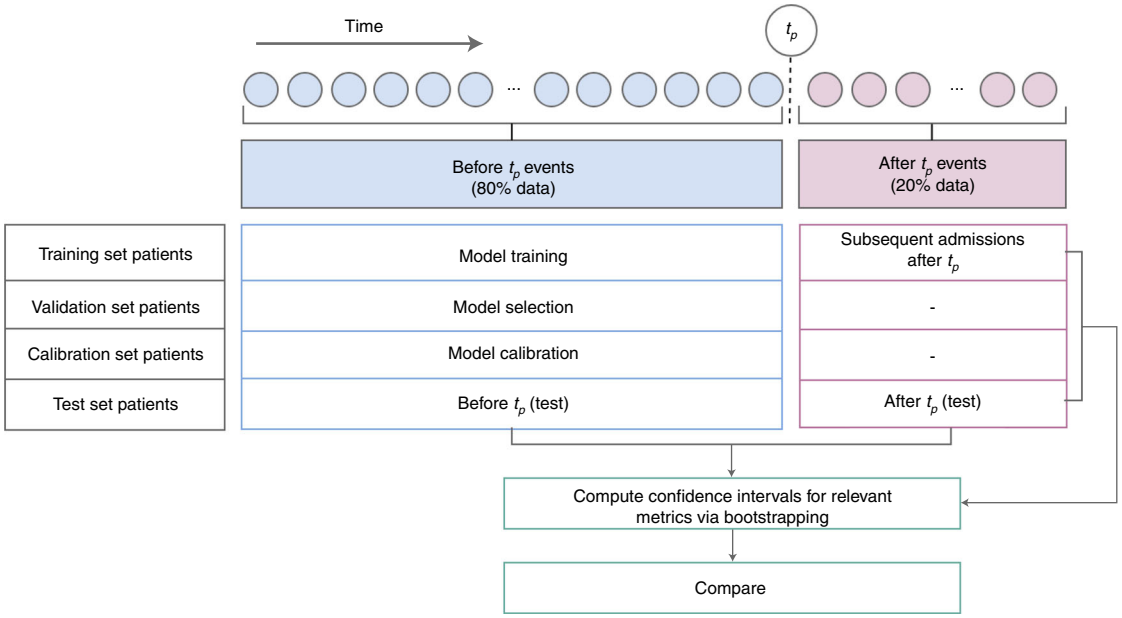

Fig. 5 | Evaluating temporal generalizability on future unseen data. Performance metrics on the test split before t_p are compared to those on the test split after t_p to determine if performance is preserved on future unseen data.

true-positive alert within a certain prior time window and (iii) did not produce any alerts. Compute the mean daily alert rate across all days in the longitudinal set. Report this metric to guide the likely resource burden in future prospective evaluation.

32 *Evaluate temporal generalizability on future unseen data*. Model performance may differ in important ways when prospectively deployed because of data drift[79]. Simulate the generalizability of models to future, unseen data. Choose a point in time t_p such that ~80% of data entries occur before time t_p and ~20% occur after time t_p (Fig. 5). Train a model using the final architecture determined in Step 25 and data only from before time t_p in the training split. Note that because hyperparameters were tuned using data from after t_p, this is an approximation, and complete re-tuning with the pre-t_p test set would be the most rigorous approach. Generate model predictions for the entire test split. Generate 95% confidence intervals of AUPRC and AUROC for predictions made before t_p and for predictions made subsequent to t_p. Compare confidence intervals to determine if model performance on future unseen data is comparable to performance on historic data.

33 *Evaluate regional generalizability in simulated cross-site deployments*. External validation, where model performance is computed on a different population/dataset, is a critical part of model evaluation. If multicenter data are available, choose a split in hospital sites such that ~80% of patient admissions occur at sites in group *A*, and ~20% occur at sites in group *B* (Fig. 6). For single-site data, this split could be done in other ways (e.g., by ward). Train a model using the final architecture determined in Step 25 and the training split, excluding data entries from admissions at sites in group *B* (note the hyperparameter leakage issue in the step above). Run inference to generate model predictions for the test split. Compare performance metrics with confidence intervals to determine if model performance for unseen sites is comparable to performance for sites used during training.

Troubleshooting

In this section, we highlight several common issues that may arise during the implementation and execution of the protocol: data issues, problem definition issues and software implementation issues.

Data issues

EHRs contain routinely collected data and are therefore prone to data entry errors and inconsistencies. Additional errors can be introduced in the process of exporting and de-identifying the data before making it available for research. These errors can adversely affect the performance of

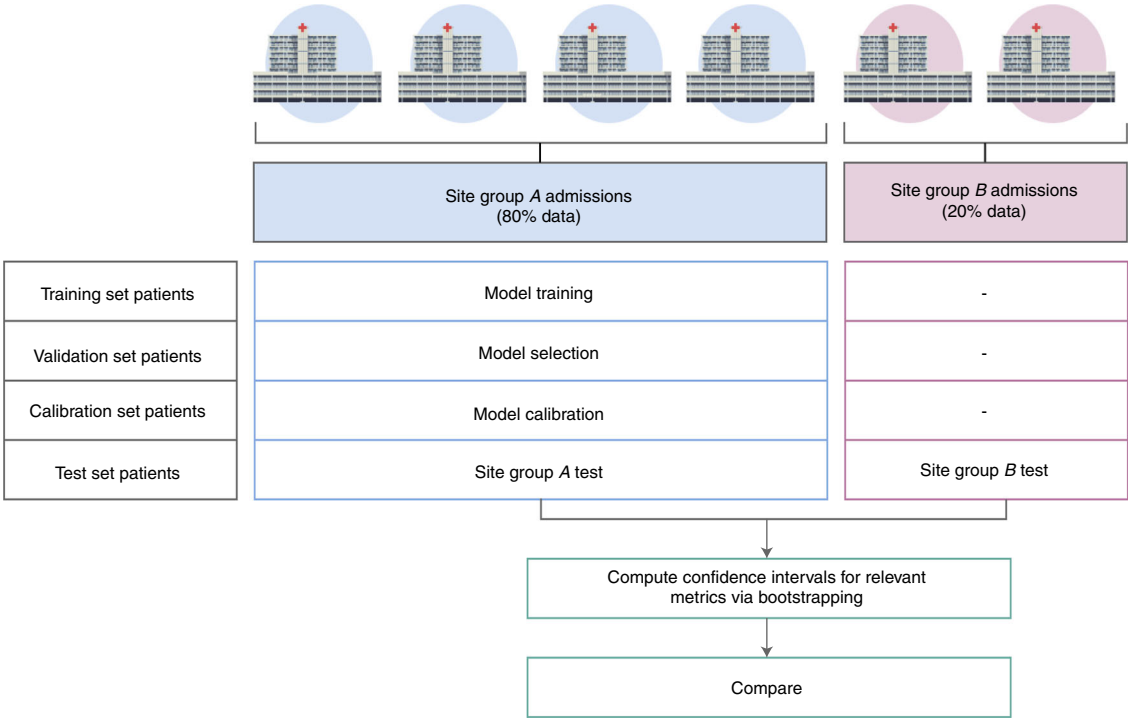

Fig. 6 | Evaluating regional generalizability in simulated cross-site deployments. Performance metrics on test split predictions made during admissions at sites in group A are compared to those for test split predictions made during admissions at sites in group B.

predictive ML models. Ideally, all data issues would be uncovered early while performing the thorough quality assessment outlined in Step 3; however, this is not always possible. It is important to regularly check the integrity of the data and labels throughout model development, as well as maintain an open communication channel to the data partners and clinical experts.

Problem definition issues

The definition of the prediction endpoints (Step 2) has a significant impact on model performance and ultimate utility. Many clinical concepts lack computable definitions, and the definitions available have often been created for large-scale epidemiological studies rather than patient-level predictions. In the case of sepsis, for example, multiple EHR definitions are available that are known to capture different patient cohorts[80]. The input of clinicians and informatics experts should be sought to ensure that the target outcomes are appropriately defined.

Software implementation issues

The protocol involves significant software engineering work to develop the data pre-processing pipeline, deep-learning architectures, baseline models and evaluation framework. The software implementation needs to be thoroughly tested in accordance with best practices of unit testing and integration testing. Automated testing should be complemented by careful error analysis (Step 24), which is critical in uncovering unanticipated implementation issues. It is worth noting that implementation issues can manifest themselves in different ways and that these errors do not always lead to lower estimated model performance metrics. For example, errors in data processing could lead to data being presented to the models out of order, artificially enhancing the perceived predictive performance.

Timing

Time estimates to execute this protocol vary greatly depending on the problem formulation, available data, team size and computational resources available. In Table 2, we provide estimates for both the design/development and the execution of different protocol steps, for a small-sized team working on a moderately-sized EHR dataset with access to appropriate infrastructure. This also assumes a full reimplementation, rather than reuse of any open-source code for component analyses.

Table 2 | Time estimates for development and execution of protocol steps

	Development	Execution
Problem definition (Steps 1–6)	Variable	–
Data pre-processing (Steps 7–16)	4–12 weeks	6–48 h
Model architecture (Steps 17–25)	2–8 weeks	1–4 d for each hyperparameter sweep
Calibration and uncertainty (Steps 26–29)	2–4 weeks	2–6 d
Generalizability evaluation (Steps 30–33)	2–4 weeks	4–8 d

Anticipated results

Detailed results for various formulations of the AKI task are provided in Tomasev et al.[15]. To demonstrate that this protocol can be generalized to other tasks and time formulations, we describe below how we used our workflow to build predictive models for three additional endpoints that were not included in Tomasev et al.[15]: mortality, LoS and hospital readmission. These endpoints were chosen because the endpoint labels were possible to define in our dataset with reasonable fidelity, numerous ML performance benchmarks exist in the literature and the endpoints are common operational targets where analytics may deliver value[81]. Below, we describe how the protocol was adapted for the additional endpoints.

Adjustments to Step 2 (outcome labels)

Inpatient mortality

The mortality label was based on a timestamped mortality flag, which included both in- and out-of-hospital mortality. To avoid label leakage, the sequential representation was masked from the 6-h time bucket in which the mortality flag occurred. The inpatient mortality rate was 2.1%.

LoS

LoS was defined as the remaining LoS from the trigger time. The median (interquartile range) LoS across all admissions was 3 (1–7) d, with a mean of 9.7 d because of a number of very long inpatient admissions. Experiments were set up as binary classification tasks, predicting remaining LoS ≤2 or ≤7 d. Previous work has defined prolonged LoS prediction as total LoS >7 d[20] or as a regression task[82]; however, for the purposes of showing multiple time formulations with a consistent lookahead window, we have chosen remaining LoS. To avoid label leakage, evaluation was not performed in the final time bucket of an admission.

30-day readmission

Readmission was defined as any inpatient admission to a VA facility within 30 d of hospital discharge. The percentage of discharged patients readmitted within this time window was 18.6%.

Adjustments to Step 5 (time formulations)

To demonstrate versatility of the architecture to time formulations, the mortality and LoS tasks were set up as both continuous predictions (triggered every 6 h) and static predictions (triggered at 24 or 48 h after admission). Readmission was modeled only as a static task at the time of discharge. For static experiments, it is possible to train as a continuous task but evaluate the model only at a single timepoint per admission (thereby converting it to a static task); however, we found that performance was higher when both training and evaluation were static tasks. Mortality models were trained using 2-, 7-, 30- and 90-d lookahead windows as well as a variable lookahead for in-hospital mortality. Remaining LoS was modeled with 2- and 7-d cutoffs; readmission used a 30-d lookahead from the time of discharge.

Adjustments to Step 6 (auxiliary tasks)

For the mortality and LoS tasks, a panel of 14 laboratory tests was identified (extending the seven auxiliary tasks used for AKI, but within the scope of the available de-identified laboratory values): hemoglobin, white blood cell count, platelets, C-reactive protein, international normalized ratio, serum protein, albumin, glucose, creatinine, urea nitrogen, potassium, sodium, chloride and pH. We swept across multiple auxiliary configurations, varying the combination of aggregating functions

(maximum, minimum, mean and standard deviation) and the combination of lookahead horizons (ranging from 6 to 72 h). In all cases, auxiliary regressions were combined to give a single loss. Where a particular laboratory value was not measured, the loss was set to zero. For consistency, the setup of auxiliary tasks was kept constant for all time formulations of mortality and LoS, although auxiliary lookaheads could readily be customized. The intuition for keeping 48 h was to capture the pattern of daily physiological trends for that patient even when modeling much longer-term clinical outcomes (e.g., 30-d mortality). No auxiliaries were used for the 30-d readmission task because this was triggered only at the time of discharge.

Adjustments to Step 21 (hyperparameter sweeps)

The highest-performing hyperparameter configurations were as follows. Continuous mortality in admission without auxiliaries, static mortality without auxiliaries and static mortality with auxiliaries used an initial learning rate of 0.0001. All other tasks used an initial learning rate of 0.001. Learning rate was decayed every 12,000 steps by a factor of 0.85, with a batch size of 128 and a back-propagation-through-time window of 128. Lookup-embedding size varied from 200 to 400 depending on the task, with constant embedding layers of size 400 each for numerical and presence features. The RNN consisted of a three-layer stacked LSTM with highway connections and cell size 300.

Results for additional endpoints

Tables 3 and 4 summarize the results obtained for the three additional endpoints: mortality, LoS and 30-d readmission. For the continuous mortality prediction, AUPRC for the RNN with auxiliary tasks ranged from 38.3% for a 48-h lookahead window to 73.8% for 90-day mortality, with AUROC of 98.6% and 95.6%, respectively. These results compare favorably to literature benchmarks for mortality prediction, although performance comparisons are difficult across datasets and experimental formulations. A recent literature review of ML models in intensive care identified 70 papers predicting mortality[83]; however, only a small subset of these used deep-learning approaches, and even fewer were designed for continuous predictions. Supplementary Table 1 provides a summary of selected ML papers predicting inpatient mortality[84–90]. Harutyunyan et al.[33,45] is one of the only studies to show results for continuous mortality predictions, specifically hourly prediction of mortality within 24 h (which the authors refer to as 'physiologic decompensation'), showing AUPRC of 31.7% and AUROC of 90.5% on a dataset with a significantly higher in-hospital mortality rate than the VA dataset (10.5% in their cohort from the Medical Information Mart for Intensive Care (MIMIC-III) dataset[91] compared with 2.1% in the VA dataset). In a related experiment, Johnson and Mark[4] simulated a real-time/continuous prediction task by training a gradient-boosting model on MIMIC-III to predict in-hospital mortality at a random timepoint, with AUPRC of 66.5% and AUROC of 92.0%. More literature benchmarks exist for static formulations—most commonly, prediction of in-hospital mortality at 24 and 48 h after admission. Our performance exceeds that reported on MIMIC-III structured data elsewhere in the literature[4,33,45], with the exception of Puroshotham et al.[34] on a feature set of 135 raw features and the use of a multimodal RNN (AUPRC of 78.6% and AUROC of 94.1% for in-hospital mortality at 24 h). A recent study by Brajer et al.[92] prospectively and externally validated a model for predicting in-hospital mortality at the time of admission, with AUPRC and AUROC of 29% and 87%, respectively, on retrospective validation and 14% and 86%, respectively, on prospective validation.

Performance for remaining LoS (AUPRC of 93.3% and AUROC of 84.3% for LoS <7 d at 24 h after admission) and 30-d readmission (AUPRC of 50.1% and AUROC of 80.8%) also compare favorably to literature benchmarks—with Rajkomar et al.[20] reporting AUROC of ≤86% for predicting total LoS >7 d at 24 h after admission and AUROC of 77% for 30-d readmission by training over both structured data and notes. Jamei et al.[93] used an MLP to predict all-cause 30-d hospital readmission and showed AUROC of 78%. Hilton et al.[94] reported AUPRC and AUROC of 38.3% and 75.8%, respectively, on 30-d readmission, with a comparable outcome prevalence to our dataset of 14.2%.

We observed a modest performance uplift from the addition of auxiliary tasks, with static formulations for mortality and LoS showing a 1–2% increase in mean AUPRC, with preserved or increased AUROC. For continuous formulations, the performance uplift from auxiliaries was consistent but small (0–1% AUPRC gain) compared with the 3.1% AUPRC uplift for the AKI task observed in Tomasev et al.[15] A different panel of auxiliary endpoints, more closely tied to the primary outcome, might improve performance. Flexible approaches to automatically identify the optimal set of auxiliary tasks are beginning to emerge and could potentially be applied[95].

Table 3 | Continuous tasks: model performance for continuous (i.e., regularly triggered) prediction tasks with variable lookahead windows

Task	Triggering	Timestep prevalence (%)	Model	AUPRC (95% CI) (%)	AUROC (95% CI) (%)
Mortality in 48 h	6-hourly	0.42	LR	11.2 (10.6, 11.8)	91.1 (90.7, 91.4)
			XGBoost	17.2 (16.2, 18.2)	94.1 (93.9, 94.3)
			RNN	37.4 (36.4, 38.3)	98.6 (98.5, 98.7)
			RNN with auxiliaries	38.3 (37.4, 39.5)	98.6 (98.6, 98.7)
Mortality in 7 d	6-hourly	1.46	LR	19.7 (18.9, 20.5)	89.5 (89.2, 89.9)
			XGBoost	25.9 (25.0, 26.9)	92.4 (92.1, 92.7)
			RNN	52.0 (51.1, 53.1)	97.9 (97.8, 98.0)
			RNN with auxiliaries	52.8 (51.8, 53.9)	98.0 (97.9, 98.1)
Mortality in 30 d	6-hourly	4.7	LR	32.5 (31.5, 33.5)	87.5 (87.1, 87.8)
			XGBoost	38.1 (37.1, 39.2)	90.0 (89.7, 90.7)
			RNN	66.8 (65.9, 67.9)	96.8 (96.6, 96.9)
			RNN with auxiliaries	67.7 (66.6, 68.8)	96.9 (96.7, 97.0)
Mortality in 90 d	6-hourly	9.0	LR	41.5 (40.6, 42.4)	85.9 (85.5, 86.4)
			XGBoost	47.1 (46.0, 48.2)	88.3 (88.0, 88.7)
			RNN	73.5 (72.6, 74.6)	95.5 (95.3, 95.7)
			RNN with auxiliaries	73.8 (72.6, 74.7)	95.6 (95.3, 95.8)
Mortality in admission	6-hourly	4.9	LR	27.5 (25.5, 29.1)	86.2 (85.2, 87.4)
			XGBoost	30.3 (28.2, 32.2)	87.3 (85.5, 89.1)
			RNN	63.5 (59.0, 67.0)	95.8 (94.8, 96.9)
			RNN with auxiliaries	64.5 (60.0, 68.8)	93.2 (89.7, 97.0)
Remaining LoS ≤2 d	6-hourly	19.9	LR	44.1 (43.8, 44.4)	78.7 (78.4, 79.0)
			XGBoost	50.5 (50.3, 50.8)	81.5 (81.2, 81.7)
			RNN	69.3 (69.1, 69.5)	90.0 (89.8, 90.1)
			RNN with auxiliaries	70.0 (69.8, 70.2)	90.2 (90.0, 90.3)
Remaining LoS ≤7 d	6-hourly	40.7	LR	73.4 (73.0, 73.7)	81.2 (80.9, 81.4)
			XGBoost	76.8 (76.5, 77.1)	83.2 (82.9, 83.4)
			RNN	86.2 (86.0, 86.4)	90.2 (90.0, 90.4)
			RNN with auxiliaries	86.4 (86.2, 86.6)	90.3 (90.1, 90.5)

A comparison is made between two baseline models (LR and XGBoost) and the deep recurrent architecture with and without auxiliary tasks. Outcome prevalence is the percentage of the positive class in the test set (timestep-level prevalence). CI, confidence interval; LR, logistic regression.

Across all tasks and time formulations, the deep-learning models trained using the above protocol outperformed baseline models (logistic regression and XGBoost). It has been suggested that performance on these canonical tasks saturates with simpler models[45]. These results suggest that there can still be significant gains in discriminative performance from deep architectures; however, the marginal benefit may be higher for more complex clinical predictions.

Further examples of the types of results that can be obtained are in Fig. 7, which shows PR and ROC curves for mortality in a 48-h model, annotated with multiple OPs (Step 29) for which the performance is further detailed in Table 5. At an OP of 33% (one true positive to two false positives), 71.2% of deaths were predicted early within a window of up to 48 h in advance (episode-level sensitivity). The shortcoming of episode-level sensitivity is that it does not account for the timeliness of predictions. To better visualize this, Fig. 8 shows early prediction histograms for various OPs, demonstrating that performance is highest closest to the time of event.

Table 4 | Static tasks: model performance for static prediction tasks (i.e., triggered at a fixed point after admission)

Task	Trigger time	Outcome prevalence (%)	Model	AUPRC (95% CI) (%)	AUROC (95% CI) (%)
Mortality in admission	24 h after admission	2.0	LR	32.7 (31.4, 34.1)	94.1 (93.8, 94.3)
			XGBoost	40.8 (39.1, 42.5)	95.7 (95.5, 95.8)
			RNN	55.0 (53.4, 56.3)	97.6 (97.4, 97.7)
			RNN with auxiliaries	56.7 (55.3, 58.4)	97.8 (97.7, 98.0)
Mortality in admission	48 h after admission	2.7	LR	23.9 (22.6, 25.0)	88.1 (87.7, 88.5)
			XGBoost	31.1 (29.8, 32.5)	91.2 (90.9, 91.5)
			RNN	58.6 (56.9, 60.1)	97.2 (97.1, 97.4)
			RNN with auxiliaries	60.8 (59.2, 62.2)	97.5 (97.4, 97.7)
Remaining LoS ≤2 d	24 h after admission	47.5	LR	55.3 (54.9, 55.8)	69.1 (68.8, 69.4)
			XGBoost	59.5 (59.1, 59.9)	72.9 (72.6, 73.1)
			RNN	73.9 (73.5, 74.3)	82.0 (81.8, 82.2)
			RNN with auxiliaries	74.7 (74.4, 75.0)	82.6 (82.4, 82.7)
Remaining LoS ≤2 d	48 h after admission	38.9	LR	50.5 (50.0, 50.9)	67.2 (66.9, 67.5)
			XGBoost	55.9 (55.4, 56.4)	71.9 (71.6, 72.2)
			RNN	71.4 (70.9, 71.7)	81.3 (81.0, 81.5)
			RNN with auxiliaries	72.1 (71.7, 72.5)	81.9 (81.6, 82.1)
Remaining LoS ≤7 d	24 h after admission	78.1	LR	86.4 (86.1, 86.6)	72.2 (71.8, 72.5)
			XGBoost	88.4 (88.2, 88.6)	75.8 (75.6, 76.1)
			RNN	93.1 (92.9, 93.2)	83.9 (83.7, 84.1)
			RNN with auxiliaries	93.3 (93.2, 93.5)	84.3 (84.1, 84.5)
Remaining LoS ≤7 d	48 h after admission	72.0	LR	83.1 (82.8, 83.5)	70.8 (70.4, 71.2)
			XGBoost	85.7 (85.4, 86.0)	74.8 (74.5, 75.1)
			RNN	91.7 (91.5, 91.8)	83.4 (83.2, 83.7)
			RNN with auxiliaries	91.9 (91.7, 92.1)	83.8 (83.6, 84.1)
Readmission in 30 d	Discharge	18.7	LR	30.2 (29.6, 30.7)	65.4 (65.0, 65.8)
			XGBoost	32.4 (31.8, 33.0)	67.1 (66.8, 67.4)
			RNN	50.1 (49.1, 50.9)	80.8 (80.5, 81.1)

A comparison is made between two baseline models (LR and XGBoost) and the deep recurrent architecture with and without auxiliary tasks. Outcome prevalence is the percentage of the positive class in the test set.

Further research is required to prepare models for clinical deployment and evaluate them prospectively. It is important to note that mortality risk models may have unintended consequences if actively deployed in real-world settings, including pathological feedback loops[96]. For example, a model may suggest removal of care for patients with a high mortality risk, in turn providing confirmatory training data for the original risk. We emphasize that any live deployment of a mortality prediction model must undergo thorough ethical review and multiple stages of user-experience research and safety evaluation. We refer the reader to refs. [14,96,97] for a more rigorous review of deployment considerations.

Conclusions

This protocol offers a versatile framework to develop deep-learning models for a range of clinical and operational use cases. We demonstrate that the architecture developed via this protocol generalizes well beyond AKI prediction, with strong performance across a range of endpoints including inpatient mortality, LoS and readmission prediction. We stress that these models are intended as prototypes for illustrating methods rather than as clinical-grade systems. Further work is required in framing appropriate clinical use cases and deploying ML models in real-world settings via prospective implementation research.

Reporting Summary

Further information on research design is available in the Nature Research Reporting Summary linked to this article.

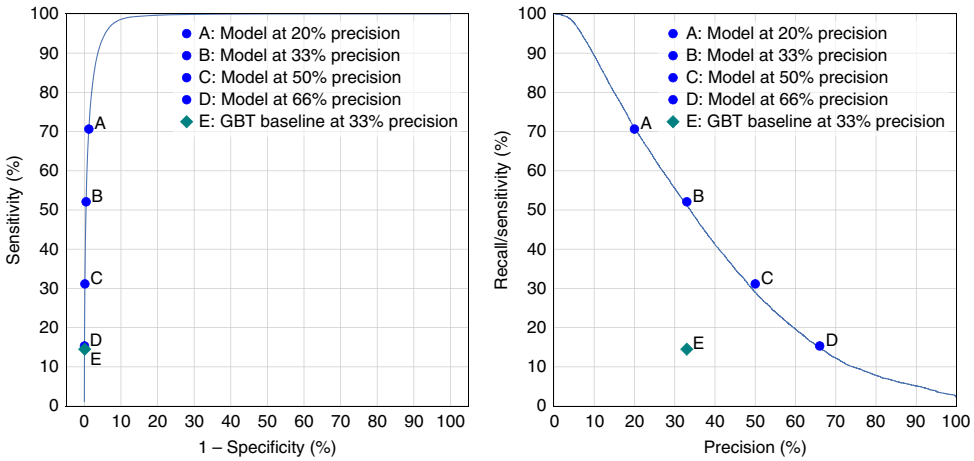

Fig. 7 | Continuous prediction of mortality in 48 h. Receiver operating characteristic (**a**) and precision–recall (**b**) curves for the risk of mortality in 48 h. Blue dots represent different model OPs on the validation set. GBT, gradient boosted tree.

Table 5 | OPs for mortality in a 48-h model: percentage of mortality events detected up to 48 h ahead of time at varying true-positive (TP) to false-positive (FP) OPs

Precision (%)	TP:FP ratio	Recall (95% CI) (%)
20	1:4	70.6 (69.6, 71.6)
33	1:2	52.1 (51.0, 53.2)
50	1:1	31.2 (30.2, 32.2)
66	2:1	15.3 (14.6, 16.0)

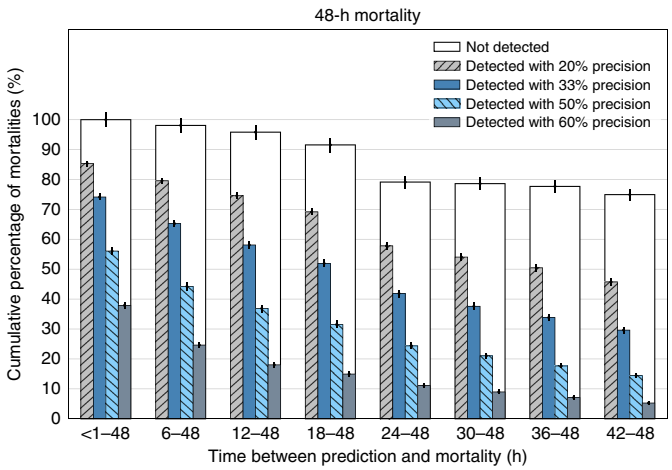

Fig. 8 | Early prediction histogram for mortality in 48 h. Model performance at timesteps before mortality. Error bars show bootstrap pivotal 95% confidence intervals; $n = 200$). The boxed area shows the upper limit on possible predictions for each time window.

Data availability

The clinical data used for the training, validation and test sets were collected at the VA and transferred to a secure data center with strict access controls in de-identified format. Data were used with both local and national permissions. The dataset is not publicly available, and restrictions apply to its use. The full results from the evaluation of our AKI model can be found in Tomasev et al[15].

Code availability

Code is available at https://github.com/google/ehr-predictions. This example code illustrates the core components of the continuous prediction architecture, task configuration and auxiliary heads. The full data pre-processing pipeline is not included here because it is highly specific to this dataset. However, we do include synthetic examples of the pre-processing stages with an accompanying data-reading notebook. We believe this exemplar code can be appropriately customized to other EHR datasets and tasks.

References

1. Royal College of Physicians. National Early Warning Score (NEWS) 2: Standardising the assessment of acute-illness severity in the NHS. Updated report of a working party. https://www.rcplondon.ac.uk/file/8636/download (2017).
2. van Walraven, C. et al. Derivation and validation of an index to predict early death or unplanned readmission after discharge from hospital to the community. *CMAJ* **182**, 551–557 (2010).
3. Sutton, R. T. et al. An overview of clinical decision support systems: benefits, risks, and strategies for success. *NPJ Digit. Med.* **3**, 17 (2020).
4. Johnson, A. E. W. & Mark, R. G. Real-time mortality prediction in the Intensive Care Unit. *AMIA Annu. Symp. Proc.* **2017**, 994–1003 (2017).
5. Barnes, S., Hamrock, E., Toerper, M., Siddiqui, S. & Levin, S. Real-time prediction of inpatient length of stay for discharge prioritization. *J. Am. Med. Inform. Assoc.* **23**, e2–e10 (2016).
6. Horng, S. et al. Creating an automated trigger for sepsis clinical decision support at emergency department triage using machine learning. *PLoS ONE* **12**, e0174708 (2017).
7. Henry, K. E., Hager, D. N., Pronovost, P. J. & Saria, S. A targeted real-time early warning score (TREWScore) for septic shock. *Sci. Transl. Med.* **7**, 299ra122 (2015).
8. Wong, A. et al. Development and validation of an electronic health record-based machine learning model to estimate delirium risk in newly hospitalized patients without known cognitive impairment. *JAMA Netw. Open* **1**, e181018 (2018).
9. Xiao, C., Choi, E. & Sun, J. Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. *J. Am. Med. Inform. Assoc.* **25**, 1419–1428 (2018).
10. Fagerström, J., Bång, M., Wilhelms, D. & Chew, M. S. LiSep LSTM: a machine learning algorithm for early detection of septic shock. *Sci. Rep.* **9**, 15132 (2019).
11. Bedoya, A. D. et al. Machine learning for early detection of sepsis: an internal and temporal validation study. *JAMIA Open* **3**, 252–260 (2020).
12. Hyland, S. L. et al. Early prediction of circulatory failure in the intensive care unit using machine learning. *Nat. Med.* **26**, 364–373 (2020).
13. Seneviratne, M. G., Shah, N. H. & Chu, L. Bridging the implementation gap of machine learning in healthcare. *BMJ Innov.* **6**, 45–47 (2019).
14. Sendak, M. P. et al. A path for translation of machine learning products into healthcare delivery. *EMJ Innov.* https://doi.org/10.33590/emjinnov/19-00172 (2020).
15. Tomašev, N. et al. A clinically applicable approach to the continuous prediction of future acute kidney injury. *Nature* **572**, 116–119 (2019).
16. Prosperi, M. et al. Causal inference and counterfactual prediction in machine learning for actionable healthcare. *Nat. Mach. Intell.* **2**, 369–375 (2020).
17. Riley, R. D. et al. Calculating the sample size required for developing a clinical prediction model. *BMJ* **368**, m441 (2020).
18. Rajkomar, A., Hardt, M., Howell, M. D., Corrado, G. & Chin, M. H. Ensuring fairness in machine learning to advance health equity. *Ann. Intern. Med.* **169**, 866–872 (2018).
19. Mitchell, M. et al. Model cards for model reporting. In *FAT* '19: Proceedings of the Conference on Fairness, Accountability, and Transparency* 220–229 (Association for Computing Machinery, 2019).
20. Rajkomar, A. et al. Scalable and accurate deep learning with electronic health records. *NPJ Digit. Med.* **1**, 18 (2018).
21. Assale, M., Dui, L. G., Cina, A., Seveso, A. & Cabitza, F. The revival of the notes field: leveraging the unstructured content in electronic health records. *Front. Med.* **6**, 66 (2019).
22. Huang, K., Altosaar, J. & Ranganath, R. ClinicalBERT: modeling clinical notes and predicting hospital readmission. Preprint at https://arxiv.org/abs/1904.05342 (2019).
23. Kemp, J., Rajkomar, A. & Dai, A. M. Improved hierarchical patient classification with language mpretraining over clinical notes. Preprint at https://arxiv.org/abs/1909.03039 (2019).
24. Chen, P.-H. C., Liu, Y. & Peng, L. How to develop machine learning models for healthcare. *Nat. Mater.* **18**, 410–414 (2019).
25. Liu, Y., Chen, P.-H. C., Krause, J. & Peng, L. How to read articles that use machine learning: users' guides to the medical literature. *JAMA* **322**, 1806–1816 (2019).
26. Wiens, J. et al. Do no harm: a roadmap for responsible machine learning for health care. *Nat. Med.* **25**, 1337–1340 (2019).

27. Ghassemi, M. et al. Practical guidance on artificial intelligence for health-care data. *Lancet Digit. Health* **1**, e157–e159 (2019).
28. Esteva, A. et al. A guide to deep learning in healthcare. *Nat. Med.* **25**, 24–29 (2019).
29. Collins, G. S. & Moons, K. G. M. Reporting of artificial intelligence prediction models. *Lancet* **393**, 1577–1579 (2019).
30. Sounderajah, V. et al. Developing specific reporting guidelines for diagnostic accuracy studies assessing AI interventions: the STARD-AI Steering Group. *Nat. Med.* **26**, 807–808 (2020).
31. Liu, X. et al. Reporting guidelines for clinical trial reports for interventions involving artificial intelligence: the CONSORT-AI extension. *Nat. Med.* **26**, 1364–1374 (2020).
32. Cruz Rivera, S. et al. Guidelines for clinical trial protocols for interventions involving artificial intelligence: the SPIRIT-AI extension. *Nat. Med.* **26**, 1351–1363 (2020).
33. Harutyunyan, H. et al. Multitask learning and benchmarking with clinical time series data. *Sci. Data* **6**, 96 (2019).
34. Purushotham, S., Meng, C., Che, Z. & Liu, Y. Benchmarking deep learning models on large healthcare datasets. *J. Biomed. Inform.* **83**, 112–134 (2018).
35. Nemati, S. et al. An interpretable machine learning model for accurate prediction of sepsis in the ICU. *Crit. Care Med.* **46**, 547–553 (2018).
36. Caicedo-Torres, W. & Gutierrez, J. ISeeU: visually interpretable deep learning for mortality prediction inside the ICU. *J. Biomed. Inform.* **98**, 103269 (2019).
37. Shickel, B. et al. DeepSOFA: a continuous acuity score for critically ill patients using clinically interpretable deep learning. *Sci. Rep.* **9**, 1879 (2019).
38. Avati, A. et al. Improving palliative care with deep learning. *BMC Med. Inform. Decis. Mak.* **18**, 122–122 (2018).
39. Hunter, J. D. Matplotlib: a 2D graphics environment. *Comput. Sci. Eng.* **9**, 90–95 (2007).
40. Pedregosa, F. et al. Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).
41. Li, R. C., Asch, S. M. & Shah, N. H. Developing a delivery science for artificial intelligence in healthcare. *NPJ Digit. Med.* **3**, 107 (2020).
42. Blecker, S. et al. Interruptive versus noninterruptive clinical decision support: usability study. *JMIR Hum. Factors* **6**, e12469 (2019).
43. Selby, N. M., Hill, R. & Fluck, R. J. Standardizing the early identification of acute kidney injury: the NHS England national patient safety alert. *Nephron* **131**, 113–117 (2015).
44. Amland, R. C. & Hahn-Cover, K. E. Clinical decision support for early recognition of sepsis. *Am. J. Med. Qual.* **31**, 103–110 (2016).
45. Wang, S. et al. MIMIC-Extract: a data extraction, preprocessing, and representation pipeline for MIMIC-III. In *CHIL '20:* Proceedings of the ACM Conference on Health, Inference, and Learning 222–235 (Association for Computing Machinery, 2020).
46. Khwaja, A. KDIGO clinical practice guidelines for acute kidney injury. *Nephron Clin. Pract.* **120**, c179–c184 (2012).
47. Ding, D. Y. et al. The effectiveness of multitask learning for phenotyping with electronic health records data. Preprint at https://arxiv.org/pdf/1808.03331.pdf (2018).
48. McDermott, M. B. A. et al. A comprehensive evaluation of multi-task learning and multi-task pre-training on EHR time-series data. Preprint at https://arxiv.org/abs/2007.10185 (2020).
49. Lipton, Z. C., Kale, D. C. & Wetzel, R. C. Directly modeling missing data in sequences with RNNs: improved classification of clinical time series. In *Proceedings of the 1st Machine Learning for Healthcare Conference, PMLR* Vol. 56, 253–270 Available at https://arxiv.org/abs/1606.04130 (2016).
50. Beaulieu-Jones, B. K. et al. Characterizing and managing missing structured data in electronic health records: data analysis. *JMIR Med. Inform.* **6**, e11 (2018).
51. Xue, Y., Klabjan, D. & Luo, Y. Mixture-based multiple imputation model for clinical data with a temporal dimension. In *2019 IEEE International Conference on Big Data (Big Data)* 245–252 (IEEE, Los Angeles, CA, USA, 2019).
52. Yoon, J., Jordon, J. & van der Schaar, M. GAIN: missing data imputation using generative adversarial nets. In *ICML '18: Proceedings of the 35th International Conference on Machine Learning* (eds. Dy, J. & Krause, A.) 5689–5698 (International Machine Learning Society, 2018).
53. Saito, T. & Rehmsmeier, M. The precision recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE* **10**, e0118432 (2015).
54. Lee, C., Yoon, J. & d. Schaar, M. V. Dynamic-DeepHit: a deep learning approach for dynamic survival analysis with competing risks based on longitudinal data. *IEEE Trans. Biomed. Eng.* **67**, 122–133 (2020).
55. Chen, T. & Guestrin, C. XGBoost: a scalable tree boosting system. In *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 785–794 (Association for Computing Machinery, 2016).
56. Efron, B. & Tibshirani, R. J. *An Introduction to the Bootstrap* (Chapman & Hall/CRC Press, 1994).
57. Miotto, R., Li, L., Kidd, B. & T. Dudley, J. Deep Patient: an unsupervised representation to predict the future of patients from the electronic health records. *Sci. Rep.* **6**, 26094 (2016).
58. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).
59. Collins, J., Sohl-Dickstein, J. & Sussillo, D. Capacity and learnability in recurrent neural networks. Preprint at https://arxiv.org/abs/1611.09913 (2017).

60. Lei, T., Zhang, Y., Wang, S. I., Dai, H. & Artzi, Y. Simple recurrent units for highly parallelizable recurrence. Preprint at https://arxiv.org/abs/1709.02755 (2017).
61. Bradbury, J., Merity, S., Xiong, C. & Socher, R. Quasi-recurrent neural networks. Preprint at https://arxiv.org/abs/1611.01576 (2016).
62. Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. Preprint at https://arxiv.org/abs/1412.3555 (2014).
63. Graves, A., Wayne, G. & Danihelka, I. Neural Turing machines. Preprint at https://arxiv.org/abs/1410.5401 (2014).
64. Santoro, A. et al. One-shot learning with memory-augmented neural networks. In *ICML '16: Proceedings of the 33rd International Conference on Machine Learning* Vol. 48 (eds. Balcan, M. F. & Weinberger, K. Q.) 1842–1850 (International Machine Learning Society, 2016).
65. Graves, A. et al. Hybrid computing using a neural network with dynamic external memory. *Nature* **538**, 471–476 (2016).
66. Santoro, A. et al. Relational recurrent neural networks. Preprint at https://arxiv.org/abs/1806.01822 (2018).
67. Zilly, J. G., Srivastava, R. K., Koutník, J. & Schmidhuber, J. Recurrent highway networks. In *ICML '17: Proceedings of the 34th International Conference on Machine Learning* Vol. 70 (eds. Precup, D. & Teh, Y. W.) 4189–4198 (International Machine Learning Society, 2017).
68. Zeiler, M. D. & Fergus, R. Visualizing and understanding convolutional networks. In *Computer Vision—ECCV 2014* (eds. Fleet, D. et al.) 818–833 (Springer, 2014).
69. Ancona, M., Öztireli, C. & Gross, M. H. Explaining deep neural networks with a polynomial time algorithm for Shapley values approximation. In *Proceedings of the 36th International Conference on Machine Learning* Vol. 97 (eds. Chaudhuri, K. & Salakhutdinov, R.) 272–281 (International Machine Learning Society, 2019).
70. Oakden-Rayner, L., Dunnmon, J., Carneiro, G. & Ré, C. Hidden stratification causes clinically meaningful failures in machine learning for medical imaging. In *CHIL '20: Proceedings of the ACM Conference on Health, Inference, and Learning* 151–159 (Association for Computing Machinery, 2020).
71. Guo, C., Pleiss, G., Sun, Y. & Weinberger, K. Q. On calibration of modern neural networks. In *ICML '17: Proceedings of the 34th International Conference on Machine Learning* Vol. 70 (eds. Precup, D. & Teh, Y. W.) 1321–1330 (International Machine Learning Society, 2017).
72. Zadrozny, B. & Elkan, C. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 694–699 (Association for Computing Machinery, 2002).
73. Niculescu-Mizil, A. & Caruana, R. Predicting good probabilities with supervised learning. In *ICML '05: Proceedings of the 22nd International Conference on Machine Learning* (eds. Raedt, L. D. & Wrobel, S.) 625–632 (Association for Computing Machinery, 2005).
74. Fauw, J. D. et al. Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nat. Med.* **24**, 1342–1350 (2018).
75. Lakshminarayanan, B., Pritzel, A. & Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems* Vol. 30, 6402–6413 (2017).
76. Gal, Y. & Ghahramani, Z. Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In *ICML '16: Proceedings of the 33rd International Conference on Machine Learning (ICML)* Vol. 48 (eds. Balcan, M. F. & Weinberger, K. Q.) 1050–1059 (2016).
77. Dusenberry, M. W. et al. Analyzing the role of model uncertainty for electronic health records. In *CHIL '20: Proceedings of the ACM Conference on Health, Inference, and Learning* (Association for Computing Machinery, 2020).
78. Romero-Brufau, S., Huddleston, J. M., Escobar, G. J. & Liebow, M. Why the C-statistic is not informative to evaluate early warning scores and what metrics to use. *Crit. Care* **19**, 285 (2015).
79. Nestor, B. et al. Feature robustness in non-stationary health records: caveats to deployable model performance in common clinical machine learning tasks. In *Machine Learning for Healthcare (MLHC)* Vol. 106, 1–23 (PMLR, 2019).
80. Johnson, A. E. W. et al. A comparative analysis of sepsis identification methods in an electronic database. *Crit. Care Med.* **46**, 494–499 (2018).
81. Bates, D. W. et al. Big data in health care: using analytics to identify and manage high-risk and high-cost patients. *Health Aff.* **33**, 1123–1131 (2014).
82. Verburg, I. W. M., de Keizer, N. F., de Jonge, E. & Peek, N. Comparison of regression methods for modeling intensive care length of stay. *PLoS ONE* **9**, e109684 (2014).
83. Shillan, D., Sterne, J. A. C., Champneys, A. & Gibbison, B. Use of machine learning to analyse routinely collected intensive care unit data: a systematic review. *Crit. Care* **23**, 284 (2019).
84. Nakas, C. T., Schütz, N., Werners, M. & Leichtle, A. B. Accuracy and calibration of computational approaches for inpatient mortality predictive modeling. *PLoS ONE* **11**, 1–11 (2016).
85. Aczon, M. et al. Dynamic mortality risk predictions in pediatric critical care using recurrent neural networks. Preprint at https://arxiv.org/abs/1701.06675 (2017).
86. Che, Z., Purushotham, S., Cho, K. & Sontag, D. Recurrent neural networks for multivariate time series with missing values. *Sci. Rep.* **8**, 6085 (2018).
87. Mayampurath, A. et al. Combining patient visual timelines with deep learning to predict mortality. *PLoS ONE* **14**, e0220640 (2019).

88. Fritz, B. A. et al. Deep-learning model for predicting 30-day postoperative mortality. *Br. J. Anaesth.* **123**, 688–695 (2019).
89. Xia, J. et al. A long short-term memory ensemble approach for improving the outcome prediction in intensive care unit. *Comput. Math. Methods Med.* **2019**, 8152713 (2019).
90. Nielsen, A. B. et al. Survival prediction in intensive-care units based on aggregation of long-term disease history and acute physiology: a retrospective study of the Danish National Patient Registry and electronic patient records. *Lancet Digit. Health* **1**, e78–e89 (2019).
91. Johnson, A. E. W. et al. MIMIC-III, a freely accessible critical care database. *Sci. Data* **3**, 160035 (2016).
92. Brajer, N. et al. Prospective and external evaluation of a machine learning model to predict in-hospital mortality of adults at time of admission. *JAMA Netw. Open* **3**, e1920733 (2020).
93. Jamei, M., Nisnevich, A., Wetchler, E., Sudat, S. & Liu, E. Predicting all-cause risk of 30-day hospital readmission using artificial neural networks. *PLoS ONE* **12**, e0181173 (2017).
94. Hilton, C. B. et al. Personalized predictions of patient outcomes during and after hospitalization using artificial intelligence. *NPJ Digit. Med.* **3**, 51 (2020).
95. Liu, S., Davison, A. J. & Johns, E. Self-supervised generalisation with meta auxiliary learning. In *32nd Conference on Neural Information Processing Systems (NeurIPS 2019)* (Neural Information Processing Systems Foundation Inc., 2019).
96. Ghassemi, M. et al. A review of challenges and opportunities in machine learning for health. Preprint at https://arxiv.org/abs/1806.00388 (2020).
97. Kelly, C. J. et al. Key challenges for delivering clinical impact with artificial intelligence. *BMC Med.* **17**, 195 (2019).

Acknowledgements

We thank the veterans and their families under the care of the VA. We also thank A. Phalen, A. Graves, O. Vinyals, K. Kavukcuoglu, S. Chiappa, T. Lillicrap, R. Raine, P. Keane, A. Schlosberg, O. Ronneberger, J. De Fauw, K. Ruark, M. Jones, J. Quinn, D. Chou, C. Meaden, G. Screen, W. West, R. West, P. Sundberg and the Google Research team, J. Besley, M. Bawn, K. Ayoub and R. Ahmed. Special thanks to K. Peterson and the many other VA staff, including physicians, administrators and researchers who worked on the data collection. Thanks to the many DeepMind and Google Health colleagues for their support, ideas and encouragement. G.R. & H.M. were supported by University College London and the National Institute for Health Research (NIHR) University College London Hospitals Biomedical Research Centre. The views expressed are those of these author(s) and not necessarily those of the NHS, the NIHR or the Department of Health.

Author contributions

M.S., T.B., J.C., J.R.L., N.T., C.N., D.H. and R.R. initiated the project. N.T., X.G., H.A., J.R.L., C.N. and C.R.B. created the dataset. N.T., X.G., A.S., H.A., J.W.R., M.Z., A.M., I.P., N.H., S.B. and S.M. contributed to software engineering. N.T., X.G., A.M., J.W.R., M.Z., A.S., S.M., N.H., S.B., M.G.S., X.G., J.R.L., T.F.O., C.N. and C.R.B. analyzed the results. N.T., X.G., A.M., J.W.R., M.Z., S.R. and S.M. designed the model architectures. J.R.L., G.R., H.M., C.L., A.C., A.K., C.O.H., M.G.S, D.K., T.F.O. and C.N. contributed clinical expertise. C.M., J.R.L., T.B., V.M., S.M. and C.N. managed the project. N.T., J.R.L., M.G.S, J.W.R., M.Z., A.M., H.M., C.R.B., S.M. and G.R. wrote the manuscript.

Competing interests

G.R., H.M. and C.L. are paid contractors of DeepMind/Google Health.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41596-021-00513-5.

Correspondence and requests for materials should be addressed to N.T., M.G.S. or J.R.L.

Peer review information *Nature Protocols* thanks Issam El Naqa and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 6 April 2020; Accepted: 25 January 2021;
Published online: 5 May 2021

Related links
Key reference using this protocol
Tomašev, N. et al. *Nature* **572**, 116–119 (2019): https://doi.org/10.1038/s41586-019-1390-1

nature research

Corresponding author(s): Joseph R Ledsam

Last updated by author(s): Oct 26, 2020

Reporting Summary

Nature Research wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Research policies, see Authors & Referees and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed	
☐	☒	The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☐	☒	A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☐	☒	The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☐	☒	A description of all covariates tested
☐	☒	A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☐	☒	A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☐	☒	For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐	For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐	For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☐	☒	Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	Data collection was performed by independent members of the VA National Data Center without involvement from research team members. Collection was performed using the Vista EHR system and associated databases.
Data analysis	The networks used the TensorFlow library with custom extensions. Analysis was performed with custom code written in Python 2.7. Please see the manuscript methods section for more detail.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors/reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Research guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A list of figures that have associated raw data
- A description of any restrictions on data availability

The clinical data used for the training, validation and test sets were collected at the US Department of Veterans Affairs and transferred to a secure data centre with strict access controls in de-identified format. Data were used with both local and national permissions. They are not publicly available and restrictions apply to their use. The de-identified dataset, or a test subset, may be available from the US Department of Veterans Affairs subject to local and national ethical approvals.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	The dataset consisted of all eligible patients during a five year period across the entire VA healthcare system in the USA. The test population was a random selection of 10% of these, totaling 70,681 individual patients and 252,492 unique admissions (please refer to methods section for more details on how test populations were selected). A sample size requirement of 179 patients would be required to detect sensitivity and specificity at 0.05 marginal error and 95% confidence. The total number of test patients exceeded this requirement by two orders of magnitude.
Data exclusions	We excluded patients below the age of 18 and above the age of 90 in accordance with HIPAA Safe Harbor criteria, and patients without any serum creatinine recorded in EHR. (See paper methods for more detail.) To protect patient privacy sites with fewer than 250 admissions during the five year time period were also excluded; four of the 1,243 health care facilities from which the VA is composed were excluded based on this criteria. All exclusion criteria were established prior to beginning the work.
Replication	All 70,681 patients in the test set were randomly selected and were not correlated in any way. The experiments can be interpreted as 70,681 replicas of the model applied to a single patient over a fifteen year period.
Randomization	The data were randomly divided into training (80% of observations), validation (5%), calibration (5%) and testing (10%) sets. All data for a single patient was assigned to exactly one of these splits. (See paper methods for more detail.)
Blinding	When assigning patients randomly to test, validation and training groups investigators were blinded to patient covariates and all features in the EHR not required to perform the research (e.g., creatinine was required to label AKI as a ground truth). Patient recruitment was conducted by independent members of the VA National Data Center; research team members were blinded to this recruitment.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology
☒	☐ Animals and other organisms
☐	☒ Human research participants
☒	☐ Clinical data

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging

Human research participants

Policy information about studies involving human research participants

Population characteristics	The data included all VA patients aged between 18 and 90 admitted for secondary care to medical or surgical services between 10/1/2011 to 9/30/2015, with laboratory data that included serum creatinine recorded in EHR and with at least one year of EHR data prior to admission data. The test set was a randomly selected 10% of all admissions included in the work. Average age was 62.3. Males represented 93.6% of the test population. Average number of inpatient admissions was 3.6; average admission duration was 9.6 days. AKI occurred in 13.4% of admissions. These figures were consistent with the population of the VA as a whole.
Recruitment	The data was recruited from the US Department of Veterans Affairs (VA). The VA is composed of 1,243 health care facilities, including 172 VA Medical Centers and 1,062 outpatient sites of care. Aggregating data from one or more of these facilities are 130 data centres, of which 114 had data for inpatient admissions used in this study. Four sites were excluded due to small numbers of patients: fewer than 250 admissions during the fifteen year time period. No other patients were excluded based on location, and no other exclusion criteria were applied. The final dataset consisted of the records for all 703,782 patients that met

inclusion and exclusion criteria.

Ethics oversight	This work, and the collection of data on implied consent, received Tennessee Valley Healthcare System Institutional Review Board (IRB) approval from the US Department of Veterans Affairs. De-identification was performed in line with the Health Insurance Portability and Accountability Act (HIPAA), and validated by the US Department of Veterans Affairs Central Database and Information Governance departments. Only de-identified retrospective data was used for research, without the active involvement of patients.

Note that full information on the approval of the study protocol must also be provided in the manuscript.