Smile: A System to Support Machine Learning on EEG Data at Scale

Lei Cao [1], Wenbo Tao [1], Sungtae An [3], Jing Jin [2], Yizhou Yan [1], Xiaoyu Liu [1]
Wendong Ge [2], Adam Sah [1], Leilani Battle [4], Jimeng Sun [3], Remco Chang [5]
Brandon Westover [2], Samuel Madden [1], Michael Stonebraker [1]

[1] *Massachusetts Institute of Technology, Cambridge, MA USA*
[2] *Massachusetts General Hospital, Boston, MA, USA*
[3] *Georgia Institute of Technology, Atlanta, GA, USA*
[4] *University of Maryland, College Park, MD, USA*
[5] *Tufts University, Medford, MA, USA*

(lcao,wenbo,yyan2,xiaoyuliu,madden,stonebraker)@csail.mit.edu
(jjing,wendong.ge,mwestover)@mgh.harvard.edu, (stan84,jsun)@gatech.edu
adam.sah@gmail.com,leilani@cs.umd.edu, remco@cs.tufts.edu

ABSTRACT

In order to reduce the possibility of neural injury from seizures and sidestep the need for a neurologist to spend hours on manually reviewing the EEG recording, it is critical to automatically detect and classify "interictal-ictal continuum" (IIC) patterns from EEG data. However, the existing IIC classification techniques are shown to be not accurate and robust enough for clinical use because of the lack of high quality labels of EEG segments as training data. Obtaining high-quality labeled data is traditionally a manual process by trained clinicians that can be tedious, time-consuming, and error-prone. In this work, we propose Smile, an industrial scale system that provides an end-to-end solution to the IIC pattern classification problem. The core components of Smile include a visualization-based time series labeling module and a deep-learning based active learning module. The labeling module enables the users to explore and label 350 million EEG segments (30TB) at interactive speed. The multiple coordinated views allow the users to examine the EEG signals from both time domain and frequency domain simultaneously. The active learning module first trains a deep neural network that automatically extracts both the local features with respect to each segment itself and the long term dynamics of the EEG signals to classify IIC patterns. Then leveraging the output of the deep learning model, the EEG segments that can best improve the model are selected and prompted to clinicians to label. This process is iterated until the clinicians and the models show high degree of agreement. Our initial experimental results show that our Smile system allows the clinicians to label the EEG segments at will with a response time below 500 ms. The accuracy of the model is progressively improved as more and more high quality labels are acquired over time.

Proceedings of the VLDB Endowment, Vol. 12, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3352063.3352138

PVLDB Reference Format:
Lei Cao, Wenbo Tao, Sungtae An, Jing Jin, Yizhou Yan, Xiaoyu Liu, Wendong Ge, Adam Sah, Leilani Battle, Jimeng Sun, Remco Chang, Brandon Westover, Samuel Madden, Michael Stonebraker. Smile: A System to Support Machine Learning on EEG Data at Scale. *PVLDB*, 12(12): 2230-2241, 2019.
DOI: https://doi.org/10.14778/3352063.3352138

1. INTRODUCTION

In contemporary medicine, a significant fraction of critically ill patients in the intensive care unit (ICU) experience non-convulsive seizures (NCS): seizures with few or no clinical manifestations [15, 24]. NCS can cause neuronal injury or worsen existing injuries, and are correlated with poor neurologic outcomes for patients.

In the Intensive Care Unit (ICU) setting, NCS can be detected by analyzing electroencephalography (EEG) data obtained through brain monitoring, because clear patterns associated with seizures and NCS can be observed in the EEG data. These patterns, known as "interictal-ictal-injury continuum" (IIC) patterns, reflect increased risk of seizures and poor outcome in critically ill patients, and can themselves damage the brain [20].

The American Clinical Neurophysiology Society (ACNS) [23] divides IIC patterns into multiple classes such as "Periodic Discharges" (PD) and "Rhythmic Delta Activity" (RDA). These patterns can be further categorized as "Lateralized" (L) or "Generalized" (G) based on whether the patterns present in a single (L) or in both (G) hemispheres. Clinicians decide which of the available treatments to administer after IIC patterns are detected and classified. In this project, we also include seizures (including NCS) among IIC patterns.

In current clinical practice, detecting and classifying IIC patterns relies on clinicians to manually examine the EEG. This is expensive and challenging. First, these patterns are often present in a short time interval, in some cases lasting only 10 seconds. Prolonged continuous EEG monitoring (cEEG) is therefore important for detecting IIC patterns. However, it is a complex and time-consuming task for clinicians to routinely scrutinize the large amount of data in cEEG. Second, correctly classifying IIC patterns requires special expertise. In some cases, even for experienced neurology specialists, it is hard to capture and classify IIC patterns. The reason is

that each type of IIC pattern can vary significantly across different patients over different times.

Therefore, there is a critical need for automating the discovery and classification of IIC patterns. Such a system can reduce the possibility of neural injury by capturing the brain diseases as early as possible. Moreover, potentially it could enable the small hospitals lack of epilepsy specialists to diagnose seizures.

In this paper, we describe a system, Smile that asks clinicians to label a few examples, using a scalable visualization and labeling system for EEG data, and then employs a neural-network based classifier to automate the task of finding IIC patterns in large scale time-series data.

State-of-the-art. Previous studies have tried to break the EEG into segments and *cluster* the EEG segments based on the features extracted from each segment [8, 2]. However, our evaluation shows that these unsupervised methods do not work well. The resulting clusters often were not related to a specific IIC pattern. This is because clustering relies on a distance function to separate the objects belonging to different classes, while the diversity within each class of IIC patterns make it hard to define an appropriate distance function.

More recent studies have used supervised methods in classifying EEG segments [16, 40]. However, due to the limited number of labeled EEG segments available to researchers, no convincing result has been reported yet.

We estimate that EEG data from at least 1000 patients is needed to cover the full range of variation encountered in practice. During the past decade, members of our group in the Neurology department of Massachusetts General Hospital (MGH) have collected unlabeled EEG data from over 2500 subjects, totaling around 30 TBytes – the largest EEG dataset in the world to date. Were it possible to label this data, this would provide sufficient data to train an accurate general-purpose IIC detection model for clinical use.

Challenges. Labeling and modeling large quantities of EEG data is challenging. First, as previously mentioned, labeling EEG data is difficult even for experts. To correctly label EEG segments, experts need to visually see the waves of the EEG data in the time domain and the corresponding spectrogram images in the frequency domain. Further, experts need to continuously pan the EEG signal over time to see how the to-be-labeled EEG segment looks in contrast to adjacent segments. However, to the best of our knowledge, no existing visualization tool can support the interactive exploration of 30T timeseries data on multiple pan/zoom views. Second, even with such a visualization tool in hand, it is still extremely time consuming for experts to label a large number of EEG segments, while the time of neurology specialists is precious. Ideally, they should only be asked to label segments that are likely to enrich the feature representation of the existing label set. Third, even if abundant labels are obtained, building an accurate classification model is challenging due to the complex and dynamic nature of IIC patterns. As the input to the classification model, the features extracted from each EEG segment have to reflect both the local characteristics of each segment itself and its temporal dynamics over longer time scales. This makes extraction of appropriate features challenging.

Approach & Contributions. In this work, we describe Smile, the system we have built to solve this problem at industrial scale. It employs scalable visualization, parallel data processing, and machine learning techniques to provide an end-to-end solution to the IIC pattern classification problem. Users can interactively label large scale EEG time-series data, and then leverage the results of a classification model trained on those labels to progressively collect new labels and improve the accuracy of the classification model. Smile is in the active use of 16 medical institutes for the labelling of EEG segments. Key contributions include:

- Smile leverages a set of scalable data processing and machine learning techniques to prepare for the interactive exploration of the big EEG data, including: (1) parallel loading of the EEG data to key-value store; (2) embedding high dimensional EEG segments for visualization in a 2D space in a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability; and (3) change point detection [21, 41] to reduce the redundancy in the data.
- We develop a labeling system that for the first time allows users to visually explore many terabytes of data (30 TB in our prototype) at interactive speed. The key techniques include (1) a big data visualization system that ensures $<$ 500 ms latency by employing a series of optimizations such as caching and spatial indexing; (2) and multiple coordinated views that allow users to simultaneously pan the EEG data and the corresponding spectrogram images.
- We design a deep neural network structure consisting of a RNN model [25] stacked on a CNN model [35, 33]. This model automatically extracts local features with respect to each segment as well as the long term characteristics of a series of adjacent EEG segments. Then, based on the output of the modeling, we use active learning to suggest candidate segments to the labeling system that could best improve the prediction of the model. This way, Smile achieves high classification accuracy with minimal labeling effort.
- Our initial experimental evaluation confirms that our Smile system is able to support the exploration and labeling of large-scale EEG data with response times below 500 ms. Further, the quality of the acquired labels and the accuracy of the classification model are shown to be improved during the iterative active learning process.
- As an example application, Smile showcases how an end-to-end system combining techniques from data processing to machine learning can transform medicine. Our methodology is of potential value to a much broader class of applications in addition to IIC classification, in particular applications dealing with time series data captured in many medical domains.

2. PRELIMINARIES

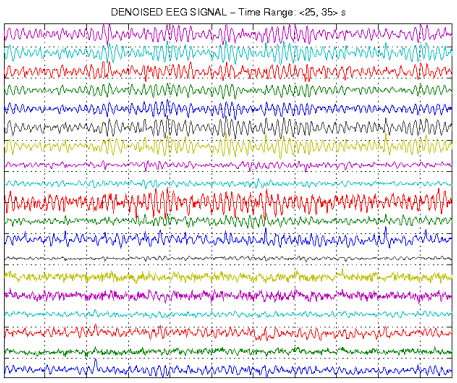


Figure 1: Example of EEG data

2.1 Electroencephalography (EEG)

Electroencephalography (EEG) is an electrophysiological method for monitoring electrical activity of the brain. It is typically non-invasive, with electrodes placed on the scalp. EEG measures voltage fluctuations resulting from ionic currents within the neurons on the brain surface [48]. In clinical use, EEG refers to recording of

the brain's spontaneous electrical activity over a period of time, as recorded from multiple electrodes placed at standardized positions on the scalp.

EEG data is one type of multivariate time series data produced from multiple channels. Each channel corresponds to one univariate time series. Fig. 1 shows an example of EEG data.

For EEG data, some signals are recognized based on their shape, spatial distribution over the scalp, and symmetry properties. Rhythmic and periodic patterns in EEG data the "ictal-interictal continuum" (IIC) [13, 29] are often associated with seizures.

Using Fourier analysis, signals can be decomposed into a spectrum of frequencies over a continuous range. After smoothing to remove noise, the resulting frequency-domain representation of the signal is called its spectrum. By using a sliding window and computing a series of spectra over time, and assembling these spectra into a time-frequency matrix, called a *spectrogram*, it is possible to provide a "compressed" or "zoomed out" view of the EEG. Whereas clinicians typically view "raw" EEG signals at a scale of 10-15 seconds / screen, spectrograms are typically viewed at a scale of 1-4 hours, providing temporal context. Spectrograms help experts to recognize patterns like seizures, and to review long EEG files efficiently [34, 47, 3].

3. SYSTEM OVERVIEW

As shown in Figure 2, our Smile system is composed of three key modules: data processing, labeling, and modeling.

The **data processing module** produces data used by the labeling module. In particular, the *EEG data processing* component divides the raw EEG data into segments of fixed duration and then loads the EEG segments into a key-value store. The *spectrogram generation* component computes the spectrum data with respect to each segment. The spectrogram images are produced using the spectrum data and stored in cloud storage. The *change point detection* component divides the EEG signal of each patient into flat chunks. In each chunk, only one segment is sampled for later labeling. This effectively reduces redundancy in the labeling effort of domain experts. All of these operations are conducted in a fully distributed fashion to cope with the big EEG data. The 2D coordinate generation component employs t-SNE [58] to embed the sampled high dimensional EEG segments into a 2D space for visualization. The 2D coordinates are stored in Postgres.

The **labeling module** corresponds to a large-scale time series data visualization system. It supports three views, namely the *2D Map view*, the *EEG view*, and the *spectrogram view*. The 2D map view allows users to interactively explore and label the sampled EEG segments. As long as an object is clicked in the 2D map view, the labeling system extracts the key of this object and then uses the key to pull out the corresponding raw EEG segment from the key-value store and spectrogram images from the cloud storage. The coordinated EEG view and spectrogram view then display the acquired data. These two views are updated simultaneously as the user pans the EEG signal through the EEG view. Labeled EEG segments are stored in a Postgres table. Besides this *free labeling mode*, the labeling module also supports a *more controlled mode* that allows users to upload a list of labeling candidates, recommended by the active learning component in modeling module.

The **modeling module** has two components, namely *deep learning* and *active learning*. First, using the labeled EEG segments produced by the labeling module, the deep learning component trains a deep neural network to classify the IIC patterns. The active learning component then recommends some EEG segments to the labeling system based on the output of the deep neural network. Modeling and labeling recur iteratively until an accurate IIC classification model is learned.

4. DATA PROCESSING

In this section, we introduce the data processing work done to label the EEG signals. Generally speaking this is a parallel data processing problem, since we are handling 30T time series data. A number of scaling techniques have to be leveraged to make the interactive labeling of EEG signals possible. Overall, the data preprocessing includes the processing of *raw EEG signals*, the *generation of the spectrogram*, *change point detection*, and *2D coordinate generation*.

4.1 Raw EEG Data

First, the raw EEG data is uploaded into cloud storage as Parquet [9] files. Each Parquet file contains part of the EEG data observed from one patient during their stay in ICU for a certain time period. One patient's data may be contained in multiple files. The files from the same patient are organized in the same directory. Each directory is named by the anonymized identity of the patient. Each file is named by the patient ID, the date, and the start time of the monitoring. For example, file 'emu100_20170517_080831' corresponds to the EEG data observed from patient emu100 starting at 08:08:31 of May 17, 2017. Each record in the file has at most 20 columns. Each column of one record corresponds to the values produced by one channel of the EEG in 5 milliseconds. Note different files maybe have different number of columns depending on the number of channels used in the monitoring of the patients.

EEG signals typically are are indexed by segments instead of at the individual record level, where each segment is composed of a fixed number of consecutive records observed over time. Accordingly, the typical lookup operation required for labeling is to find a segment based on the patient ID and the segment ID. Based on this observation, we store the EEG segments using Google Bigtable, which as key-value store, supports the key-based look up operation in constant time. Note the parquet files are still needed as the input to train the deep learning model.

The Structure of the Bigtable. The table is composed of rows. Each row describes a single entity with multiple columns. Each row is indexed by a single row key. Columns that are related to one another are typically grouped together into one column family. Each column is identified by a combination of the column family and a column qualifier unique within the column family.

Our Bigtable structure is shown in Figure 3. It contains one column family (eeg) and 20 columns under this column family. The 20 column qualifiers are ['fp1', 'f3', 'c3', 'p3', 'f7', 't3', 't5', 'o1', 'fz', 'cz', 'pz', 'fp2', 'f4', 'c4', 'p4', 'f8' ,'t4' ,'t6' ,'o2' ,'ekg'] representing data from 20 different channels. We use the column names in the Parquet file as the column qualifiers of Bigtable. In this way, when we ingest the data from file system, we are able to quickly map the columns in the Parquet file to the corresponding columns in the Bigtable. If one column does not exist in a Parquet file, the corresponding column in the Bigtable will be set as null. Given one record, each of its columns contains 400 values, corresponding to the signals produced by one channel over 2 seconds.

The Design of the Row Key. The design of the row key is essential for the efficiency of the look up operation. When labeling one EEG segment, the users typically also need to see the adjacent segments as the context. Therefore, ideally we want to make sure the adjacent segments are also next to each other in the Bigtable such that the neighboring segments could be pulled out via one single look up operation. Since in Bigtable the records are automatically sorted

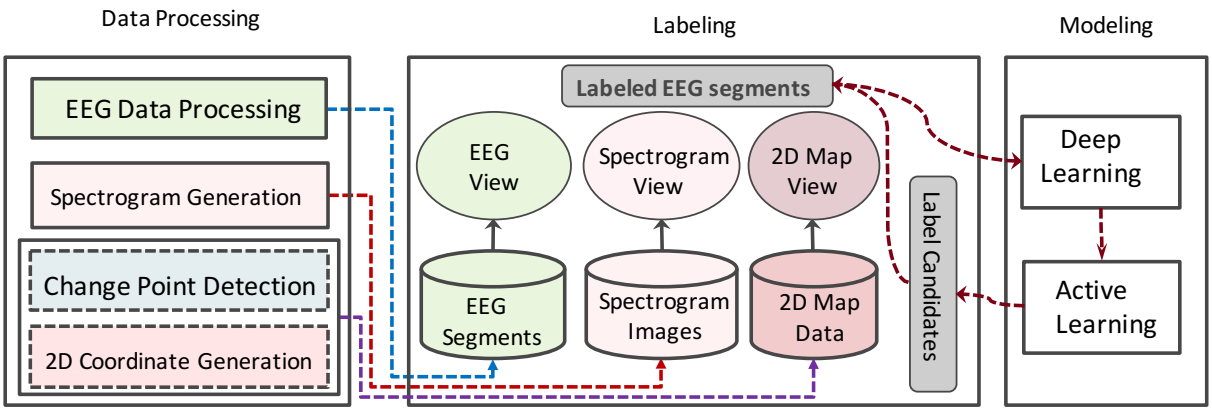


Figure 2: An overview of Smile system

	Column family: eeg			
Row Key	fp1	f3	...	ekg
abn10000_20140117_093552_000000	400 values	400 values	400 values	400 values
abn10000_20140117_093552_000001	400 values	400 values	400 values	400 values
abn10000_20140117_093552_000002	400 values	400 values	400 values	400 values

Figure 3: eegTable Structure

	COI-1	COI-2	COI-3	COI-4
Region-1	fp1 – f7	f7 – t3	t3 – t5	t5 – o1
Region-2	fp1 – f3	f3 – c3	c3 – p3	p3 – o1
Region-3	fp2 – f4	f4 – c4	c4 – p4	p4 – o2
Region-4	fp2 – f8	f8 – t4	t4 – t6	t6 – o2

Figure 4: Spectrograms of different regions

in the alphabetical order of the row keys, for any two adjacent segments, we have to make sure their row keys also are next to each other alphabetically. To satisfy this requirement, we design the row key as follows.

The row key is composed of two parts, namely patient identifier and segment id. The key lesson learned here is that the segment_id should be represented as a string in fixed length instead of a numerical integer value. For example, in the Bigtable, we expect the segment next to 'abn10000_20140117_093552_1' to be 'abn10000_20140117_093552_2'. However, if we used integer to represent segment_id, the segment next to 'abn10000_20140117_093552_1' would be 'abn10000_20140117_093552_10'.

Parallel Data Loading. Based on our experiments on a small sample set, sequentially loading all 30 TB of EEG data into BigTable will take more than 100 days. Therefore, we use a parallel mechanism to speed up the loading process. That is, we first divide the Parquet files in the cloud storage into multiple groups. Each group contains the similar number of files. Then the file groups are distributed to different compute nodes in a computing cluster to make sure that each processor of each node has one group of files to process. More specifically, we use a computing cluster with 40 compute nodes, each of which has 4 processors. We then divide the 42,609 files into 160 groups and distribute 4 groups to one node.

However, since the sizes of the Parquet files vary dramatically, this mechanism does not ensure a balanced workload across the compute nodes even if the same number of files are assigned to them. To solve this problem, we monitor the list of completed files for each node. If one node becomes idle, we re-distribute the work load by moving some of the files on the busy nodes to the idle nodes.

Using parallel data loading, we are able to load all 30 TB of EEG data within 24 hours.

4.2 Spectrogram Generation

In this section, we introduce the generation of the spectrograms of EEG data, i.e., 2D time-frequency maps. The frequency spectrum of spectrogram image varies with time. Different colors in the image represent different energy values. It corresponds to another form of feature representation of the raw EEG signals. Using such representations, we are able to visually demonstrate the temporal dynamics of EEG signals. In our application, it is considered as the long term context needed for the understanding of the to-be-labeled EEG segment. The spectrogram is produced in two steps: producing the spectrum data and then drawing the spectrogram images. Both steps are conducted in a fully distributed fashion, using a strategy similar to the one used in the processing of raw EEG data.

4.2.1 Spectrum Data

We generate 4 spectrograms with respect to 4 different regions. Each region contains multiple channels. Accordingly, its spectrogram is generated based on the EEG signals produced by these channels. As shown in Fig. 4, each region uses 4 channels of interest (COI in short). For instance, in the first region COI-1 represents the difference between channels 'fp1' and 'f7'.

The multitapering method [36] is then applied to compute spectrum estimates with respect to each channel. Essentially the orthogonal tapers used in the multitapering method correspond to Discrete Prolate Slepian Sequences (or DPSS for short) on the windowed time series. In our application, we set both the window size

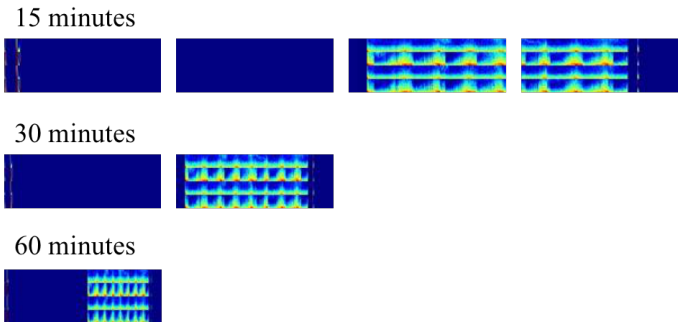


Figure 5: Spectrogram Example.

and the step size to 2 seconds. Each 2-second segment is decomposed into 50 frequencies. Given a patient file containing 1,200,000 records, where each record represents the signals produced in 5-milliseconds, the multitapering method will generate 3000 $\times$ 50 data points for each channel, where 3000 stands for the number of the 2-second segments and the 50 values per 2-second segment stand for the amplitudes of the 50 frequencies.

At the end, the regional averages are computed by averaging the spectrum data of all channels in this region.

4.2.2 *Spectrogram Images*

The spectrogram images could be produced by the visualization tool on the fly. That is, we could first pre-compute all spectrum data and then store the data in Bigtable. The visualization tool then fetches the corresponding data from Bigtable and draws the spectrograms for the requested EEG segment. However, this method cannot meet the response time requirement of interactive exploration. The reason is that drawing the 4 spectrograms requires a large number of data points. Specifically, to produce the spectrograms for one hour of EEG signals, we would have to retrieve 4 $\times$ $\times$ 50 $\times$ 1800 = 360K data points. Fetching and transferring these many points from Bigtable takes around 4 seconds, while rendering these points takes even longer on the front-end.

To solve this performance issue, we generate all spectrogram images beforehand store them in cloud storage. The visualization tool is then able to directly load the spectrogram images from the cloud storage via public image URLs provided by Google Cloud.

To increase the flexibility of the front end, we generate 3 independent sets of spectrogram images for consecutive windows at 15min, 30min, and 60min scales respectively. In this process, several decisions are made mainly for the saving of storage. First, no overlap is allowed between the adjacent windows. Second, the window always starts from t=0. Zero pads for the right-end residual if necessary. Each time scale results in images of the same dimensions, which are [150x450 pixels, 96dpi, 24bd]. The sizes of the images vary from 1K to 60K. Moreover, in order to reduce the storage costs, we format all images in '.jpg' format instead of '.png'. Finally, we remove both boarders and white-spaces for the convenience of seamless stitching. Fig. 5 shows an example of the generated spectrogram images for an 1 hour EEG segment.

In order to produce the 2D map, the high dimensional EEG segments have to be mapped to a 2D space. This is a two-step process including change point detection and dimension reduction.

4.3 Change Point Detection

Given the EEG signals from one patient, many adjacent 2-second segments in fact have very similar distribution characteristics. Therefore, they tend to share the same label. It is thus not necessary to label each of these segments one by one. Instead, only the segments that show different characteristics should be labeled. In this work, we use change point detection to discover these segments. The redundancy among the EEG segments from different patients is resolved in the active learning component as discussed later in Sec. 6.

A change point is a time instant at which some statistical property of a signal changes abruptly. Change point detection (CPD) [21, 41] is a general method to find abrupt changes in time series. The property in question can be the mean of the signal, its variance, or a spectral characteristic, among others.

More formally, assume we have an ordered sequence of data $y = (y_1, y_2, \ldots, y_n)$. Suppose there are m change points together with their positions, $\tau = (\tau_1, \tau_2, \ldots, \tau_m)$ where (1) each change point position is an integer between 1 and n 1 inclusive; (2) $\tau_0 = 0$ and $\tau_{m+1} = n$; and (3) the change points are ordered such that $\tau_i < \tau_j$ if and only if $i < j$. Consequently the m change points will split the data into $m+1$ chunks, with the i-th chunk containing $y_{(\tau_{i-1}+1):\tau_i}$. In this work, Pruned Exact Linear Time (Pelt) [31] is applied to detect change points. Pelt is designed to minimize:

$$\sum_{i=1}^{m+1} [\mathrm{C}(y_{(\tau_{i-1}+1):\tau_i}) + \beta] \quad (1)$$

Here C is a cost function for a chunk. Pelt involves a pruning step which reduces the computational cost of the method, while not impacting the exactness of the final results. The essence of pruning in this context is to remove those values of τ which can never be minima.

In this work, Pelt is applied on the spectral estimate data. As described in Sec. 4.2.1, 4 sets of spectral estimate data have been computed with respect to 4 different regions. We first convert the 4 sets of spectral estimate data to decibel scale and compute the total power average over the 4 regions. Then the total power average is smoothed by the Savitzky-Golay Filter [51] with window size = 10-sec (5 2-sec segments) and order = 3. The smoothed total power average is further clipped to range [-1000dB, 1000dB]. A random additive white noise is added to make sure the change point detection method works properly. Finally, the Pelt method is applied with segment model 'rbf'.

Between each pair of adjacent change points, only one EEG segment will be sampled for the later labeling. By this, in total 17M segments are selected out of the 350M segments.

Again, the change point detection is executed in parallel using the similar strategy to handling raw EEG data and producing the spectrogram images.

4.4 2D Coordinate Generation

Next, we produce the 2D coordinate for each sampled EEG segment, which is used in the 2D map view for the experts to visually select the segment to be labeled.

In visual analytics, t-SNE [58] is widely used to reduce the high dimensional feature vectors to 2D for display purpose. However, t-SNE scales quadratically in the number of objects N. Therefore, its applicability is limited to data sets with only a few thousand input objects. Beyond that, learning becomes too slow to be practical and the memory requirements become too large.

To solve this problem, many variations have been proposed to improve the efficiency of t-SNE, such as tree-based t-SNE [57], UMAP [45]. The key lesson learned here is that none of these approaches are able to handle the 17M EEG segments. These methods typically failed due to out-of-memory issues. Based on our experience, only parametric t-SNE [42] can handle data at this scale,

Column Name	Description	Type	Modifiers	Example
recordid	Record identifier, each record represents one segment	character varying(255)	not null	sid825_20141024_075301_024967
longitude	Longitude at the 2D map	numeric(6,3)		687.246
latitude	Latitude at the 2D map	numeric(6,3)		643.728
color	Predicted label	character varying(20)		Others

Figure 6: Table Structure of 2D Coordinates.

Column Name	Description	Type	Modifiers	Example
recordid	Record identifier, each record represents one segment	character varying(255)	not null	sid1228_20141113_134623_019810
doctorname	The doctor initials	character varying(20)	not null	ah
label	Doctor's Label	numeric(6,3)		Other
timestamp	The date & time for this label	timestamp with time zone	not null	2018-10-30 23:18:42.303335+00

Figure 7: Table Structure of Labeled Segments.

because it uses a neural network to reduce dimension. Since neural networks use batch processing, it is not necessary to hold all data in memory. However, parametric t-SNE is extremely slow. Using 10 high-end GPUs, it still takes one week to process the 17M objects.

Finally, the 2D coordinates produced by parametric t-SNE are re-scaled into range [0, 10000] for the display of the 2D bubble map. These coordinates are then loaded into Postgres with a table structure shown in Fig. 6. In this table, the 'color' attribute represents the label of the segment. Currently, we see 6 classes of label, namely 'Others', 'Seizure', 'GPD', 'LRDA', 'GRDA' and 'Artifact'. More classes could be supported in the future.

One important observation here is that to date no distributed t-SNE exists in the literature. Designing effective distributed solution to scale t-SNE to big data is a promising research problem.

4.5 The Label Table

The labeled segments are stored in a Postgres table. The table structure is shown in Fig. 7. In this table, one segment might correspond to multiple rows, because multiple doctors might label the same segment. In addition, one doctor might label one segment multiple times to correct the mistakes they made before. Therefore, to distinguish the labels marked by different doctors at different time, in this table we maintain the name of the doctor who labels the segment and the timestamp when the segment is labeled. Essentially we are recording the whole history of labeling for the future analysis.

5. A WEB-BASED VISUALIZATION PLATFORM FOR LABELING

To facilitate model training, we built a visualization platform for efficient labeling. This platform was created using Kyrix [56], a web-based toolkit for developing large-scale data visualization applications. Kyrix makes use of a simple client-server architecture. We host the server on Google Cloud.

In this section, we first describe the basic user interface in Section 5.1. We then describe in Section 5.2 how we apply Kyrix to build our labeling platform for visualizing large amounts of data. Lastly, Section 5.3 describes how our unique requirements drive the development of several new features of Kyrix.

5.1 UI Design

We designed our user interface (Figure 8) to enable quick retrieval of information necessary to make labels, and to accommodate doctors' habits when using commercial EEG viewers. Our UI is composed of four coordinated views as highlighted by the dashed boxes in Figure 8. We describe them in detail in the following.

2D bubble map. In the upper left-hand corner, we visualize the 2D bubble t-SNE map. This multi-class map shows two-second segments as colored circles. Each color represents a particular IIC pattern category. This 2D map is pannable and zoomable. The labeler can use simple mouse-based controls to navigate in this map (e.g. double click to zoom in). Clicking on any of the segment populates the EEG and spectrogram view with EEG/spectrogram centered at the selected segment.

Spectrogram view. The spectrogram view (lower left-hand corner) displays the spectrogram images stored in Google Cloud. Kyrix stitches the image segments to form a consecutive spectrogram. The panning of this view is coordinated with the panning of the EEG view.

EEG view. The EEG view displays eight two-second segments at a time. The segment in the middle is highlighted which indicates to the labeler that a label can be made to this segment. We implemented a set of fine controls over this EEG viewer to assist the labeler in tweaking the time series to make better labeling decisions. For example, by pressing up/down arrows, the labeler is able to increase/decrease the magnitude of the time series. The labeler can also switch between different montages of the EEG by pressing the key M.

Labeling box. The box in the upper right-hand corner is where the labeler makes labels. On the top, an identifier of the highlighted 2-second segment is shown. Following that are radio buttons for making the labels. Every time a radio button is clicked, a request is sent to the server to record the label in the database. The labeler can click on a cancel button to revert his/her last made label.

5.2 Using Kyrix to Handle Large Data

The key challenge in building this web UI is to enable interactive browsing of large amounts data for the labeler. As demonstrated by prior research [38], it is important to bound the response times to user interactions (e.g. pan and zoom) within 500 ms to obtain a

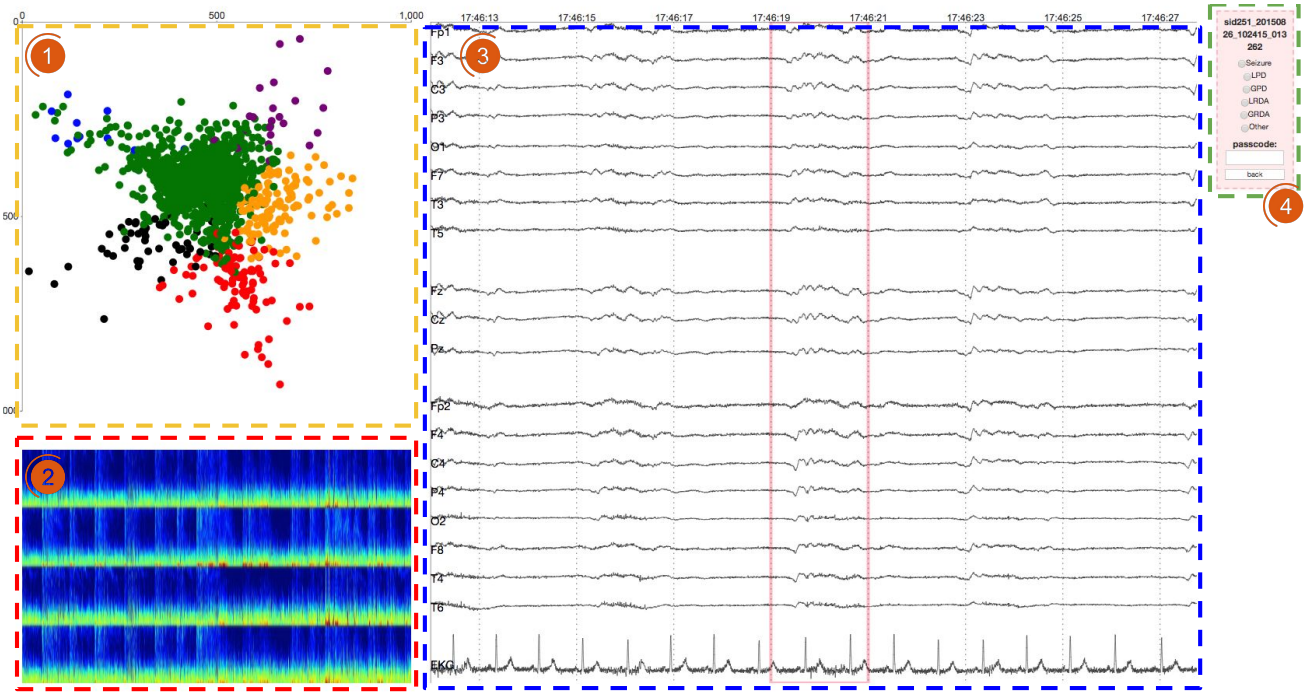


Figure 8: An illustration of our web-based platform, created using Kyrix. Four main parts of the UI: (1) the 2D bubble map showing 2-second segments; (2) the spectrogram view centered at a selected 2-second segment; (3) EEG time series centered at a selected 2-second segment; (4) the panel for making labels.

fluid visual experience. To meet this latency requirement, we apply Kyrix [56], an open-source visualization system designed to ease the creation of large-scale pan/zoom data visualizations. Kyrix employs a client-server architecture, where the client communicates with the server to fetch data in the user's viewing region to render the visualizations. The server then fetches data from storage systems (i.e. PostgreSQL and Bigtable in our application).

Kyrix allows the visualization developer to declaratively specify the visualizations using a visualization grammar. Our UI perfectly fits into the type of visualization that Kyrix supports, so it is straightforward to write the specifications. The Kyrix backend server applies a suite of optimization techniques (e.g. prefetching, spatial indexing) to ensure only data in the viewport is fetched. These optimization techniques are crucial to ensure the 500 ms latency requirement and are completely transparent to the visualization developer. Interested readers can refer to the paper [56] or the Github repo[1] for more technical details of the Kyrix system.

5.3 Extending Kyrix to Support New Features

When we started building our visualization, there were several desired features that were not supported by Kyrix. In the following, we describe these features in detail and discuss how we extended Kyrix to support them.

Multiple coordinated views. Kyrix was originally only able to produce single-view visualizations. In our UI design, we desire to have multiple coordinated pan/zoom views so that it is easier for the labeler to grab information. More specifically, we require a "load coordination" (e.g. load the EEG view when the labeler clicks on the 2D map view) and "synchronized scrolling" between the spectrogram view and the EEG view. The change made to support coordinated view has been contributed back to the open source community via GitHub.

To support multiple views with aforementioned coordinations, we have made a major extension to Kyrix. Here we give an overview of the changes we made to the three main components of Kyrix: (1) the compiler that parses visualization specifications, (2) the backend that precomputes database indexes and (3) the frontend for rendering the visualizations.

The compiler offers a concise declarative visualization grammar, which employs two main abstractions *canvas* and *jump*. A canvas can be seen as a zoom level, whereas a jump indicates there is a zoom transition between two canvases. To support multiple coordinated views, we add to the grammar a *view* abstraction that allows specifications of views, their sizes and relative positions, canvases initially assigned to a view, etc. We also allow specifications of view coordinations by associating a jump to some views.

The backend server precomputes indexes on a per-canvas basis. Also, the frontend only requests data in a particular canvas, although the canvas may be displayed in different views. Therefore, there was little change made to the backend code, except some part of the data fetching logic where view-specific information (e.g. size, position) was needed.

The frontend underwent two major changes. First, we needed to set up multiple views based on user specifications and add one *view_id* argument to almost every frontend function to enable view-dependent rendering. Second, we enriched the implementation of the jump function to support coordinations between views.

An automatic drill-down system. To organize the 2D bubble map into multiple canvases (zoom levels) where the labeler can zoom in (out) to see more (fewer) segments, we needed to manually sample

[1]https://github.com/tracyhenry/Kyrix

the segments for each canvas. This turned out to be a tedious process because the distribution of the segments was fairly skewed and simple random sampling could easily lead to crowded display. We desired to avoid large visual density because it could slow down both the frontend and the backend. We ended up repeatedly experimenting with different setups (e.g. number of zoom levels, zoom factors between levels) in order to find a satisfying configuration.

This experience has driven us to look for algorithms that can automatically organize a potentially skewed dataset into multiple zoom levels. We are working on using sampling methods [6, 11, 17] to generate non-overlapping designs where visual objects on each zoom level do not occlude each other. We plan to investigate different methods, compare their performance and integrate the algorithms into Kyrix's declarative grammar as high level abstractions.

Parallel PostgreSQL. To handle larger datasets which cannot be accommodated by one single machine, we needed to support clusters of machines running in parallel. At the time, Kyrix only supported single-node PostgreSQL and MySQL, so we had to hard-code some logic to fetch data from Bigtable. We have experimentally extended Kyrix to support Citus [1], an open source system for running PostgreSQL across clusters of machines and automatically parallelizing queries. For system administration and deployment, we have extended Citus to inside Docker containers on Kubernetes, which allows any developer to quickly deploy this multi-node system on rented cloud computing facilities, including Amazon Web Services, Google Cloud, Microsoft Azure, Digital Ocean and others. Note that Citus is implemented as a standard Postgres extension, and virtually all single-node features continue to work as normal, including the spatial indexing methods required by Kyrix as well as hundreds of traditional RDBMS features. Because of this, basic support for parallelism only required one additional SQL statement to distribute the data across the cluster.

In the future the EEG Bigtable will be migrated from Bigtable to Citus, while the Kyrix team will add configuration options and tools for development and system administration, etc.

One interesting decision is how to distribute ("shard") the Kyrix data across a cluster. One method is to shard geospatially, where visual objects "near" each other are co-located in the same shard (i.e. on the same server). On one hand, this improves multi-user performance by only burdening a smaller number of servers with that user's request traffic. On the other hand, if there is skew in the data or skew in the popularity of the data, then a large percentage of servers will be idle. Moreover, in the case where the Kyrix deployment has ample resources (nodes) per user, then geospatial sharding does not improve that individual user's performance. Another method is random sharding (e.g. round-robin), which has the benefit of potentially improving per-user performance by performing less work per node. The final sharding option we are considering is data-dependent sharding, where the Kyrix administrator who uploads the dataset chooses a per-dataset method to shard, optionally including a function (code) to determine the shard. In terms of performance, data dependent sharding is the best option and can be used to implement the geospatial and random options, but it comes at the cost of adding substantial complexity for administrators. Benchmark testing is needed to evaluate these strategies.

6. ACTIVE LEARNING

Although we have large volumes of EEG data obtained in the course of monitoring ICU patients, high-quality labels (Section 4.5) are not initially available. Moreover, inter-rater agreement among multiple experts regarding EEG segment patterns is often low. Therefore, we need an efficient framework to collect and update the segment labels iteratively, using human time as judiciously as possible. Inspired by active learning [53], we use a framework where the trained machine learning model, a deep neural network in our case, suggests candidate objects to be labeled again, and multiple domain experts review and relabel them. During each iteration, we improve the quality of the labels as well as inter-rater agreement, leading to more accurate classification.

Figure 9: An overview of the active learning framework. This process is iteratively performed until it reaches a certain level of convergence.

Figure 9 shows the overall process of our relabeling framework: Assume we have a small set of labels provided by the physicians during an initial labeling pass. We then cluster the EEG segments and propagate these labels within the "pure" clusters, namely the clusters in which the labeled segments share the same label. By this we get a set of "noisy" labels.

First, we use a subset of the labeled data as training set to train a deep neural network model. All labeled segments are then examined at inference time by the trained model. We then extract the k segments where the neural net had highest confidence but disagreed with the labels. These segments are given to domain experts to review and potentially update their labels.

In addition, the inference results can also be used to acquire new labels, such as suggesting the experts to label the unlabeled segments on which the model has a low confidence.

6.1 Model Training

Deep neural networks. We use a deep neural network model in the active learning framework for relabeling the EEG segments. Overall, the model consists of two parts: a convolutional neural network (CNN) module [35, 33] to recognize local patterns and a recurrent neural neural network (RNN) module [25] to capture the global characteristics of the given EEG signal.

CNNs have shown impressive representational power over the past decade, especially in computer vision applications [35, 33]. Inspired by previous works that successfully apply CNNs in clinical signal processing applications such as arrhythmia detection using electrocardiogram (ECG) [22] and EEG analysis [52], we use a CNN module to extract local pattern features that represents short term patterns over a few seconds.

While CNNs have shown their great representational power for local feature extraction, it has been shown that RNNs, especially the gated variants such as Long Short-term Memory (LSTM) [25] and Gated Recurrent Units (GRUs) [14], typically have better abilities than CNNs in settings with sequential or temporal dynamics such as machine translation [55]. Therefore, we utilize a RNN module to summarize long term (over a few minutes) trends in EEG segments. In addition, we stack the RNN module on top of our CNN module to take advantage of each module simultaneously, following the motivation of the previous work on sleep staging using spectrograms of EEG signals [7].

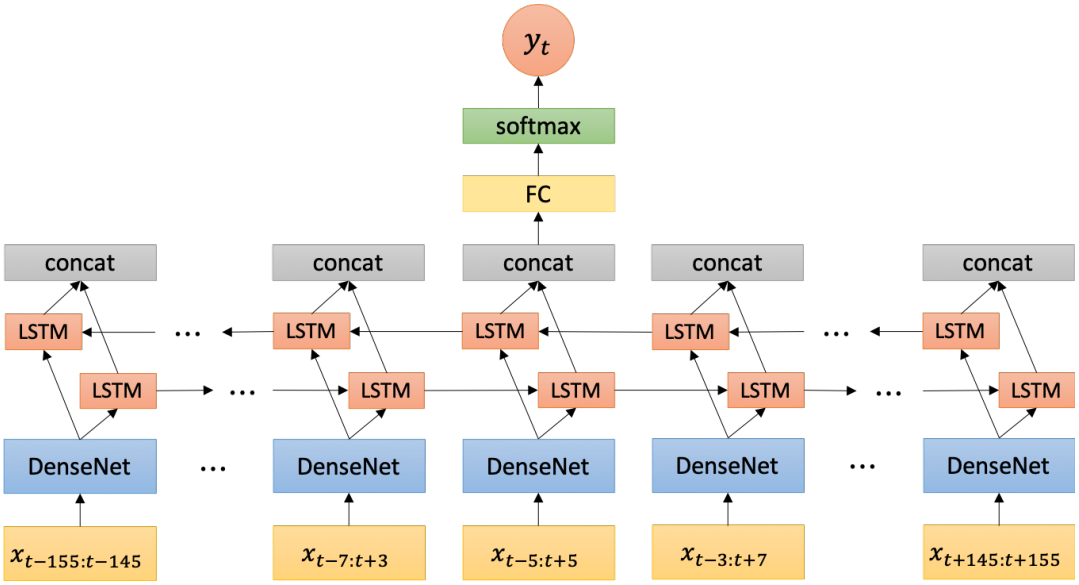


Figure 10: Unrolled depiction of the deep neural network architecture used in the active learning for relabeling process. Here a `concat` block represents a concatenation of outputs from each direcion of LSTM module and a `FC` block refers to a fully-connected layer.

Implementation. Figure 10 depicts the architecture of the deep neural network used. For the CNN module, we adapt a DenseNet-BC architecture [26] with 7 dense blocks each of which consists of 4 layers with a growth rate of $k = 32$. On top of it, a layer of bidirectional long short-term memory (LSTM), which contains 8 hidden units for each direction, is stacked for the RNN module.

To label each 2 second target EEG segment x_t, 77 segments before and after x_t are provided as context, meaning a total of 155 segments, each 310 seconds in length ($x_{t-155:t+155}$) are used as input data to the model. Specifically, these segments are split into 5-segment chunks, and each chunk is passed into the DenseNet module that generates 255-dimensional local features. Then, the bidirectional LSTM module uses 31 local features to calculate a global temporal characteristic. The hidden states of each direction at the middle of the sequence are and input to a 16-dimensional fully connected layer. Finally, the class probabilities are calculated using a softmax function.

6.2 Relabeling

Once the model is trained on the current dataset, all labeled 2-sec EEG segments (in training, validation, and test sets) are evaluated by the model and relabeled through the following procedure: First, all segments having incorrect model predictions are gathered into groups having the same pair of label and model prediction. Second, a clustering algorithm, e.g., k-means [44], is applied to each group to create a certain number of clusters, e.g, 1000, in total across the groups. Third, the data points closest to each cluster center are extracted, while all cluster membership information is also recorded for updating labels in a later phase. Fourth, the domain experts are asked to review and relabel only the closest data points from the cluster centers. Similar to change point detection introduced in Sec. 4.3 which avoids the labeling of the similar segments from *the same* patient, this reduces the labeling load by avoiding review of many similar segments from *different* patients. Finally, the labels for all cluster members are updated together using the labels reassigned to the center-closest data points by the domain experts. Then, the model is fine-tuned or retrained using the updated dataset. The similar strategy can also be applied to label the unlabeled segments on which the model has low confidence.

7. EXPERIMENTAL EVALUATION

In the experiments, we evaluate both the effectiveness of our active learning strategy and the response time of the visualization component for labelling EEG segments.

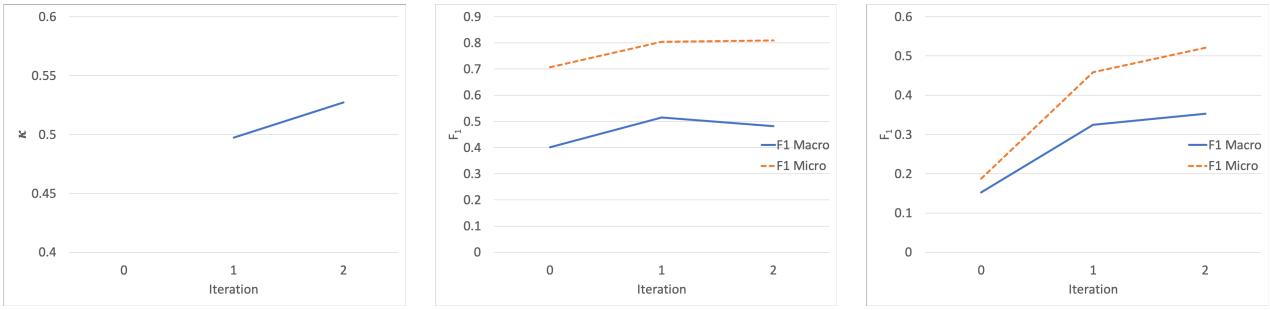
Active Learning Evaluation. We conducted experiments that clinical experts iteratively relabel the EEG segments through the active learning framework described in the previous section. At the beginning of the experiment, called iteration 0, the deep neural network model described in the Section 6.1 was trained using the labeled portion of the EEG segments data with the initial labels. Combining the two classes of Others and Artifact, we used the five classes: 'O/A', 'Seizure', 'LPD', 'GPD', 'LRDA', and 'GRDA'. In addition, we split the entire labeled dataset into three disjoint subsets of training, validation, and test set with a ratio of 5:1:2 respectively. Although the EEG segments from any of those three subsets can be relabeled over the iterations, their membership is not changed.

After the initial model had been trained at iteration 0, the relabeling process was executed for the next two iterations with three neurological experts. Each expert reviewed and relabeled the same segments suggested by the model at each iteration; majority votes were used when the labels were actually updated. We use Fleiss' kappa (κ) [19] to assess the inter-rater agreement among the three experts. Figure 11a shows that the inter-rater agreement measured by Fleiss' kappa increases over the two iterations of relabeling.

For the model performance on the given dataset at each iteration, we use F_1 score, which is a popular metric on a classification problem especially with an unbalanced dataset, i.e., the distribution of the classes is not uniform. In a binary classification problem, F_1 score is given by the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

In a multi-class problem like our case, one may use either macro-average or micro-average to obtain a single score for each metric,



(a) Inter-rater agreement by Fleiss' κ (b) F_1 scores on the entire test set (c) F_1 scores on the relabeled test set only

Figure 11: Performance metrics measured on the inter-rater agreements and the model predictions.

precision and recall, in the equation above. Macro-average is obtained by calculating a metric for each class separately and taking their unweighted mean value. On the other hand, micro-average can be found by counting the total true positives, false negatives and false positives for all classes and calculating a metric. While macro-average gives an overview of the performance without considering the class imbalance, micro-average implicitly considers the contribution of each class. We report both macro-averaged and micro-averaged F_1 score to avoid either overestimation or underestimation of the model performance. Figure 11b shows both F_1 scores for entire test data. While both scores increase as more relabeling iterations are performed, the micro-averaged F_1 score is consistently higher than the macro-averaged one. This is because the model has a better prediction performance on the majority class than the other classes. Figure 11c shows the same F_1 scores but on the test segments that were relabeled. Both scores were very low before any relabeling process, and they rapidly increase as relabeling progresses. This gives us important intuition that the relabeled segments were initially either incorrectly labels or very difficult for the model to correctly classify. These labels have been corrected by the experts as more relabeling iterations are executed, and the model becomes more accurate as it is trained and evaluated on the corrected labels. Overall, this shows that we are able to achieve better model performance by incrementally reviewing and correcting the labels.

Visualization Response Times. We also conducted experiments to evaluate the average response times (to client requests) of our visualization platform. Response time is composed of two parts: network and data fetching. Data fetching time is defined as the time elapsed from the Kyrix backend receiving the request to the Kyrix backend getting the data from the storage systems. Network time is defined as the time it takes to send the data back to the frontend.

Table 1: Average response times of the visualization platform.

View	Avg network time (ms)	Avg data fetching time (ms)	Avg response time (ms)
2D Map	67.1	23.8	90.9 ($\sigma = 174.4$)
EEG	202.2	211.5	413.7 ($\sigma = 174.7$)
Spectrogram	0.2	331.8	331.9 ($\sigma = 117.1$)

A clinical expert labeled 63 2-second segments using the visualization platform. The average response time on each of the three main views (i.e. 2D map, EEG, and spectrogram) is reported in Table 1. We can see from the table that our average response time was below 500 ms, which ensured interactive visual browsing for the labeler. Note that the data fetching took much longer than network for the spectrogram view. The reason was that the Kyrix backend needed to communicate with Google Cloud whether a spectrogram image exists (which contributes to the long data fetching time) and that only small-sized image links were sent back to the frontend.

8. RELATED WORK

IIC pattern classification. Small scaled analysis has been conducted on IIC pattern classification. In [28], unsupervised machine learning methods were applied to pre-clustering the EEG segments to facilitate the labeling by experts. Although this method could significantly speed up the labeling process, the produced labels tend to be noisy due to inter-rater disagreement and ambiguous cluster boundaries. In this work, we use a large scale visualization system and active learning to improve the quality of labeling.

Labeling. As the key component of a machine learning system, labeling has drawn a lot of research attentions in recent years. In [50], Snorkel was proposed to efficiently produce a large amount of labels based on the business rules written down by the users. Statistical methods are then applied to remove the noises in the produced labels. However, in our complex IIC classification context, it is extremely hard even for the Neurology specialists to write down rules to explicitly define the characteristics of different IIC patterns. Following a different direction, Snuba [59] first uses a small number of labels to train a bunch of lightweight machine learning models. A large number of labels are then produced leveraging the inferences results from these small models. More specifically, given one testing object, if most of the models agree on the prediction, then the prediction will be used as the ground truth of this object. However, this methodology tends to produce a large number of redundant labels which not necessarily lead to an enriched model covering complex and subtle cases. In many cases, labels from the boundary or ambiguous cases are more desired than the ease cases to improve the predication accuracy of a classification model. In this work, our Smile system employs active learning to discover and resolve the boundary cases. Furthermore, none of these approaches consider the labeling of time series data.

Active Learning. Active learning [53, 27, 46, 61, 18] studies the problem of recommending the most promising objects for labeling based on the output of the modeling. Typically, these techniques recommend objects based on the uncertainty, namely labeling the objects that the current model is unsure about. The diversity of the labeled objects are also taken into consideration. In this project, we will further investigate how to leverage various active learning techniques to reduce the noises in the labels and save the labeling efforts of the experts.

Visualization. Big data visualization has been a hot research topic for more than a decade. Multiple efforts have studied or built systems/toolkits to support scalable visualizations of various kinds. A long line of research has studied how to apply data cube or sampling algorithms to efficiently render aggregate visualiza-

tions (e.g. bar charts) [39, 54, 37, 49, 60, 32, 43]. A number of systems also focus on pan/zoom-based details-on-demand visualizations [4, 10, 12, 30]. Despite the seemingly abundance of the tools/systems, very few of them can fully support the unique requirements posed by our design. For one, a great deal of general visual analytics tools cannot support data size larger than main memory [5]. For another, many pan/zoom systems (e.g. ForeCache [4], HiGlass [30] and Google Maps) are hardcoded for specific data types and visualizations. In addition, most of them do not support multiple coordinated views. We pick Kyrix [56] for implementing our visualization platform due to its support for general-purpose pan/zoom applications at scale. New features such as multiple coordinated views are designed to enhance Kyrix to meet the customized requirements of exploring and labeling EEG data.

9. CONCLUSION

In this work, we explore big data analysis, visual analytics, and cutting-edge deep learning techniques to solve a hard yet important medical problem, namely automatically detecting the seizures and non-convulsive seizures by analyzing the EEG signals. Big data techniques are leveraged to slice the EEG data and load the EEG segments to key-value store, transfer the time series to frequency domain for producing the spetrogram images, and embed the EEG segment to 2D space for visualization, etc. A visualization based labeling system is developed for the users to interactively explore and label the big EEG data with the support of multiple coordinated views. Active learning techniques are then employed that based on the output of a deep learning model, iteratively produce more and more high quality labels and improves the classification accuracy.

10. ACKNOWLEDGMENTS

We would like to thank Google, Microsoft, and Intel for supporting our Data System and AI Lab (DSAIL). This project is also supported in part by MGH-MIT Strategic Initiative Grand. We are also grateful to Google Cloud for the GCP credits.

11. REFERENCES

[1] Citus data. `https://www.citusdata.com/`.

[2] R. Agarwal, J. Gotman, D. Flanagan, and B. Rosenblatt. Automatic eeg analysis during long-term monitoring in the icu. *Electroencephalography and clinical Neurophysiology*, 107(1):44–58, 1998.

[3] E. Amorim, C. A. Williamson, L. M. Moura, M. M. Shafi, N. Gaspard, E. S. Rosenthal, M. M. Guanci, V. Rajajee, and M. B. Westover. Performance of spectrogram-based seizure identification of adult eegs by critical care nurses and neurophysiologists. *Journal of clinical neurophysiology: official publication of the American Electroencephalographic Society*, 34(4):359–364, 2017.

[4] L. Battle, R. Chang, and M. Stonebraker. Dynamic prefetching of data tiles for interactive visualization. In *SIGMOD*, pages 1363–1375, New York, NY, USA, 2016.

[5] M. Behrisch, D. Streeb, F. Stoffel, D. Seebacher, B. Matejek, S. H. Weber, S. Mittelstaedt, H. Pfister, and D. Keim. Commercial Visual Analytics Systems-Advances in the Big Data Analytics Field. *TVCG*, pages 1–1, 2018.

[6] C. Beilschmidt, T. Fober, M. Mattig, and B. Seeger. A linear-time algorithm for the aggregation and visualization of big spatial point data. In *SIGSPATIAL*, page 73. ACM, 2017.

[7] S. Biswal, H. Sun, B. Goparaju, M. B. Westover, J. Sun, and M. T. Bianchi. Expert-level sleep scoring with deep neural networks. *Journal of the American Medical Informatics Association*, 25(12):1643–1650, 2018.

[8] G. Bodenstein and H. M. Praetorius. Feature extraction from the electroencephalogram by adaptive segmentation. *Proceedings of the IEEE*, 65(5):642–652, 1977.

[9] A. Boufea, R. Finkers, M. van Kaauwen, M. Kramer, and I. N. Athanasiadis. Managing variant calling files the big data way: Using HDFS and apache parquet. In *BDCAT, Austin, TX, USA, December 05 - 08, 2017*, pages 219–226, 2017.

[10] S.-M. Chan, L. Xiao, J. Gerth, and P. Hanrahan. Maintaining interactivity while exploring massive time series. In *VAST*, pages 59–66, 2008.

[11] H. Chen, W. Chen, H. Mei, Z. Liu, K. Zhou, W. Chen, W. Gu, and K.-L. Ma. Visual abstraction and exploration of multi-class scatterplots. *TVCG*, 20(12):1683–1692, 2014.

[12] D. Cheng, P. Schretlen, N. Kronenfeld, N. Bozowsky, and W. Wright. Tile based visual analytics for twitter big data exploratory analysis. In *IEEE Big Data, 2013*, pages 2–4. IEEE, 2013.

[13] D. J. Chong and L. J. Hirsch. Which eeg patterns warrant treatment in the critically ill? reviewing the evidence for treatment of periodic epileptiform discharges and related patterns. *Journal of Clinical Neurophysiology*, 22(2):79–91, 2005.

[14] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[15] J. Claassen, S. Mayer, R. Kowalski, R. Emerson, and L. Hirsch. Detection of electrographic seizures with continuous eeg monitoring in critically ill patients. *Neurology*, 62(10):1743–1748, 2004.

[16] M. C. Cloostermans, C. C. de Vos, and M. J. van Putten. A novel approach for computer assisted eeg monitoring in the adult icu. *Clinical neurophysiology*, 122(10):2100–2109, 2011.

[17] A. Das Sarma, H. Lee, H. Gonzalez, J. Madhavan, and A. Halevy. Efficient spatial sampling of large geographical tables. In *SIGMOD*, pages 193–204, 2012.

[18] B. Du, Z. Wang, L. Zhang, L. Zhang, W. Liu, J. Shen, and D. Tao. Exploring representativeness and informativeness for active learning. *IEEE Trans. Cybernetics*, 47(1):14–26, 2017.

[19] J. L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[20] N. Gaspard, L. J. Hirsch, S. M. LaRoche, C. D. Hahn, M. B. Westover, and C. C. E. M. R. Consortium. Interrater agreement for critical care eeg terminology. *Epilepsia*, 55(9):1366–1373, 2014.

[21] V. Guralnik and J. Srivastava. Event detection from time series data. In *KDD*, pages 33–42, 1999.

[22] A. Y. Hannun, P. Rajpurkar, M. Haghpanahi, G. H. Tison, C. Bourn, M. P. Turakhia, and A. Y. Ng. Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nature medicine*, 25(1):65, 2019.

[23] L. Hirsch, S. LaRoche, N. Gaspard, E. Gerard, A. Svoronos, S. Herman, R. Mani, H. Arif, N. Jette, Y. Minazad, et al. American clinical neurophysiology societys standardized critical care eeg terminology: 2012 version. *Journal of clinical neurophysiology*, 30(1):1–27, 2013.

[24] L. J. Hirsch. Continuous eeg monitoring in the intensive care unit: an overview. *Journal of clinical neurophysiology*, 21(5):332–340, 2004.

[25] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[26] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, pages 4700–4708, 2017.

[27] S. Huang, R. Jin, and Z. Zhou. Active learning by querying informative and representative examples. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(10):1936–1949, 2014.

[28] J. Jing, E. Angremont, S. Zafar, E. S. Rosenthal, M. Tabaeizadeh, S. Ebrahim, J. Dauwels, and M. B. Westover. Rapid annotation of seizures and interictal-ictal continuum eeg patterns. In *EMBC*, pages 3394–3397. IEEE, 2018.

[29] E. L. Johnson and P. W. Kaplan. Population of the ictal-interictal zone: The significance of periodic and rhythmic activity. *Clinical neurophysiology practice*, 2:107–118, 2017.

[30] P. Kerpedjiev, N. Abdennur, F. Lekschas, C. McCallum, K. Dinkla, H. Strobelt, J. M. Luber, S. B. Ouellette, A. Azhir, N. Kumar, et al. Higlass: web-based visual exploration and analysis of genome interaction maps. *Genome biology*, 19(1):125, 2018.

[31] R. Killick, P. Fearnhead, and I. A. Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012.

[32] A. Kim, E. Blais, A. G. Parameswaran, P. Indyk, S. Madden, and R. Rubinfeld. Rapid sampling for visualizations with ordering guarantees. *PVLDB*, 8(5):521–532, 2015.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[34] J. S. Kumar and P. Bhuvaneswari. Analysis of electroencephalography (eeg) signals and its categorization–a study. *signal*, 25:26.

[35] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS*, pages 396–404, 1990.

[36] J. M. Lees and J. Park. Multiple-taper spectral analysis: A stand-alone c-subroutine. *Computers & Geosciences*, 21(2):199–236, 1995.

[37] L. Lins, J. T. Klosowski, and C. Scheidegger. Nanocubes for real-time exploration of spatiotemporal datasets. *IEEE TVCG*, 19(12):2456–2465, 2013.

[38] Z. Liu and J. Heer. The effects of interactive latency on exploratory visual analysis. *TVCG*, 20(12):2122–2131, 2014.

[39] Z. Liu, B. Jiang, and J. Heer. imMens: Real-time visual querying of big data. *Comput. Graphics Forum*, 32:421–430, 2013.

[40] S. S. Lodder and M. J. van Putten. Quantification of the adult eeg background pattern. *Clinical neurophysiology*, 124(2):228–237, 2013.

[41] R. Lund, X. L. Wang, Q. Q. Lu, J. Reeves, C. Gallagher, and Y. Feng. Changepoint detection in periodic and autocorrelated time series. *Journal of Climate*, 20(20):5178–5190, 2007.

[42] L. Maaten. Learning a parametric embedding by preserving local structure. In *Artificial Intelligence and Statistics*, pages 384–391, 2009.

[43] S. Macke, Y. Zhang, S. Huang, and A. G. Parameswaran. Adaptive sampling for rapidly matching histograms. *PVLDB*, 11(10):1262–1275, 2018.

[44] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[45] L. McInnes and J. Healy. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[46] P. Melville and R. J. Mooney. Diverse ensembles for active learning. In *ICML*, pages 74–, New York, NY, USA, 2004.

[47] L. M. Moura, M. M. Shafi, M. Ng, S. Pati, S. S. Cash, A. J. Cole, D. B. Hoch, E. S. Rosenthal, and M. B. Westover. Spectrogram screening of adult eegs is sensitive and efficient. *Neurology*, 83(1):56–64, 2014.

[48] E. Niedermeyer and F. L. da Silva. *Electroencephalography: basic principles, clinical applications, and related fields*. Lippincott Williams & Wilkins, 2005.

[49] C. A. L. Pahins, S. A. Stephens, C. Scheidegger, and J. L. D. Comba. Hashedcubes: Simple, low memory, real-time visual exploration of big data. *TVCG*, pages 671–680, 2017.

[50] A. Ratner, S. H. Bach, H. R. Ehrenberg, J. A. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. *PVLDB*, 11(3):269–282, 2017.

[51] A. Savitzky and M. J. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.

[52] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball. Deep learning with convolutional neural networks for eeg decoding and visualization. *Human brain mapping*, 38(11):5391–5420, 2017.

[53] B. Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

[54] C. Stolte, D. Tang, and P. Hanrahan. Multiscale visualization using data cubes. In *INFOVIS*, pages 7–14, 2002.

[55] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[56] W. Tao, X. Liu, Ç. Demiralp, R. Chang, and M. Stonebraker. Kyrix: Interactive visual data exploration at scale. In *CIDR*, 2019.

[57] L. van der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of Machine Learning Research*, 15:3221–3245, 2014.

[58] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[59] P. Varma and C. Ré. Snuba: Automating weak supervision to label training data. *PVLDB*, 12(3):223–236, 2018.

[60] M. Vartak, S. Rahman, S. Madden, A. G. Parameswaran, and N. Polyzotis. SEEDB: efficient data-driven visualization recommendations to support visual analytics. *PVLDB*, 8(13):2182–2193, 2015.

[61] Z. Wang and J. Ye. Querying discriminative and representative samples for batch mode active learning. In *SIGKDD*, pages 158–166, 2013.