TemPredict: A Big Data Analytical Platform for Scalable Exploration and Monitoring of Personalized Multimodal Data for COVID-19

1st Shweta Purawat
SDSC
UCSD, La Jolla, USA
shpurawat@ucsd.edu

2nd Subhasis Dasgupta
SDSC
UCSD, La Jolla, USA
sudasgupta@ucsd.edu

3rd Jining Song
SDSC
UCSD, La Jolla, USA
jis269@ucsd.edu

4th Shakti Davis
Lincoln Laboratory
MIT, Boston, USA
shakti@ll.mit.edu

5th Kajal T. Claypool
Lincoln Laboratory
MIT, Boston, USA
claypool@ll.mit.edu

6th Sandeep Chandra
SDSC
UCSD, La Jolla, USA
chandras@sdsc.edu

7th Ashley Mason
Department of Psychiatry
UCSF, San Francisco, USA
Ashley.Mason@ucsf.edu

8th Varun Viswanath
HDSI
UCSD, La Jolla, USA
varunv9@eng.ucsd.edu

9th Amit Klein
Department of Bioengineering
UCSD, La Jolla, USA
a3klein@ucsd.edu

10th Patrick Kasl
Department of Bioengineering
UCSD, La Jolla, USA
pkasl@ucsd.edu

11th YingJing Wen
HDSI
UCSD, La Jolla, USA
y2wen@ucsd.edu

12th Benjamin Smarr
HDSI
UCSD,La Jolla, USA
bsmarr@eng.ucsd.edu

13th Amarnath Gupta
SDSC
UCSD, La Jolla, USA
a1gupta@ucsd.edu

14th Ilkay Altintas
SDSC and HDSI
UCSD, La Jolla, USA
ialtintas@ucsd.edu

2021 IEEE International Conference on Big Data (Big Data) | 978-1-6654-3902-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/BigData52589.2021.9671441

***Abstract*—A key takeaway from the COVID-19 crisis is the need for scalable methods and systems for ingestion of big data related to the disease, such as models of the virus, health surveys, and social data, and the ability to integrate and analyze the ingested data rapidly. One specific example is the use of the Internet of Things and wearables (i.e., the Oura ring) to collect large-scale individualized data (e.g., temperature and heart rate) continuously and to create personalized baselines for detection of disease symptoms. Individualized data, when collected, has great potential to be linked with other datasets making it possible to combine individual and societal scale models for further understanding the disease. However, the volume and variability of such data require novel big data approaches to be developed as infrastructure for scalable use. This paper presents the data pipeline and big data infrastructure for the TemPredict project, which, to the best of our knowledge, is the largest public effort to gather continuous physiological data for time-series analysis. This effort unifies data ingestion with the development of a novel end-to-end cyberinfrastructure to enable the curation, cleaning, alignment, sketching, and passing of the data, in a secure manner, by the researchers making use of the ingested data for their COVID-19 detection algorithm development efforts. We present the challenges, the closed-loop data pipelines, and the secure infrastructure to support the development of time-sensitive algorithms for alerting individuals based on physiological predictors illness, enabling early intervention.**

***Index Terms*—wearables, COVID-19, secure IoT, time series data, big data workflows, big data system**

I. Introduction

A general goal of Artificial Intelligence is to develop automated techniques by which machines can perceive, interpret and respond to natural and human phenomena. Several subdisciplines of AI, like Natural Language Processing, Image/Video Understanding, and Speech Perception attempt to analyze surface signals to construct semantic symbols and utilize the context where the symbols occur to arrive at a semantic interpretation that is actionable. The problem of actionable signal interpretation is much harder in the domain of biomedicine, where multiple physiological and behavioral signals must be concurrently analyzed, and the context for analysis involves a large number of measurable as well as latent factors (e.g. skin temperature vs mood or endocrine dynamics). For example, a high-amplitude rise in body temperature can be used to detect fevers [1], which is a cardinal symptom for COVID-19 infection. However, to date, such approaches fail to generalize, in part because of inter-personal differences in dynamics, and in part because a rise in body temperature can be associated with an individual's immediate environment, prior state of activity, and may also depend on hidden states such as her circadian rhythm and menstrual cycle, as well as the general physiological characteristics of this individual. In the absence of very large data sets of continuous temperature measurements with all of the appropriate labels for these and other potential confounds, AIs struggle to make clinically actionable insights across heterogeneous populations. One approach to overcome the lack of well-labeled, population-scale databases is to train AIs within individuals. For the TemPredict research effort aimed at using wearable devices to develop and deploy COVID-19 detection algorithms, we took a holistic approach, including across- and within-individual comparisons, using daily symptom reports, demographics surveys, and test/diagnosis results as labels. We

enhanced our detection strategy by considering multiple real-time signals including the heart rate, heart rate variability, respiration, activity, and temperature, considered both for the current time frame and over a longer time-period. With this approach, we developed a *personalized physiological model* for the individual. This model then serves as the context against which the current physiological observations can be evaluated. To make such models more useful on patients with limited data histories, the personalized model must be complemented by the development *population models* that would characterize the physiological signatures of people who get the disease versus those who do not, to assess an a priori estimate of vulnerability for different population subgroups.

Joint construction of personalized as well as population models with streaming physiological data is an "AI at scale" problem that requires a machine learning system that operates within a large-scale data acquisition and processing architecture. To our knowledge, although the ability to perform continuous risk assessment is a fundamental need of the hour for a raging pandemic like COVID-19, there is no "big data"-style system today that allows monitoring, analysis and interpretation of multiple physiological signals to construct personalized and population-based physiological profiles with a goal of improving real-world actionable advice generated by appropriate AIs.

The goal of this paper is to present the cyberinfrastructure (CI) undergirding TemPredict (TPCI): an analytical platform we are developing to fill this gap, and to show how even with the initial version of the system, we have started enabling personalized model development and continuous risk assessment for several thousand users.

A. The TPCI System

The TPCI originated from the observation that pre-symptom changes in physiological data obtained from wearable devices could be used to detect some COVID-positive cases. [1] demonstrated that it is possible to algorithmically detect and predict fever from continuous temperature data generated by wearable devices, and was on average detectable *before* symptom onset was reported by participants. Over the year 2020, TemPredict grew into an international collaboration with $<$65,000 participants sharing survey and wearable data to feed algorithm development and test deployment. Here we describe the TPCI that was established to enable this initial COVID-19 algorithm effort. Knowing the barriers to generalizable health algorithms, we developed TPCI to support multiple parallel AI development schemes so that these and other data sets could fuel the future model developments and refinements necessary for TemPredict and related efforts to become productive tools serving Smart Public Health efforts.

The TPCI is based on the following research desiderata.

1) The system should enable the development of explicable "dynamic" models of physiological characteristics of human subjects. The models must adapt to temporal variations in human physiology and lead towards deeper understanding of the physiology of COVID-19.
2) Critical to a physiological model development is the exploration phase that surfaces data properties and constraints which can be exploited for predictive modeling. This phase informs the selection of model algorithms based on its mathematical characteristics to learn from data. The system must make the exploration step easier.
3) Multiple physiological signals required for constructing these models come from IoT devices and are subject to unreliability and disruption. The model development process must be robust to signal noise.
4) For scaling in a resource efficient manner, the system should support frugal selection of a minimal feature set that matches latency constraints and accurately generates *daily updatable personalized predictions*.
5) The system should support contextually aware modeling that accounts for both medical history (e.g., the existence of hypertension) and periodically updated contextual signals such as the COVID-19 test results.

Thus, we view TPCI as a large-scale AI-based model development platform that will enable a holistic understanding of the COVID-19 physiology, and not just as a system that implements a machine learning technique.

B. Related Work

Most research papers covering AI techniques for COVID-19 research fall in the "Early Detection and Diagnosis" focus area [2], and formulate their specific task as a classification problem. Typically, the classification task is binary – whether a subject is or is not COVID positive. For example, [3], analyzes the HRV (heart rate variability) to determine the relationship between changes in HRV and the presence or development of COVID-19–related symptoms and to evaluate how HRV changed throughout the infection and symptom period. However, this study limited itself to a small number of subjects. Among research groups [4]–[6] that have considered large-scale data, [4] considered the resting heart rate and sleep duration of 200,000 subjects to develop a prediction model for influenza-like symptoms. However, their log linear model is not personalized but developed as an aggregate for the state.

Researchers have demonstrated how wearable sensors play a significant role in collecting real-time multi-sensor physiological signals for developing models [7], [8]. Three primary data source categories include: (*i*) continuous monitoring data from platforms (e.g. Fitbit, Google Fit, Oura rings, Applewatch) [9], [10], (*ii*) self-reported symptoms [5], and (*iii*) physical/demographic parameters (e.g, gender, age, ethnicity, BMI). The continuously acquired data go through the steps of segmentation into time intervals, denoising, and feature extraction. There is a wide variability in the features used, including statistical properties like mean and standard deviation [11], [12], wavelet transforms [1] and spectral analysis [13]. These features are combined with symptoms and demographic information to create full a feature set for classification.

While all of these research attempts make valuable contributions, the research reported by these groups do not take an *AI-systems* approach to design their analysis architecture.

Consequently, processes for data acquisition, data integrity, support for large-scale computation, feature extraction and storage, and model computation, storage, refinement and reuse, and how these processes impact the implementation, accuracy, and efficiency of the AI research are not deeply investigated.

C. Contributions

The primary intent of this paper is not in designing an AI-method for COVID-19 prediction, but in developing an end-to-end platform for AI research on COVID-19 physiology, starting from data collection and the development of individual and community models of COVID-19 physiology, to the delivery of results to clinical users.

Specifically, this paper makes the following contributions.

1) We present a reference architecture for disease-specific physiology understanding platform, and apply it toward COVID-19 research. An important feature of this architecture is a large-scale data processing processing pipeline that feeds multiple intelligent inference modules.
2) Most AI systems expect "clean" data. We develop a data cleaning/feature extraction framework based on a combination of integrity constraints, sampling and "alignment" for multi-temporal data.
3) We demonstrate the utility of a polystore for storing heterogeneous data and computed features. The polystore adopts a temporal data model that natively supports temporal operations that is intrinsic to all physiological and evolving health-state data.
4) We present several cases of AI-centric research that is supported by our generic architecture.

II. System Design

A. Architecture

The TPCI, shown in Figure 1 consists of 6 modules.

1) The *data collection module* gathers multi-device real-time physiological signals, one-time demographic information and slow-moving health-state data with privacy constraints. Currently, the physiological data comes from Oura rings, multi-sensor wearable devices developed by Oura Health Inc. under a research agreement with participants who donate their data. The data from the participants' devices are gathered at the device provider's server via the Internet. The provider uploads the data periodically (daily) to a Cloud Storage Service (Amazon S3). The data, available from January 2020 including nightly heart rate, heart rate variability, respiration rate, and sleep staging, as well as 24h continuous skin temperature and activity, was pushed from Oura's data store to the secure S3 in the TPCI. In addition, the system collects participant responses from an "on-boarding" survey, including information about demographics and medical history. The health signals are gathered by monthly surveys assessing conditions such as stress or temporary medical conditions. Additionally, Oura smartphone app delivers daily questions to cover symptoms, tests, and diagnoses related to illness, with especial focus on COVID-19 related symptoms. As of this writing, the system contains around 3.3 million survey responses.
2) The *secure data management module* transforms the raw data into polystore records [14], performs anonymization and deanonymization (for specific tasks described later in the paper), and sends secure alerts to clinical researchers for at-risk subjects. The module is situated inside Sherlock (sherlock.sdsc.edu), a FISMA-compliant secure information environment that implements strict protocols for monitoring the ingress, access and egress of all information flowing through the system. This poses a system design challenge for the TemPredict architecture – while information security is important for personal data and is guaranteed by Sherlock, Sherlock is, by design, decoupled from the high-performance computing and robust big-data computing resources required for the rest of the data processing, model training and processes. To address this problem, we replicate a portion of the polystore inside Sherlock as well as outside it, and create a bridge to *push* anonymized data from the secure side to the public side of the architecture. Data transport in the reverse direction is performed by executing a *pull* operation from Sherlock to access model execution results into the information dissemination module, thus ensuring that a non-secure component of the architecture has no control over the data handshake.
3) The *feature management module* is populated with anonymized data and applies a series of processes to compute features. The features (i.e., derived data) are stored back into the polystore for next steps of the analysis. The module is discussed in more detail in Section III.
4) The *high-performance computation module* is used when feature computation routines are too expensive to run locally. In this case, the computation is performed on a parallel, GPU-enabled platform whose performance justifies the data transport cost between the two modules. Section III-A presents an example of this feature.

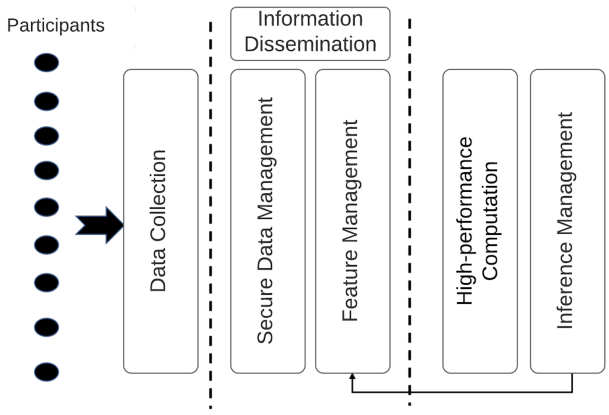

Fig. 1: A High-Level Diagram of the TPCI Architecture

5) The *inference management module* is responsible for training and testing AI models, storing the trained models and executing the models on new data. Both individual-based and population-based models are managed by this module. Section IV presents various inference management processes enabled by TPCI.
6) Finally, the *information dissemination module* transmits the results of the inference to the secure data management module, where the results are deanonymized and then then sent out securely to approved recipients. The results of the inference are passed back to the information dissemination module, which, as mentioned previously, resides within the secure component of the system. The information dissemination module uses the subject identifiers associated with the prediction results to recover personal information which is then used to securely notify the appropriate clinical personnel to perform requisite medical tests on the subject.

B. Design Decisions

The steps from acquisition of physiological data to the computation of features require several design decisions. One set of design decisions centers on 4 consistency problems related to data ingestion from the raw data to the creation of database records from the observations.

1) *Inter-batch Inconsistency.* The physiological data is delivered in batches where each batch covers a subset of subjects and a time-interval of observations. Inter-batch inconsistency refers to a situation where two batches have different schemas and/or different value domains. The schema inconsistency can be caused by multiple factors. Very often a part of the data attributes are encoded in the fully qualified file path in the Cloud Storage. Different encoding for two batches effectively lead to a schema inconsistency. A value-domain inconsistency occurs when two batches have different representation of the same attribute leading to downstream processing errors. For example, if time is represented with different formats, it is a value-domain inconsistency.
2) *Bitemporal Discrepancy.* The physiological data is bitemporal because the valid (capture) time and the transaction time (when it is a record at the data collection site) are different. Both times are available for a data record. The primary source of physiological data is an individual device which transmits data to a participant-side data accumulator that pushes the data to the provider. However, due to several issues like the lack of internet connectivity, not all data of each attribute of each person are pushed at the same time, making the data batches partially complete for any one individual, that may not get completed till several more batches have been processed, possibly after days of unavailability. This compromises the subjects for whom the discrepancy exists.
3) *Survey-Physiology Inconsistency.* Ideally, the physiological data and the survey will be in perfect synchrony, i.e., at any time, the latest survey record for every subject is available before the physiological signals of the subjects are collected to ensure that the two can be joined in the prediction algorithm, failing which the prediction accuracy may be compromised.
4) *Differences in Time Granularity and Data Volume.* The sampling rates for different physiological signals differ significantly from 60 Hz to 5 minute intervals, producing a large volume of personally identifiable temporal data. While the data does come into our secure server, the secure server does not have the capability of performing the requisite computations needed for the predictive analysis. This requirement has a significant impact on our system architecture.

Algorithm 1: The "Checkpointing" logic

Input: $s_{min}, t_{max}, t_n, ws$
Output: $res[][]$ is the output array, which store the sketch count for each time point
Result: Resketch recommendation at any time point

```
for each i in res do
    if res[i] < s_min then
        remove res[i];
    else
        for each i in t_n do
            if i not in res then
                res[i][j] ← count(t_n[i], ws)
            else
                α ← res[i][j];
                if α < t_max then
                    res[i][j] ← count(t_n[i], ws)
                else
                    return 0
                end
            end
        end
    end
end
```

The ingestion architecture applies a set of *data transformers*, which read data files from the cloud storage and convert the data to the standard schema implemented in the Secure AWESOME [14], [15] system that internally uses the TimeScaleDB (www.timescale.com) due to its native support of temporal operations including operations for time-series analysis. A transformer is designed to operate with a specific configuration of the input defined by its folder structure and the data definition used in the Apache Parquet (parquet.apache.org) files at S3. There is a family of transformers for every timestamped physiological attribute. The inter-batch discrepancy is managed through a logging procedure managed by the secure AWESOME system. When a specific batch fails to load, an error signal engaged a human data administrator to resolve the issue by creating a new transformer, such that all future batches that present the erstwhile-erroneous configuration can be processed.

As data gets ingested, the system maintains a monitoring

mechanism to manage the two forms of temporal discrepancy. We view this problem as a conceptual analog of the transaction management and checkpointing logic Algorithm 1 applied in a database platform. The purpose of the checkpoint algorithm is to help to avoid unnecessary sketch computing. The inputs are as follows, s_{min} is the minimum acceptable number of samples for a time bucket, t_{max} is the upper bound of the time, t_n is the array of time buckets, and ws is the working set. Using a data structure to record the number of data points per time window, and during each execution, the algorithm verifies the total number of records per sketch window. Too old or duplicate data due to re-transmission are ignored. Sketch computation algorithm uses the data structure to efficiently execute the sketch per data point.

A subject's data record is in a *temporally consistent* state at time T if all physiological and survey variables of the subject is in the database at T. Further, the data of a subject is *quantitatively consistent* at T if the number of data items received between the previous checkpoint time T_0 and T is consistent with the expected data rate for each variable. The ingestion system implements a monitoring procedure to periodically check for both forms of consistency for a random sample of subjects. If a subject's data passes the check, the checkpoint is advanced from T_0 to T. If it fails either test, the subject's ID is placed on a watch list, until it is removed upon a later update, and all pending feature computations on the subject are executed. Subjects staying long on the watch list are, per policy, reported back to the data provider.

III. Feature Engineering

The ultimate goal of the TPCI is to analyze and report within a guaranteed time-bound, (measured in hours) for a target population of 65,000 subjects. At this point, we have not reached this capability. However, our system design is primarily geared toward achieving a time-bounded execution guarantee. We take a two-pronged approach to achieve this goal. First, we compute *coarse sketches*, features that (i) perform a *coarsen* operation to reduce the time granularity, (ii) replace that actual data with a statistical representation. Secondly, we develop a parallel computational pipeline that (i) computes *higher-order sketches*, sketches that contextualize a subject's coarse-sketches over an observation period by using the subject's historical data and survey results, and (ii) performs a prediction computation. These features include the sample size per time interval, the total number of samples, the minimum and the maximum signal values and basic signal statistics – the mean and the first four moments of the frequency distribution of signal values for each physiological variable, as well as the 20, 40, 60 and 80 percentile values of the cumulative frequency distribution. One target of the pre-prediction data pipeline is to strike a balance between prediction accuracy and reduction of the data processing cost. Our strategy is informed by a number of data-centric and domain-specific considerations.

1) As in the case of most IoT data, the quality of data must be validated before performing any operation. In our case the validation is expressed as constraints of the following form:

$$valid(val(a_i, T)) \textbf{ if } (\theta_1 \leq val(a_{j\neq i}, T') \leq \theta_2) \\ \wedge (\theta_3 \leq val(a_{k\neq i, k\neq j}, T') \leq \theta_4) \ldots$$

where $val(a_i, T)$ represents the value of the i-th attribute (i.e., signal) at time interval T, θ_k represents the upper and lower bounds and interval $T' = T \pm \epsilon$. In other words, the validity of the value of a specific attribute during a time interval depends on the range of other signals during an ϵ-window around time interval T. For example, the metabolic rate captures a person's accelerometry or movement when they are wearing the ring (and during sleep). This property can be used for validating other variables like temperature. All the temperature values are dropped when the same time point's corresponding metabolism is less than the thresholds and the person is considered not wearing the ring that time.
This is the only step in the data processing where a temporal join operation is computed across multiple signals, following which the attributes are treated completely independently and in parallel.
2) The data features to be used for predictive analytics must capture the signal variability within and across observation windows. However, choosing a very narrow time-interval for windowing does not guarantee sufficient independence across consecutive observations. This precludes the use of point sampling (which would have lowered cost) and implies a lower bound on Δ_o, the duration of the *observation interval*. If a sliding window is applied, shift size δ_s must also been determined. Based on an empirical study to monitor the impact of the interval parameters on prediction accuracy, it was determined that setting $\Delta_o = 1800s$ and $\delta_s = 0s$ (i.e., a tumbling window) produces reasonably accurate results.
3) The features to represent signal variation capture the value statistics within the observation interval. The feature set consists of the maximum and minimum values, the first four moments, and k-th percentile statistics (practically, $mod(k - 20) = 0$), and the number of samples, which can vary between observation windows. We exploit TimeScaleDB's functions for this computation. TimeScaleDB performs this operation by efficient constructing a single statistical aggregate from which all our metrics are derived.
4) The data from the devices are timestamped but they are not programmed to be emitted at fixed intervals – hence the signals are not periodic. However, the precise time of the data is not materially significant for sketch computation. Therefore, for each signal, the data is *time-aligned* to the nearest t-th minute ($t < \Delta_o$). If n samples are collected within a span of t minutes, they are averaged and mapped to the t-th minute, leading to the first data reduction opportunity. Since the time alignment effectively creates $\frac{\Delta_o}{t}$ independent time slices, the alignment operation can be executed in parallel.

As the size and processing of the observation intervals are determined by the above considerations, we now need to compute sketches for N subjects for $I(= 48)$ intervals. With enough resources, the sketch computation, implemented as relational views (with functions) on the observation can be computed in an embarrassingly parallel manner. However, since the sketches are computed on a large volume of nearly-raw time-aligned data, the data cannot be moved to a high-performance parallel computing platform, because the data transmission cost overshadows the benefit of parallelization. Since the higher-order sketch computation and the prediction algorithm must have the sketches of all intervals of a single individual over a 24-hour period, we can only parallelize the sketch computation over the set of subjects because all intervals of an individual must be computed. Our current solution is to partition the subject pool into batches and perform the sketch computation of the members of a batch in parallel.

A. Higher Level Features and AI Model Preparation

In this part, we dive deeper into the infrastructure that exploits computational and data parallelism to allow quickest execution of higher order features computation and Machine Learning algorithms. Two mission critical aspects of our solution to COVID-19 are (a) prediction accuracy, and (b) speed and efficiency of execution.

Once data had been sketched within single observation intervals, a *sketch normalization* step is executed over the sketches for every individual subject to compute what we call the *higher-order sketches*. This "normalization" is to ensure that the computed sketch values are biologically informative for use in the biological time-series applications of illness detection. For example, physiological dynamics and ranges within variables are dynamic within individuals, with important differences across individuals. For this reason, the historical sketch data each individual must be used for his/her *own* baseline of comparison, and when needed, informed by survey data. Thus, a few weeks (or months, in the case of ovarian-cycling women, and years in the case of seasonal changes) of each individual's pre-illness data is used to generate normalizations along daily ranges and other comparisons as appropriate. A more detailed rationale and procedure for normalization can be found in some of the observations about fever detection in the initial TemPredict paper [1]. Note that, while this requires additional DB access, this normalization is still parallelizable over subjects and variables and there is no "join" operation between the three data categories. Further, the survey data for an individual is queried only once during the normalization process. An outcome of the higher order sketch computation is based on frequency of exceptional time windows within a wider timeframe using the history of sketches of the same person, allowing comparisons over time. The normalized sketch is fed to the prediction algorithm to determine subjects who may have a potential onset of illness. Details of the prediction algorithm is out the scope of this paper.

Our integrated big data infrastructure has three major characteristics: (1) A simple interface for users with different domain expertise to capitalize advanced compute and data resources; (2) An automated back end that accelerates the end-to-end process of scientific discovery from the exploration-to-scalable deployment of AI algorithms without human involvement/ intervention; and (3) A collaborative environment for Team Science [16]. Fig.2 presents the TPCI Big Data System for AI Model Development and Scalable Deployment for COVID-19. The Scalable Data Analysis System consist of the following key components: (i) Exploratory Workflow Development Interface, and (ii) Auto-deployment and auto-scale of AI Models.

(i) Exploratory AI Development Workflow User Interface:

The first component a user interacts with is the JupyerHub, which provides a Python notebook-based exploratory workflow development interface (https://tempredict.sdsc.edu/jupyterhub). TPCI allows authorized users to leverage this tool on scalable hardware transparently while supporting efficient data access from a variety of data systems. Users can use their existing identity providers, such as University, Google, or Github credentials to login to TPCI. Upon login, using a Spawner Options form, users can submit their resource requests such as *number of CPUs*, *number of GPUs*, *Memory in GB*, *Container Image*, and *Persistent Storage*. The workflow involves a domain expert getting resources on Jupyterhub via the Kubernetes-based (kubernetes.io) Nautilus portal at UCSD (nautilus.optiputer.net), opening their Jupyter notebook by spawning an instance, and establishing database connectivity. The user can mount a persistent storage system of 400TB capacity which provides a shared storage environment per group. The project lead manages access control to shared persistent storage. A project can have more than one shared persistent volumes for different groups. The shared persistent storage gives an opportunity for collaboration among authorized group members. Behind the scene, the TPCI platform leverages advancements in Kubernetes without exposing scientists to the complexities of coding for creating deployments of pods, volumes, services, and API triggers. The TPCI platform exposes a Jupyter Notebook interface for exploration and development of algorithms so users can leverage advanced data science and AI libraries without needing to do repetitive software stack installations. Further, TPCI exposes simple APIs that researchers can run from their Jupyter notebooks to access data efficiently from a variety of data systems, such as remote database servers and cloud storage such as Amazon S3, Swift Object Storage, and Next Cloud. Users can configure their personal individual storage systems as well. Once the algorithm has been established, users execute their algorithms in a parallelized manner. The APIs provide a path to submit to code repositories such as Github. From Github, the TPCI system auto-deploys and auto-scales algorithms on GPU and CPU clusters managed by Kubernetes. The entire system is designed and built to empower users to develop and test their algorithms through an agile development methodology. A data scientist may develop their algorithm for one unit of data (in our case for a subject with a PID) without worrying about scalability in Jupyter Notebook. Using the TPCI APIs, users can then push their algorithms or Jupyter Notebooks to Github or other code repositories. Next, we discuss how TPCI

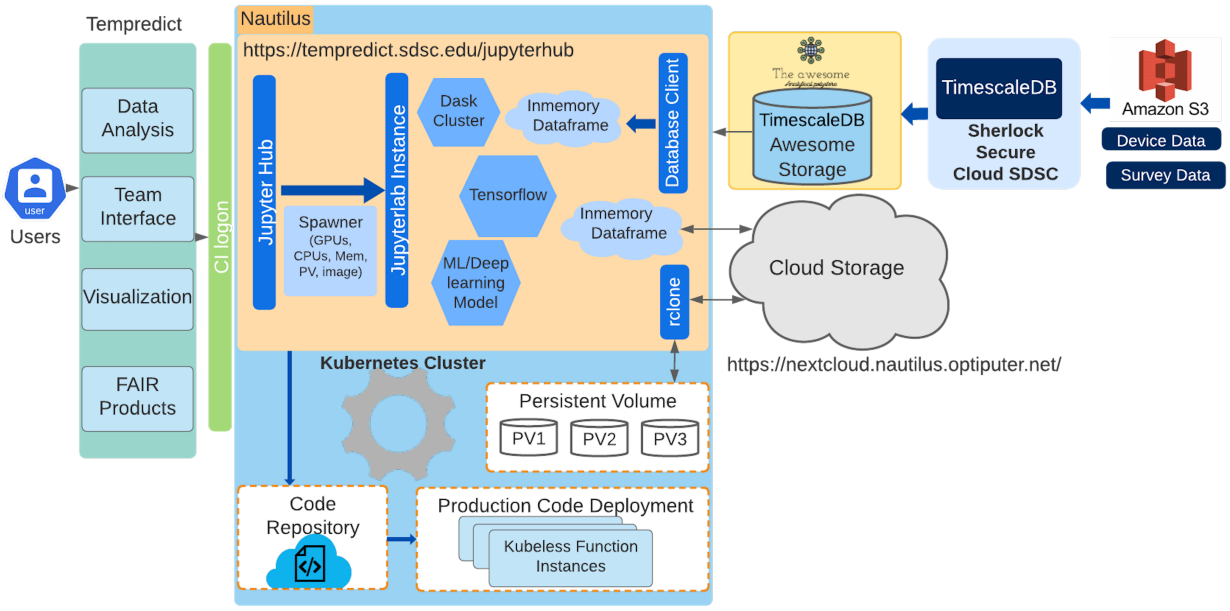

Fig. 2: AI Model Development and Scalable CI Deployment

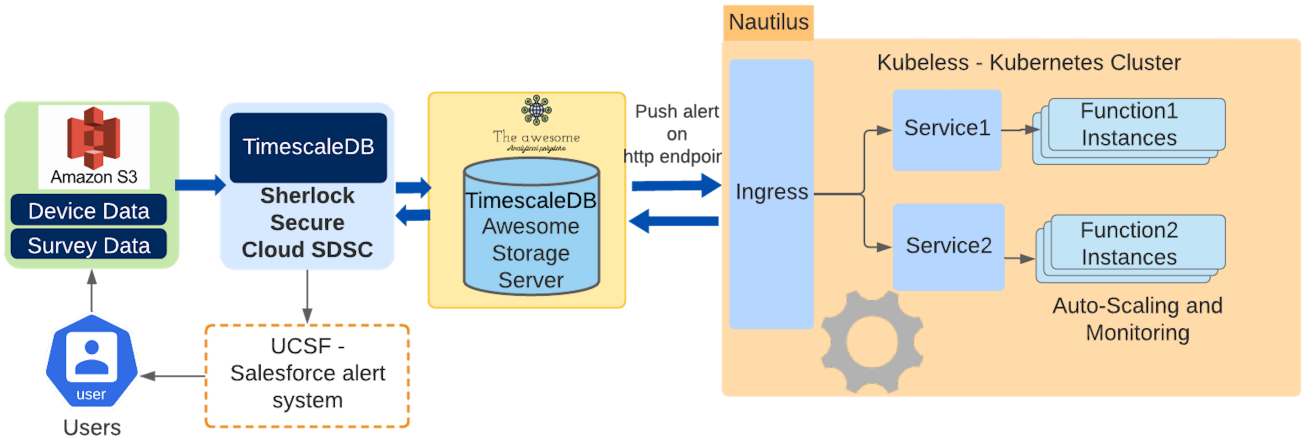

Fig. 3: Production Deployment and Auto-scaling Environment

scales these algorithms using different parallelism techniques.

(ii) Auto-deployment and Auto-scale of AI Models:

First analyzed are the algorithm, input load, and associated constraints. Parallel computing techniques are applied such as task parallelism and data parallelism to utilize all the CPU cores optimally. Once the algorithm is optimized for the number of CPU cores on a single node, it is deployed as a Kubeless (kubeless.io) function on Kubernetes to further parallelize using Kubernetes auto-scale capability on multiple nodes.

- **Single Node Parallel Computing:** TPCI APIs will apply appropriate parallelism and scaling based on the algorithm itself, input, and system constraints such as data transfer to data systems. Task parallelism is applied when multiple tasks can concurrently execute on the same data, i.e., when one function is applied to many pieces of data independently such as loops. For example, the platform provides a concurrent mapping capability to parallelize sequential for loops on multiple cores and reduce execution time.
- **Kubeless Function Deployment:** TPCI provides APIs

to execute code in Serverless Kubernetes Frameworks using Kubeless based infrastructure. Once the algorithm is optimized for a single node, TPCI APIs were used to deploy it on the Kubernetes cluster as a Kubeless function and create the required service API and ingress deployments in the backend as shown in the Fig. 3. This serverless mechanism facilitates rapid deployment and the service API gives a handle to invoke the AI Models programmatically for the automated live pipeline.
The service API triggers the function execution over HTTP protocol. We can configure the APIs to conveniently trigger the execution of functions upon the occurrence of certain critical events, such as, when first order sketches are available. Based on an event or a cron trigger, the users can initiate a parallelized execution of their algorithm remotely from anywhere on any device. This gives users the capability to quickly execute and test functions based on source code developed by their team members. Further, autoscale rules can be applied on Kubeless functions to scale up and down.

- **Kubernetes Autoscale:** Kubeless autoscale functionality is used to scale Kubeless pod horizontally based on metric targets such as CPU utilization, number of requests, and others. The function execution will have the ability to auto-scale up and down without intervention based on computing needs and input data size.

IV. Progress Toward AI Research

The complexity of human physiology and pathology are, at least for the time being, literally beyond description. As such, health problems can benefit from AI support when those AIs have been developed at the intersection of specific populations and conditions. Additionally, AIs can be developed to help map approximations of the hyper-complex health space, as in identifying similar patients or latent physiological indications. For example, in developing algorithms for the early detection of COVID-19 from wearable device data, the following (non-exhaustive) gaps in knowledge presented barriers: lack of basic description of illness progression, and how this manifests in and perturbs normal physiological dynamics; absence of normal physiological dynamics models, let alone how these dynamics change by age, sex, ethnicity, condition, and circumstance to create dynamic baselines; lack of models for physiological dynamic baselines, and how to optimize anomaly detection in such models; lack of ability to detect (confirm) positive COVID-19 infection (across 2020, tests were developed to fill this gap, though not it real-time), and how to confirm infection onset (still missing) and recovery (still missing).

To address these issues with sufficient speed to be of service during the COVID-19 pandemic, UCSF launched TemPredict in March 2020, in collaboration with UC San Diego. TemPredict, initially sponsored by OuraRing, Inc., was subsequently funded by the DoD and was supported by MIT Lincoln Lab collaborators. Here, we present additional inference management processes enabled by TPCI. A full description of recruitment, data gathering, and subsequent algorithm development appears in another manuscript, currently in production.

(i) *Random Forest-Based Prediction of At-Risk Subjects*: In this method, researchers are collecting health sensing data from wearable devices from confirmed COVID-19 patients and define baseline measurements of physiological dynamics to ultimately capture anomalies and fluctuations associated with COVID-19 infections. The platform enables this team to create features using rolling windows, perform normalization and train a Random Forest model across the physiological variables.

(ii) *Bayesian Networks-Based Prediction of Reported Health Status*: This supported scenario is leveraging Bayesian Networks to make probabilistic estimations from an individual's symptom profile states. The Bayesian approach gives insights about the dependencies among the symptoms, however incomplete data poses a critical challenge. The TPCI enabled temporal analysis by mapping disease progression to probabilities.

(iii) *Autoencoder-Based Classification of Physiological Signals*: TPCI enables rapid exploration of Deep Learning Models that can infer outcomes from time-series signals. Researchers are using the TPCI to map time-series physiological signals to low dimensional embeddings using auto-encoders (variational, sparse, and temporal). This compressed embeddings enable rapid exploration of Recurrent Neural Network (RNN) architectures to arrive at optimal health prediction algorithms.

(iv) *Exploratory Analysis of Paired Survey Response and Physiological Data*: In this experiment, researchers leveraged features such as variations in skin surface temperature and duration of elevated temperature to develop and compare Hidden Markov Models against baseline Logistic Regression Classifiers. The simplicity of this feature and usability of the TPCI allowed rapid implementation and comparison.

V. Results and Future Work

The fundamental design idea behind the TemPredict system is simplicity coupled with computational scalability. TPCI provides users with a simplified user interface to develop predictive algorithms and decouples the responsibility of scaling a given algorithm for a large number of subjects. This section illustrates the performance of the TPCI system by means of the sketch computation and the higher-order feature extraction steps for 200 candidate users, and a data time window of the last twenty-four hours. As a part of the live study, the system collected, ingested, and processed users' data incrementally with time in near real-time.

A mid-size server for computing the sketches was used. The execution time, CPU usages, and memory utilization are reported in Fig. 4, in which the graph represents the users' batch size (x-axis), and the measurement (y-axis). The sketch computation is a CPU-bound process, so a python parallel thread was used to compute the batches. The execution time linearly increases with the number of batches, and the CPU and memory utilization is almost stable for the different batches. Fig. 5 shows Runtime, CPU Usage, Memory Usage (y-axis) for higher order features computation with the number of cores(x-axis) for different batch sizes. The original algorithm calculated

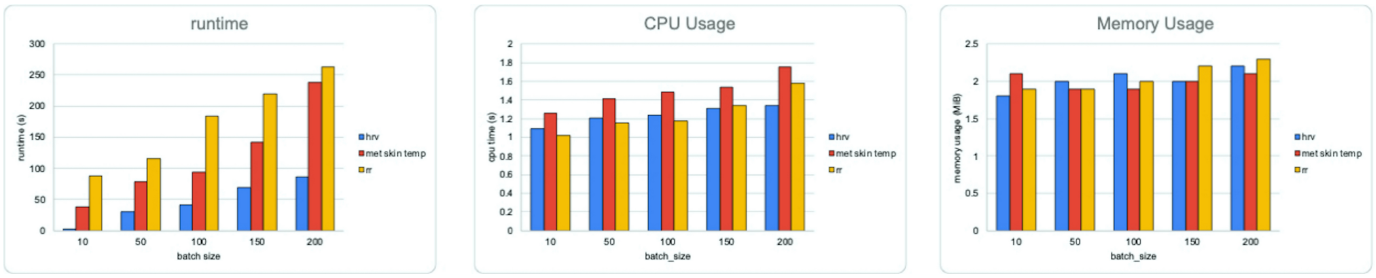

Fig. 4: Runtime, CPU, Memory usages for Sketch computation batch sizes.

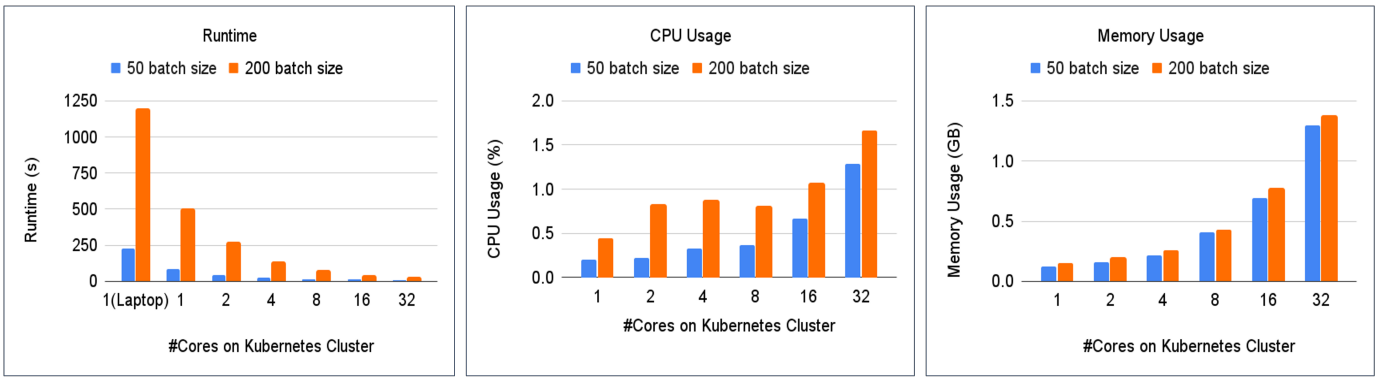

Fig. 5: Runtime, CPU, Memory usages for Higher Order Feature Extraction Step.

high order features for each subject (PID) in a sequential way on a laptop, taking 20 mins to complete higher-order sketches for nearly 200 subjects (batch size). The application of data parallelism across 16 cores to concurrently run the algorithm on each subject (PID) improved the execution speed by 32 times, reducing the execution time to ∼37sec.

As part of future work, we plan to focus on the team science for AI functionality of TPCI, allowing multiple users to collaborate on parts of the analysis through a chain of Python notebooks. Initial framework was developed to chain together notebooks with constraints on metrics of performance and accuracy, but left out of the scope of this paper. In addition, future work will be on deployment of various studies and metadata capabilities to ensure utilization of the insight gained from analysis of the data within new science questions related to the disease onset and development.

ACKNOWLEDGMENT

This effort was funded under MTEC solicitation MTEC-20-12-Diagnostics-023 and is funded by the USAMRDC under the Department of Defense. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government. TPCI was built on Cognitive Hardware and Software Ecosystem Community Infrastructure (CHASE-CI) supported by NSF award numbers 1730158, 2100237, and 2120019. We thank the UC San Diego CHASE-CI team for computing support.

REFERENCES

[1] B. L. Smarr, K. Aschbacher, S. M. Fisher, A. Chowdhary, S. Dilchert, K. Puldon, A. Rao, F. M. Hecht, and A. E. Mason, "Feasibility of continuous fever monitoring using wearable devices," *Scientific Reports*, vol. 10, no. 1, p. 21640, Dec 2020. [Online]. Available: https://doi.org/10.1038/s41598-020-78355-6

[2] H. B. Syeda, M. Syed, K. W. Sexton, S. Syed, S. Begum, F. Syed, F. Prior, and F. Yu Jr, "Role of machine learning techniques to tackle the covid-19 crisis: Systematic review," *JMIR medical informatics*, vol. 9, no. 1, p. e23811, 2021.

[3] R. P. Hirten, M. Danieletto, L. Tomalin, K. H. Choi, M. Zweig, E. Golden, S. Kaur, D. Helmus, A. Biello, R. Pyzik *et al.*, "Use of physiological data from a wearable device to identify sars-cov-2 infection and symptoms and predict covid-19 diagnosis: Observational study," *Journal of medical Internet research*, vol. 23, no. 2, p. e26107, 2021.

[4] J. M. Radin, N. E. Wineinger, E. J. Topol, and S. R. Steinhubl, "Harnessing wearable device data to improve state-level real-time surveillance of influenza-like illness in the usa: a population-based study," *The Lancet Digital Health*, vol. 2, no. 2, pp. e85–e93, 2020.

[5] G. Quer, J. M. Radin, M. Gadaleta, K. Baca-Motes, L. Ariniello, E. Ramos, V. Kheterpal, E. J. Topol, and S. R. Steinhubl, "Wearable sensor data and self-reported symptoms for covid-19 detection," *Nature Medicine*, vol. 27, no. 1, pp. 73–77, 2021.

[6] M. Poongodi, M. Hamdi, M. Malviya, A. Sharma, G. Dhiman, and S. Vimal, "Diagnosis and combating covid-19 using wearable oura smart ring with deep learning methods," *Personal and ubiquitous computing*, pp. 1–11, 2021.

[7] D. R. Seshadri, E. V. Davies, E. R. Harlow, J. J. Hsu, S. C. Knighton, T. A. Walker, J. E. Voos, and C. K. Drummond, "Wearable sensors for covid-19: a call to action to harness our digital infrastructure for remote patient monitoring and virtual assessments," *Frontiers in Digital Health*, vol. 2, p. 8, 2020.

[8] X. Wang, N. Vouk, C. Heaukulani, T. Buddhika, W. Martanto, J. Lee, and R. J. Morris, "Hopes: An integrative digital phenotyping platform for data collection, monitoring, and machine learning," *Journal of Medical Internet Research*, vol. 23, no. 3, p. e23984, 2021.

[9] H. Jeong, J. A. Rogers, and S. Xu, "Continuous on-body sensing for the covid-19 pandemic: Gaps and opportunities," *Science Advances*, vol. 6, no. 36, p. eabd4794, 2020.
[10] A. Henriksen, E. Johannessen, G. Hartvigsen, S. Grimsgaard, and L. A. Hopstock, "Consumer-based activity trackers as a tool for physical activity monitoring in epidemiological studies during the covid-19 pandemic: Development and usability study," *JMIR Public Health and Surveillance*, vol. 7, no. 4, p. e23806, 2021.
[11] M. A. Mehrabadi, S. A. H. Aqajari, I. Azimi, C. A. Downs, N. Dutt, and A. M. Rahmani, "Detection of covid-19 using heart rate and blood pressure: Lessons learned from patients with ards," *arXiv preprint arXiv:2011.10470*, 2020.
[12] R. Buchhorn, C. Baumann, and C. Willaschek, "Heart rate variability in a patient with coronavirus disease 2019," in *International Cardiovascular Forum Journal*, vol. 20, 2020.
[13] B. Fyntanidou, M. Zouka, A. Apostolopoulou, P. D. Bamidis, A. Billis, K. Mitsopoulos, P. Angelidis, and A. Fourlis, "Iot-based smart triage of covid-19 suspicious cases in the emergency department," in *2020 IEEE Globecom Workshops (GC Wkshps*. IEEE, 2020, pp. 1–6.
[14] S. Dasgupta, K. Coakley, and A. Gupta, "Analytics-driven data ingestion and derivation in the awesome polystore," in *2016 IEEE International Conference on Big Data (Big Data)*, 2016, pp. 2555–2564.
[15] A. Gupta, S. Dasgupta, and M. Roberts, "Data ingestion into a polystore," Jun. 6 2019, uS Patent App. 16/209,397.
[16] I. Altintas, S. Purawat, D. Crawl, A. Singh, and K. Marcus, "Toward a methodology and framework for workflow-driven team science," *Computing in Science Engineering*, vol. 21, no. 4, pp. 37–48, 2019.