Exploring Edge Machine Learning-based Stress Prediction using Wearable Devices

Sang-Hun Sim
Arizona State University
ssim6@asu.edu

Tara Paranjpe
Arizona State University
tparanjp@asu.edu

Nicole Roberts
Arizona State University
nicole.a.roberts@asu.edu

Ming Zhao
Arizona State University
mingzhao@asu.edu

***Abstract*—Stress is a central factor in our daily lives, impacting performance, decisions, well-being, and our interactions with others. With the development of IoT technology, smart wearable devices can handle diverse operations, including networking and recording biometric signals. The enhanced data processing capability of wearables has also allowed for increased stress awareness among users. Edge computing on such devices enables real-time feedback which can provide an opportunity to prevent severe consequences that might result if stress is left unaddressed. Edge computing can also strengthen privacy by implementing stress prediction on local devices without transferring personal information to the public cloud.**

This paper presents a framework for real-time stress prediction, specifically for police training cadets, using wearable devices and machine learning with support from cloud computing. We developed an application for Fitbit and the user's accompanying smartphone to collect heart rate fluctuations and corresponding stress levels entered by users and a cloud backend for storing data and training models. Real-world data for this study was collected from police cadets during a police academy training program. Machine learning classifiers for stress prediction were built using this data through classic machine learning models and neural networks. To analyze efficiency across different environments, the models were optimized using model compression and other relevant techniques and tested on cloud and edge environments. Evaluation using real data and real devices showed that the highest accuracy came from XGBoost and Tensorflow neural network models, and on-edge stress prediction models produced lower latency results than in-cloud prediction.

I. Introduction

Stress often occurs when we become pressured by events that overwhelm our capacity to deal with those events [1]. This can be particularly consequential in the workplace. Stress is inversely related to job performance and negatively affects decision-making [2]. Law enforcement work in particular comes with many unexpected threats and challenging demands. Chronic stress increases the perception of threat and level of aggression in response [3]. Identifying and addressing symptoms of acute stress among officers is critical, as stress can otherwise become chronic, leading to loss in job performance, poor officer health, and problematic behaviors [4]. Moreover, despite wide variation from person to person [5], increased cardiovascular activation (e.g., heart rate) generally is associated with more intense physical and emotional states, including stress [6].

There is utility in understanding whether frequent or cumulative increases in heart rate can predict feelings of stress; which features or properties of the heart rate signal are predictive; and the time course for optimal predictions. This has real-world implications for police officers, who must be aware of and manage their stress and physiological arousal in real time [7]. To address and ultimately mitigate the potentially detrimental effects of stress in police officers as they navigate their ever-changing field, we investigated stress prediction through the collection of biometric data via wearables, used that data to create machine learning models, and analyzed the success of those models on the edge.

As described in this paper, we leveraged machine learning and edge computing techniques for stress prediction. The machine learning approach allowed us to draw relevant predictions from a large volume of data collected over time. With the growth of machine learning techniques and models, this paper investigates the use of classic models as well as modern neural networks. Edge computing, combined with a machine learning approach, can create a faster and more reliable stress prediction mechanism than the cloud. With all the computing power moved to the local devices, users can achieve strong data privacy and fast prediction speed.

We developed a framework for real-time stress prediction using wearable devices and machine learning with support from cloud computing. We utilized Fitbit, a commercial wearable device, to collect biometrics including heart rate, and developed a custom application to collect stress level ratings that participants could manually input into the Fitbit. We used Amazon Web Services (AWS) to securely store data and train machine learning classifiers. Our custom Fitbit application generated prompts, triggered by heart rate fluctuation thresholds (35% increase above resting heart rate, based on previous research and pilot testing), with 5 stress level buttons to collect user input (adapted from [8]). We created a secure web application with a dashboard to allow users and researchers to monitor each Fitbit device and view all the collected data.

We worked closely with a local metropolitan police training academy to collect biometric data and stress perception scores from police training cadets. For this initial study, data was collected over 4 months across 15 different cadets. Our team worked with each cadet to prepare and maintain their data collection from Fitbit devices and to gather overall study feedback.

Using the data collected, we built classification models to categorize stress experiences from features of heart rate using machine learning. We employed binary classification methods ("not stressed" vs. "stressed") using both classic models and a neural network. Then, we deployed the stress classification models on smartphones to predict stress ratings in proximity to the user without sending the data to the cloud. To optimize the neural network for edge use, we applied quantization to reduce the model size.

The most significant results of our study are as follows:

- The best accuracies for stress prediction were 96.98% from XGBoost and 95.98% from a feedforward neural network using an over-sampled data extraction approach.
- Quantization reduced the stress prediction model size by 74.2% and maintained the accuracy as before the quantization.
- Several trials proved that on-edge stress prediction with the quantized model saved nearly 200ms compared to in-cloud stress prediction.

The rest of the paper is organized as follows: Section 2 investigates the related works; Section 3 presents our data collection strategies; Section 4 details our approach to machine learning-based stress prediction; Section 5 presents our edge computing-based stress prediction solution, and Section 6 concludes the paper.

II. Related Works

A. *Efficacy of Wearable Devices in Clinical Domains*

Wearable devices as a health monitoring and management tool are increasing in popularity and functionality. Seamless integration with smartphone devices and real-time data collection have made it simple and convenient for users to learn more about their health, mental health, and physiological states without any external involvement. As wearable devices continue to grow in popularity, studies have begun to leverage their health data collection capabilities to learn more about human behaviors and tendencies.

For example, a study conducted by Beniczky et al. [9] investigated the use of wearable devices to detect and predict seizures in patients with epilepsy. Through the use of the sensors on the device, the researchers were able to collect electrocardiogram (ECG), heart rate variability (HRV), and accelerometer data. These data were then fed into machine learning models to create predictive models for generalized tonic-clonic seizures. Results showed that the noninvasive wearable devices were able to detect such seizures with 90%-96% accuracy. Rykov et al. [10] were able to leverage wearable devices in the same capacity to screen for depression-related biomarkers, collecting sleep patterns, physical activity, and psychological measurements to evaluate mental states. Models created in this depression detection domain were able to reach 82% accuracy. The ability to make diagnoses and predictions like these in real-world contexts demonstrates the feasibility of using wearable devices to extend clinical research beyond controlled environments.

B. *Stress Prediction using Machine Learning and Wearables*

Several studies have leveraged data collected from wearable devices in machine learning-based stress prediction. Lawanont et al. [11] and Dai et al. [12] both investigated the use of wearable devices to detect stress in controlled stress scenarios. Using the Fitbit and the Fossil Gen4 Explorist commercially available wearable devices, both studies collected several different data points during controlled experiments. Inputs included heart rate data—specifically root mean square of successive differences (RMSSD) and inter-beat interval; sleep data—including rapid eye movement (REM), deep and light sleep time; and other metrics such as calories, steps, and intensity of movement. Can et al. [13] and Martinez et al. [14] similarly leveraged wearable devices in the stress prediction domain, collecting heart rate variability, accelerometer, and temperature data, but took the research a step further by attempting to detect stress *in situ*. The authors underscore the promise and challenges of data collection in unconstrained daily-life contexts.

These studies employed classification-based models to detect stress. Among these, the Lawanot et al. study [11] achieved an 84.10% accuracy with the Decision Tree model (no F1-score reported), the Dai et al. study [12] achieved an 82.3% accuracy and 62.3% F1-score with the Support Vector Machine (SVM) model, and the Can et al. [13] study achieved 92.19% accuracy and 90.30% F1-score among other significant results. Through such key findings, we can learn that there is great promise in leveraging empirically-chosen classification models to predict stress experiences in highly stressful situations, even unconstrained daily-life contexts. Our work is one of few studies to date to investigate stress prediction in the real-world context of law enforcement training.

With an increased demand for stress management in law enforcement domains, a handful of studies involving wearable devices, machine learning, and behavior prediction have been conducted. Tiwari et al. [15] proposes stress prediction through the use of heart rate variability and breathing analyses during three waves of data collection. Each wave consisted of a different assessment—beginning with daily wear to gauge baseline levels, and moving on to shooting range exercises and intervention simulation exercises. Similarly, in the study by Erickson et al. [16], heart rate and sleep data were collected through in-class, day, and night field training. Both studies used the collected data as inputs into classification models, including SVM (most accurate for [15]), Logistic Regression, Random Forest (most accurate for [16]), and Adaboost.

Building upon the findings of these stress prediction studies in both general and law enforcement domains, our work makes several new contributions: 1) we developed an end-to-end system with a custom wearable app to prompt users for real-time inputs based on physiological signals; 2) we used this system to collect multiple months of data from real users in real-world scenarios; and 3) we implemented the use of edge computing for real-time stress prediction.

C. *Machine Learning on Edge Devices*

Due to the immediate and dynamic nature of stress and its consequences, the use of on-edge machine learning can be pivotal in real-time stress prediction.

Chen et al. [17] and Mauldin et al. [18] provided in-depth research about the capability of mobile devices for training and inferencing with deep learning models. Chen et al. leveraged the CIFAR-10 dataset, containing millions of images used to train computer-vision models. These data were fed into various deep learning models, significantly, the Convolutional Neural Network (CNN). Mauldin et al., proposing a real-time fall detection system, leveraged external wearable datasets in several models, including SVM, Naive Bayes Classifier, and a Deep Neural Network (DNN) which was the most accurate. The former study found that training operations contribute most significantly to the latency, especially for the gradient calculation of the backward path. As a result, the study concluded that it is possible to run both training and inference on mobile devices provided that the models' complexity is reduced. The latter study similarly concluded that it is possible to run machine learning inference on mobile devices, providing a lightweight model. It becomes clear from these works that mobile inference is possible in real-time machine learning deployments.

Ogden et al. [19] and Guo et al. [20] delved deeper into on-edge vs. in-cloud latency with their respective deployments. Ogden et al. primarily focused on which model compression techniques to use, which model to choose for mobile inference, and when to depend on servers. They proved that the quantized models have significantly smaller model sizes; moreover, loading the 8-bit quantized model did not contribute significantly to inference time, compared with other models. Additionally, through an analysis of various devices and Internet connection strength, they determined that cloud-based inference does still consume less power and deliver faster response times compared to on-edge inference, but discovers promising results for the future of on-edge inference.

Guo et al. [20] tested on-edge vs. in-cloud latency, but observed varying results. The evaluation metrics used in this study were latency, power consumption, and resource usage. They found that the in-cloud approach outperforms on-edge. They share, however, that this conclusion is dependent on the performance capabilities of the smartphone and the model size. With the manipulation of these two parameters, there is a high possibility for low latency through on-edge deployment. Though both conclusions differ, this lays the groundwork for additional research between on-edge and in-cloud model deployments to improve real-time feedback.

In this paper, we study an edge computing-based solution for stress prediction that addresses the unique challenges brought by developing an accurate and fast model for predicting stress in real time from real users' biometric inputs.

III. Data Collection

A. Overview

This project collected heart rate and stress rating data from Fitbit devices worn by 15 police cadets during four continuous months in a rigorous training academy. Stress ratings were collected directly into the Fitbit when prompted based on increases in the user's heart rate. Stress level rating options ranged from one to five, with one as "not at all stressed" and five as "extremely stressed". We recorded a zero when the cadet did not respond to a prompt. This scale was adapted from the 5-point Perceived Stress Scale (PSS) [8], a widely used self-report measure of subjective stress.

The architecture of our Stress Management system comprises several key components, as shown in Figure 1. There are two types of data collection pipelines for heart rates and stress responses, respectively. Heart rate data is transmitted to Fitbit's remote server by itself (once a Fitbit gets synchronized to the smartphone), and can be retrieved via Fitbit Web API with user account credentials. Stress levels, on the other hand, need a private database as they are measured and collected by the custom application. These are stored in key-value format. After the data uploading and processing phases, a data archive is kept with cadets' heart rates and stress responses for model development.

B. Fitbit Application

Fitbit's network architecture includes the Fitbit watch and the smartphone companion application, a supplementary runtime environment built to extend application capabilities. Fitbit needs to be paired with the companion through a Bluetooth connection to transfer Fitbit-collected data. Because Fitbit does not have Internet connection by itself, it depends on the companion for any other operations, like fetching information or storing data, except for recording bio-signals. Upon syncing to the companion, Fitbit sends all the biometric signals recorded to the remote server. Fitbit also provides an application on the smartphone to display statistical data of the recording. While Fitbit focuses mainly on recording biometric measurements, the companion can do more complex operations; together, they can build a more elaborate application.

Fitbit supplies a Software Development Kit (SDK) and various Application Programming Interfaces (APIs) to developers for custom application development. As this initial study intended to catch stress occurrence mainly by heart rate fluctuation, we built a Fitbit application (in the form of a custom clock face) to detect patterns indicating possible stressful situations and prompt users for stress inputs as shown in Figure 2. The prompting generation mechanism is when the heart rate goes above the resting heart rate by 35 percent for two minutes, the clock's face changes to the prompted face with five buttons ranging from five levels, "No Stress", "A Little", "Moderate", "A Lot", and "Extremely". Users enter their stress levels subjectively. A 30-minute pause period was programmed to occur between prompts to avoid continuous prompting. If a user is unable to respond to a given prompt after 7 minutes, the value is stored as a zero representing a "missed" prompt in our database.

Through this custom application, once the user enters the stress level, Fitbit keeps the stress input data in the file storage in the CBOR format and sends it to the companion as soon as the Bluetooth connection is established. The companion receives the stress file and concatenates it to the existing data in a key-value format, so the data is temporarily preserved in the

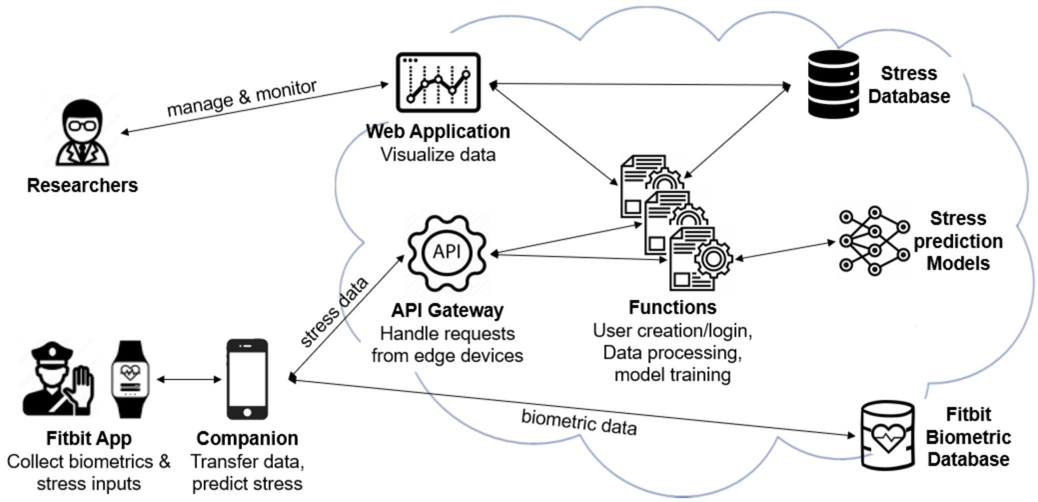

Fig. 1. Architecture of the proposed edge machine learning-based stress management system. Through the custom-built app on Fitbit and companion on the smartphone, police cadets can provide stress perception scores alongside Fitbit-collected biometric data. The companion requests various cloud resources through the API Gateway to process data and train models. With the trained models, the companion can perform stress prediction on the edge in real time. The system also provides a web application for researchers to manage users and monitor data.

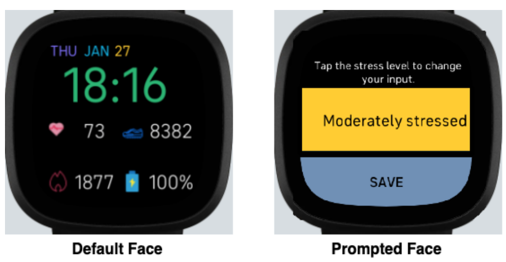

Fig. 2. Custom-built Fitbit clock face built to collect self-reported stress levels from cadets in real-time while also serving general watch functionality with a traditional default face. On prompt, the watch face displays 5 stress levels on a rotating cycle as options for stress reporting from the user.

companion. The companion is responsible for sending the key-value dataset to the cloud when the smartphone is connected to the Internet. It sends the dataset to the cloud database using Fitbit Fetch API. The Fetch API allows developers to make GET and POST requests to HTTPS endpoints. Additionally, to distinguish each cadet's data from the others, the Fitbit-provided unique device identification (ID) number was used as the partition key in the database.

Fitbit automatically stores biometric data, through the companion, in its remote server, and we can retrieve this data using the Fitbit Web APIs. However, it requires specific authorization credentials, including access tokens and user identification. To validate the access tokens, users must go through the authorization code grant process, which depends on OAuth 2.0, a protocol to allow a third-party user to access the resources. The OAuth application creates the client ID and secret for use to invoke the access token. Each access token is only valid for 8 hours before it needs to be refreshed again.

C. Web Application

To handle multiple devices and data collection pipelines, we created a web application dashboard. The web application enables monitoring of the data collection for each user and shows the device information on the dashboard, such as battery status and the last synced time to their smartphone. It also provides tools for data retrieval and processing, including downloading interfaces that convert JSON data to CSV format and a visualization page to analyze the heart rate data with stress inputs for each cadet, as shown in Figure 3.

We implemented this web application using a serverless static web application through AWS. The benefit of static web hosting is that it minimizes the initial cost and eliminates the need for a hosting server, such as an EC2 instance. Instead, the web application runs in the S3 bucket, where HTML, CSS, and JS files are stored. The bucket has a feature to host an application with a designated AWS domain. However, since Fitbit only allows the HTTPS protocol to communicate with the outside, we purchased the private domain for the website and connected it to the S3 bucket using Route53, CloudFront, and Certificate Manager.

Most importantly, we utilized AWS Lambda functions to allow our static web application to act as a standard server. AWS Lambda is a serverless computing service that executes code without establishing a server. It is event-driven, executed only when the service is requested. Lambda helps the application build data processing functions by accessing other resources provided by AWS such as DynamoDB and S3. It can handle up to 250MB of code, and the execution time cannot be more than 15 minutes, which is sufficient for our needs. In the case of our application, we built lambda functions for creating and logging in user accounts, uploading and retrieving recorded data to the interface, and, in the case of in-cloud prediction,

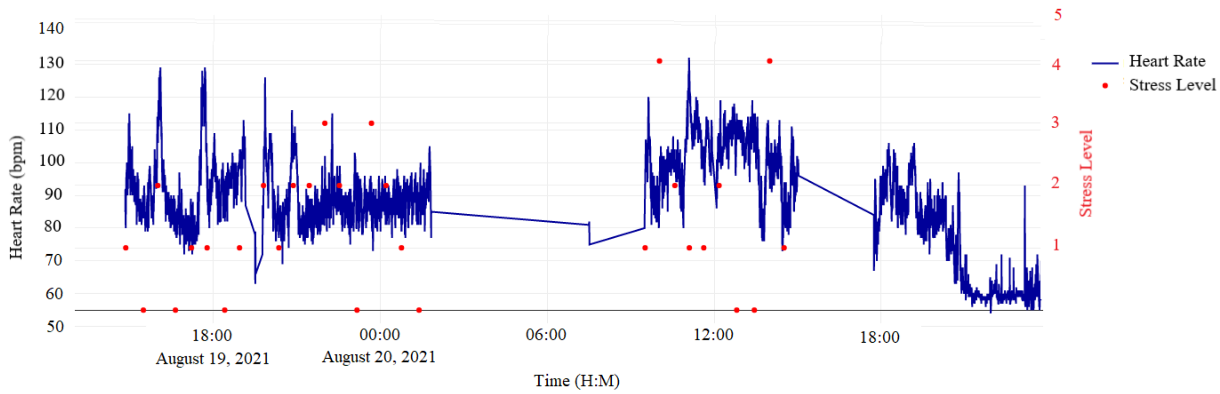

Fig. 3. A graph from the web application visualizing 24 hours of heart rate and stress data from one user. The blue lines represent heart rate in bpm, and the red dots represent stress prompts, ranging from no-answer (shown as 0) to 5.

kicking off inference tasks on pre-trained models.

D. Database

We employed DynamoDB, a non-relational key-value NoSQL database, to store all of our collected stress data. The key-value data is easy to query by the partition key or sort key, like extracting heart rate values for a specific user given start and end dates. Additionally, DynamoDB allows scaling both vertically and horizontally, which is important to support data generated continuously from many wearable devices. As mentioned earlier, biometric data is stored and accessed separately through the Fitbit-owned database.

Note that in this research, we obtained the cadets' permissions to store their data in the cloud and used the security mechanisms provided by AWS to protect the data. In real scenarios, users may not permit their private data to leave their personal devices. Our system can be extended to employ privacy-preserving machine learning methods such as federated learning to support such scenarios.

E. API Gateway

We utilized the API Gateway to handle requests from the companion using resources provided by the cloud. Our application runs in RESTful APIs, which requests and responds in JSON format with four methods, including CREATE (post), READ (get), UPDATE (put), and DELETE (delete). AWS's API Gateway provides users in the backend with the endpoints to access other services such as Lambda and DynamoDB. In our application, we built endpoints and mapped them to Lambda functions so that the companion can upload and process data, and the system can use the data to train models.

IV. Machine Learning-Based Stress Management

A. Problem Definition

As stated earlier, this research aimed to build a real-time stress prediction solution using machine learning. Our initial goal was to detect the patterns of heart rate that could indicate stress. Heart rate was collected through the Fitbit Versa 3, and the user supplied real-time stress perception scores using the clock face application. Our dataset, comprised of these data points, proved to be suitable for supervised learning algorithms, due to its labeled nature. The labels were also categorical variables, so we decided to employ classification algorithms to determine the input instances as "stressed" or "not stressed". Table I shows the distribution of stress levels from all the prompts that the users received. Stress level 0 indicates no response at the time of the stress prompt, while 1-5 indicates the increasing scale of stress.

Stress level	Numerical value	Number of responses
No Response	0	9578
Not Stressed	1	2935
A Bit Stressed	2	355
Moderate	3	89
A Lot	4	19
Extremely Stressed	5	5

TABLE I
Distribution of Stress Responses

B. Data Processing

As mentioned above, our work is appropriate for classification models of supervised learning given its labeled nature. We began by extracting heart rate segments from the original, continuous heart rate data for each user. To capture heart rate leading up to the stress prompt (programmed to generate when the current heart rate exceeded 35% above the baseline for 2 minutes), we extracted the 2 minutes of data before the prompt. Each heart rate segment was then mapped to corresponding stress levels to label each of them as "stressed" or "not stressed". We converted the stress level to "0" for "not stressed" and "1" for "stressed" for binary classification. Since users indicated their stress in five different levels, we could map them into "not stressed" and "stressed" in different ways. Given the ambiguity of stress level 2, we mapped and processed the data in two separate ways: 1) including data with stress levels from 2 to 5 as "stressed" while level 1 was "not stressed", and 2) including data with stress levels from 3 to 5 as "stressed" while levels 1 to 2 were "not stressed".

The pre-processed raw dataset includes the prompting threshold, resting heart rates, and 40 samples of heart rates.

	"Stressed'" levels	"Not Stressed" levels	Resampling method
1	2 to 5	1	Under-sampling
2	2 to 5	1	SMOTE
3	3 to 5	1 to 2	Under-sampling
4	3 to 5	1 to 2	SMOTE

TABLE II
FOUR APPROACHES FOR TRAINING MODELS

The granularity of heart rates recorded by Fitbit is between 5 to 10-seconds, so the 40 samples of heart rates represent about 2 to 3 minutes of data before the prompt. Since the resting heart rates are calculated by Fitbit automatically, we did not have to go through feature extraction steps to get those values.

The next step is feature extraction. Since we set the prompting algorithm based on how the heart rate fluctuates compared to the resting heart rate, we extracted the features representing distribution and fluctuation-related aspects of heart rate. There were seven features extracted from the heart rate. The primary five were 1) mean, 2) standard deviation, 3) minimum and 4) maximum value of the heart rate, and 5) resting heart rate. The additional two features were 6) the difference between mean heart rate from the resting heart rate by percentage (DiffRest) [11] and 7) the root mean square of successive differences between normal heartbeats (RMSSD) which Fitbit uses for calculating heart rate variability.

One issue we faced was the unbalanced nature of the dataset. Over 85% of the dataset contained "not stressed" instances. To avoid a learning bias across our models, we explored two approaches for dataset balancing: 1) To under-sample the majority class by randomly choosing as many instances as the number of the minority, and 2) To over-sample the minority instances using Synthetic Minority Over-Sampling Technique (SMOTE) [21]. The former method actually reduced the size of the data, proving to be ineffective for training models; however, it presented the ability to learn both "stressed" and "not stressed" equally. The latter increased the dataset size and the effectiveness of the learning stage but created similar but synthetic sampled instances. Lastly, to ensure all attributes had equal, unbiased influence, we implemented standard scaling on all the dataset columns to make them have the same distribution with 0 means and the unit standard deviation.

C. Training Models

We considered four approaches for training models as specified in Table II. We trained models separately by how we defined "stressed" (level 2 to 5 or 3 to 5) and how we balanced the dataset (under-sampling or over-sampling). Regarding classification algorithms, we considered four classic models and TensorFlow's feed-forward network. The set of classic models includes Decision Tree, Random Forest, Adaboost, and XGBoost. The first three models were imported from Scikit-learn (Sklearn) [22] which provides not only reliable classification models but also useful built-in functions for the machine learning process, such as *train_test_split* or *standard scaler*. Before running the dataset for training, we split the dataset into a training set (80%) and a testing set (20%) to evaluate the models' capability to handle unseen data.

	Model	Accuracy	F1 Score
Approach 1	Decision Tree	61.23%	60.51%
	KNN	60.14%	58.33%
	Random Forest	64.85%	65.72%
	AdaBoost	60.86%	58.33%
	XGBoost	70.28%	70.50%
	Neural Networks	56.52%	56.52%
Approach 2	Decision Tree	85.29%	85.63%
	KNN	81.08%	83.18%
	Random Forest	87.63%	88.14%
	AdaBoost	71.22%	73.37%
	XGBoost	88.60%	88.75%
	Neural Networks	87.77%	88.38%
Approach 3	Decision Tree	77.27%	76.92%
	KNN	77.27%	72.72%
	Random Forest	81.72%	81.83%
	AdaBoost	72.12%	71.64%
	XGBoost	78.78%	78.12%
	Neural Networks	86.36%	88.46%
Approach 4	Decision Tree	91.79%	91.73%
	KNN	88.34%	88.73%
	Random Forest	94.41%	94.34%
	AdaBoost	79.25%	80.15%
	XGBoost	**96.98%**	**96.95%**
	Neural Networks	**95.98%**	**96.02%**

TABLE III
EVALUATIONS BY ACCURACY AND F1-SCORE

We developed a feed-forward network on TensorFlow with six hidden layers activated by the ReLu function. The output layer was set with a Sigmoid function returning a value ranging from 0 to 1. If the final return value from an input instance is lower or equal to 0.5, it is "not stressed", which is labeled as 0. Otherwise, it is "stressed" and labeled as 1. Since we used binary classification, we used binary cross-entropy for the loss function.

D. Evaluation

Table III shows the accuracy and F1 scores for each approach. Approach 1 gave us the worst result. The models could not distinguish between "not stressed" and "stressed" by heart rate features. This might be because of either an insufficient number of instances or ambiguity of features from stress level 2. We could gain better results from Approach 2, which over-sampled the minority class. Approach 3 showed more reliable results than Approach 1 by not using over-sampling with synthetic samples. The dataset size was reduced even more than Approach 1, as it excluded stress level 2 from "stressed" and re-sampled the "not stressed" data as many as the number of "stressed" data samples. Although it does not have a sufficient dataset, it showed pretty good accuracy and F1 scores by properly inferencing on the testing set. Lastly, Approach 4 gave us the best results. It excluded stress level 2 and over-sampled the minority class. The best accuracy it reached is 96.98% from XGBoost and 95.98% from the Tensorflow neural network.

V. Edge Computing For Stress Prediction

A. In-Cloud vs. On-Edge Stress Prediction

Edge computing enables the operation of classification models on edge devices such as smart wearables and smartphones, alleviating the burdens of reaching the server in the cloud. Before we established the edge-based stress prediction, as a baseline, we implemented in-cloud stress prediction by running pre-trained models on EC2 instances with the input from the companion. On-edge stress prediction, in comparison, uses the companion, instead of the cloud server, to load models and predict stress to save the communication time between the companion and server.

Our Fitbit app always maintains the latest 2 to 3 minutes of heart rates using the queue data structure. When it needs to request stress prediction, it generates an input instance by calculating the seven features specified in Section B. It sends a request with a feature vector to either the companion or the cloud after extracting the features.

For on-edge stress prediction, the companion loads the pre-trained models and runs the inference. Unlike typical Android applications, the Fitbit companion cannot load files from the device's storage, and thus it has to retrieve the pre-trained models from the cloud. Specifically, the companion uses the Fitbit Fetch API to retrieve the models from the AWS S3 buckets via HTTPS. There is an initial overhead when the companion loads the models from the cloud into its memory when it performs the first inference; the following inferences can directly use the models that are already in memory (until the app restarts). Since the Fitbit app and the companion run in JavaScript, we had to convert the models, originally written in Python for TensorFlow, to JavaScript versions, utilizing the TensorFlow library. Among the four approaches we tested, we chose Approach 4's model, which achieved the highest accuracies and F1 scores. Because the smartphone has limited computing power and memory capacity, we considered model optimization techniques such as quantization for reducing the complexity and size of the models.

For in-cloud stress prediction, we used the API Gateway that handles requests from the companion, runs model inference, and returns the prediction result. The companion communicates with the Gateway using the Fitbit Fetch API to upload, retrieve, and manipulate data from our storage tables. When requesting stress prediction, the companion uses the *POST* method of the Fetch API to send the inference input containing heart rate features. Upon receiving the request, the Gateway triggers Lambda functions to proceed with stress prediction using pre-trained models fetched from the S3 bucket. Since the cloud has sufficient resources, far superior to the companion, the server-side models have not gone through model optimization processes.

B. Evaluation

Since the companion has limited computing resources, we compressed the model to optimize for the edge device. We applied quantization to reduce the precision of model parameters to 8-bits from the original model. There are two approaches to quantization, post-quantization and quantization-aware training. In this work, we used the quantization-aware training approach because it generally creates more accurate quantized models by considering the error introduced by quantization during the training process. Table IV shows the accuracy and the size of the models. The topology size slightly increases from the original to the quantized model but decreases by 74.2% for the overall weight size. At the same time, the accuracy across both models only varies by 2%.

Model	Type	Value
Original Model	Accuracy	85.98%
	Topology Size	0.005 MB
	Weight Size	3.0 MB
Quantized 8-bit Model	Accuracy	84.10%
	Topology Size	0.0067 MB
	Weight Size	0.774 MB

TABLE IV
ACCURACY AND THE SIZE OF MODELS

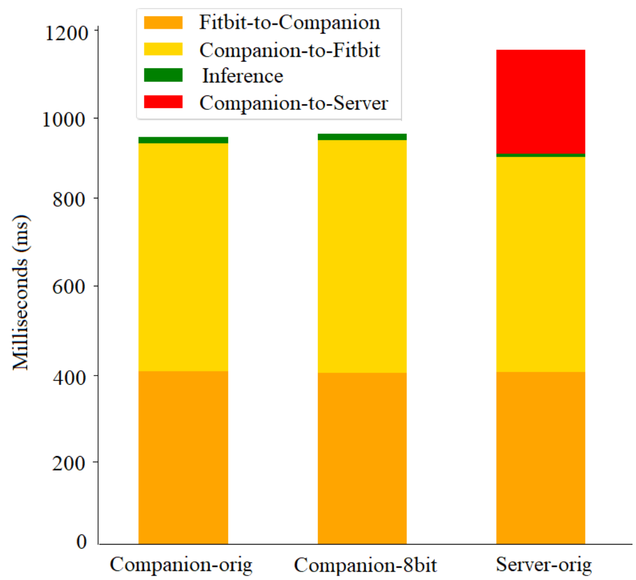

Fig. 4. Latencies of in-cloud vs. on-edge stress prediction with their breakdowns. Fitbit-to-Companion is the latency of transferring input from Fitbit to the smartphone companion; Inference is the time for stress prediction using model inference; Companion-to-Fitbit is the latency of sending the prediction result back to Fitbit. In addition, for in-cloud prediction, Companion-to-Server is the latency for transferring input and response between the companion and the cloud.

Figure 4 illustrates the stress prediction latencies and their breakdowns. We compared three scenarios: original model on-edge ("Companion-orig"), quantized 8-bit model on-edge ("Companion-8bit"), and original model in-cloud ("Server-orig"). On one hand, the overall latency difference for both on-edge scenarios proves to be negligible, with small variations across data transfer steps. Nonetheless, the quantized model on the edge still saves nearly 74.2% of memory usage compared to the traditional server, which is still valuable for resource-limited smartphones. On the other hand, it can be observed that the in-cloud deployment takes an extra 200ms on average, mainly contributed by the additional data transfer overhead between the companion and server. This result confirms that on-edge deployment of stress prediction produces lower latency than in-cloud deployment.

VI. Conclusion

In this paper, we studied the feasibility of edge machine learning-based stress prediction using wearable devices with police cadets during a training academy. We focused on heart rate as an important starting point and verified that heart rate recorded by commercial wearable devices can be effectively used for stress prediction. Unlike other equipment which records ECG data in milliseconds, Fitbit reads heartbeats in 5 to 10-second granularity (with a 1-second segment extraction option). However, with features extracted from the heartbeats and participant-reported stress levels, we found that Fitbit-based heartbeat data also can be a validated data type indicating stressful or non-stressful circumstances, even when measured under real-world conditions.

Next, we employed five classic classification models and one neural network for binary classification. After segmenting heart rate data into 2 to 3-minute windows and extracting five statistical features and two features representing heart rate variability, we could apply the dataset to machine learning algorithms. We also resolved the imbalance of the dataset by either under-sampling or over-sampling. The best accuracies were 96.98% from XGBoost and 95.98% from a 6-layer feed-forward neural network using an approach that over-sampled the minority instances and only used levels 3 to 5 as "stressed" while categorizing levels 1 and 2 as "not stressed".

Lastly, we optimized our neural network model for edge deployment (on the Fitbit companion) using quantization which reduced the model size by 74.2% and maintained the accuracy as before the quantization. On-edge deployment of the quantized model to the companion saved approximately 200 milliseconds, proving to be promising as we move forward.

VII. Acknowledgement

We thank the anonymous reviewers for their feedback. This research is sponsored by the U.S. National Science Foundation awards OAC-2126291 and CNS-1955593, with support from Arizona State University's Institute for Social Science Research and School of Social and Behavioral Sciences. We also thank the Phoenix Regional Police Academy for their support and engagement.

References

[1] R. S. Lazarus and S. Folkman, *Stress, appraisal, and coping*. Springer publishing company, 1984.

[2] N. Atsan, "Decision-making under stress and its implications for managerial decision-making: A review of literature," *International Journal of Business and Social Research*, vol. 6, p. 38, 04 2016.

[3] S. P. Griffin and T. J. Bernard, "Angry aggression among police officers," *Police quarterly*, vol. 6, no. 1, pp. 3–21, 2003.

[4] ——, "Angry aggression among police officers," *Police Quarterly*, vol. 6, no. 1, pp. 3–21, 2003.

[5] K. Hoemann, Z. Khan, M. J. Feldman, C. Nielson, M. Devlin, J. Dy, L. F. Barrett, J. B. Wormwood, and K. S. Quigley, "Context-aware experience sampling reveals the scale of variation in affective experience," *Scientific reports*, vol. 10, no. 1, pp. 1–16, 2020.

[6] S. Pieper and J. F. Brosschot, "Prolonged stress-related cardiovascular activation: Is there any?" *Annals of Behavioral Medicine*, vol. 30, no. 2, pp. 91–103, 2005.

[7] J. M. Reingle Gonzalez, K. K. Jetelina, S. A. Bishopp, M. D. Livingston, R. A. Perez, and K. P. Gabriel, "The feasibility of using real-time, objective measurements of physiological stress among law enforcement officers in dallas, texas," *Policing: Int'l J. Police Strat. & Mgmt.*, vol. 42, p. 701, 2019.

[8] S. Cohen, T. Kamarck, R. Mermelstein *et al.*, "Perceived stress scale," *Measuring stress: A guide for health and social scientists*, vol. 10, no. 2, pp. 1–2, 1994.

[9] S. Beniczky, P. Karoly, E. Nurse, P. Ryvlin, and M. Cook, "Machine learning and wearable devices of the future," *Epilepsia*, vol. 62, pp. S116–S124, 2021.

[10] Y. Rykov, T.-Q. Thach, I. Bojic, G. Christopoulos, J. Car *et al.*, "Digital biomarkers for depression screening with wearable devices: cross-sectional study with machine learning modeling," *JMIR mHealth and uHealth*, vol. 9, no. 10, p. e24872, 2021.

[11] W. Lawanont, P. Mongkolnam, C. Nukoolkit, and M. Inoue, *Daily Stress Recognition System Using Activity Tracker and Smartphone Based on Physical Activity and Heart Rate Data*, 01 2019, pp. 11–21.

[12] R. Dai, C. Lu, L. Yun, E. Lenze, M. Avidan, and T. Kannampallil, "Comparing stress prediction models using smartwatch physiological signals and participant self-reports," *Computer Methods and Programs in Biomedicine*, vol. 208, p. 106207, 2021.

[13] Y. S. Can, N. Chalabianloo, D. Ekiz, and C. Ersoy, "Continuous stress detection using wearable sensors in real life: Algorithmic programming contest case study," *Sensors*, vol. 19, no. 8, p. 1849, 2019.

[14] G. J. Martinez, T. Grover, S. M. Mattingly, G. Mark, S. D'Mello, T. Aledavood, F. Akbar, P. Robles-Granda, and A. Striegel, "Alignment between heart rate variability from fitness trackers and perceived stress: Perspectives from a large-scale in situ longitudinal study of information workers," *JMIR human factors*, vol. 9, no. 3, p. e33754, 2022.

[15] A. Tiwari, R. Cassani, J.-F. Gagnon, D. Lafond, S. Tremblay, and T. H. Falk, "Prediction of stress and mental workload during police academy training using ultra-short-term heart rate variability and breathing analysis," in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, 2020, pp. 4530–4533.

[16] M. L. Erickson, W. Wang, J. Counts, L. M. Redman, D. Parker, J. L. Huebner, J. Dunn, and W. E. Kraus, "Field-based assessments of behavioral patterns during shiftwork in police academy trainees using wearable technology," *Journal of Biological Rhythms*, vol. 37, no. 3, pp. 260–271, 2022.

[17] Y. Chen, S. Biookaghazadeh, and M. Zhao, "Exploring the capabilities of mobile devices supporting deep learning," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 17–18. [Online]. Available: https://doi.org/10.1145/3220192.3220460

[18] T. R. Mauldin, M. E. Canby, V. Metsis, A. H. H. Ngu, and C. C. Rivera, "Smartfall: A smartwatch-based fall detection system using deep learning," *Sensors*, vol. 18, no. 10, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/10/3363

[19] S. S. Ogden and T. Guo, "MODI: Mobile deep inference made efficient by edge computing," in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*. Boston, MA: USENIX Association, Jul. 2018. [Online]. Available: https://www.usenix.org/conference/hotedge18/presentation/ogden

[20] T. Guo, "Cloud-based or on-device: An empirical study of mobile deep inference," in *2018 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2018, pp. 184–190.

[21] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.