E-stress detector: predicting stress level from computer usage patterns

Fabian Okeke* FNO2@CORNELL.EDU
Cornell University, Ithaca, NY 14850, USA

Vincent Tseng* WT262@CORNELL.EDU
Cornell University, Ithaca, NY 14850, USA

Abstract

Stress is inevitable. Sometimes people fall back to their old habits or perform a series of actions that can be directly linked to their stressful states. Identifying stress is important but the approach scales poorly when invasive sensors have to be used or the when the research method relies on induced stress in a timed lab experiment. Our solution aims to identify stressful states based on continuous logging of keystroke dynamics, mouse patterns, and foreground application usage. We propose a privacy-aware system, E-stress detector, that logs participants' computer activities alongside their stress self report. After extracting relevant features, our top classifiers provide highest accuracy when training-test split is 70% to 80% for predicting user future stress level based on historical data.

1. Introduction

As we know, accomplish daily tasks at work can lead to stress and when too much becomes overstress. Overstress undermines our productivity, and more seriously, it might bring harm to our health, even cause mental diseases. However, sometimes we tend to feel exhausted and stressed, but unaware of what cause us to feel stressed exactly. Therefore, it is crucial to find out the sources of stress, and at some appropriate time, give users some reminders that you are overstressed and you should take a break and relax yourself right away.

When people are under stress, their behaviors tend to change accordingly. In particular, we have observed that people's computer usage patterns vacillate—the way they type, click, browse webs change with regard to their stress, or anxiousness level. For example, when people feel stressed, they click the mouses with higher frequency; they tend to have more typos, and thus the frequency of doing error correction raises; the frequency of switching between different web pages also increases; they might check their phones more frequently but unconsciously, etc. We suppose these are good indicators to users stress level, and by analyzing users baseline stress level and detecting abnormal stress level, we will be able to understand whether they are overstressed and give proper reminders.

Since the computing age has encouraged working on personal computers, stress detection correlated to usage of these electronic devices could lead to insightful results. This research aims to discover which computer activities, if any, correlate with individual stress levels. Can one's typing pattern, duration on a browser, rate of mouse clicking or overall computer usage be linked to their stress level? Insightful results could have multiple applications such as personalized health recommendation systems for handling stress, timely interventions for detecting stress at an individuals peak, holistic feedback for doctors during diagnosis, among other health driven and personal purposes. This could also be an infomative tool for deploying new students to a tool introduced in a course or introducing employees to a new software just adopted in a company. [1]

2. Related Work

Computer activities such as keystroke dynamics and mouse click patterns have been used in detecting user

*Both authors contributed equal amount of work

[1] *both authors contributed equal amount to the work

emotional and stress states. Related works can be broadly grouped into four major categories: Affective Computing, Keystroke Dynamics, Mouse Usage, and Computer Applications.

2.1. Affective Computing

Epp Clayton et al, arguably did the first major work connecting keystroke dynamics and emotional states. This research measured 15 emotional states of users using periodical self report and keystroke features throughout the day (Epp et al., 2011). Hong proposed StressSense, which can detect human stress by focusing on paralinguistic features instead of actual vocal contents (Lu et al., 2012). StudentLife utilized SurveyMonkey and mobileEMA to measure participants stress as ground truth (Wang et al., 2014). Similarly, we will use the same approach as our ground truth. Instead of 15 emotional states, our research primarily focuses on user's level of stress.

2.2. Keystroke Dynamics

Hernandez, Javier, et al in Under Pressure use a pressure sensitive keyboard and capacitive mouse to discriminate between stressful and relaxed states in a lab setting (Hernandez et al., 2014). To allow our application to scale easily, we do not rely on using custom hardware.

2.3. Mouse Usage

Early work done by Ark Wendy S. et al in The Emotion Mouse relate the mouse touch to the emotion attached to a computer task. GSR and chest sensors were used to collect heart rate, temperature, galvanic skin response (GSR), and somatic movement, which were measured against the six Ekmans facial emotions (Ark et al., 1999). In more recent work, Sun, David et al in MouStress proposed an approach to detect stress from mouse motion, but from a physiological perspective—a model of the arm (Sun et al., 2014). Our approach is less intrusive as we do not propose an external sensor; rather, we focus on non-invasive continuous mouse tracking activites such as click rate and coordinates.

2.4. Computer Applications

Karpathy on his blog shows how he measures computer activities for self tracking. However, this is done with the idea of quantifying productivity without linking user stress state. Some of the ideas used in our project design were inspired by work on Karpathy's system (Karpathy).

Our work differs from all other works in 3 major ways: a custom hardware does not need to be installed; data collection is not done in a stress-induced lab setting for a short period of time, rather data is continuously logged with the intent of long-term continuity; we are the first, as far as we know, to incorporate computer usage application as part of stress sensing.

Unlike previous works, which focus on only keystrokes, only mouse, or both, our work leverages foreground application usage as an extra data source. This is based on the intuition that certain applications might immensely affect user emotional state compared to others. For instance, a non-programmer using Matlab for the first few days might hit the backspace key to delete wrongly written code; click on the menu buttons several times, perhaps in search of a functionality or switch applications multiple times between Matlab software and a browser page showing "how to perform matrix multipication in Matlab".

It is worthwile to note that previous works have logged user computer activities without explicitly discussing how user data is protected. As such, a significant amount of work went into building the tool necessary for collecting data while preserving user privacy. Our application takes a different approach by only logging specific keys when users type. Every other key is blocked with the symbol "$$". In addition, every data logged is not uploaded to a cloud server as the data is locally stored on the user's computer.

For project scope, we do not use body sensors as our work focuses on self-report of stress state as with existing literature (Epp et al., 2011), (Wang et al., 2014).

3. Methodology

The work involves two major components: the data logging process done in-situ – as participants carried out their daily activities, and the data processing required to classify stress states. For the data collection, users' keystrokes, mouse activities, and computer applications usage are timestamped and logged. The data processing involved extracting relevant keystroke features in order to build Machine Learning classifiers.

3.1. Experience Sampling of Computer Activities

Experience Samping Method (ESM) involves continuously logging computer activities while periodically collecting user responses to self-reports of their level of stress (Hektner et al., 2007). Stress states were measured using Photographic Affect Meter (PAM) (Pollak et al., 2011)and three-question likert-scale survey.

Figure 1 shows the PAM survey. PAM existed for only mobile phones but we have ported it to computers. During periodic interval—we arbitrarily select 30 minutes interval—a webpage automatically opens up and a user is asked *"How do you feel right now?"* Then the user selects one of 16 grid pictures that reflects their internal emotional state. Each grid represents each of 16 emotions and there are 3 possible images that could be displayed per grid. On clicking *You can load more images button*, some of the grid images change based on randomized selection. In our study, we divide the positive and negative emotions into stress and not-stressed respectively. (Pollak et al., 2011) provides detailed information on valence and arousals and how these map to user emotions, which we then use as a stress self-report.

Figure 2 shows a self-report questionnaire. After selecting a picture in PAM, a 3-question survey page opens up where the user gets to pick one response out of five categories.

Like StudentLife, (Wang et al., 2014), our groundtruth is the 3-question survey. StudentLife precisely used a one-question survey which is the same as the third question in our 3-question survey; the only difference in our approach is that we add two extra questions—the first and the second—in order to capture more input from the user. Input from PAM is compared to the responses from the 3-question survey and this can help control the amount of irregular reports made by an end user. For instance, if a user selected the ice-cream picture on the top right of figure 1 but "very stressed" in the 3-question survey, then we can easily identify that the input is incorrect.

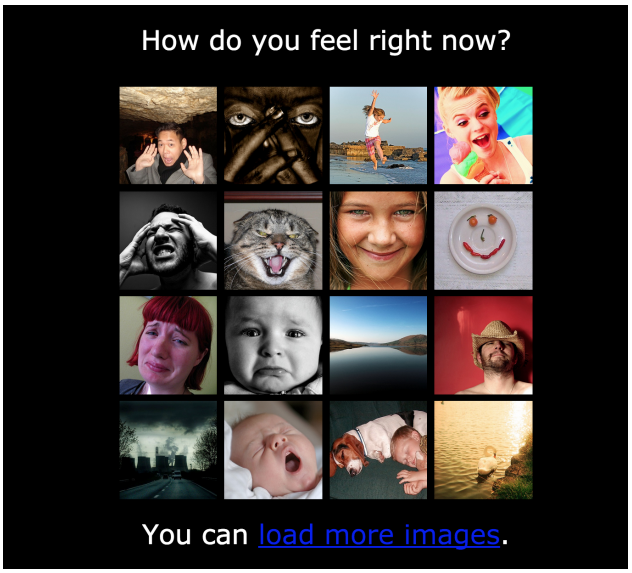


Figure 1. PAM: the user selects one of 16 grid pictures that reflects their internal emotional state.

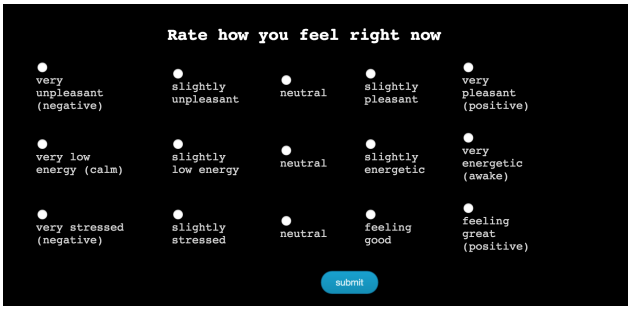


Figure 2. A 3-question survey: the user gets to pick one response out of five categories for each question.

3.2. Data Collection Software

For visualizing keystrokes heatmap and mouse heatmap, we use WhatPulse (wha). This can easily be downloaded and installed on the end user system. Since WhatPulse does not provide data to the granularity necessary for conducting our research, a system was built that logs user keystrokes, mouse activities and applications used throughout the day. While there are a few open sourced tools that could have been adopted, a custom application was built because existing tools suffer from poor documentation and/or incomprehensible programming approaches.

Upon installation, our software tool contains a keystroke logger, a mouse logger, an application logger and a self-report logger. The keyboard and mouse applications continuously logs keystrokes and mouse activities; the foreground application is logged every 3 seconds; while the stress self-report is logged every 30 minutes. The self-report application starts up a PHP server, opens PAM webpage with the user's default browser. Upon selecting a choice from the 16-grid pictures, the response is saved in a file. Thereafter, the 3-question survey is opened as the next webpage, which upon answering a "thank-you" page loads. After about 5 seconds, the "thank-you" page automatically closes. Self-report application logs user response based on when they responded and not when the page opens. For instance, if the Web page opens at 4pm but the user responds at 7pm, then the response is logged at 7pm.

Every 15 minutes, a background script checks if all the applications are running or if they have been stopped either manually by the user, another running process, or through a power down of the user computer. This ensures that user logging continues non-intrusively as

users do not have to manually start the applications if they were shut down. In order to maintain user access control, every activity application has an icon that appears in the dock so that the user can close the application whenever they want to; users can also delete the data directory if they don't want to partake in the experiment. These intervals for each application were arbitrarily chosen.

In order to obtain personal visualization of data collected so far, a user has to run the "view-my-stats" script. This automatically starts the server and opens up the visualization webpage based using default browser. Due to space limitation, details of this including sample visualization are left out as this feature exists to keep the user informed of their computing habits.

3.3. Data Format

Each activity logged had specific data formats that aided the analysis and visualizations.

mouse logging format:
time | mouse_event | mouse_direction | y=y_coordinate x=x_coordinate

where mouse_event could be *MouseButtonHooker* and mouse_direction is either of *NSRightMouseUp, NSRightMouseDown, NSLeftMouseUp, NSLeftMouseDown.*

For instance:
2015-05-03 12:18:55.438360 MouseMoveHooker NSMouseMoved y=606.19140625 x=683.0390625

keyboard logging format:
time | key_event | key_direction | char= char_pressed key=numeric_value mods=[$'fcn\_value'$] is_repeat=boolean_value.

where key_event is *KeyHooker*; key_direction is either of *NSKeyUp, NSKeyDown*; char_value is the actual key typed (this shows as $$ to protect user privacy), mods is any key command combined with an alphabetic key such as SHIFT, CONTROL, ALTERNATE, etc, which are referred to as fcn_value i.e. function value; is_repeat is either of *True, False* depending on if a key is held down or not.

For instance:
2015-05-03 12:20:28.903225 | KeyHooker | NSKeyDown | char=$$ mods=[$'COMMAND'$] key=$$ is_repeat=False

application logging format:
time | application

For instance:
2015-05-07 22:39:09.113787 | Google Chrome

pam logging format: time, image_id: img_num, cell_id: id_num, pam_pa: pa_num, pam_na: na_num, arousal: arousal_num, valence: valence_num.

For instance:
time: 2015-04-29 23:47:11, image_id: 11, cell_id: 4, pam_pa: 16, pam_na: 3, arousal: 4, valence: 4.
where each image has an associated id that identifies the exact picture selected by the user. Details about the scoring mechanism can be found in the PAM literature (Pollak et al., 2011).

ema logging format:
time: 2015-04-27 03:59:49, pleasant_state: p_state, energy_state: e_state, stress_state: s_state.
where each of the states are one out of the five options as shown in the 3-point survey question above.

For instance:
time:2015-04-2703:59:49, pleasant_state: slightly_pleasant, energy_state: slightly_energetic, stress_state: feeling_good.

3.4. Feature Extraction

We segmented the raw computer usage activities with sliding window size = 1 minute and the extracted three types of computer usage features, **M**, **K**, and **A** for mouse clicks, keystrokes, and application usage within each sliding window. $\mathbf{M} = [d_m, r_m, t_m]$, where d_m is the total distance of mouse movement, r_m is the mouse click rate, and t_m is the average mouse button pressed duration. $\mathbf{K} = [r_{k_1}, t_{k_1}, r_{k_2}, t_{k_2}, t_{k_i}, ..., r_{k_N}, t_{k_N}]$, i $\in N$, where k_i stands for different keys and N stands for total number of key types we logged, and $N = 9$ in our case, including function keys, letter keys, enter key, tab key, space key, delete key, escape key, arrow key, and all the keys that were pressed (that is, the previous key types were also taken into account), and t_{k_i} is key pressed time for key k_i. $\mathbf{A} = [t_{a_1}, t_{a_2}, t_{a_j}, ..., t_{a_P}, x]$, j $\in P$, where a_i stands for different applications, P is the total number of applications users might open when using computers, t_{a_j} is the time users spent on application a_j, and x is the number of active applications within each time window. To know decide what P is, we first iterated through the users' historical application usage logs.

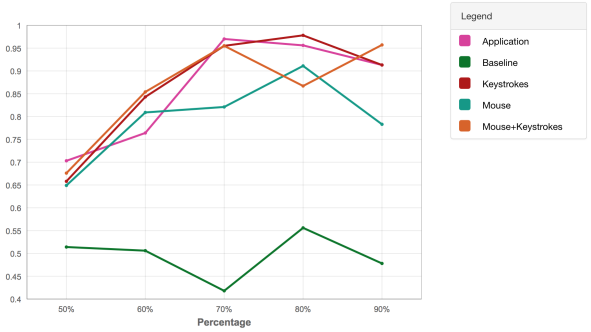
3.5. Algorithms

Since the self-report survey popped up every 30 minutes, we only considered raw computer usage features within the 30-minute duration, if the computer was active, before the survey is filled out. That is, assuming the user filled out the survey at timestamp T_end, and the computer got active at timestamp T_c, we only considered raw features with timestamps in $[T_{begin}, T_s)$, where T_{begin}=$min(T_{end-1800}, T_c)$. And according to the previous section, we extracted high level features from the raw features among each period and got computer usage samples $S_{T_{begin}}, S_{T_{begin+60}}, ..., S_{T_{end-60}}, S_{T_{end}}$, where each sample S was labeled with the corresponding binary results of users' stress level based on the participants' EMA self-report score. Then we used Decision Tree, Random Forest, support vector machine (SVM), as well as participants' historical data for future stress prediction.

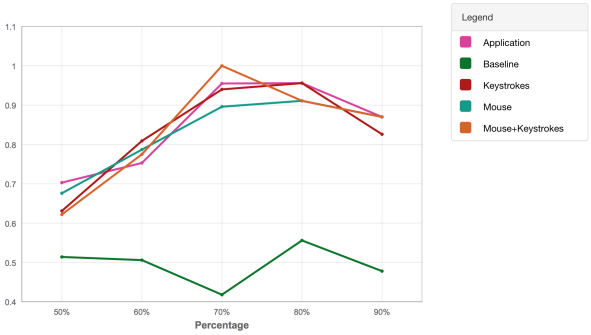
4. Results

Due to time limit and the sensitivity of data initially collected, we could only collect data and train from one participant for consecutive 3 days after the deployment, and we got 352 samples from the deployment. We tried different ways of splits for training and testing, from 50-50 split to 90-10 split, well as different combinations of features (M, K, A, $M + K$). But we found that the prediction accuracy was very low using only A, because during each time window, the participant tended to open just a couple of applications, which made the feature vector of A become very sparse and in turn diminished A's ability for prediction. As a consequence, we decided not to consider all the applications users might use as features. Instead, we only took into account the top 5 frequently used applications as features. That is, the application usage features became $A' = [t_{a'_1}, t_{a'_2}, t_{a'_3}, t_{a'_4}, t_{a'_5}, t_{a'_6}, x]$, where $t_{a'_1} \sim t_{a'_5}$ are the amount of time users spend on the top 5 most frequently used applications during each 1-minute window, and $t_{a'_6}$ is the amount of time users spend on the rest of the applications. After the features were adjusted, the accuracy significantly improved. Table 1 is the accuracy of stress prediction with different algorithms, different combinations of features, as well as different ways of data splits, and Figure 3 is the visualization of the comparison of different modalities. For the baseline algorithm, it randomly guesses whether the user is stressed or not.

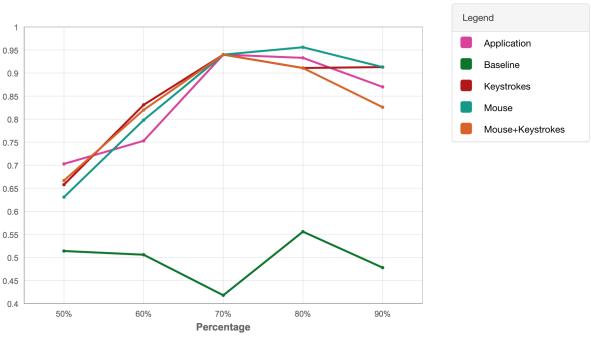
In general, the algorithms achieve the highest accuracy when 70% $\sim$ 80% data is used as training data, and the accuracy goes down if more data is used as training



a. Decision Tree



b. Random Forest



c. SVM

Figure 3. The accuracy of stress prediction with different algorithms: (a) Decision Tree, (b) Random Forest, (c) Support Vector Machine (SVM) and with different ways of data splits.

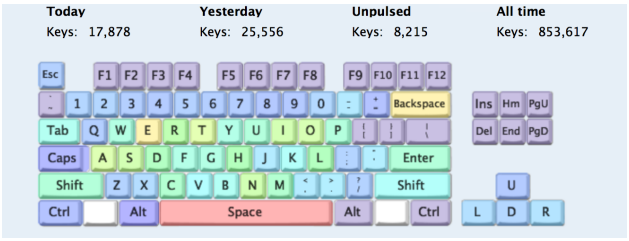


Figure 4. WhatPulse: Keyboard Heatmap

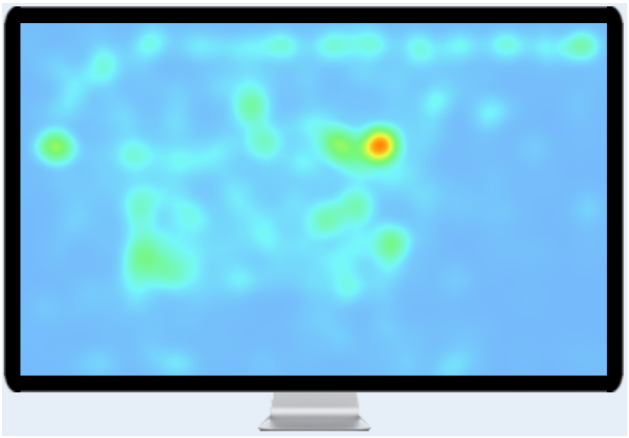


Figure 5. WhatPulse: Mouse Heatmap

data. We have to collect more data and calculate their p-values in order to know what kind of data splits can really achieve the best accuracy and yet the result it obtains is statistical significance, since it is a must for the system to know to how many historical data it has to acquire in order to predict future stress level correctly for the next whole day.

Initially, we assumed that the more features we used, the better accuracy we would get, but it turned out we got worse accuracy if we built the models using multiple features at the same time, $M + K$ for instance. According to Figure 3, the models trained with $M+K$ performed worse than models trained only with single feature during most of the time. Our speculation is that owning to the limited amount of data, our models tend to over-fit the data if too many features are given. On the other hand, we also noticed that different algorithms perform differently given different types of features. For example, Decision Tree gives the highest prediction accuracy with application usage features, whereas Random Forest somewhat does better with keystrokes features. Therefore, for our next step, we are considering using ensemble methods, combining all these different models together and generating the final prediction with the linear combination of the results given by individual algorithms, where each algorithm is only trained with single type of features with which it is most discriminative.

Modalities	M			K		
Algorithms	DT	RF	SVM	DT	RF	SVM
50%	0.649	0.676	0.631	0.658	0.631	0.658
60%	0.809	0.787	0.798	0.843	0.809	0.831
70%	0.821	0.896	0.94	0.955	0.94	0.94
80%	0.911	0.911	0.956	0.978	0.956	0.911
90%	0.783	0.87	0.913	0.913	0.826	0.913
Modalities	A			M+K		
Algorithms	DT	RF	SVM	DT	RF	SVM
50%	0.703	0.703	0.703	0.676	0.622	0.667
60%	0.764	0.753	0.753	0.854	0.775	0.82
70%	0.97	0.955	0.94	0.955	1	0.94
80%	0.956	0.956	0.933	0.867	0.911	0.911
90%	0.913	0.87	0.87	0.957	0.87	0.826
Modalities	M+A			K+A		
Algorithms	DT	RF	SVM	DT	RF	SVM
50%	0.622	0.631	0.613	0.622	0.649	0.658
60%	0.809	0.775	0.764	0.787	0.798	0.831
70%	0.925	0.91	0.955	0.94	0.94	0.955
80%	0.889	0.911	0.933	0.933	0.933	0.933
90%	0.783	0.87	0.913	0.87	0.957	0.913
Modalities	M+K+A					
Algorithms	DT	RF	SVM			
50%	0.613	0.64	0.631			
60%	0.775	0.787	0.742			
70%	0.896	0.955	0.925			
80%	0.889	0.956	0.956			
90%	0.783	0.826	0.913			

Table 1. Prediction accuracy with different modalities, algorithms and data splits. M stands for mouse usage, K for keystrokes, and A for application usage.

5. Limitations

Our dataset is limited. Since recruited participants were PhD students, our prediction model performance cannot be generalized to a broader audience. Since self-report surveys are subject, the definition of stress for different participants will have varying likert scale values.

Self-report periodically popping up can become annoying to users and/or boring overtime; this invariably means providing misleading training data for our algorithm should users absent-mindedly select a response or none at all. Moreover, emotional states are short-lived and a user can have multiple emotions in a short time so the surveys cannot fully capture the end user's

entire emotional states but only a portion.

6. Future Work

Future work will involve external sensors for measuring HR, BP, EEG. The current application focuses on batch learning of data already amassed so next iteration will involve online learning especially for making efficient recommendation systems.

We also plan to use active learning during training and testing in order to overtime reduce the amout of self-reports that occur: in this future approach, the algorithm starts by collecting self-reports periodically. Later, it only prompts the user for self-report after detecting anomalies during data logging.

Further, we intend to collect data for a broader audience and for a longer time to demonstrate more applications of our system and algorithm.

7. Conclusion

In this work, we present a system that predicts user future level of stress based on historical data of keystroke dynamics, mouse activities, application usage, and self report of stress state. Major contributions of our work to existing literature include: building a custom privacy-aware open-sourced stress keylogger, adding computer applications as a contextual feature, deploying diverse machine learning prediction techniques, and providing personalized visualization, which the end user can control. Even though our project is currently limited by the amount of data collected so far, our system and algorithm still demonstrates its functionality; we are confident that our system will demonstrate its full capabilities upon collecting data for a longer period of time. Our work mildly addresses personalized recommendation so future work can building upon more tailored result for each end user.

Acknowledgments

Special thanks to Prof Daniel Cosley, students of INFO 6010, reviewers of Advanced Machine Learning course (Spring 2015), members of the PAC lab at Cornell University Information Science, and Prof Tanzeem Choudhury, whose guidance was instrumental in this project.

References

Keystroke. https://github.com/fnokeke/keystroke-stress. Accessed: 2015-05-01.

WhatPulse. https://whatpulse.org/. Accessed: 2015-05-01.

Ark, Wendy S, Dryer, D Christopher, and Lu, Davia J. The emotion mouse. In *HCI (1)*, pp. 818–823, 1999.

Epp, Clayton, Lippold, Michael, and Mandryk, Regan L. Identifying emotional states using keystroke dynamics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 715–724. ACM, 2011.

Hektner, Joel M, Schmidt, Jennifer A, and Csikszentmihalyi, Mihaly. *Experience sampling method: Measuring the quality of everyday life*. Sage, 2007.

Hernandez, J, Paredes, P, Roseway, A, and Czerwinski, M. Under pressure: Sensing stress of computing users. *Proc. of Computer and Human Interaction*, 2014.

Karpathy, Andrej. Quantifying Productivity. http://karpathy.github.io/2014/08/03/quantifying-productivity/. Accessed: 2015-05-01.

Lu, Hong, Frauendorfer, Denise, Rabbi, Mashfiqui, Mast, Marianne Schmid, Chittaranjan, Gokul T, Campbell, Andrew T, Gatica-Perez, Daniel, and Choudhury, Tanzeem. Stresssense: Detecting stress in unconstrained acoustic environments using smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pp. 351–360. ACM, 2012.

Pollak, John P, Adams, Phil, and Gay, Geri. Pam: a photographic affect meter for frequent, in situ measurement of affect. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 725–734. ACM, 2011.

Sun, David, Paredes, Pablo, and Canny, John. Moustress: detecting stress from mouse motion. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pp. 61–70. ACM, 2014.

Wang, Rui, Chen, Fanglin, Chen, Zhenyu, Li, Tianxing, Harari, Gabriella, Tignor, Stefanie, Zhou, Xia, Ben-Zeev, Dror, and Campbell, Andrew T. Studentlife: assessing mental health, academic performance and behavioral trends of college students using smartphones. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 3–14. ACM, 2014.