# Representation Learning for Emotion Recognition from Smartphone Keyboard Interactions

**Conference Paper** · September 2019

DOI: 10.1109/ACII.2019.8925518



**5 authors**, including:

Surjya Ghosh
Centrum Wiskunde & Informatica
**22** PUBLICATIONS **61** CITATIONS

SEE PROFILE

Niloy Ganguly
Indian Institute of Technology Kharagpur
**315** PUBLICATIONS **3,878** CITATIONS

SEE PROFILE

Pradipta De
Georgia Southern University
**89** PUBLICATIONS **1,225** CITATIONS

SEE PROFILE

# Representation Learning for Emotion Recognition from Smartphone Keyboard Interactions

Surjya Ghosh*‡, Shivam Goenka*, Niloy Ganguly*, Bivas Mitra*, Pradipta De†
*Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, INDIA 721302
†Department of Computer Science, Georgia Southern University, USA
‡Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
Email: {surjya.ghosh, shivamgoenka}@iitkgp.ac.in, {niloy, bivas}@cse.iitkgp.ac.in, pde@georgiasouthern.edu

***Abstract*—Characteristics of typing on smartphone keyboards among different individuals can elicit emotion, similar to speech prosody or facial expressions. Existing works on typing based emotion recognition rely on feature engineering to build machine learning models, while recent speech and facial expression based techniques have shown the efficacy of learning the features automatically. Therefore, in this work, we explore the effectiveness of such learning models in keyboard interaction based emotion detection. In this paper, we propose an end-to-end framework, which first uses a sequence-based encoding method to automatically learn the representation from raw keyboard interaction pattern and subsequently uses this representation to train a multi-task learning based neural network (MTL-NN) to identify different emotions. We carry out a 3-week in-the-wild study involving 24 participants using a custom keyboard capable of tracing users' interaction pattern during text entry. We collect interaction details like touch speed, error rate, pressure and self-reported emotions (*happy, sad, stressed, relaxed*) during the study. Our analysis on the collected dataset reveals that the representation learnt from the interaction pattern has an average correlation of 0.901 within the same emotion and 0.811 between different emotions. As a result, the representation is effective in distinguishing different emotions with an average accuracy (AUCROC) of 84%.**

***Index Terms*—Representation learning, Emotion detection, Keyboard interaction, Smartphone interaction**

## I. Introduction

The keyboard interactions on smartphones have been researched as an effective modality for emotion detection [1]–[9]. However, the underlying patterns are complex enough that require extensive feature engineering to construct an accurate emotion prediction model from smartphone keyboard interactions. Some of the recent works on emotion detection based on other modalities, such as facial expressions, and speech characteristics, showed that automatic feature extraction can be as effective as feature engineering [10], [11]. Hence, applying automatic feature extraction to detect patterns from smartphone keyboard interactions to build the predictive models presents itself as a promising approach.

The existing literature indicates that many emotion detection techniques adopted advanced techniques such as *representation learning, multi-task learning (MTL)* motivated by the success of deep learning in different domains [10]–[13]. For example, Ghosh et al. [13] applied representation learning to automatically extract the features from speech and glottal flow signals. Li et al. [10] proposed an attention pooling based representation learning mechanism to determine emotion from speech utterance. They used an end-to-end deep convolutional neural network (CNN) on the spectrogram extracted from speech utterances, thus overcoming the requirement of manual feature extraction. On the other hand, in existing literature, it is shown that the performance of emotion detection from acoustic signals improves when valence and arousal are modeled together using MTL [12]. In Emo2Vec [11], Xu et al. showed that word-level representations obtained using MTL return superior performance for different emotion related tasks (e.g. emotion analysis, stress detection) from text data. While representation learning reduces the feature engineering effort [14], MTL often returns superior performance by sharing the training knowledge among different related tasks [15], [16]. However, to the best of our knowledge, no prior work investigates the effectiveness of these learning models for emotion detection from keyboard interaction pattern on smartphone.

We, in this paper, propose an end-to-end framework to determine human emotion based on keyboard interaction pattern leveraging on the aforesaid learning algorithms. It comprises of two phases. In the *first phase*, we deploy a sequence-based encoder using Long Short-Term Memory (LSTM). It automatically learns the representation from raw keyboard interaction pattern, thus reduces the feature engineering overhead. We collate all the keyboard interactions in a typing *session*. The interaction details within a session like pressure, speed, duration, key type (deletion, special character, alphanumeric etc.) are fed as input to the framework to obtain the session-level representation. In the *second phase*, we deploy a multi-task learning (MTL) based deep neural network (DNN) model for emotion detection using the learnt representation. In MTL, learning multiple tasks together helps to share knowledge among similar tasks, thereby often yielding superior performance. In our context, emotion detection of an individual user is a separate *task*. As a result, the underlying similarity in keyboard interaction behavior of different users is leveraged by MTL to improve the emotion detection performance.

We conduct a 3-week in-the-wild study involving 24 participants using an Android based custom keyboard, capable of tracing users' keyboard interactions. Based on the text entry, a self-report probing mechanism collects four types of emotions (*happy, sad, stressed, relaxed*). We utilize the collected keyboard interactions and self-report details for model construc-

978-1-7281-3888-6/19/$31.00 ©2019 IEEE

tion and evaluation of (i) learnt representations (ii) emotion classification. Our key experimental results indicate that the representation vector obtained using the proposed model has a similarity value of 0.901 within the same emotion, in comparison to the similarity value of 0.811 between different emotions. The obtained representation helps to achieve user-wise average AUCROC of 84%, and emotion-wise average AUCROC of at least 83%.

## II. Background

In this section, we provide a brief overview of the representation learning and multi-task learning (MTL).

### A. Representation Learning

In representation learning, the focus is on learning the compact representation (often expressed as a vector) of the data so that it can be useful for building models for different prediction and recommendation problems [14]. The automatic learning of representation from the data reduces the feature engineering effort for the model development.

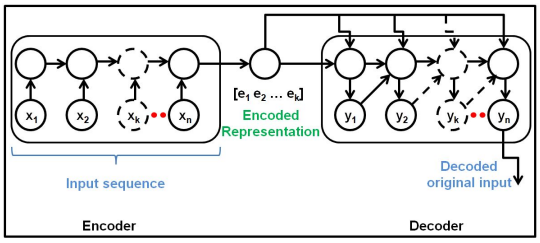



**Fig. 1:** LSTM based *encoder - decoder*. The encoder gets the input sequence and produces a representation vector. The decoder takes the representation as input and produces the output sequence. During training, the loss between actual input and decoded output is minimized and the model parameters are learned. At the end of training, the decoder is removed and the output of the encoder is used as representation.

There exist various algorithms to learn the representations [14], out of which, in the following, we briefly illustrate a LSTM based *encoder - decoder* model [17] (see Fig. 1). It consists of two LSTM blocks - (a) *encoder* takes the input sequence $X_n = [x_1, x_2, .., x_n]$ of size $n$ and produces the representation vector $E_k = [e_1\ e_2\ ..\ e_k]$ of fixed dimension $k$, while (b) the *decoder* produces the output sequence $Y_n = [y_1, y_2, .., y_n]$ from the representation vector $E_k$. The encoder learns the function $f$: $X_n \rightarrow E_k$ for generating the representation vector, while the decoder learns the function $f'$: $E_k \rightarrow Y_n$ to produce the output sequence. The loss between the input and output sequence is minimized using gradient-based algorithms [18]. Once the model is trained, the output of the LSTM encoder is used to obtain the representation.

### B. Multi-task Learning

In multi-task learning (MTL), multiple *related* tasks are learned simultaneously. Learning multiple tasks together helps to share knowledge among similar tasks, thereby often yielding superior performance [15], [16]. We show the generic architecture of a MTL based neural network in Fig. 2. The initial layers are used to transform the input vector to learn the generalized embedding, while the final layers are used to obtain task-specific embedding. The initial layers are *shared* across tasks to improve the learning by using the information contained in training samples of other related tasks. These *shared layers* allow the model to eavesdrop from one task to learn the features for another task. On the other hand, the final *task-specific layer* allows to learn task-specific embedding. This layer realizes the personalization and the specific traits of an individual task are taken into account. Subsequently, this layer ignores inputs from other tasks (as shown in different colors in the Fig. 2) by assigning them small weights, as these may not be useful. This task-specific layer utilizes the embedding obtained from the shared layers and makes it specific to individual task so that final task-specific predictions are made at the output layer.

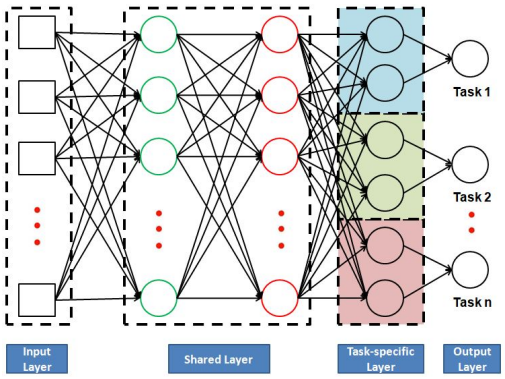



**Fig. 2:** Multi-task Learning based Neural Network (MTL-NN) model for training multiple *related* tasks. The input layer takes the input representation as a vector and passes it through the shared layers to obtain generalized embedding, whereas task-specific layer uses this embedding and make amendments to obtain output for each task.

## III. Framework Design for Emotion Detection

In this section, we explain the technique for data collection for the problem, followed by framework that uses the data to build emotion prediction model.

### A. Data Collection Process

The data collection process during keyboard interactions is shown in Fig. 3. We define a text entry *session* as the uninterrupted time duration where a user engages with an application without switching to any other application. In Fig. 3, the elapsed time between the interval `t1` to `t2` is defined as a *session*, when user performs text entry in WhatsApp [19]. The text entry starts at time instance `t1` and finishes at `t2`. Similarly, time `t3` to `t4` is also a session, when user performs text entry in Hangout [20]. Every touch instance within a session is shown with a small bar. Once the user completes text entry in a session and changes the application, she is probed to record her perceived emotion during this session. We issue the emotion self-report collection probes using Experience Sampling Method (ESM) [21]–[23]. Finally, the user provided emotion self-reports are used as the ground truth labels for the associated sessions.

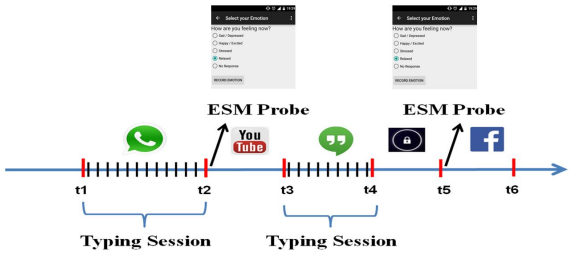



**Fig. 3:** Collecting touch interaction details and emotion self-reports based on keyboard interaction pattern. For example, time interval between `t1` and `t2` is considered a *session*, where each small bar within this *session* is a touch instance. Emotion self-report is collected via ESM probe as soon as user changes the app at `t2`. The same is associated with this *session*.

**TABLE I:** Description of different touch interaction details recorded within a session. These are used to organize a session.

| Interaction detail | Interaction description |
|---|---|
| Inter-tap duration (ITD) | Time elapsed since last touch interaction |
| Alphanumeric | One-hot representation to denote current touch interaction alphanumeric or not |
| Spl. Char | One-hot representation to denote current touch interaction special character or not |
| Backspace | One-hot representation to denote current touch interaction backspace or not |
| Touch pressure | Pressure at current touch interaction |
| Touch speed | Speed at current touch interaction |
| Touch time | Duration of current touch interaction |

Based on the touch interactions performed in a session, we organize each session as follows. For a session $s_i$, if there are $n$ touch instances, the input sequence of the session $s_i$ is expressed as $S_i = [s_i^1, s_i^2, .., s_i^n]$, where $s_i^j$ indicates different touch interaction details like pressure, speed, key type (backspace, alphanumeric) at interaction $j$ of $s_i$. Precisely, at each touch interaction, we capture the details as described in Table I; hence $s_i^j$ becomes a column vector of dimension 7.

### B. Emotion Detection Framework

The proposed framework is shown in Fig. 4. It consists of two phases - (i) we take the raw keyboard interaction data as input and learn a representation for every session, (ii) we use this representation as input to a Multi-task Learning based Neural Network to detect different emotions.

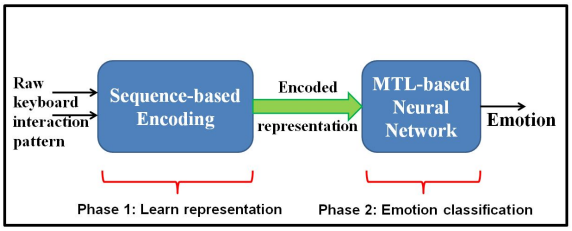



**Fig. 4:** End-to-end framework to infer emotion from raw keyboard interaction pattern. It consists of two phases, in first phase, representation is learnt from unprocessed interaction data, while in second phase, the representation is used for emotion classification.

*1) Representation Learning on Keyboard Interactions:* We obtain the representation for keyboard interaction pattern using an LSTM encoder-decoder model as shown in Fig. 1. The LSTM encoder takes the session-wise keyboard interaction sequence as input and produces a fixed length representation for every session. The keyboard interaction data in each session is organized as described earlier and fed as input to the encoder. The LSTM encoder in turn produces a representation vector of size 8 from every sequence of input. For a session $s_i$, consisting of $n$ touch interactions, the encoder learns the encoding function $f$: $(s_i^1, s_i^2, .., s_i^n) \rightarrow E_8^i$, which converts the sequence of these touch interactions to a representation vector $E_8^i$ of dimension 8. This way, for every session, a representation is obtained from the LSTM encoder. The output of the LSTM encoder is fed as the input of the next phase for emotion classification.

*2) Emotion Detection Model Generation:* We show the architecture of the proposed Multi-task Learning Neural Network (MTL-NN) for emotion detection in Fig. 2. In the MTL implementation, we use 2 shared layers and 1 task-specific layer. The choice of relatively few hidden layers, unlike image or video classification network, is driven by our observation that low-level common features for detecting emotion from keyboard interactions are few. Therefore, increasing the number of hidden layers does not help. In this MTL model, predicting emotion for every user is considered a *task*. The input layer takes the derived representation vectors as the input (see Fig. 4). The initial shared layers try to improve learning by leveraging on the commonality in text-entry behavior of different users. For example, this layer learns by sharing information between those users, who may have high typing speed in *happy* state. On other hand, the final task-specific layers add personalization to the generalized representation. This layer uses the embedding obtained from the shared layers and makes it specific to individual user. For example, this layer uses the common representation for the set of users having relatively high typing speed in *happy* state, then makes changes for specific user, who has very high touch pressure in *happy* state.

As building model for every user is a separate task, while training the model, every batch of training data is used from a specific user. This helps to predict the emotions of that user only and as a result, the errors made during prediction are backpropagated to adjust the weights in every layers (shared and task-specific). In this way, by selecting every user and continuously updating the person-specific and shared weights, the model will learn the generic representation for every user. MTL itself works as a regularization tool to avoid overfitting [24]. Additionally, we use dropout after final shared layer. Categorical cross entropy is used as the loss function and softmax is used as the activation function.

## IV. User Study

### A. Experiment Apparatus

We develop a custom QWERTY keyboard (Fig. 5) using the Android Input Method Editor (IME) facility [25]. It identifies the keyboard interaction sessions and records the timestamp, application name, any non alphanumeric character typed, pressure and speed during every touch interaction.

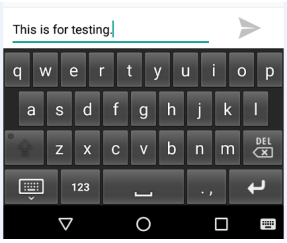



**Fig. 5:** App keyboard

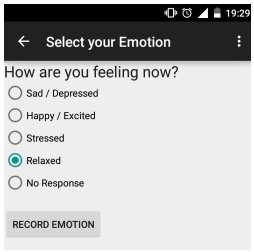



**Fig. 6:** Emotion collection UI

Once the user completes text-entry in a session, she is probed to record her perceived emotion during the session. The user is provided with the option to record any of the four emotions - *happy, sad, stressed, relaxed* (Fig. 6). These emotions are having unambiguous valence and arousal in Circumplex plane [26], hence easy to discriminate during self-reporting. The user can skip self-reporting by selecting the *No Response* option.

### B. Study Procedure

*Participants:* We recruited 30 university students (25 male, 5 female, aged between $18-35$ years) to participate in our experiments. We installed the app on their smartphones and instructed them to use it for 3 weeks. 4 participants left the study in the middle and 2 participants entered less than 40 labels during entire period. Finally, we collected data from the remaining 24 users (20 male, 4 female). The average age of the selected set of participants is 23.3 years (std dev. 4.23).

*Instructions to the Participants:* We instructed the participants to select the custom keyboard as the default one and use the same for typing. We informed the participants that once they switch from their current application after typing, they may receive a survey questionnaire as a pop-up, where they need to report their emotion state. We also advised participants to record *No Response* label if they want to skip self-reporting.

## V. Dataset

### A. Data Overview

We collected a total of $203,386$ keyboard interactions spanning across $2,476$ sessions. All the sessions tagged with *No Response* are eliminated. Similarly, we eliminate small sessions (having less than 80 interactions) as they may not provide sufficient detail for obtaining representation [4]. We summarize the final dataset in Table II.

**TABLE II:** Final dataset

| | |
|---|---|
| Total touch interactions | 203,386 |
| Total sessions | 2,486 |
| Per user sessions (mean, std dev.) | 103, 68 |
| Minimum number of session | 41 |
| Maximum number of session | 305 |

### B. Emotion Distribution

We also exhibit the frequency distribution of different emotions for each user. It is observed that all but 5 users ($U11, U14, U16, U18, U21$) have recorded four emotions (Fig. 7). We identify *relaxed* as the most commonly recorded emotion state. Overall, we record $18\%, 7\%, 24\%$ and $51\%$ sessions tagged with *happy, sad, stressed,* and *relaxed* respectively, which reveals the class imbalance across different emotions.

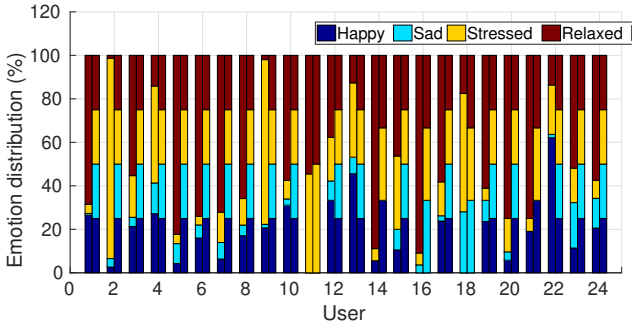



**Fig. 7:** Emotion distribution of each user. All but 5 users ($U11, U14, U16, U18, U21$) have recorded every emotion state. For every user, there are two bars, where the first bar indicates the distribution of emotion samples in original data as recorded by the participants. The corresponding second bar indicates the distribution after balancing the emotion distribution using SMOTE.

We rely on Synthetic Minority Over-sampling Technique (SMOTE) [27] to address the problem of data imbalance in emotion samples. SMOTE re-samples the class with the least number of instances so that all the classes are equally balanced. We remove class imbalance for every user by applying SMOTE on her data. On the collected data of every user, we apply SMOTE separately so that all the four classes have almost equal number of instances. We show the distribution of emotion samples for every user in the original data and in the over-sampled data using SMOTE in Fig. 7.

## VI. Evaluation

In this section, first we describe the experiment setup and then evaluate the proposed framework. The evaluation is carried out in two steps. In first step, we evaluate the quality of learnt representation and in second step, we measure the emotion classification performance using that representation.

### A. Experiment Setup

We consider the oversampled data of each user separately and split it into 60:40 ratio, where the initial $60\%$ is used to train the model and the remaining $40\%$ is used for testing. We learn the representation using training data only. To find the values of the hyperparameters, we experimented with the following batch sizes $(8, 16, 24)$, epochs $(50, 75, 100, 150)$ and dropout rates $(0.15, 0.20, 0.25, 0.30)$. We obtain the optimal values for these parameters using grid search. It is observed that for a batch size of 8, epoch of 100 and dropout of $0.20$, best classification performance is obtained. So, we have used these values for the hyperparameters.

*1) Performance Metric:* We implement cosine similarity to measure the similarity between the representation vectors obtained for different emotions, which we use to evaluate the quality of learnt representations. We measure AUCROC

(Area under the Receiver Operating Characteristic curve) as the performance metric to evaluate the emotion classification performance. We compute the weighted average of AUCROC ($auc_{wt}$) for every user as follows, $auc_{wt} = \sum_{\forall i \in \{happy,sad,stressed,relaxed\}} f_i * auc_i$, where $f_i$, $auc_i$ indicate the fraction of samples and AUCROC for emotion state $i$ respectively. We also report the mean and standard deviation of AUCROC for every emotion state as a measure to understand the emotion-wise classification performance.

*2) Baseline Models:* We compare the proposed MTL-NN emotion detection model with the following baseline models.

- *Most Represented Emotion Model (MRE):* In our dataset, we observe for most of the users *relaxed* state is dominantly present. As a result, the proposed model needs to be compared with an emotion detection model, which always predicts the mostly represented emotion state. In this approach, we build a personalized emotion prediction model, which always predicts the most frequent emotion.
- *Single-Task Learning Model (STL):* This is a personalized DNN model. While building this model for a user, we do not use data from any other user. We compare the proposed model with this *Single-Task Learning* model to understand how accurately we can determine the user emotion based on personal keyboard interaction details only. We use the same representation to build the model.
- *Feature-driven Emotion Model (FTR):* We develop this personalized model by extracting a set of hand-crafted features from every text-entry session. We use the same set of features as defined in [28]. These are primarily derived from typing speed, error rate, special characters used in a session, session length and session duration. Comparison with this model brings the efficacy of automatic feature learning over manual feature extraction.
- *Combined Emotion Model (COM):* This is also a personalized DNN model developed using MTL. We train this model by combining both representation (obtained from the encoder) and the hand-crafted features (used in the *FTR* model). Comparison with this baseline helps to understand whether combining automatically learnt representations with hand-crafted features boosts emotion prediction performance.

For all the baselines, we split the data in 60:40 ratio, where initial $60\%$ is used for training and remaining $40\%$ for testing. We use same network configuration like MTL-NN in *STL, FTR* and *COM* models.

### B. Quality of Keyboard Interaction Representation

We evaluate the quality of the automatically learnt representation using the cosine similarity between the vectors obtained for different emotion pairs. Ideally, the vectors obtained for similar emotions should have higher similarity than that of the different emotions.

For every user and every emotion-pairs, we compute the average cosine similarity. We compute the mean of these similarity values and show them in Table III. It is observed that for any emotion, the intra-emotion representation vector similarity is higher than that of the inter-emotion similarity.

**TABLE III:** Average cosine similarity of the representation vectors obtained for every pair of emotions reveals that the representations for the similar emotions are more alike than the different ones.

| | **Happy** | **Sad** | **Stressed** | **Relaxed** |
|---|---|---|---|---|
| **Happy** | **0.829** | 0.739 | 0.801 | 0.804 |
| **Sad** | 0.739 | **0.843** | 0.810 | 0.806 |
| **Stressed** | 0.801 | 0.810 | **0.926** | 0.911 |
| **Relaxed** | 0.804 | 0.806 | 0.911 | **0.918** |

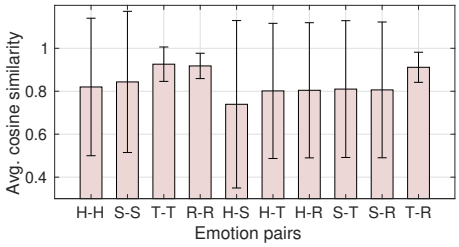


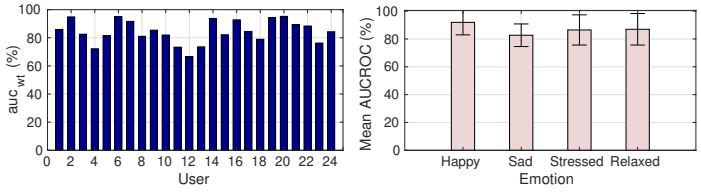

**Fig. 8:** Comparison of average cosine similarity across different emotion pairs reveals that all different emotion pairs (except *T-R*) are having less similarity than that of the similar emotion pairs. *H, S, T, R* indicate *happy, sad, stressed, relaxed* respectively. Error bar indicates std dev.

We also compare the average cosine similarity across every emotion pairs in Fig. 8. It reveals that all the different emotion pairs (except *stressed-relaxed*) are having less similarity than that of the similar emotion pairs. It also points that the different emotion pairs are having very high standard deviation, thus indicating that for users there are variations in the similarity values. So, for different emotion pairs, these similarity values could have been even lower unless influenced by few users' high similarity. On other hand, for three of the users, the *happy-happy* and *sad-sad* pairs are having very low similarity value, causing the average similarity to go down for similar emotion pairs. Otherwise, these values would have been higher than all the different emotion pairs. In summary, the representations obtained for different emotion-pairs are less alike than that of the similar emotion-pairs.

### C. Performance of Emotion Detection Model

*1) Emotion Classification Performance:* The emotion classification performance is shown in Fig. 9. We show the user-wise classification accuracy ($auc_{wt}$) in Fig. 9a. We observe, for $75\%$ of the participants, the value of $auc_{wt}$ is greater than 80% and we obtain an average AUCROC of $84\%$ (std dev. $8\%$). The $95\%$ confidence interval for the values of $auc_{wt}$ is $[81.1\%, 87.7\%]$. The emotion-wise classification performance (AUCROC) is shown in Fig. 9b. All the emotions are identified with an average AUCROC of more than $83\%$, while *happy* state is identified with highest average AUCROC ($92\%$).

*2) Comparison with Baselines:* We compare the performance of the proposed MTL-NN model with the baselines in Fig. 10. We observe that the proposed MTL-NN model outperforms two baselines (*MRE, STL*). The *MRE* model achieves an average $auc_{wt}$ of $75\%$ (std dev. $12\%$), while the *STL* model




**(a)** User-wise AUCROC **(b)** State-wise mean AUCROC

**Fig. 9:** Emotion classification performance of the proposed MTL-NN model. Error bar indicates std dev.

achieves an average $auc_{wt}$ of 74% (std dev. 11%). We also observe that mean $auc_{wt}$ for proposed model is significantly ($p < 0.05$) higher than that of the MRE model using t-test. We have similar observation for *STL* model. These observations indicate that always predicting the most represented emotion as the outcome is not a good choice. Similarly, relying only on personal data is not effective either. This primarily happens because lack of enough training samples to train a neural network. On the contrary, when MTL is used to train the model using all users data, it can leverage on the similarity in the text-entry pattern across users and adjust the weights in the shared layers, thus producing the superior performance.

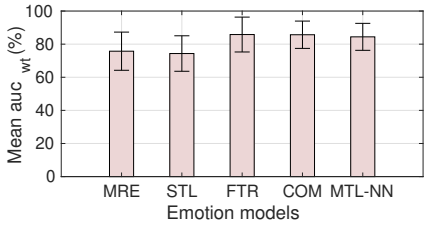



**Fig. 10:** Comparison with baselines reveals that the proposed MTL-NN model outperforms *MRE, STL* models and produces comparable performance like *FTR, COM* models. Error bar indicates std dev.

However, we also observe that the proposed model exhibits comparable performance with *FTR* and *COM* models. We do not observe significant ($p < 0.05$) difference in average $auc_{wt}$ for these two models with respect to the proposed model using t-test. These reveal two important findings - (1) the proposed method of automatically learning the representation is comparable with manually designed feature extraction and (2) combining both representation and handcraft features together does not return substantial improvement in emotion classification. This happens because both features and representations provide similar information. As a result, by using the proposed end-to-end framework, we can reduce the effort of manual feature extraction and yet obtain similar classification performance like feature-driven models.

*3) Impact of DNN Layers:* We show the impact of the number of shared layers on emotion classification performance in Fig. 11. We vary the number of layers and measure the mean $auc_{wt}$. It is observed that superior performance is obtained if the number of shared layers are 2. However, the performance starts to deteriorate if number of layers are increased beyond 4 as the model starts to overfit. So, we use the number of shared layers as 2 in the proposed MTL-NN model.

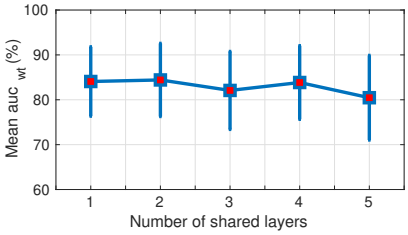



**Fig. 11:** Variation in mean $auc_{wt}$ with varying number of shared layers. Error bar indicates std dev.

## VII. Discussion

This work demonstrates that feature engineering can be replaced by automatic representation learning with comparable emotion recognition accuracy from smartphone keyboard interaction. The other important takeaway is that the use of MTL can reduce the data collection effort from each user. MTL is especially effective in this case where groups of users exhibit similarity in their keyboard interaction pattern.

We note two factors which may play a role in determining the quality of the results. First, the length of the sessions were not very long, thereby the assumption that a single emotion label represents the entire session works out to be true. For longer sessions, a single label may not represent transitions in emotion during a session. Secondly, the benefit of MTL stems from the fact that there is similarity in behavior across users. For groups with more heterogeneity, this could be challenging. Perhaps increasing the group sizes could help in the solution in such a scenario.

## VIII. Conclusion

The use of representation learning in extracting complex patterns has been beneficial in emotion prediction from speech, and facial expressions. In this work, we show that representation learning can replace the efforts of feature engineering for emotion detection based on keyboard interaction pattern on smartphone by proposing an end-to-end framework. We used the raw keyboard interaction details like speed, duration, character type (special character, alphanumeric etc.) as input and automatically learned a condensed representation using an LSTM-based encoder. This representation is used subsequently to train a multi-task learning (MTL) based neural network model for emotion classification. The use of MTL reduces the burden of data collection by leveraging similarity across users. We validate the proposed framework with 24 participants in a 3-week in-the-wild study, during which we collect four types of emotion self-reports (*happy, sad, stressed, relaxed*) along with keyboard interaction pattern. We have shown how to learn a representation from raw keyboard interaction data. The representation is effective in determining four emotions with an average accuracy (AUCROC) of 84%.

## Acknowledgement

This research was supported by TCS under the project titled "Behavior Modeling In Multi-sensor Environments - Integrating Environment Sensing, Human Sensing & Social Sensing for Rich Insights" (VN/BK/18-19/AUG/65).

## REFERENCES


[1] A. Mottelson and K. Hornbæk, "An affect detection technique using mobile commodity sensors in the wild," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2016, pp. 781–792.

[2] S. Ghosh, N. Ganguly, B. Mitra, and P. De, "Evaluating effectiveness of smartphone typing as an indicator of user emotion," in *2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*. IEEE, 2017, pp. 146–151.

[3] H. Lee, Y. S. Choi, S. Lee, and I. Park, "Towards unobtrusive emotion recognition for affective social communication," in *Proceedings of IEEE CCNC*, 2012.

[4] S. Ghosh, N. Ganguly, B. Mitra, and P. De, "Tapsense: Combining self-report patterns and typing characteristics for smartphone based emotion detection," in *Proceedings of the ACM MobileHCI*, 2017.

[5] M. Ciman and K. Wac, "Individuals' stress assessment using human-smartphone interaction analysis," *IEEE Transactions on Affective Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[6] H.-J. Kim and Y. S. Choi, "Exploring emotional preference for smartphone applications," in *Proceedings of the IEEE CCNC*, 2012.

[7] R. Bixler and S. D'Mello, "Detecting boredom and engagement during writing with keystroke analysis, task appraisals, and stable traits," in *Proceedings of the ACM IUI*, 2013.

[8] K. Wac, M. Ciman, and O. Gaggi, "isensestress: Assessing stress through human-smartphone interaction analysis," in *Proceedings of the PervasiveHealth*, 2015.

[9] S. Ghosh, K. Hiware, N. Ganguly, B. Mitra, and P. De, "Emotion detection from touch interactions during text entry on smartphones," *International Journal of Human-Computer Studies*, vol. 130, pp. 47 – 57, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1071581918304889

[10] P. Li, Y. Song, I. McLoughlin, W. Guo, and L. Dai, "An attention pooling based representation learning method for speech emotion recognition," *Proc. Interspeech 2018*, pp. 3087–3091, 2018.

[11] P. Xu, A. Madotto, C.-S. Wu, J. H. Park, and P. Fung, "Emo2vec: Learning generalized emotion representation by multi-task training," in *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, 2018, pp. 292–298.

[12] R. Xia and Y. Liu, "A multi-task learning framework for emotion recognition using 2d continuous space," *IEEE Transactions on Affective Computing*, vol. 8, no. 1, pp. 3–14, 2017.

[13] S. Ghosh, E. Laksana, L.-P. Morency, and S. Scherer, "Representation learning for speech emotion recognition." in *Interspeech*, 2016, pp. 3603–3607.

[14] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[15] M. L. Hutchinson, E. Antono, B. M. Gibbons, S. Paradiso, J. Ling, and B. Meredig, "Overcoming data scarcity with transfer learning," *arXiv preprint arXiv:1711.05099*, 2017.

[16] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[17] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[18] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[19] "Whatsapp," 2019, https://www.whatsapp.com/.

[20] "Google hangout," 2019, https://hangouts.google.com/.

[21] R. Larson and M. Csikszentmihalyi, "The experience sampling method," in *Flow and the foundations of positive psychology*. Springer, 2014, pp. 21–34.

[22] N. V. Berkel, D. Ferreira, and V. Kostakos, "The experience sampling method on mobile devices," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, p. 93, 2017.

[23] S. Ghosh, N. Ganguly, B. Mitra, and P. De, "Designing an experience sampling method for smartphone based emotion detection," *IEEE Transactions on Affective Computing*, pp. 1–1, 2019.

[24] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich, "To transfer or not to transfer," in *NIPS 2005 workshop on transfer learning*, vol. 898, 2005, pp. 1–4.

[25] 2016, http://developer.android.com/guide/topics/text/creating-input-method.html.

[26] J. A. Russell, "A circumplex model of affect," *Journal of Personality and Social Psychology*, vol. 39, no. 6, pp. 1161–1178, 1980.

[27] 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.

[28] S. Ghosh, N. Ganguly, B. Mitra, and P. De, "Effectiveness of deep neural network model in typing-based emotion detection on smartphones," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom)*. ACM, 2018.