Few-Short Learning for Detecting Affective States from Keyboard and Mouse Data

Ramu Gautam*, Beiyu Lin†, Mei Yang*
*Department of Electrical and Computer Engineering, University of Nevada, Las Vegas, Las Vegas, USA
†School of Computer Science, University of Oklahoma, Norman, USA
Emails: ramu.gautam@unlv.edu, beiyu.lin@ou.edu, mei.yang@unlv.edu

***Abstract*—Detecting human emotional states is crucial for improving employee well-being and productivity. Traditional methods for emotion recognition often require obtrusive sensors or raise privacy concerns. Unobstructive approaches like using keyboard and mouse data to measure affective states have been shown to be accurate. However, due to the inherent class imbalance in human affect data, the methods developed for affective state detection struggle in identifying less-common affective states. We investigate few-shot learning techniques to address this issue. Specifically, we evaluate the performance of prototypical networks with various neural network backbones on a time-series emotion recognition task with limited labeled data. The results demonstrate that the prototypical network with an RNN backbone achieves the best classification performance among the few-shot learning models evaluated. Notably, one-shot trained prototypical networks outperform traditional supervised machine learning approaches across all metrics, highlighting their ability to learn discriminative class representations from minimal labeled samples, even with class imbalance. By enabling accurate emotion detection from readily available keyboard and mouse inputs without specialized hardware or privacy trade-offs, the proposed few-shot learning approach shows promise for developing unobtrusive affect-aware tools to enhance employee productivity and well-being in diverse work environments.**

***Index Terms*—few-shot, one-shot, emotion, affect, keyboard, mouse, PAM**

I. Introduction

Around 49% of people working with a computer for more than 4 hours a day and 30% of those working with a computer for 2-4 hours a day report stress due to computer use [1]. Affects such as stress and fatigue have been shown to cause attention loss, slow reaction, and tiredness leading to diminished productivity and burnout [2]. Decrease in productivity has been observed in subjects during a negative emotional state. The study in [3] found that 70% of users showed decreased typing speed in negative emotional state and 83% of users showed increase in typing speed in positive emotional state compared to the neutral state. With the increase in popularity of work from home, the blurring of lines between professional and personal lives in work-from-home may cause new forms of stress and other emotional afflictions [4]. Accurate reading of human affective states could help make decisions that improve employee well-being and productivity. Developing tools to detect emotional states and enabling proper emotional well-being strategies in workplace is thus of vital importance [5].

Different approaches have been studied for detecting human emotional states. Many of them require sensors directly attached to the user [6] [7]. These methods require specialized hardware, which can be expensive and not feasible due to budget constraints. Other methods require using external sensors such as microphones or webcams [8] [9] [10]. In both of these methods, the users are aware of being monitored, which in itself can cause changes in the emotional state. Attaching physical sensors to the subjects continuously throughout the day may be considered obtrusive or invasive. Furthermore, both of these methods raise privacy concerns. This creates a need for an unobtrusive and non-invasive way of measuring affects.

One unobtrusive way of detecting affects is measuring human emotional states from keyboard and mouse usage data. This approach does not require any additional hardware and is suitable for both traditional office environments and work-from-home scenarios. In addition, it can also be easily integrated into the work environment at scale. Various machine learning methods have been applied to identify emotional states based on keyboard and mouse usage data and shown to be effective [3] [7] [11]. However, these methods still have some limitations. Human affect states (e.g., very low, low, neutral, high, and very high stress) do not have a balanced distribution. Some states are represented more frequently than others [12]. The imbalanced labeled data may result in the tools developed for detecting human affective states underperform in detecting less common states.

In this paper, we explore few-shot learning techniques and various machine learning methods to address this issue. The remainder of the paper is structured as follows: Section II reviews relevant literature. Section III describes the methodology, including details on the dataset, problem formulation as a few-shot learning task, and the prototypical network architecture with different neural network backbones. Section IV presents the experimental setup, results, and analysis comparing the classification performance of the few-shot models against traditional supervised learning approaches. Finally, Section V concludes with a summary of findings and potential future directions.

II. Literature Review

Keyboard and mouse usage data have been used by many studies to measure users' affective states. In [13], a behavioral method is developed to investigate the influence of induced film clips on users' motor-behavioral parameters while completing a computer task. It measures five affective states: Low Valence Low Arousal, High Valence Low Arousal, Neutral, Low Valence High Arousal, and High Valence High Arousal. Keystroke time (time between the press and release of a key) and flight time (time between a key release and next key press) have been proposed to detect the user's emotional states [3] [14]. Various classifiers like Logistic Regression, Multilayer Perceptron (MLP), and Random Tree are used to detect affects. In [11], Multiple Instance Learning (MIL) is applied to Random Forest (RF) classification to distinguish three stress levels (Low, Medium, High). They collect data using a purpose-built application structured to evoke stress by requiring each participant to complete eight computer tasks. A similar method is used in [3] to recognize three emotion categories (neutral, positive and negative) from keyboard stroke patterns. In [15], a method to classify emotional states based on keystroke pattern and the type of texts typed by the user is proposed. Keyboard and mouse usage data has been used in several other studies to detect stress [16] [17], drowsiness [18] and fatigue [19] [20].

Information such as sensor readings, human actions and keyboard and mouse activity is present as sequential data. The order of the information is a key characteristic of sequential data. Sequential data classification tasks are common in real-life situations such as stock trading, trend prediction, anomaly detection, user activity recognition, action recognition, and real-time physical and/or mental health tracking. Because of this, there has been a significant interest in sequential data classification research.

In [21], a Support Vector Machine (SVM) classifier is used for the classification of heartbeat data into different categories using the statistical features extracted from the heart rate variability (HRV) of the signals. A K-nearest-neighbor (KNN) based time-series classification technique for classifying time-series data is proposed in [22]. In [**?**], they propose approximating the nearest neighbor(s) with the help of clustering as a preprocessing step to mitigate the problem of kNN high computational cost. Deep learning techniques, particularly Convolutional Neural Networks (CNNs), have been extensively explored for classification of sequential data [23] [24] [25]. Recurrent neural networks (RNNs) are a class of artificial neural networks well-suited for processing sequential data, such as text, speech, and time series. By incorporating a feedback loop to maintain an internal state, RNNs can capture dependencies between elements in a sequence, enabling them to effectively model contextual outputs. RNNs have been widely adopted for time series classification over the years [26] [27] [28]. Long Short-Term Memory (LSTM) networks are a type of RNN designed to be able to learn long-term dependencies in sequential inputs, making them effective at tasks like sequential data classification and time series prediction [29] [30] [31].

Despite the extended work in sequential data classification, obtaining labeled data can be expensive, time-consuming, or even infeasible. Few-shot learning aims to develop models that can effectively learn and generalize from a limited number of labeled samples. Prototypical networks [32] are a common choice for few-shot classification of sequential data [33] [34] [35].

III. Methodology

This section first explains the dataset and data preprocessing steps. It then formally defines the problem as a few-shot learning task. The subsequent subsection describes the few-short learning technique and the prototypical network.

A. Data Preprocessing

In this study, we use a publicly available dataset, Stress Detection by Keystroke, App & Mouse Changes [36], that has two parts: 1) a sequential dataset captured activities of a keyboard and computer mouse when two participants are using computers, 2) users' affect metrics.

For the sequential dataset, we list a sample of the dataset in Table I that includes timestamps (denoted as "Time"), both keyboard and mouse activities (denoted as "Event_Type"), cursor positions in X and Y coordinates (denoted as "X" and "Y"), and parts of the day when those activities happened (denoted as "Daylight"). The keyboard and mouse activities (denoted as "Event_Type") include a total of 6 mouse activities: Move, Left_Pressed, Left_Released, Right_Pressed, Right_Released, and Scroll. As shown in Table II, the keystroke data captures the press time and release time of the key activated by the user. The key activated by the user is also recorded. The mouse speed data calculated from the raw mouse activity data is also available in the dataset. Mouse inactivity duration and keyboard inactivity duration derived from the mouse activity data and keystroke data are also available in the dataset as the user inactivity data. Table III and IV show mouse speed and user inactivity, respectively. As shown in Table V, the application usage data records the users' active application and the number of background applications open at the moment, captured every several seconds or whenever the active application changes.

The user affect dataset includes the users' photographic affect meter (PAM) values among other user affect metrics measured at 5-30 minute intervals. Other metrics include Fatigue (denoted as Fatigue_Val), Stress (denoted as Stress_Val), Energy (denoted Energy_Val), and Pleasantness (denoted Pleasant_Val). Table VI shows a sample of the user affect condition dataset. PAM is a tool for measuring affect (emotional state) in which users select from a grid of photos the one that best represents their current mood [37]. User's PAM value can take values 1 to 16 arranged in a 4×4 grid as shown in Fig. 1. The PAM scores of 1-16 map to PANAS Positive Affect [38] scale. The PAM value 1 represents the Negative Valence Low Arousal peak and the PAM value 16 represents the Positive Valence, High Arousal peak. The PAM grid is divided into four valence/arousal quadrants with quadrant scores of 1 (NVLA: Negative valence/Low Arousal), 2 (NVHA: Negative Valence/High Arousal), 3 (PVLA: Positive Valence/Low Arousal) and 4 (PVHA: Positive Valence/High

Arousal) to indicate the subject's affect [37]. As highlighted by four colors in Fig. 1, the four quadrants include PAM values 1-4, 5-8, 9-12 and 13-16, respectively.

From the keystroke, mouse and application usage data, a range of derived metrics are calculated for non-overlapping 30-minute intervals, as shown in Table VII. These include the inactivity time, total keystroke count, average mouse speed, and the most frequently used application. Only mouse inactivity time is used to calculate the total inactivity time. The "PAM_Val" column includes the PAM quadrant score measured during each interval. We define a *trajectory* as a 4-hour window that includes 8 adjoining 30-minute intervals. Table VII presents an example of such a trajectory, showcasing the array of features extracted from the raw data.

For each trajectory in our dataset, the PAM value of the final interval is assigned as its label. The combined 62 hours of data from two users is transformed into 62 trajectories, out of which 2 (3.2%) of trajectories have label 4 (PVHA) while 41 (66.1%) of trajectories have label 2 (NVHA). 10 (16.2%), and 9 (14.5%) trajectories have labels 1 (NVLA) and 3 (PVLA), respectively.

Before splitting the dataset into training and testing sets, data augmentation is performed for Class 4. Two trajectories were added to the existing set of trajectories. After adding those two examples, our final dataset consists of 64 trajectories. To ensure that each class is represented at least twice in the training set and the test set, stratified sampling is used so that both the training and test sets get 50% of total trajectories.

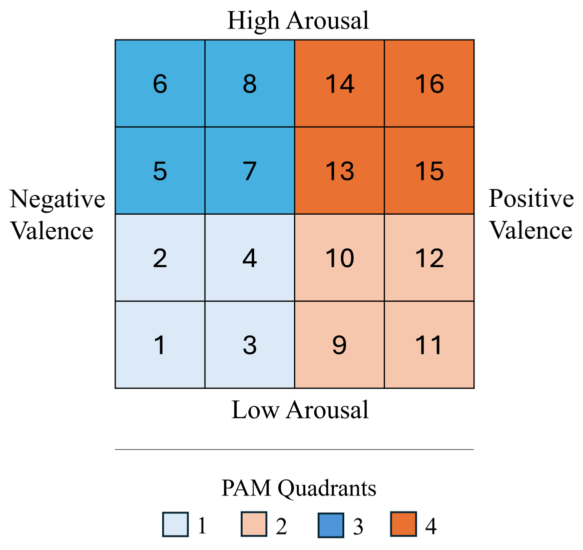

Fig. 1: Photographic Affect Meter (PAM) Grid

B. Problem Definition

In this study, we aim to leverage the keyboard and mouse activities from users to identify their emotional status via PAM. Specifically, users' activity data from 4-hour windows will be used to predict their PAM quadrant score. Since the dataset has a limited number of labeled data and an imbalanced class distribution, this study approaches the problem as a few-shot classification task.

TABLE I: Mouse activity data

Time	Event_Type	X	Y	Daylight
2021-09-10 11:59:42.515770	Move	518	381	Afternoon
2021-09-10 11:59:42.523750	Move	511	388	Afternoon
2021-09-10 11:59:42.531727	Move	509	393	Afternoon
2021-09-10 11:59:42.539705	Move	505	397	Afternoon
⋮	⋮	⋮	⋮	⋮
2021-09-10 11:59:44.892417	Left_Pressed	720	505	Afternoon

TABLE II: Keyboard activity data

Key	Press_Time	Release_Time
$	2021-09-10 12:05:31.488128	2021-09-10 12:05:31.567916
$	2021-09-10 12:05:31.810266	2021-09-10 12:05:31.910000
enter	2021-09-10 12:05:32.074560	2021-09-10 12:05:32.208202
arrow_key	2021-09-10 12:06:59.935683	2021-09-10 12:07:00.112209
arrow_key	2021-09-10 12:07:00.618858	2021-09-10 12:07:00.801367

C. Few-Shot Learning (FSL)

Unlike traditional supervised learning, which requires a large volume of labeled data to train robust models, few-shot learning techniques focus on leveraging prior knowledge to learn from a small number of samples. By framing our classification task as a few-shot learning problem, we can leverage approaches that are specifically designed to handle small and imbalanced datasets.

1) Episodic Training in FSL: Few-shot learning algorithms are trained using an episodic training procedure where each episode consists of a support set $\mathcal{S}$ with M labeled samples from a subset of N classes in the base set. The model is trained to classify another set of samples (called the query set $\mathcal{Q}$) from the same set of classes in the base set. Such a training episode is referred to as an N-way, M-shot training episode. Fig. 2 illustrates a training episode in a 3-way 2-shot setting. As shown in the figure, in 3-way 2-shot training procedure, a support set $\mathcal{S}$ in each episode is created by selecting exactly two (M = 2) random samples from three randomly selected classes (N = 3) in the base set. Query set $\mathcal{Q}$) is created by selecting one random sample from the same three classes. One-shot learning is a specific type of few-shot learning where

TABLE III: Mouse speed data

Time	Speed (ms)	Daylight
2021-09-10 11:59:43.521084	0.9937275914428129	Afternoon
2021-09-10 11:59:45.192614	0.9937285789383203	Afternoon
2021-09-10 11:59:46.255772	0.9976843745666308	Afternoon
2021-09-10 11:59:47.699912	1.9953667583870254	Afternoon
2021-09-10 11:59:48.708217	0.9986777506581286	Afternoon

TABLE IV: User inactivity data

Type	Stopped_Time	Duration (s)	Daylight
Keyboard	2021-09-10 11:59:39.834937	351.647207	Afternoon
Mouse	2021-09-10 12:05:44.520289	7.195763	Afternoon
Mouse	2021-09-10 12:06:14.316669	6.46169	Afternoon
Keyboard	2021-09-10 12:05:32.202218	87.727479	Afternoon
Mouse	2021-09-10 12:06:59.080968	17.090314	Afternoon

TABLE V: Active windows data

Time	App_Name	BG_App_Cnt
2021-09-10 11:59:39.449966	TextLogger.exe	8
2021-09-10 11:59:46.178977	TextLogger.exe	8
2021-09-10 11:59:59.178226	TextLogger.exe	8
2021-09-10 12:00:00.847763	explorer.exe	8
2021-09-10 12:00:02.272954	explorer.exe	8

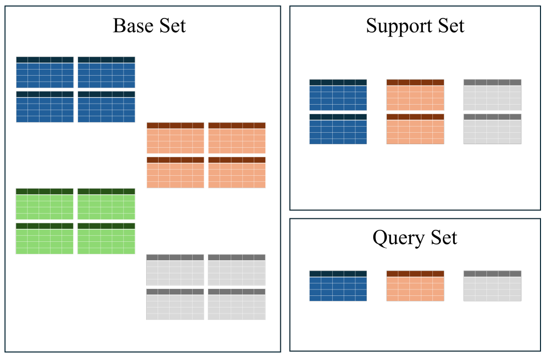

Fig. 2: Illustration of a 3-way 2-shot task in a few-shot training episode.

the model is trained and evaluated on tasks with only a single sample per class in the support set.

Prototypical networks [32] are a popular approach for few-shot learning. The prototypical networks learn a discriminative embedding space where samples from the same class are closer to each other than to samples from other classes. The basic idea behind a prototypical network is shown in Fig. 3. Transformation of input samples into the embedding space is performed by an embedding function (also referred to as a backbone or a feature extractor). The backbone is typically a pre-trained neural network model that has been trained on a large dataset for a related task, for example, a CNN model trained on ImageNet is used in few-shot image classification tasks. Using a pre-trained backbone can help in learning more effective representations from the limited labeled samples.

In few-shot learning, a class prototype refers to a representation that characterizes or summarizes the class. These prototypes are typically computed from the limited labeled samples available during the few-shot training phase, often by calculating the mean or centroid of the feature vectors of the few samples from each class. At each training episode the prototype of class k, $\vec{c}_k$, is calculated as the mean of the embeddings of M samples of class k in the support set, i.e.,

$$\vec{c}_k = \frac{1}{|\mathcal{S}_k|} \sum_{(x_i,y_i)\in\mathcal{S}_k} f_\theta(x_i) \tag{1}$$

where $\mathcal{S}_k$ represents all samples from class k in the support set, (x_i, y_i) is one sample in $\mathcal{S}_k$ and f_θ is the embedding function parameterized by θ.

To classify query samples, the Euclidean distance is computed between their embeddings and the class prototypes derived from the support. Each query sample is then assigned the label of the class whose prototype lies closest to it in the embedding space. Prototypical networks produce the distribution over classes for a query set sample based on the distances between the query embedding and class prototypes. The softmax function transforms the negative Euclidean distances into a probability distribution over the classes, allowing the network to make a prediction by selecting the class with the highest probability. Specifically, the probability of the query sample x belonging to class k is computed as:

$$p_\theta(y = k|x) = \frac{exp(-d(f_\theta(x), c_k))}{\sum_{k'} exp(-d(f_\theta(x), c_{k'}))}, \tag{2}$$

where y is the predicted class of the query sample x, $f_\theta(x)$ is the embedding of the query sample x produced by the embedding function f_θ, and c_k is the prototype for class k. The model is trained by minimizing the negative log-probability of the query samples' true class labels shown in eq. (3).

$$J(\phi) = -\log p_\phi(y = k|x) \tag{3}$$

RNNs, LSTMs, and 1-D convolutional networks are well-suited for processing sequential data, such as the trajectories in our dataset. RNN, LSTM, and CNN are thus common choices as a prototypical network backbone.

IV. Experiments and Results

Various ML models are trained using traditional supervised learning procedures. This is followed by one-shot classification with prototypical networks with various backbones. All models are evaluated in terms of accuracy, precision, recall, and F1 scores.

A. Experiment Setup

1) Traditional Supervised Learning: We start with traditional machine learning algorithms, specifically Logistic Regression, Naive Bayes, SVM, Naive Bayes, kNN, and RF. Following the traditional machine learning algorithms, classification is performed with RNN, CNN, LSTM, and ResNet. The models are trained on the training set and evaluated on the test set. The RNN model consists of a recurrent layer followed by a fully connected layer. The RNN layer has 128 hidden units and 2 stacked layers. This choice of relatively small network size works well in scenarios with limited labeled samples, capturing relevant information from the sequential data while avoiding overfitting. The CNN model consists of two convolutional layers with a ReLU activation. The first layer has 32 output channels and a kernel size of 3, with padding. The second convolutional layer has 64 output channels and a kernel size of 2, also with padding. After the convolutional layers, a max-pooling layer with a kernel size of 2 is applied, and the feature maps are flattened and passed through a dropout layer with a rate of 0.2. This convolutional unit is followed by a fully connected layer. The LSTM model consists of 2 LSTM layers with 128 hidden units followed by fully connected layers.

TABLE VI: User condition data

Time	Fatigue_Val	PAM_Val	Stress_Val	Energy_Val	Pleasant_Val	Daylight
2021-09-10 12:03:49.599397	Below_Avg	14	Neutral	Neutral	Neutral	Afternoon
2021-09-10 12:05:18.375074	Avg	3	Neutral	Neutral	S_Unpleasant	Afternoon
2021-09-10 12:56:42.248174	Below_Avg	4	S_Stressed	S_Low_Energy	Neutral	Afternoon
2021-09-10 13:27:00.550791	Low	2	Neutral	S_Low_Energy	Neutral	Afternoon
2021-09-10 13:57:10.866776	Low	7	V_Stressed	V_Low_Energy	V_Unpleasant	Afternoon

TABLE VII: A trajectory

Time	App_Name	Inactivity (s)	Keystorke_count	Speed	PAM_Val
2021-09-13 14:00:00	opera.exe	1210.531504	215.0	6.644607	2
2021-09-13 14:30:00	opera.exe	1021.323029	358.0	6.498835	1
2021-09-14 10:30:00	opera.exe	1002.831345	19.0	14.601119	1
2021-09-14 11:00:00	Skype.exe	1293.995927	829.0	11.215676	3
2021-09-14 11:30:00	opera.exe	714.970757	1078.0	10.216914	4
2021-09-14 12:00:00	opera.exe	552.622641	256.0	9.928773	1
2021-09-14 12:30:00	opera.exe	1055.705093	140.0	13.519175	1
2021-09-14 13:00:00	opera.exe	777.054938	175.0	7.069363	**2**

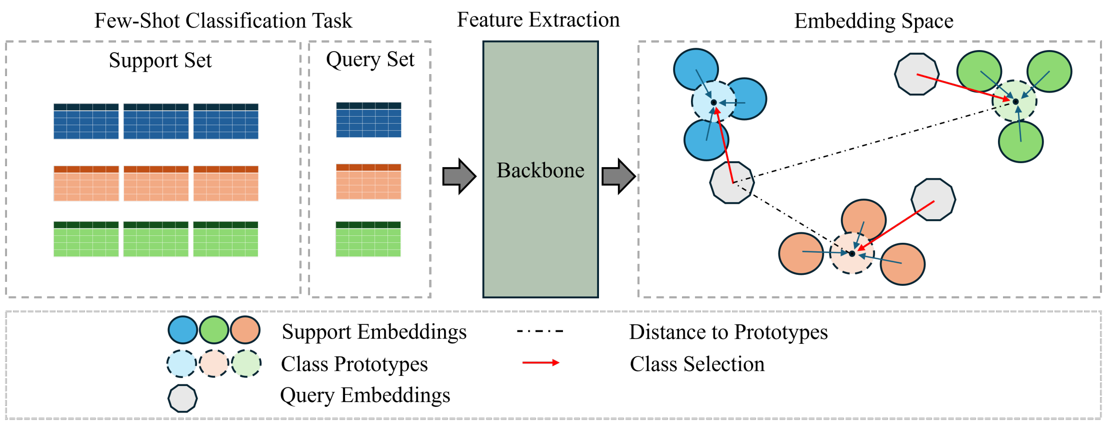

Fig. 3: Prototypical Network.

2) Few-shot Learning with Prototypical Network: The pre-trained RNN, LSTM, and CNN models from above are used as the backbones of the prototypical networks after removing the fully-connected heads. For our few-shot classification problem, we adopt a 4-way one-shot episodic training procedure and a 2-way one-shot episodic training procedure. One-shot training simulates limited labeled data availability and the model is forced to learn from one labeled sample per class in the support set. After training, the prototypical network models are evaluated on the test set. Similar to the episodic training procedure, testing also follows a 4-way one-shot approach. The embeddings of the support set samples serve as the class prototypes and query samples are classified to the closest prototype class. The models are evaluated on the test set over 50 episodes.

B. Results

Table VIII and Fig. 4 show the one-shot classification performance of our prototypical network with various backbones in terms of accuracy, Precision, Recall, and F1 scores. The table also includes the classification performance of various machine learning models trained with traditional supervised learning. The results show a mixed performance across the different models and few-shot learning settings. Among the 4-way one-shot trained models, the prototypical network with an RNN backbone has achieved the highest accuracy, precision, recall, and F1 scores at 0.68, 0.68, 0.68, and 0.67, respectively. This indicates the RNN is able to effectively learn discriminative features from the time-series data. The prototypical network with a CNN backbone shows slightly lower performance in the 4-way one-shot task, with scores of 0.58, 0.58, 0.58, and 0.57 for accuracy, precision, recall, and F1 scores. The LSTM model lags behind the RNN and CNN in the 4-way one-shot setting, with the test scores of 0.54, 0.54, 0.53, and 0.51, respectively. Using Linear Discriminant Analysis (LDA) as a feature extractor, the test score of 0.49 is achieved for accuracy, precision, recall, and F1 scores, respectively.

In the 2-way one-shot training setting, the prototypical network with an RNN backbone again exhibits the best performance with scores of 0.58, 0.56, 0.58, and 0.56 for accuracy, precision, recall, and F1 on the test set. The prototypical network with the CNN backbone shows slightly lower test scores of 0.55,0.54, 0.55, and 0.52, respectively. Similar to the 4-way, 1-shot setting, the 2-way one-shot trained prototypical network with an LSTM backbone does not perform well compared to RNN and CNN backbones with test scores of 0.49, 0.46, 0.49 and 0.46, respectively. The model with an

TABLE VIII: Classification performance

Model	Accuracy 4-way	Accuracy 2-way	Precision 4-way	Precision 2-way	Recall 4-way	Recall 2-way	F1 4-way	F1 2-way
Proto. Net. (RNN)	**0.68**	**0.61**	**0.68**	**0.61**	**0.68**	**0.60**	**0.67**	**0.60**
Proto. Net. (CNN)	0.58	0.55	0.58	0.54	0.58	0.55	0.57	0.52
Proto. Net. (LSTM)	0.53	0.48	0.53	0.48	0.51	0.47	0.52	0.48
Proto. Net. (LDA)	0.49	0.46	0.49	0.46	0.49	0.46	0.49	0.45
RNN	0.68		0.22		0.30		0.25	
CNN	0.44		0.57		0.36		0.30	
LSTM	0.59		0.35		0.44		0.39	
Random Forest	0.59		0.32		0.41		0.31	
kNN	0.50		0.15		0.25		0.17	
SVM	0.50		0.30		0.36		0.32	
Logistic Regression	0.56		0.26		0.33		0.29	
Naive Bayes	0.47		0.26		0.34		0.29	
MLP	0.56		0.17		0.23		0.16	
ResNet	0.59		0.46		0.40		0.35	

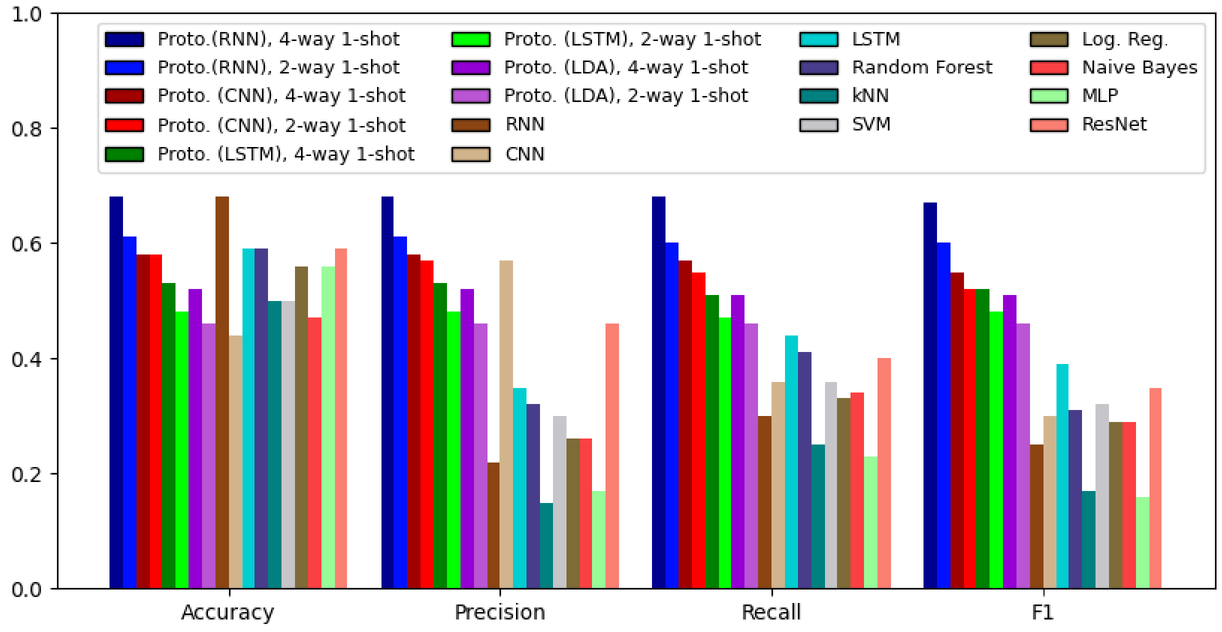

Fig. 4: Accuracy, Precision, Recall and F1 Scores for one-shot trained prototypical networks and traditional machine learning models.

LDA backbone performed the worst, with test scores of 0.46, 0.46, 0.46, and 0.45, respectively. The combined results from 4-way one-shot and 2-way one-shot settings demonstrate the RNN's strong ability to learn class representations of multivariate time-series data. All four models perform significantly better in the 4-way one-shot setting than in the 2-way one-shot setting.

To ensure that the traditional methods are evaluated on the same number of samples from each class as the prototypical networks, a test set is generated by selecting 50 random samples of each class from the test set. The table shows the performance of these models on that set. The RNN model shows the highest accuracy at 0.68. The LSTM model performs the best among these models in terms of recall and F1 scores. The CNN model shows the highest precision score among these models. The results also show that one-shot trained prototypical networks with RNN, CNN, and LSTM backbones all outperform the traditional machine learning approaches in terms of Precision, Recall, and F1 scores. The traditional training procedure for supervised learning models typically requires abundant labeled training data across each target class. The relatively lower performance of the traditional machine learning methods in our experiments can be attributed to the limited number of labeled data and the severe class imbalance present in our dataset.

Due to data constraints, the traditional models are unable to effectively learn robust and discriminative representations for the under-represented classes in the dataset. This inherent limitation of the traditional approaches likely hindered their ability to generalize well to unseen samples during the evaluation phase, resulting in lower classification accuracy compared to the one-shot trained prototypical networks. In contrast, with one-shot learning, the prototypical networks are able to leverage the feature transformation capability of the same models to learn more effective class-level representations from just a few labeled samples per class. The ability to learn discriminative class representations from just a few samples is particularly valuable in real-world applications where data is limited or class distribution is imbalanced.

V. Conclusion

This study explored few-shot learning techniques, specifically prototypical networks with various neural network backbones, for classifying human emotional states from time-series data representing keyboard and mouse usage. The superior performance of few-shot learning methods like prototypical networks highlights their ability to learn discriminative class representations from limited labeled data, even in the presence of class imbalance. This capability is particularly valuable for detecting human emotional states, where certain affective states may be underrepresented in the data. By enabling accurate emotion recognition from unobtrusive keyboard and mouse inputs, without requiring specialized hardware or raising privacy concerns, the proposed few-shot learning approach paves the way for developing tools to improve employee well-being and productivity in both traditional office and work-from-home environments. This approach can also be used to analyze the affective states of students taking remote learning classes.

References

[1] A. Ellahi, M. S. Khalil, and F. Akram, "Computer users at risk: Health disorders associated with prolonged computer use," *J. Bus. Manage. Econ.*, vol. 2, no. 4, pp. 171–182, 2011.

[2] N. Siddiqui, R. Dave, and N. Seliya, "Continuous user authentication using mouse dynamics, machine learning, and minecraft," in *Proc. IEEE Int. Conf. Elect. Comput. Energy Technol. (ICECET)*, 2021, pp. 1–6.

[3] P. Khanna and M. Sasikumar, "Recognising emotions from keyboard stroke pattern," *Int. J. Comput. Appl.*, vol. 11, no. 9, pp. 1–5, 2010.

[4] B. E. Ashforth, G. E. Kreiner, and M. Fugate, "All in a day's work: Boundaries and micro role transitions," *Acad. Manage. Rev.*, vol. 25, no. 3, pp. 472–491, 2000.

[5] S. Koldijk, M. A. Neerincx, and W. Kraaij, "Detecting work stress in offices by combining unobtrusive sensors," *IEEE Trans. Affective Comput.*, vol. 9, no. 2, pp. 227–239, 2016.

[6] G. Giannakakis, D. Grigoriadis, K. Giannakaki, O. Simantiraki, A. Roniotis, and M. Tsiknakis, "Review on psychological stress detection using biosignals," *IEEE Trans. Affective Comput.*, vol. 13, no. 1, pp. 440–460, 2019.

[7] A. M. Khan, "Personal state and emotion monitoring by wearable computing and machine learning," Ph.D. dissertation, Universität Bremen, 2017.

[8] M. Magdin, M. Turcani, and L. Hudec, "Evaluating the emotional state of a user using a webcam," *Int. J. Interact. Multim. Artif. Intell.*, 2016.

[9] A. Dingli and A. Giordimaina, "Webcam-based detection of emotional states," *Vis. Comput.*, vol. 33, pp. 459–469, 2017.

[10] H. Rahman, M. U. Ahmed, S. Begum, and P. Funk, "Real time heart rate monitoring from facial rgb color video using webcam," in *Proc. 9th Annu. Workshop Swedish Artif. Intell. Soc.(SAIS)*, 2016, pp. 15–46.

[11] L. Pepa, A. Sabatelli, L. Ciabattoni, A. Monteriu, F. Lamberti, and L. Morra, "Stress detection in computer users from keyboard and mouse dynamics," *IEEE Trans. Consum. Electron.*, vol. 67, no. 1, pp. 12–19, 2020.

[12] M. De Choudhury, S. Counts, and M. Gamon, "Not all moods are created equal! exploring human emotional states in social media," in *Proc. Int. AAAI Conf. Web Soc. Media*, 2012, pp. 66–73.

[13] P. Zimmermann, S. Guttormsen, B. Danuser, and P. Gomez, "Affective computing—a rationale for measuring mood with mouse and keyboard," *Int. J. Occupat. Saf. Ergonom.*, vol. 9, no. 4, pp. 539–551, 2003.

[14] C. Epp, M. Lippold, and R. L. Mandryk, "Identifying emotional states using keystroke dynamics," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2011, pp. 715–724.

[15] A. N. H. Nahin, J. M. Alam, H. Mahmud, and K. Hasan, "Identifying emotion by keystroke dynamics and text pattern analysis," *Behav. Inf. Technol.*, vol. 33, no. 9, pp. 987–996, 2014.

[16] Y. M. Lim, A. Ayesh, and M. Stacey, "Detecting cognitive stress from keyboard and mouse dynamics during mental arithmetic," in *Proc. Sci. Inf. Conf.*, 2014, pp. 146–152.

[17] M. Rodrigues, S. Gonçalves, D. Carneiro, P. Novais, and F. Fdez-Riverola, "Keystrokes and clicks: Measuring stress on e-learning students," in *Manage. Intell. Syst.: 2nd Int. Symp.*, 2013, pp. 119–126.

[18] S. Natnithikarat, S. Lamyai, P. Leelaarporn, N. Kunaseth, P. Autthasan, T. Wisutthisen, and T. Wilaiprasitporn, "Drowsiness detection for office-based workload with mouse and keyboard data," in *Proc. 12th IEEE Biomed. Eng. Int. Conf. (BMEiCON)*, 2019, pp. 1–4.

[19] A. Pimenta, D. Carneiro, P. Novais, and J. Neves, "Monitoring mental fatigue through the analysis of keyboard and mouse interaction patterns," in *Proc. Hybrid Artif. Intell. Syst.: 8th Int. Conf.*, 2013, pp. 222–231.

[20] V. Parekh, D. Shah, and M. Shah, "Fatigue detection using artificial intelligence framework," *Augment. Hum. Res.*, vol. 5, no. 1, p. 5, 2020.

[21] A. Kampouraki, G. Manis, and C. Nikou, "Heartbeat time series classification with support vector machines," *IEEE Trans. Inf. Technol. Biomed.*, vol. 13, no. 4, pp. 512–518, 2008.

[22] Y.-H. Lee, C.-P. Wei, T.-H. Cheng, and C.-T. Yang, "Nearest-neighbor-based approach to time-series classification," *Decis. Supp. Syst.*, vol. 53, no. 1, pp. 207–217, 2012.

[23] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Exploiting multi-channels deep convolutional neural networks for multivariate time series classification," *Frontiers Comput. Sci.*, vol. 10, pp. 96–112, 2016.

[24] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *J. Syst. Eng. Electron.*, vol. 28, no. 1, pp. 162–169, 2017.

[25] F. J. Ordóñez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.

[26] M. Hüsken and P. Stagge, "Recurrent neural networks for time series classification," *Neurocomputing*, vol. 50, pp. 223–235, 2003.

[27] P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff, "Timenet: Pre-trained deep recurrent neural network for time series classification," *arXiv preprint arXiv:1706.08838*, 2017.

[28] W. Yu, I. Y. Kim, and C. Mechefske, "Analysis of different rnn autoencoder variants for time series classification and machine prognostics," *Mech. Syst. Signal Process.*, vol. 149, p. 107322, 2021.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "Lstm fully convolutional networks for time series classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2017.

[31] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate lstm-fcns for time series classification," *Neural Networks*, vol. 116, pp. 237–245, 2019.

[32] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[33] W. Tang, L. Liu, and G. Long, "Interpretable time-series classification on few-shot samples," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*. IEEE, 2020, pp. 1–8.

[34] C. Huang, X. Wu, X. Zhang, S. Lin, and N. V. Chawla, "Deep prototypical networks for imbalanced time series classification under data scarcity," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, 2019, pp. 2141–2144.

[35] J. Huang, B. Wu, P. Li, X. Li, and J. Wang, "Few-shot learning for radar emitter signal recognition based on improved prototypical network," *Remote Sens.*, vol. 14, no. 7, p. 1681, 2022.

[36] C. Weerasinghe, "Stress detection by keystroke, app mouse changes," 2021. [Online]. Available: https://www.kaggle.com/dsv/2948977

[37] J. P. Pollak, P. Adams, and G. Gay, "Pam: a photographic affect meter for frequent, in situ measurement of affect," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2011, pp. 725–734.

[38] D. Watson, L. A. Clark, and A. Tellegen, "Development and validation of brief measures of positive and negative affect: the panas scales." *J. Personality and Soc. Psychol.*, vol. 54, no. 6, p. 1063, 1988.