7-15-2017

Deep Long Short-term Memory Structures Model Temporal Dependencies Improving Cognitive Workload Estimation

Ryan G. Hefron
Air Force Institute of Technology

Brett J. Borghetti
Air Force Institute of Technology

James C. Christensen
Air Force Research Laboratory

Christine M. Schubert Kabban
Air Force Institute of Technology

Recommended Citation

Hefron, R. G., Borghetti, B. J., Christensen, J. C., & Schubert Kabban, C. M. (2017). Deep long short-term memory structures model temporal dependencies improving cognitive workload estimation. Pattern Recognition Letters, 94(15 July), 96–104. https://doi.org/10.1016/j.patrec.2017.05.020

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Deep long short-term memory structures model temporal dependencies improving cognitive workload estimation

Ryan G. Hefron[a,*], Brett J. Borghetti[a], James C. Christensen[b], Christine M. Schubert Kabban[c]

[a] Deptpartment of Electrical & Computer Engineering, Air Force Institute of Technology, WPAFB, OH 45433, USA
[b] Air Force Research Laboratory, WPAFB, OH 45433, USA
[c] Deptpartment of Mathematics & Statistics, Air Force Institute of Technology, WPAFB, OH 45433, USA

ARTICLE INFO

Article history:
Received 30 November 2016
Available online 20 May 2017

MSC:
41A05
41A10
65D05
65D17

Keywords:
Psychophysiological workload estimation
EEG
Electroencephalograph
LSTM
Long short-term memory
Temporal nonstationarity
Temporal dependence
Day-to-day variability
Time-series analysis
Recurrent neural network
Operator functional state assessment
Human-machine teams

ABSTRACT

Using deeply recurrent neural networks to account for temporal dependence in electroencephalograph (EEG)-based workload estimation is shown to considerably improve day-to-day feature stationarity resulting in significantly higher accuracy ($p < .0001$) than classifiers which do not consider the temporal dependence encoded within the EEG time-series signal. This improvement is demonstrated by training several deep Recurrent Neural Network (RNN) models including Long Short-Term Memory (LSTM) architectures, a feedforward Artificial Neural Network (ANN), and Support Vector Machine (SVM) models on data from six participants who each perform several Multi-Attribute Task Battery (MATB) sessions on five separate days spread out over a month-long period. Each participant-specific classifier is trained on the first four days of data and tested using the fifth's. Average classification accuracy of 93.0% is achieved using a deep LSTM architecture. These results represent a 59% decrease in error compared to the best previously published results for this dataset. This study additionally evaluates the significance of new features: all combinations of mean, variance, skewness, and kurtosis of EEG frequency-domain power distributions. Mean and variance are statistically significant features, while skewness and kurtosis are not. The overall performance of this approach is high enough to warrant evaluation for inclusion in operational systems.

1. Introduction

Teams composed of both humans and machines can potentially work together to mitigate their respective inherent weaknesses. A computer's strength is manifested in its ability to quickly and correctly compute answers, while humans exhibit superior flexibility of response to unexpected situations. Thus, Human-Machine Teams (HMTs) promise to mitigate inherent limitations on computational decision-making in all-human teams while simultaneously reducing the brittleness and inflexibility of fully-autonomous systems [11]. Team outcomes are improved when one agent (human or computer) assists another in the right way at the right time [7]. For computers to help humans in HMTs, they must know the human's cognitive state; this knowledge can be obtained through operator functional state assessment (OFSA) [45]. Several methods of OFSA exist, which can generally be broken into two classes of measures–objective and subjective. Subjective measures usually ask the operator to evaluate themselves either during or after the task, while objective measures use a physiological sensor such as electroencephalograph (EEG) or electrocardiogram (ECG) to provide inputs to an algorithm that assesses the operator's functional state. The benefit of objective measures is that they do not interrupt the operator while performing the task [41,42]. Continuous non-interrupting state assessment is an important characteristic for viable HMTs outside the laboratory.

A key subarea of research within OFSA is mental workload estimation. Enabling the machine in a human-machine team to unobtrusively and continuously ascertain the operator's mental workload is the first step in closing the machine-to-human augmentation loop. In order for augmentation to be effective, it must be driven by an accurate estimate of mental workload [7]. A common

* Corresponding author.
E-mail address: ryan.hefron@afit.edu (R.G. Hefron).

http://dx.doi.org/10.1016/j.patrec.2017.05.020

method for estimating mental workload is to first use statistical machine learning to fit a model which enables prediction of mental workload from the physiological signals, and then use that model to make mental workload estimates from newly-gathered physiological signals [44].

The utility of an OFSA system will depend on the benefits of accurate assessment and the costs of errors. This cost-benefit tradeoff will be application-specific and different for correctly identifying high and low workload states depending on the types of augmentation tied to a given state and the consequences of incorrect/inappropriate activation or lack of activation. These errors directly impact an operator's trust in the automation, in-turn affecting future utility of that automation in a closed loop-fashion [28]. Rouse et al.[35] indicated that a 95% accuracy rate for workload estimation may be required for a system to be acceptable. Parasuraman et al. [31] went further and suggested that if the system does not approach 100% accuracy then the costs of inaccuracy and lack of trust may lead to the system being unacceptable, especially in safety-critical environments.

Unfortunately, current state-of-the-art systems are not yet able to achieve the required accuracy, due in part to the challenge of temporal non-stationarity in psychophysiological signals. This challenge relates to variation over longer periods of time and dependence within shorter periods. Both can negatively impact the generalizable long term accuracy of workload assessment systems [7]. Within shorter spans of time, signals tend to exhibit hysteresis or serial dependence. This suggests that there is inherent structure in the statefulness in the brain that can be exploited with appropriate machine learning techniques. While it is difficult to attribute this dependence to any discrete set of factors, some of the likely possibilities include consistency in default mode activity [34] and hysteresis exhibited by most physiological systems.

In the context of machine learning, temporal non-stationarity can be addressed in two ways. The first is through feature generation or selection. A better set of features will exhibit less long-term non-stationarity and will lead to better model performance. In this work, we examine several feature generation techniques to determine empirically if certain feature sets are superior to others. The second way to address non-stationarity with machine learning is to use algorithms that make different assumptions about the nature of the data being processed. As it stands, most published research on operator workload estimation implicitly assumed temporal independence from one time segment to the next. This is likely a poor assumption due to both the factors discussed above as well as longer term effects such as fatigue and performance hysteresis with mental workload transitions [22]. An example from aviation illustrates this nicely. If a pilot has just completed flying an instrument approach in instrument meteorological conditions (IMC) when an unexpected emergency requires attention, pilot workload will increase differently than if the pilot had the same unexpected emergency arise following a period of autopilot-on flight at cruising altitude in visual meteorological conditions (VMC). This simple example illustrates that what has happened in the recent past temporally, matters for operator workload assessment.

Machine learning algorithms that consider past information as well as current information when fitting models should perform better. Such algorithms must be able to learn a temporal representation of the data. A common model used for modeling temporal data is the Recurrent Neural Network (RNN). RNNs are neural networks that are able to learn sequences that are not composed of independent, identically distributed observations [16]. Rather, they are able to elicit the context of observations within sequences and accurately classify sequences that have strong temporal correlations [16]. Historically, RNNs had limitations when training models with more than 10–20 time steps which led to poor performance. Incorporating longer time-series data streams would cause computational sensitivity problems that stymied RNN training.

Recent developments have resulted in RNN architectural and training advances which mitigate these computational problems and allow much longer temporal sequences to be processed. One approach is the Long Short-Term Memory (LSTM) layer. LSTM architectures extend the length of sequences that can be considered by a RNN by overcoming computational sensitivities encountered during backpropagation [21]. For these reasons, they may offer improved workload classification accuracy over other methods when using EEG data. With these improvements in machine learning, there is no longer a reason to avoid incorporating temporal context in a workload model. We capitalize on these machine learning developments in our research.

The primary contribution of this research is demonstration of significantly improved cross-day workload classification accuracy by integrating contextually relevant algorithmic architectures with improved feature generation techniques. We statistically evaluate all combinations of mean, variance, skewness, and kurtosis of frequency-domain power distributions and contrast a variety of RNN architectures, to include deeply stacked LSTMs, with baseline algorithms and features. Both linear and Radial Basis Function (RBF) Support Vector Machines (SVMs) and single-layer feed-forward Artificial Neural Network (ANNs) using mean-only features are used as baseline cases. We show that by accounting for temporal dependence using deep LSTM models trained with new feature combinations, we can maximize cross-day workload estimation accuracy resulting in a 58% reduction in classification error over baseline methods and a 59% decrease in error compared to the best published results for this dataset.

2. Background and related work

Temporal non-stationarity of electroencephalograph (EEG) signals within individuals is likely caused by a large number of intrinsic and extrinsic factors. Participant motivation and mental or physical readiness are examples of some intrinsic factors; extrinsic factors include significant differences in EEG electrode placement, changes in conductance, and different motion artifacts [8,23,30]. Due to the challenge of handling these factors, cross-day non-stationarity of EEG signals has motivated a number of related studies including several using the same dataset described below.

2.1. Dataset

Data for our investigation was used in the 2011 Cognitive State Assessment Competition [13] and was recorded during a prior human research study performed by Wilson et al. [43]. Eight participants completed scenarios within the Multi-Attribute Task Battery (MATB) [10] environment across five test days spread out over a month-long period. Monitoring, communication, resource management, and tracking tasks were presented and manipulated to induce three levels of difficulty: low, medium, and high [8,43]. Resource allocation errors, monitoring task reaction times, and communication response times were recorded and used to validate that participants experienced distinct low and high difficulty levels. Participants were trained to asymptotic proficiency prior to the first test day [43].

For each participant, horizontal electrooculogram (HEOG), vertical EOG (VEOG), and 19 channels of EEG voltages (according to the International 10–20 System) were sampled at 256 Hz. On each of the five days, each participant performed three five-minute trials at low, medium, and high difficulty for a total of nine trials per day. Trials were presented in a random ordering with transition periods in between. Each participant completed a 30 s resting baseline at the start of each session prior to the MATB task. Only six of the participants were used in our study due to missing data from two of the original eight participants [19]. Similar to Christensen et al.

[8] and Casson [5], we selected data from the low and high workload conditions resulting in a total of 30 conditions–15 low and 15 high–for each individual.

2.2. Within-participant cross-day variability

Many researchers using this dataset focused their efforts on cross-participant variability or a combination of cross-participant and cross-day models rather than independently investigating cross-day variability within individuals. This section will only consider those works that exclusively examined within-participant cross-day variability.

In the workload literature using this dataset, three research teams focused solely on the performance of classifiers in cross-day analyses. Hefron and Borghetti evaluated the variance of frequency-domain power distributions as a feature and found it to improve accuracy for random forest, Linear Discriminant Analysis (LDA), and K-Nearest Neighbors (KNN) classifiers for the two-class–high versus low workload–problem. Models built with variance and mean features exhibited a 5.8% increase in accuracy over models built using only the mean of frequency-domain power distributions [19]. In their study, it was postulated that the use of skewness and kurtosis as features could generate further gains in classification accuracy [19].

In another study, Christensen et al. evaluated cross-day classification performance of low versus high workload by training three classifiers: LDA, Support Vector Machine (SVM), and a single-hidden-layer feedforward Artificial Neural Network (ANN) [8]. Two SVM implementations were used–one used a radial basis function kernel and the other a linear kernel. The authors noted the linear kernel performed best and was used for reporting results. One portion of the experimental design used leave-one-out cross-validation, holding out one day's data in each case. This procedure produced five separate testing periods that were used to evaluate algorithmic performance. Of note, the SVM achieved 68% accuracy, while the ANN attained 83% accuracy [8]. The results demonstrated significantly better cross-day performance for the neural network compared to more traditional machine learning techniques, namely LDA and SVM classifiers. These results indicated that neural networks may have more capacity to handle non-stationarity of feature-to-target mappings.

In a study very similar to Christensen's, Casson [5] evaluated the effect of temporal stability of feedforward ANNs trained on EEG signals with differences of seconds, minutes, hours, and days between collection of training data and testing data. While other interesting results were presented in the paper, the most relevant to our work concerned participant-specific, leave-one-session-out, cross-validated models. These models were aimed at producing excellent cross-session predictive performance and attained an average classification accuracy of 73%. Differences in preprocessing, training, and network architecture contributed to the differential between these results and ours.

In all of these investigations, the demonstrated classification accuracy fell short of recommendations for use in many operational settings. While accuracy requirements will be task specific, as discussed in Section 1, neither Rouse's desired 95% accuracy rate [35] nor Parasuraman's near 100% accuracy [31] condition were met for workload estimation. Since all of these methods assumed temporal independence between time segments, we believe new models that account for temporospatial dependencies and the use of new features should be explored.

2.3. RNNs and LSTMs

Deep neural networks have been used extensively to achieve state-of-the-art results across a wide range of categories including image recognition, speech recognition, translation, and image captioning [16,17,26,40]. Recurrent Neural Networks (RNNs) are a type of deep network where depth is added via a recurrent connection in the hidden layer which is used to process sequential data. Processing sequential data is fundamentally different than processing independent identically distributed observations because of the distributional dependence on the sequence. In a traditional feed-forward ANN, each observation is processed individually and the network state does not persist while other potentially sequentially-dependent points are processed. Conversely, in a RNN, recurrent connections pass state information across time steps allowing previously processed observations to affect the subsequent observations [29].

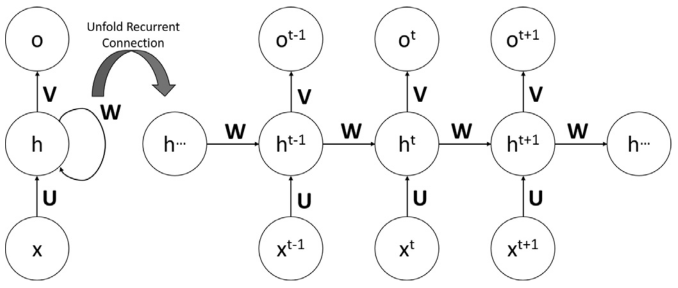

Fig. 1. Temporal unfolding of the recurrent unit in the computational graph. On the left side is the cyclic graph. All cycles have been removed from the graph on the right side by unfolding the cyclic graph in time. Note the weight matrices **U**, **V**, and **W** are shared across time.

Feedforward ANNs have a directed acyclic computational graph–one layer feeds to another with no cycles. RNNs can be understood as an extension to the feedforward structure allowing for cycles in the graph structure. Fig. 1 shows how very deep computational graphs can be created when the recurrent connections are unfolded in time. The process of unfolding a recurrent network simply requires removal of all cycles in the graph to form a directed acyclic graph [15]. To allow these networks to learn important pieces of information that may be located at different positions in the sequential data, the input, recurrent, and output weight matrices' parameters are shared [15]. If different parameters were used for each time step, no generalization across time would be possible; sharing parameters allows the network to keep track of state. Since temporal non-stationarity of EEG signals across days is a challenge, we needed to ensure the temporal sequences supplied to our model did not exceed periods of time over which feature-to-target non-stationarity would occur.

In addition to the depth of a RNN due to unfolding over time, depth can be added to the network when recurrent layers are stacked in a sequence-to-sequence fashion. This means that the output from one layer of a RNN returns a sequence of vectors which form the input to the next layer. Graves et al. demonstrated that deeply layering RNNs has a more beneficial effect than merely adding memory cells [17]. The lower layers transform input sequences into more easily learned representations in the higher layers, resulting in better classification accuracy [15].

One problem with the simple recurrent structure shown in Fig. 1 is a result of shared weights which produce vanishing and exploding gradients during backpropagation through time. This limits the sequential depth of simple recurrent units to sequences of no greater than 10–20 observations because the signal from distant positions will not propagate to the current time [15]. A significant innovation which solved this problem was Long Short-Term Memory (LSTM) [14,21]. LSTM provides the algorithm fine control over what is put into memory and removed from memory in the hidden layer. This is achieved by a combination of three gates which control flow into and out of the memory cell: an in-

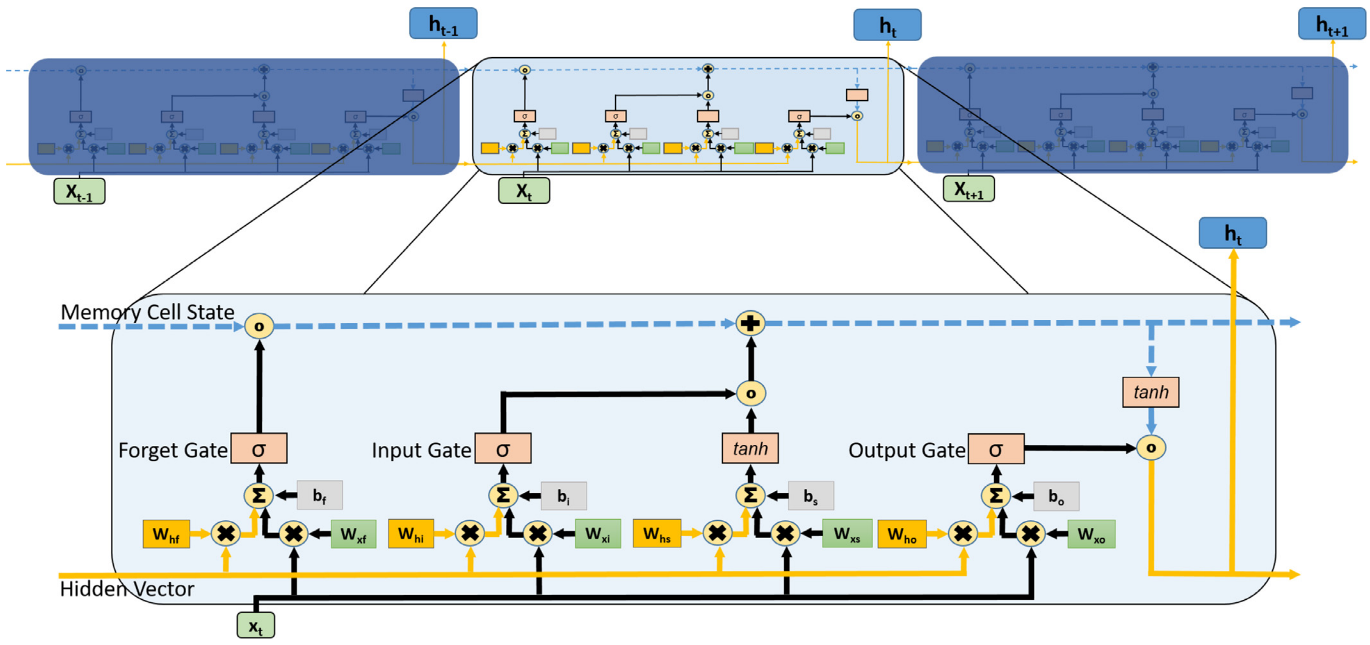

Fig. 2. This illustrates the internal structure of a single LSTM memory cell unfolded in time where x_t is the windowed sequence input from time t, and h_t is the hidden-vector state at time t. Weight matrices are labeled using a from-to subscript convention, $W_{from,\, to}$. Biases, b, have subscripts associated with their corresponding gate or activation. The following operations are annotated symbolically: **X** represents matrix-vector multiplication, $\circ$ is a Hadamard (element-wise) product between two vectors, **Σ** is used for summations of 3 or more vectors, $+$ indicates the addition of two vectors, finally σ and *tanh* are respectively a sigmoid function and hyperbolic tangent function applied element-wise to a vector. The memory cell state vector propagates along the dashed blue line at the top while the input vector and hidden vector buses are connected as shown by the gold and black lines respectively. All three of these vectors should be thought of as column vectors. The size of the hidden vector and memory cell state vector are equal to the number of "memory cells" specified during initialization of the LSTM layer. Three sigmoid functions act as gates and are labeled: forget, input, and output. If a LSTM returns a sequence, the sequence is a series of the hidden vector states with each corresponding to the output from a particular timestep within the supplied temporal window. If a sequence is not returned, typically the last hidden state is returned. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

put gate, a forget gate, and an output gate. Each gate works by using an element-wise sigmoid function, σ, to scale each element of the gate vector to a value between 0 and 1. The gating functionality is accomplished by taking this vector of values between 0 and 1 and performing an element-wise multiplication with another vector, thus specifying what proportion of the second vector passes through the gate; and conversely, determining which parts are blocked. Fig. 2 illustrates the internal structure of the LSTM. For clarity, we will describe the state of a memory cell at time t with recurrent connections from time $t-1$, and to time $t+1$.

As shown in Fig. 2, there are two vectors that persist from time $t-1$: the hidden vector, h_{t-1}, and the memory cell state, s_{t-1}. The forget gate f_t, as shown in Eq. (1), determines what to remove from the memory cell state. In other words, it forces the memory cell to forget things that are not important based on error backpropagation:

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f), \tag{1}$$

where the weight matrix W_{xf} is the weight matrix from the input x to the forget gate f_t, x_t is the input at time t, W_{hf} is the weight matrix from the previous hidden vector h_{t-1} to the forget gate f_t, and b_f is the forget gate bias. The from-to subscript convention is used to describe each weight matrix. The input gate i_t determines how much each element of the candidate update vector, $\tilde{s_t}$, should be added to the corresponding memory cell element at time t based on the recurrent connection from the hidden vector h_{t-1} and the sequential input at time t, x_t. The gate scales the candidate update vector which is the output from a fully-connected *tanh* layer:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{2}$$

$$\tilde{s_t} = tanh(W_{xs}x_t + W_{hs}h_{t-1} + b_s). \tag{3}$$

The delicate balance required to maintain memory cell state over long sequences is attained by forgetting old information and incorporating new information. Forgetting is accomplished by multiplying the old memory cell state by the output of the forget gate, while the new information is supplied by adding the portion of each value specified by the element-wise product of the input gate with the candidate update:

$$s_t = i_t * \tilde{s_t} + f_t * s_{t-1}. \tag{4}$$

Finally, the output gate determines what should be output to the hidden vector from the memory cell state, given the temporal context of the time-step, to minimize error. The output gate, o_t, and hidden vector, h_t, are given by:

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{5}$$

$$h_t = o_t * tanh(s_t). \tag{6}$$

2.4. RNN models for EEG analysis

Despite excellent results in other fields, relatively few researchers have applied deep neural network techniques to classify EEG data. Bashivan et al. [2] trained a deep recurrent-convolutional neural network accounting for both temporal and spatial dependencies in the network. They began by performing a Fast Fourier Transform (FFT) on each time-series signal from each electrode and estimating the power in the theta (4–7 Hz), alpha (8–13 Hz), and beta (13–30 Hz) frequency bands over 3.5 s working memory experiment trials with four classes of task difficulty. Then, they created a time-series of images by performing a 2-d Azimuthal Equidistant Projection (AEP) for power in each of the three different bands. This preserved distances from each electrode to the center point of the EEG cap, thus accounting for some spatial dependencies. These images were fed into a series of convolutional and max pooling layers. Their best performing model fed the output from the convolutional portion of the architecture into a 1-d convolutional layer, as well as a LSTM layer, and merged the output from both of these into a fully connected layer. This was then connected to a softmax layer which provided the final classification result. This complex architecture used 1.62 million parameters and was able to reduce the classification error from 15.34% to 8.89%; an impressive 42% reduction compared to a baseline radial basis function SVM [2]. While a large reduction in error over baseline methods was achieved, the complexity and amount of computation required to train such a deep network is significant. Our research sought to examine if similar reductions in error over baseline methods could be achieved using different preprocessing techniques and a less complex deep architecture which only used recurrent neural networks.

Two other researchers used a form of the LSTM to analyze EEG data. Davidson, et al. found that a small single-layer LSTM was able to identify lapses in attention based on EEG spectral data better than a tapped delay-line Multilayer Perceptron (MLP) during performance of a visuomotor tracking task [12]. They trained their networks in a leave-one-out, cross-subject manner. Their results evaluated temporal dependencies out to a length of 6 s and showed that accounting for temporal dependence of up to 4 s prior to a lapse improved detection. We use an extended temporal window of 30 s in this study. In other work, Binz et al.[3] used a derivative of the LSTM unit, called Dynamic Cortex Memory (DCM), to classify imagined sensorimotor imagery data from a Brain Computer Interface (BCI) workshop competition [4]. A single hidden layer was used with eight DCM units followed by a softmax output layer. While their results did not achieve state-of-the art accuracy, several advantages were present in their solution. Their network was able to provide real-time results and their solution did not need to specify a time window, since the network learned appropriate temporally-dependent sequences [3]. Binz's work largely differed from ours since a mixture of within-participant and cross-participant models were used to evaluate trials that were separated from the training data by only seconds-to-minutes rather than days. The only similarity between our studies was that we both used forms of the LSTM architecture to account for temporal dependencies in EEG signals.

In a medical application, several research groups [18,27,36,39] successively improved performance of epilepsy diagnosis using RNNs. All four groups used various input features to train small Elman RNNs in a cross-subject manner using the dataset described by Andrzejak et al. [1]. The Elman RNN architecture has limited temporal representational capacity compared to the networks trained in our investigation. Like our study, each research group demonstrated the superiority of an RNN compared to other neural networks that did not incorporate temporal dynamics. However, unlike our experiment, the data sequences analyzed did not span temporal lengths great enough to examine day-to-day variability. Finally, Guler and Ubeyli both demonstrated that summary statistics (min, max, mean, standard deviation) of time-varying feature distributions can be useful input features for a recurrent model; however, no comparison to a baseline feature set was provided in either case.

The primary difference between previous research and ours is that we demonstrate improved within-participant, day-to-day feature stationarity by accounting for temporal dependencies in cognitive activity; whereas the aforementioned studies do not address day-to-day variability. Another difference is that the preceding research focused on either medical uses or state estimation unrelated to workload. A final differentiating factor was that many previous studies were conducted prior to breakthroughs that enabled drastic improvements in network performance such as advances in initialization, optimization, and regularization techniques, as well as the rise of abundant computational capacity via Graphics Processing Unit (GPU) computing which enables the training of larger, deeper networks.

Table 1
Test matrix of all combinations of mean, variance, skewness, and kurtosis features. * Denotes feature sets that were included in models that incorporated the mean while all others are models that did not incorporate the mean.

Test run	Features included in dataset
1*	Mean
2	Variance
3	Skewness
4	Kurtosis
5*	Mean, Variance
6*	Mean, Skewness
7*	Mean, Kurtosis
8	Variance, Skewness
9	Variance, Kurtosis
10	Skewness, Kurtosis
11*	Mean, Variance, Skewness
12*	Mean, Variance, Kurtosis
13*	Mean, Skewness, Kurtosis
14	Variance, Skewness, Kurtosis
15*	Mean, Variance, Skewness, Kurtosis

3. Methodology

The goal of producing several workload models to evaluate the efficacy of new features and LSTM based classification algorithms required preprocessing of the EEG data to convert it into the frequency domain and to extract power in different frequency bands as outlined by [19]. Raw EEG data was transformed into features in clinical frequency bands (delta (1–4 Hz), theta (4–8 Hz), alpha (8–14 Hz), beta (15–30), and gamma (30–55 Hz) to conduct time-frequency analysis using the following process: The power spectral density was determined for 30 points spread out over a logspace from 3 Hz to 55 Hz by extracting power from complex Morlet wavelets [9]. Each wavelet was 2 s in length and the number of wavelet cycles increased logarithmically from 3 to 10 in conjunction with the frequencies. Mean power in each band was determined by averaging each power value for the evaluated frequencies within each of the clinical bands. Power was then aggregated over a ten second sliding window with 9 s of overlap, allowing for a new update each second.

Final features were generated by determining the mean, variance, skewness, and kurtosis of the power distribution in each of these ten second windows for all 19 EEG electrode sites across the five frequency bands. This process yielded 380 features for each second and approximately 9000 observations per individual for the five day period. These features were then centered and scaled by session so that each session had a mean of zero and variance of one since none of the algorithms used for analysis were scale invariant. The data were split such that the first four days were used for training and cross-validation while the last day was reserved for testing.

To examine the effect of inclusion/exclusion of particular types of features, the test matrix shown in Table 1 was constructed to document features included in each test run. A total of six algorithms were used to train models: linear SVM (SVM-L), Radial Basis Function (RBF) SVM (SVM-R), feedforward ANN (ANN), deeply stacked simple RNN (RNN-D), single LSTM (LSTM-S), and deeply stacked LSTM (LSTM-D). All model development used 4-fold cross-validation to select hyperparameters for each model and then final models were trained using all data from days 1–4. This process was repeated for each feature set in Table 1 resulting in 90 final models for each algorithm. Final models were trained using each set of features and differences in classification accuracy due to choice of algorithm and feature set were considered. It is important to note that the cross-validation data was split so that each fold was a full day rather than splitting the folds by random selection of observations from the entire dataset. There were two compelling reasons for this choice. The first reason was that randomly selected cross-validation points allow for too many temporally adjacent points to be split between the training and test sets which artificially inflates cross-validation accuracy due to the non-stationarity of the datasets. The second reason was to preserve temporal context of the data for processing when using RNNs.

Once final models were produced, classification accuracies for the holdout day 5 test set were determined for each individual and the algorithm average was calculated across participants. This enabled by-algorithm and by-feature set comparisons. Due to confounding effects of algorithm selection and feature sets on classification accuracy, an ANOVA test with five factor outcomes was performed to elicit the effect of varying levels of each factor. The first factor was algorithm selection which had six levels corresponding to each of the aforementioned algorithms. The remaining four factors grouped the feature sets in a binary fashion based on whether a feature was included or excluded from a particular test run. For example in Table 1, all asterisked runs produced models that included the mean, while the others were models that excluded mean features. Interaction effects were not examined. A significance level of $\alpha = 0.05$ was used to indicate if a factor had a statistically significant impact on classification accuracy. The Tukey Honest Significant Difference (HSD) test was performed following the ANOVA to determine which classification accuracies were different from each other in the six-level algorithm factor, and to determine the direction and magnitude of differences across all two-level factor comparisons.

All neural networks were created using the Keras [6] and Theano [38] frameworks. The feedforward ANN had a single hidden layer that was fully connected with the input layer and the output layer. The input consisted of the appropriate number of features for a given test run, while the output layer was a single node with a sigmoid activation function which forced a classification as either high or low workload. Mini-batch gradient descent was performed using 600 observations per batch for all neural network implementations, and the *Adam* optimizer was chosen to optimize mini-batch gradient descent due to its ability to handle non-stationary targets and noisy data [25]. A binary cross-entropy loss function was selected as the cost function [15]. The number of nodes in the hidden layer was tuned by performing 4-fold cross-validation while varying the number of nodes from 50 to 800 in steps of 50. This resulted in 3960 cross-validation models being trained. The lowest cross-validation error rate was used to determine both the number of nodes to use in the hidden layer and how many epochs to train the network.

As illustrated in Fig. 3, the deep LSTM architecture consisted of an input layer, the first sequence-to-sequence LSTM layer, a many-to-one LSTM layer, a 20% dropout layer, and a final sigmoid activation function for binary classification. The first hidden layer contained 50 LSTM units while the second hidden layer used 10 units. With unlimited resources, we would have tuned the number of hidden layer nodes exhaustively using grid cross-validation. However, due to computational resource constraints, several hidden layer sizes were tested for each layer on smaller representative sets of data and it was found that reducing the number of LSTM units in each layer improved generalization and that empirically, 50 and 10 appeared to work well. Each network had a lookback of 30 s of pre-processed features or, for the case of the second layer, features generated from the output of the first LSTM layer. Dropout

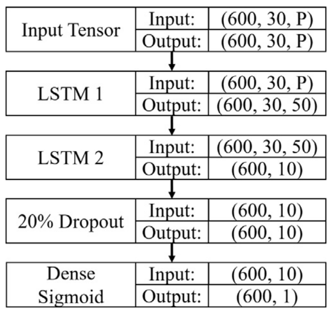

Fig. 3. Deep LSTM Architecture: This illustrates the size of the tensor in terms of batch size, temporal depth in seconds, and number of features used at each level of the network. *P* represents the number of features used based on the test matrix shown in Table 1, and ranged from 90 to 380 features. Since LSTM 2 does not return a sequence, the tensor becomes two-dimensional at that point.

on the input gates to each LSTM layer and between the final LSTM and fully-connected sigmoid layer served as a method of regularization and was set to 20% [15,37]. Dropout prevents co-adaptation of the hidden units by temporarily removing a percentage of randomly selected nodes, including their input and output, in a given layer during a training pass [20]. This forces hidden units to learn features without depending upon particular nodes to correct mistakes made during learning [20]. While dropout is the most widely used regularization method for deep neural architectures, it is also important to understand when it is not appropriate to use. The recurrent connections within the LSTM structure are one such case. The purpose of the recurrent connection in a LSTM is to store important long-term dependencies. Pham et al. [33] showed that if dropout is applied to the recurrent connection, then the long term memory becomes corrupted and inhibits learning rather than improving generalization. Similar to Zaremba et al. [46], we found using a dropout of 20% on the input gates and between the final LSTM output and the sigmoid classification layer provided better results than the typically recommended 50% dropout for fully connected layers. The number of training epochs was tuned using cross-validation as previously specified in this section. The length of training with the lowest cross-validation error rate for each participant/feature set combination was selected for final network training. All training data was then used to retrain the network. To ensure that anomalous behavior was not present in the reported results from day 5, all other combinations of cross-validation with hold-out test day were performed for the deeply stacked LSTM model. No significant deviations from the reported upon results in Section 4 were observed due to permuting the test and training days.

Two recurrent networks were trained which were derivatives of the deep LSTM architecture. The differences between the architectures are detailed next. The deeply stacked simple RNN used the same architecture as the deep LSTM except that instead of using LSTM units, simple recurrent units were used. Due to parameter sharing in the weight matrix for the recurrent connections and a lack of gating functions, the vanishing gradient problem was present and restricted the effective temporal period to 10–20 s for this model despite 30 s of data being provided as input [15]. The second network consisted of an input layer, a many-to-one LSTM layer using 50 hidden units, and a sigmoid output layer. Training and classification methods using these networks mirrored those of the deep LSTM architecture.

Two categories of SVMs were trained, one with a linear kernel as recommended by Christensen et al. [8], and one using a RBF kernel. The resulting models were used to compare neural network results to those from a more traditional machine learning algorithm. To train the linear SVM, the appropriate number of features

Table 2
ANOVA table with five factors. Algorithm is a six-level factor indicating the algorithm used: LSTM-D, LSTM-S, RNN-D, ANN, SVM-L, or SVM-R. The final four factors have two levels and indicate the feature was either included or excluded.

Source	DF	Sum of squares	R^2	F Ratio	Prob >F
Algorithm	5	0.7876	0.2774	29.4052	<0.0001
Categorized by mean	1	1.2113	0.4266	226.1277	<0.0001
Categorized by variance	1	0.2579	0.0908	48.1448	<0.0001
Categorized by kurtosis	1	0.0033	0.0011	0.6235	0.4301
Categorized by skewness	1	0.0028	0.0009	0.5252	0.4689

Table 3
Tukey HSD results for all pairs of algorithm levels.

Model 1	Model 2	Diff	Std Err	t Ratio	Prob>|t|	95% Conf Int
LSTM-D	ANN	0.089	0.011	8.19	<0.0001	0.058 to 0.121
LSTM-D	SVM-L	0.088	0.011	8.06	<0.0001	0.057 to 0.119
LSTM-D	SVM-R	0.078	0.011	7.12	<0.0001	0.046 to 0.109
LSTM-D	RNN-D	0.020	0.011	1.82	0.4536	−0.011 to 0.051
LSTM-D	LSTM-S	0.010	0.011	0.89	0.9493	−0.022 to 0.041
LSTM-S	ANN	0.080	0.011	7.31	<0.0001	0.049 to 0.111
LSTM-S	SVM-L	0.078	0.011	7.17	<0.0001	0.047 to 0.109
LSTM-S	SVM-R	0.068	0.011	6.23	<0.0001	0.037 to 0.099
LSTM-S	RNN-D	0.010	0.011	0.93	0.9382	−0.021 to 0.041
RNN-D	ANN	0.070	0.011	6.37	<0.0001	0.038 to 0.101
RNN-D	SVM-L	0.068	0.011	6.24	<0.0001	0.037 to 0.099
RNN-D	SVM-R	0.058	0.011	5.30	<0.0001	0.027 to 0.089
SVM-R	ANN	0.012	0.011	1.08	0.8906	−0.019 to 0.043
SVM-R	SVM-L	0.010	0.011	0.94	0.9351	−0.021 to 0.041
SVM-L	ANN	0.001	0.011	0.13	1	−0.03 to 0.033

for a given test run were provided for each observation supplied to the SVM. The tuning parameter *C* set a tolerance for number and severity of margin and hyperplane violations–effectively determining smoothness of the decision surface [24,32]. This hyperparameter was optimized using a cross-validated grid search across exponentially spaced values of *C* from 10^{-4} to 10^{2} resulting in 2520 cross-validation models for the linear SVM. Nearly all *C* values selected for the final models were either 10^{-4} or 10^{-3}. All training data was then used to retrain the linear SVM with the selected values for *C*. The procedure for training the RBF SVM was nearly the same as in the linear case, except a cross-validated grid search over values of γ and *C* was performed where the hyperparameter γ controlled the radius of influence for a single observation. The same range of *C* values were evaluated while γ was exponentially varied from 10^{-3} to 10^{2}, resulting in 15,120 cross-validation models for the RBF SVM. The best hyperparameters were selected and the final models were trained. This procedure ensured proper tuning to establish a valid baseline for comparison with new techniques.

Table 4
Cross-participant averaged classification accuracy for each model and feature set. Mean, variance, skewness, and kurtosis features are denoted by M, V, S, and K respectively. Bold values are models with greater than 90% classification accruacy.

Feature set	SVM-L	SVM-R	ANN	LSTM-S	RNN-D	LSTM-D
M	0.823	0.834	0.816	0.871	0.884	0.891
V	0.762	0.769	0.754	0.865	0.850	0.861
S	0.680	0.694	0.678	0.757	0.760	0.765
K	0.672	0.686	0.662	0.732	0.736	0.762
M/V	0.836	0.846	0.844	**0.911**	0.866	**0.911**
M/S	0.823	0.836	0.828	0.878	0.861	0.897
M/K	0.831	0.843	0.830	0.896	0.884	**0.908**
V/S	0.762	0.771	0.748	0.847	0.869	0.862
V/K	0.758	0.769	0.757	0.863	0.833	0.882
S/K	0.716	0.733	0.709	0.834	0.770	0.807
M/V/S	0.839	0.840	0.848	**0.909**	0.890	**0.927**
M/V/K	0.835	0.837	0.838	**0.907**	**0.918**	**0.930**
M/S/K	0.824	0.838	0.831	0.897	**0.911**	**0.918**
V/S/K	0.759	0.771	0.753	0.846	0.847	0.863
M/V/S/K	0.835	0.842	0.836	**0.913**	0.899	**0.930**

4. Results

Results of the five-factor ANOVA indicate that algorithm type, mean features, and variance features had a statistically significant effect on classification accuracy (all *p*-values < 0.0001). Skewness and kurtosis in the presence of mean and variance were not significant (*p*-values > 0.43). Table 2 summarizes the results from the ANOVA while Tables 3 and 5 display the post hoc Tukey HSD results. Results from the Tukey HSD test showed that the ANN and both SVM models' mean classification accuracies were not statistically different with *p*-values ranging from 0.8906 to 1. The deep LSTM models demonstrated statistically significant accuracy increases of 8.9% over ANN results as well as 8.8% and 7.8% over linear and RBF SVM results respectively (all *p*-values < 0.0001). The participant averaged classification accuracies for each model and feature set in Table 4 show that there are 6 feature combi-

Table 5
Tukey HSD results comparing models where the feature of interest was included versus those where it was excluded.

Model 1	Model 2	Diff	Std Err	t Ratio	Prob>|t|	95% Conf Int
Mean	-Mean	0.096	0.006	15.04	<0.0001	0.083 to 0.108
Var	-Var	0.044	0.006	6.94	<0.0001	0.032 to 0.057
Skew	-Skew	0.005	0.006	0.72	0.4689	−0.008 to 0.017
Kurt	-Kurt	0.005	0.006	0.79	0.4301	−0.007 to 0.018

nations that result in classification accuracies greater than 90% using the deep LSTM architecture. The highest accuracy feature set was attained using all features with the deep LSTM model. This model achieved an average classification accuracy of 93.0% across participants. This compares favorably to 84.8% and 84.6% for the best ANN and SVM cross-participant feature set performance. Furthermore, this represents a 58% decrease in error compared to the best baseline case–the mean-only RBF SVM. These results illustrate the importance of accounting for temporal dependencies in workload data. Further examining the results shows that the influence of using a temporally-stateful model on classification accuracy for workload estimation cannot be understated. In all cases tested, every temporally-stateful model outperformed the best performing non-stateful model and were found to be significantly different than all non-stateful models with Tukey HSD p-values all < 0.0001 (Table 3).

Statistically, inclusion of mean and variance were significant, while skewness and kurtosis were not. ANOVA results show a significant effect dependent upon inclusion or exclusion of mean features ($p < 0.0001$). Post hoc comparisons using the Tukey HSD test indicate that average classification accuracy for models including the mean features results in an accuracy increase of 9.6% versus models excluding the mean features (Table 5). A significant effect is also present dependent upon inclusion or exclusion of variance features ($p < .0001$). The Tukey HSD test shows that average classification accuracy for including the variance features increased 4.4% versus excluding the variance features. ANOVA and Tukey HSD results for skewness ($p = 0.4689$) and kurtosis ($p = 0.4301$) were not significant. However, by comparing models with and without a single feature, these results merely indicate that kurtosis does not add significantly to a model with skewness included and vice-versa. We hypothesize that skewness and kurtosis may be more important in situations involving workload transitions. We did not investigate workload transitions in our research, but expect that transitions between different workload levels may first become apparent in the tails of the distributions.

5. Conclusion and future work

Cross-day workload estimation based on EEG is a difficult domain due to temporal non-stationarity of feature-to-target mappings. Previous research on cross-day workload estimation implicitly assumed independence of a participant's workload from one instance to the next due to the algorithms used for analysis. Theory and practical experience show that workload can build in a cumulative fashion and that a temporal dependence exists. Recurrent Neural Network (RNN) models, in particular those that use Long Short-Term Memory (LSTM) architectures, can account for both long-term and short-term temporal dependencies inherent in brain activity data. This work also statistically evaluated the utility of mean, variance, skewness, and kurtosis of frequency-domain power distributions and found only the mean and variance to be statistically significant.

This research demonstrated the utility of deep RNN models and particular feature sets for cross-day workload estimation and showed that drastically improved model accuracy can be achieved over Support Vector Machine (SVM) and feedforward Artificial Neural Network (ANN) models when working with non-independent data. Previously, the best accuracy achieved using this dataset was 83%. Our models built with deep LSTMs increased that accuracy to 93.0%, representing a 58% reduction in classification error over baseline methods and a 59% decrease in error compared to the best published results for this dataset. This pushed us closer to the 95% threshold where adaptive operator augmentation may become feasible.

There is an abundance of future work to be pursued in this area. Due to time constraints and computational complexity, only a select number of deep architectures were examined during this research. A thorough evaluation of different deep RNN architectures to include variations in the depth of hidden layer recurrent connections, stacking of different sized LSTM layers, and interleaving fully-connected feedforward layers between sequence-to-sequence recurrent layers may yield additional improvement.

Another enhancement for future work would be to include workload transitions. We believe that skewness and kurtosis may be relevant in datasets where the target workload is transitioning across high and low workload conditions since distributional changes may first become evident in the tails of the distributions. Other ideas to pursue include creating ensembles of deep RNNs. This would almost certainly improve results as long as enough diversity could be added to the ensemble. In our research, we only supplied 30 s sequences to the recurrent networks. Exploring variations in temporal length supplied to a RNN to examine stationarity of target-to-feature mapping for workload estimation using electroencephalograph (EEG) would also be an interesting subject for future work. Deep RNN architectures could also be used to improve cross-participant and cross-day classification simultaneously by training and testing on all participants grouped together rather than individually. Finally, significant improvement could be realized if time-series data augmentation methods are developed capable of forcing a learned invariance to the sources of temporal non-stationarity.

Acknowledgment

We thank Justin Estepp and the Air Force Research Laboratory 711th Human Performance Wing (711HPW/RHCPA) for providing the data from their MATB experiment and other support for this research. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government.

References

[1] R.G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, C.E. Elger, Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: dependence on recording region and brain state, Phys. Rev. E 64 (6) (2001) 061907.
[2] P. Bashivan, I. Rish, M. Yeasin, N. Codella, Learning Representations from EEG with Deep Recurrent-Convolutional Neural Networks. International conference on learning representations (2016).
[3] M. Binz, S. Otte, A. Zell, On the applicability of recurrent neural networks for pattern recognition in electroencephalography signals, in: Workshop New Challenges in Neural Computation 2015, 2015, p. 85.
[4] B. Blankertz, G. Dornhege, M. Krauledat, K.-R. Müller, G. Curio, The non-invasive berlin brain–computer interface: fast acquisition of effective performance in untrained subjects, Neuroimage 37 (2) (2007) 539–550.
[5] A.J. Casson, Artificial neural network classification of operator workload with an assessment of time variation and noise-enhancement to increase performance, Front. Neurosci. 8 (2014) 372, doi:10.3389/fnins.2014.00372.
[6] F. Chollet, Keras, 2015, (https://github.com/fchollet/keras).
[7] J.C. Christensen, J.R. Estepp, Coadaptive aiding and automation enhance operator performance, Hum. Factors (2013) 965–975.
[8] J.C. Christensen, J.R. Estepp, G.F. Wilson, C.A. Russell, The effects of day-to-day variability of physiological data on operator functional state classification, Neuroimage 59 (1) (2012) 57–63, doi:10.1016/j.neuroimage.2011.07.091.
[9] M.X. Cohen, Analyzing Neural Time Series Data: Theory and Practice, The MIT Press, 2014.
[10] J.R. Comstock, R.J. Arnegard, The Multi-attribute Task Battery for Human Operator Workload and Strategic Behavior Research, NASA Technical Memorandum 104174, 1992. January
[11] M.L. Cummings, A. Clare, C. Hart, The role of human-automation consensus in multiple unmanned vehicle scheduling, Hum. Factors (2010).
[12] P.R. Davidson, R.D. Jones, M.T. Peiris, Eeg-based lapse detection with high temporal resolution, IEEE Trans. Biomed. Eng. 54 (5) (2007) 832–839.
[13] J.R. Estepp, S.L. Klosterman, J.C. Christensen, An assessment of non-stationarity in physiological cognitive state assessment using artificial neural networks, Proceedings of the Annual International Conference of the IEEE Engineering

in Medicine and Biology Society, EMBS 2011 (2011) 6552–6555, doi:10.1109/IEMBS.2011.6091616. URL: http://www.ncbi.nlm.nih.gov/pubmed/22255840
[14] F.A. Gers, J. Schmidhuber, F. Cummins, Learning to forget: continual prediction with lstm, Neural Comput. 12 (10) (2000) 2451–2471.
[15] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[16] A. Graves, Neural networks, in: Supervised Sequence Labelling with Recurrent Neural Networks, Springer, 2012, pp. 15–35.
[17] A. Graves, A.-r. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013, pp. 6645–6649.
[18] N.F. Güler, E.D. Übeyli, I. Güler, Recurrent neural networks employing Lyapunov exponents for EEG signals classification, Expert Syst. Appl. 29 (3) (2005) 506–514.
[19] R.G. Hefron, B.J. Borghetti, A new feature for cross-day psychophysiological workload estimation, in: Machine Learning and Applications (ICMLA), 2016 15th IEEE International Conference on, IEEE, 2016, pp. 785–790.
[20] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, arXiv:1207.0580(2012).
[21] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[22] B.M. Huey, C.D. Wickens, et al., Workload Transition: Implications for Individual and Team Performance, National Academies Press, 1993.
[23] D.W. Jahns, Operator workload: what is it and how should it be measured, in: K.D. Cross, J.J. McGrath (Eds.), Crew System Design. Proceedings of an Interagency Conference on Management and Technology in the Crew Systems Design Process, Santa Barbara, California, 1973.
[24] G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning, vol. 112, Springer, 2013.
[25] D. Kingma, J. Ba, Adam: a method for stochastic optimization, International conference on learning representations arXiv:1412.6980 (2014).
[26] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp. 1097–1105.
[27] S.P. Kumar, N. Sriraam, P. Benakop, B. Jinaga, Entropies based detection of epileptic seizures with artificial neural network classifiers, Expert Syst. Appl. 37 (4) (2010) 3284–3291.
[28] J.D. Lee, K.A. See, Trust in automation: Designing for appropriate reliance, Hum. Factors 46 (1) (2004) 50–80.
[29] Z.C. Lipton, J. Berkowitz, C. Elkan, A critical review of recurrent neural networks for sequence learning, arXiv:1506.00019 (2015).
[30] S.R. Liyanage, C. Guan, H. Zhang, K.K. Ang, J. Xu, T.H. Lee, Dynamically weighted ensemble classification for non-stationary eeg processing, J. Neural Eng. 10 (3) (2013) 036007. URL: http://stacks.iop.org/1741-2552/10/i=3/a=036007
[31] R. Parasuraman, T. Bahri, J.E. Deaton, J.G. Morrison, M. Barnes, Theory and Design of Adaptive Automation in Aviation Systems, Technical Report, DTIC Document, 1992.
[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (Oct) (2011) 2825–2830.
[33] V. Pham, T. Bluche, C. Kermorvant, J. Louradour, Dropout improves recurrent neural networks for handwriting recognition, in: Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on, IEEE, 2014, pp. 285–290.
[34] M.E. Raichle, A.Z. Snyder, A default mode of brain function: a brief history of an evolving idea, Neuroimage 37 (4) (2007) 1083–1090.
[35] W.B. Rouse, W.R. Cody, K.R. Boff, The human factors of system design: understanding and enhancing the role of human factors engineering, Int. J. Hum. Factors Manuf. 1 (1) (1991) 87–104, doi:10.1002/hfm.4530010108.
[36] V. Srinivasan, C. Eswaran, N. Sriraam, Approximate entropy-based epileptic eeg detection using artificial neural networks, IEEE Trans. Inf. Technol. Biomed. 11 (3) (2007) 288–295.
[37] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting., J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[38] Theano Development Team, Theano: a Python framework for fast computation of mathematical expressions, arXiv:1605.02688 (2016).
[39] E.D. Übeyli, Analysis of eeg signals by implementing eigenvector methods/recurrent neural networks, Digital Signal Process. 19 (1) (2009) 134–143.
[40] O. Vinyals, A. Toshev, S. Bengio, D. Erhan, Show and tell: a neural image caption generator, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3156–3164.
[41] G. Wilson, In-flight psychophysiological monitoring, Progr. Ambulatory Monit. (2001) 435–454.
[42] G.F. Wilson, Psychophysiological test methods and procedures, in: Handbook of Human Factors Testing and Evaluation, 2002, pp. 127–156.
[43] G.F. Wilson, C. Russell, J. Monnin, J. Estepp, J. Christensen, How does day-to-day variability in psychophysiological data affect classifier accuracy? in: Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol. 54, SAGE Publications Sage CA: Los Angeles, CA, 2010, pp. 264–268.
[44] G.F. Wilson, C.A. Russell, Real-time assessment of mental workload using psychophysiological measures and artificial neural networks., Hum. Factors 45 (4) (2003) 635–643, doi:10.1518/hfes.45.4.635.27088.
[45] G.F. Wilson, C.A. Russell, Performance enhancement in an uninhabited air vehicle task using psychophysiologically determined adaptive aiding., Hum. Factors 49 (6) (2007) 1005–1018, doi:10.1518/001872007X249875.
[46] W. Zaremba, I. Sutskever, O. Vinyals, Recurrent neural network regularization, arXiv:1409.2329 (2014).