This paper was accepted to the Intelligent Vehicles Symposium (IV), 2021, Nagoya, Japan.

Multivariate Time Series Analysis for Driving Style Classification using Neural Networks and Hyperdimensional Computing

Kenny Schlegel[1], Florian Mirus[2], Peer Neubert[1], and Peter Protzel[1]

***Abstract*— In this paper, we present a novel approach for driving style classification based on time series data. Instead of automatically learning the embedding vector for temporal representation of the input data with Recurrent Neural Networks, we propose a combination of Hyperdimensional Computing (HDC) for data representation in high-dimensional vectors and much simpler feed-forward neural networks. This approach provides three key advantages: first, instead of having a "black box" of Recurrent Neural Networks learning the temporal representation of the data, our approach allows to encode this temporal structure in high-dimensional vectors in a human-comprehensible way using the algebraic operations of HDC while only relying on feed-forward neural networks for the classification task. Second, we show that this combination is able to achieve at least similar and even slightly superior classification accuracy compared to state-of-the-art Long Short-Term Memory (LSTM)-based networks while significantly reducing training time and the necessary amount of data for successful learning. Third, our HDC-based data representation as well as the feed-forward neural network, allow implementation in the substrate of Spiking Neural Networks (SNNs). SNNs show promise to be orders of magnitude more energy-efficient than their rate-based counterparts while maintaining comparable prediction accuracy when being deployed on dedicated neuromorphic computing hardware, which could be an energy-efficient addition in future intelligent vehicles with tight restrictions regarding on-board computing and energy resources. We present a thorough analysis of our approach on a publicly available data set including a comparison with state-of-the-art reference models.**

I. INTRODUCTION

Assistance systems in modern series cars are becoming increasingly complex on the road to future intelligent vehicles. Particularly, reliable classification of driving styles, such as aggressive, inattentive or drowsy driving, in automotive context is a research area of growing interest for both, academic and industrial scientists. Potential applications for such classification systems span from Advanced Driver Assistance Systems (ADAS) such as driver distraction or drowsiness detection to increase driving safety over applications improving ride comfort and assessing the performance of the driver (e.g., economical driving) to insurance applications [1]. The majority of driving style classification systems uses the dynamics of the vehicle to detect the current driving style either relying on internal sensors available from the vehicle's CAN-bus [2] or from external devices such as smartphones [3]. Irrespective of the concrete input data, modern machine learning approaches such as Artificial Neural Networks (ANNs) offer an appealing tool set to tackle the complexity of classification problems such as driving style classification, especially since the number of available sensing modalities in future intelligent vehicles is likely to increase. Instead of using hand-crafted features of the input data fed into a prediction model (e.g., a neural network) classifying the driving style, Saleh et al [4], for instance, consider driving style classification a problem based on time series sensory input and therefore use an end-to-end learning approach employing Recurrent Neural Networks (RNNs), particularly LSTMs [5], to detect the current driving style. This approach is appealing since it avoids the necessity to manually design features, which is prone to human bias, and rather let the neural network extract those features into a temporal representation directly from the input data. On the other hand, this temporal representation of the input data results in an embedding in the form of an internal high-dimensional vector learned by the RNN, which is not comprehensible for human users anymore. Furthermore, LSTMs are rather complex learning models, which typically require long times as well as large amounts of data for successful training and additionally could be, due to their complexity, prohibitive in terms of in-vehicle deployment with respect to limited on-board resources.

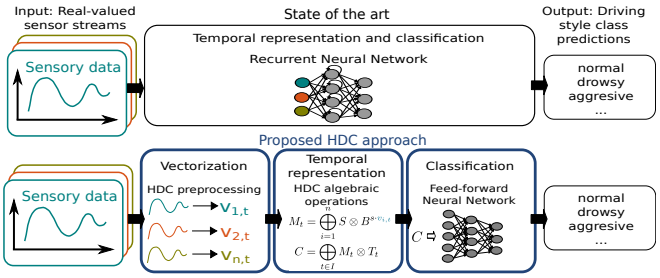

Fig. 1. Schematic visualization of the flow of information in a driving style classification system comparing our proposed approach (boxes highlighted through dark blue borders) with decoupled temporal representation and classification modules to an end-to-end learning approach employing RNNs.

To tackle some of the aforementioned issues, we propose a novel approach for driving style classification based on HDC [6], which refers to a family of modelling approaches representing entities of interest in high-dimensional vectors. One of the strengths of this approach is the ability to combine symbolic and numeric information in a common representational substrate while it also allows to impose structure by manipulating the representational vectors through algebraic operations. In contrast to [4], our modular approach employs HDC to encapsulate the time series input data into an embedding vector and thus decouples this step from the actual classification (see Fig. 1), which allows us to

[1]Kenny Schlegel, Peer Neubert, and Peter Protzel are with Chemnitz University of Technology name.surname@etit.tu-chemnitz.de

[2]Florian Mirus is with the department Research, New Technologies, Innovations at BMW Group, Munich, Germany

explore a plethora of (much simpler) prediction models such as feed-forward neural networks for the actual driving style classification. We evaluate our approach with several prediction models for classification on the publicly available UAH-DriveSet data set [3] and compare it to state-of-the-art models employing LSTMs [4] not only regarding prediction accuracy, but also in terms of training time, data efficiency and model complexity. Finally, we show that our simple feed-forward classification network as well as our HDC-based data representation can be translated into a SNN while maintaining similar results in terms of prediction accuracy compared to the rate-based ANN [7], [8]. Spiking neurons as computational substrate offer the opportunity to deploy our model on neuromorphic hardware, which could be an interesting addition to future intelligent vehicles promising to be orders or magnitude more energy-efficient than traditional CPU/GPU architectures [9].

To the best of our knowledge, this is the first time that these two worlds, Hyperdimensional Computing and Artificial Neural Networks, are combined in this or a similar way for multivariate time series analysis. We demonstrate the application to driving style classification. However, the presented approach is potentially applicable to many other prediction tasks with similar structure. To enable other researchers to reproduce and extend our results, we make the source code of our implementation publicly available[1].

II. RELATED WORK

A. Driving style classification

With the increasing focus on ADAS, driving style classification becomes more important in terms of safety and energy efficiency. The survey from [10] gives a broad overview of this research field, discusses the relevance of influencing factors (mainly divided by environmental or human factors) and the role of used sensors. Typical sensors are inertial measurement units (IMUs) or global navigation satellite systems (GNSS) [11]. Smartphones are well suited for data collection due to their low cost, multiple built-in sensors, and high processing capabilities [12]–[14]. The smartphone-based dataset from [13] is available, in addition, [15] presented a similar data recording only using an IMU. Romera et al. [3] provide a basis for further, more complex work in the field of driving style recognition by introducing a publicly available data set consisting of smartphone data and information about other vehicles and lane changes obtained from pre-processed camera images.

For supervised learning for driving style classification, Vaitkus et al. [2] use a simple hand-crafted statistical feature extraction (mean, median, etc.) from time series data in combination with a k-nearest neighbors (kNN) classifier. [16] uses a combination of K-means and Support Vector Machine (SVM) to explore the relevance of manoeuvres and recognize the driving style. Especially when the number of sensor modalities increases, the usage of Artificial Neural Networks (ANNs) can improve the driving style classification performance [10]. For instance, Zhang et al. [15] proposed an ANN with multiple convolutional layers to exploit the spatial-temporal context for classification. Beside such feed-forward networks, recurrent LSTM networks can be used to capture temporal features of multivariate time series [17]. A similar implementation of LSTMs for driving style classification was shown by [4], which classifies driving style based not only on IMU and GPS, but also on more diverse data such as number of vehicles or distance to the vehicle in front. However, recurrent LSTM models tend to suffer from high learning effort compared to feed-forward ANNs. Unfortunately, the convolutional model [15] without recurrent units requires even more training time than the LSTM models.

B. Hyperdimensional Computing (HDC)

Hyperdimensional Computing (HDC) (also known as Vector Symbolic Architectures) is an established class of approaches to solve numeric and symbolic computational problems using mathematical operations on large numerical vectors with thousands of dimensions [6], [18]–[20]. We provide a short introduction to HDC working principles in Sec. III-A. HDC has been applied in various fields including addressing catastrophic forgetting in deep neural networks [21], medical diagnosis [22], robotics [23], fault detection [24], analogy mapping [25], reinforcement learning [26], long-short term memory [27], text classification [28], and synthesis of finite state automata [29]. High-dimensional vectors are also the core representational element in most ANNs and a combination with HDC is straightforward. For example, [23] used HDC in combination with deep-learned descriptors for sequence encoding for localization or [30], [31] build semantic and spatial representations for visual place recognition based on HDC.

The usage of HDC or vector computation in general is still rather unusual in intelligent vehicles. One example of an abstract vector representation in an automotive context called Drive2Vec is presented in [32]: similar to word embedding approaches in natural language processing [33], Drive2Vec learns an embedding from a high-dimensional vector space (i.e., language or in this case several sensor streams) to a vector representation of comparably low dimension (representational space), which is subsequently used to make predictions about the vehicle's state. HDCs as representational substrate employing fractional binding [34] have been applied in automotive context to tasks such as driving context classification [35], vehicle trajectory prediction [36], [37], and anomaly detection [38]. There are several HDC implementations available, e.g., [18]–[20], [39]. A comparison is available in [6], a theoretical analysis is provided in [40].

III. APPROACH

Fig. 1 illustrates the structure of our proposed approach. Instead of performing all computations in a single recurrent neural network, we decouple the creation of vector representations of sensor values and the temporal encoding from the classification task. This allows to incorporate model

[1] https://github.com/TUC-ProAut/HDC_driving_style_classification

knowledge and to use simpler classifiers (e.g., a feed-forward network instead of a recurrent network). To avoid bias and limitations by manually created intermediate representations, we use high-dimensional distributed vector representations at all stages, very similar to the internal representations in artificial neural networks. To perform systematic computations on these vectors, we use Hyperdimensional Computing (HDC) [18] [19]. Sec. III-A will provide a short introduction to the relevant concepts and operations, i.e., bundling and (fractional) binding from this field. Fig. 3 illustrates how we use them for encoding multivariate time series data in a single vector representation (Sec. III-B). Finally, Sec. III-C provides details on the combination with classifiers for driving style recognition.

A. Hyperdimensional Computing and Fractional Binding

In this work, we adopt Holographic Reduced Representation in the Frequency Domain (FHRR) [19], which is one special case of HDC, a family of modeling approaches based on high-dimensional vector representations. This decision is based on the experimental comparison from [6] and the compatibility of FHRR vectors with fractional binding [34] for systematically encoding scalar sensor values to vectors. The resulting complex-valued high-dimensional vectors are one example of distributed representations in the sense that information is captured over all dimensions of the vector instead of one single number, which allows to encode both, symbol-like and numerical structures in a unified representational substrate. Although it is sometimes desirable to encode some sort of similarity when mapping entities to the representational vector space, it is also common to simply use random vectors to represent unrelated entities [18]. The enabling feature of high-dimensional vector spaces allowing random vectors as representational substrate is that two random vectors have a high probability (only depending on the dimension of the vector space) of being almost orthogonal and thus the chance of unintentionally confusing vectors representing different entities (such as different sensor types in Sec. III-B) is very low. Additionally, the algebraic operations of FHRR allow manipulation and combination of represented entities into structured representations. Those algebraic operations are *bundling* $\oplus$ and *binding* $\otimes$:

- **Bundling** $\oplus$ is used to store multiple input vectors in a set-like representation, where the result is a vector that is similar to each input vector. The implementation of the bundling $\oplus$ operator is an *element-wise addition* of the complex vector-values.
- **Binding** $\otimes$ is used to store variable-value pairs. The result of binding is dissimilar to each input vector, but each of the input vectors can be (approximately) restored. In case of the complex FHRR, the **binding** $\otimes$ operation is implemented as an *element-wise multiplication* of the complex values.

Komer et al. [34] define a **fractional binding** mechanism, which is originally implemented for real valued vectors $\mathbf{v} \in \mathbb{R}^D$ with dimensionality D as their power

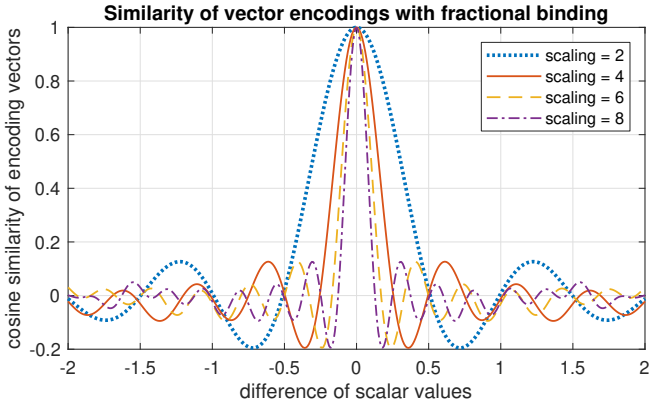

Fig. 2. Encoding similar scalar values with fractional binding results in high-dimensional vectors with high cosine similarity. A scaling parameter s can be used to influence the decay of the vector similarities.

$$\mathbf{v}^p := IDFT\left(\left(DFT_j\left(\mathbf{v}\right)^p\right)_{j=0}^{D-1}\right). \tag{1}$$

where DFT and IDFT denote the Discrete Fourier Transform and Inverse Discrete Fourier Transform respectively. This can be used to systematically encode a scalar p into a high-dimensional vector. "Systematically" means that similar scalar values (small euclidean distance) are encoded to similar vectors (small cosine distance). Fig. 2 shows the resulting vector similarities for varying differences of encoded scalar values. Since we use complex-valued vectors in the FHRR, it is not necessary to apply the Fourier Transform and Eq.(1) can be simplified to an element-wise exponentiation of a random complex-valued vector $\mathbf{c} \in \mathbb{C}^D$ by exponent p:

$$encoding(p) = \mathbf{c}^{s \cdot p} \tag{2}$$

A scaling value s can be used to set the similarity curve slope – the larger the scaling, the less similar the neighboring scalar-encoded vector is to the output vector (cf. Fig. 2). The visible oscillation is a result of the periodic behavior of exponentiation in the complex domain, see [34] for more details.

For efficient implementation, [19] suggests to restrict the vector space to complex numbers of magnitude 1. With this assumption of only selecting complex numbers on the unit circle, it is sufficient to use values in the range of $(-\pi, \pi]$ to define the angles of the complex values. Therefore, the complex vector $\mathbf{c} = (c_1, \cdots, c_D)$ with $c_j \in \mathbb{C}$ can be stored by using only the real vector of angles θ_j with $c_j = e^{i \cdot \theta_j}$. Calculating with angles instead of the whole complex number has the effect of simplifying the binding operation to element-wise addition and the fractional binding to element-wise multiplication instead of exponentiation. For bundling, however, the angles need to be converted into a complex number before addition.

B. Vector representation of time series sensor signals

Fig. 3 visualizes our approach of encoding a time series of sensor signal in the substrate of high-dimensional vectors using fractional binding. In the first step, we represent each

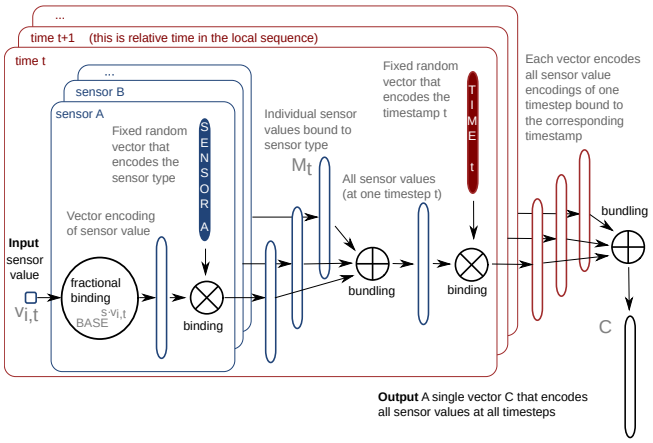

Fig. 3. Schema of context encoding with HDC operations. Inputs are (scalar) sensor values from different sensors at different time-steps. The blue part shows the encoding across multiple sensors, the red part across multiple time-steps. All shown vectors are from the same vector space. The output is a single vector that encodes all information and can, e.g, be fed to a (feed-forward neural network) classifier.

sensor type by a randomly chosen vector $\mathbf{SENSOR}_i$. Given values $v_{i,t}$ measured by sensor $\mathbf{SENSOR}_i$ for $i = 1, \cdots, n$ at time-step t, we use fractional binding to encode all sensor measurements into one representational vector by

$$\mathbf{M}_t = \bigoplus_{i=1}^{n} \mathbf{SENSOR}_i \otimes \mathbf{BASE}^{s \cdot v_{i,t}}, \tag{3}$$

where $\mathbf{BASE}$ is the initial base vector for the scalar-value encoding, which is also randomly chosen and the same for all sensor-types. The parameter s is the aforementioned scaling factor. In the second step, we again assign a randomly chosen vector $\mathbf{TIME}_t$ to represent each step t of the time series to be encoded. To encode the temporal context in high-dimensional vectors, we bind each measurement vector $\mathbf{M}_t$ with the time stamp vector $\mathbf{TIME}_t$ representing the time-step t, i.e., $\mathbf{M}_t \otimes \mathbf{TIME}_t$. Ultimately, bundling or summation of all vectors representing the sensor values at each time-step yield our final context vector, i.e.,

$$\begin{aligned} \mathbf{C} &= \bigoplus_{t \in I} \mathbf{M}_t \otimes \mathbf{TIME}_t \qquad (4) \\ &= \bigoplus_{t \in I} \left(\bigoplus_{i=1}^{n} \mathbf{SENSOR}_i \otimes \mathbf{BASE}^{s \cdot v_{i,t}} \right) \otimes \mathbf{TIME}_t, \end{aligned}$$

where I denotes the time interval, i.e., all time-steps t of the current sliding window. The result is a high-dimensional vector $\mathbf{C}$ containing all sensor values combined with the corresponding sensor type for every time-step in the current sliding window I. Finally, we standardize the resulting vectors per dimension by their mean and standard deviation, since [41] showed promising results in applying this approach.

C. Prediction Models

This section describes two prediction models, a simple feed-forward Artificial Neural Network and a Spiking Neural Network, that can be used to predict the driving style class using the high-dimensional vector C from the previous Sec. III-B.

1) Artificial Neural Networks (ANNs): We use an ANN in a supervised training setup to map the high-dimensional vector C to a driving style class. Since the different sensor types and the temporal context are already encoded in the input vector, a simple fully-connected feed-forward network with one hidden layer is sufficient. Although we work in the space of complex numbers, it should be noted that the vector C only stores the angles of the complex numbers on the unit circle. Therefore, it is a vector of real numbers, that can be processed by standard neural networks. Sec. IV-A provides implementation details.

2) Spiking Neural Networks (SNNs): In contrast to rate-based ANNs used in "traditional" machine learning, Spiking Neural Network (SNN) employ discrete spikes rather than average firing rates to encode information and are therefore often referred to as the third generation of neural networks. In theory [42], SNNs have at least the same computational power as traditional neural networks of similar size. One major hindrance for the widespread adoption of SNNs however, is that, due to the binary output of the spiking neurons, the neural network is not differentiable and thus standard training procedures such as backpropagation are not directly applicable. One option to circumvent this problem is to train the network with a differentiable approximation of the spiking neurons and replace this approximation with the actual spiking neurons during inference [7]. Spiking neurons as algorithmic substrate are appealing for tasks with tight restrictions regarding power consumption, particularly automotive applications, since they promise to be more energy-efficient when deployed on neuromorphic hardware. For our SNN in this paper, we adopt the ANN's architecture with some alterations in terms of hyper-parameters.

IV. EXPERIMENTS

We will first present the experimental setup including datasets, compared algorithms, as well as training and parameter settings. This is followed by evaluation of driving style classification performance, data efficiency, parameter influences, and runtime.

A. Experimental setup

1) UAH-DriveSet data set: Since we primarily compare our proposed method with the LSTM approach from [4], we use the same data set for experimental evaluation: Romera et al. [3] introduced a data set with more than 8 h driving scenes for driving behavior analysis and classification. The sensor values are collected using a smartphone with a monitoring application (shown in Fig. 4). Six different drivers generate data for two types of roads: secondary and motorway. All recorded sequences are labeled with one of three classes: normal, aggressive, or drowsy driving style.

The provided data contains raw sensor values from GPS and IMU, but also additional pre-processed data such as road lanes and vehicle detections from the smartphone camera. We use the same 9 different sensor values as [4]: IMU data

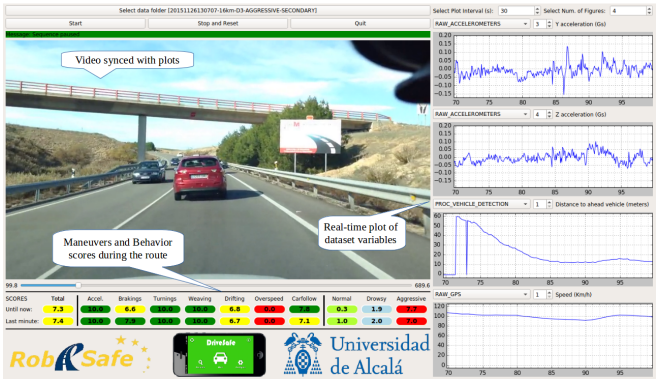

Fig. 4. Screenshot from the UAH-DriveSafe reader tool, which recorded the data with a smartphone. Image source [3].

for acceleration Acc_X, Acc_Y, Acc_Z, $Roll$, $Pitch$, and Yaw; $Speed$ from GPS; as well as $Distance\ to\ ahead\ vehicle$ and $number\ of\ detected\ vehicles$ obtained from the camera. We also use the same data preprocessing [4]: resampling to constant frequency followed by per-dimension standardization to mean 0 and standard deviation 1. To generate the sequence samples, Saleh et al. [4] used rolling windows with a size of 64 time-steps and 50 % overlap. The resulting total number of sequence samples is 9499. For training of the prediction models (HDC does not require training) we use the same random training and test split with a ratio of 2:1.

2) Parameters and training setup: The default setup for the HDC encoding uses 2048 dimensional vectors and scaling $s = 6$ in fractional binding. The ANN classifier uses 40 neurons in the hidden layer and 75 % dropout between the input and the hidden layer. These parameters result in a very similar number of parameters for ANN training compared to the LSTM [4] (cf. Table IV). An evaluation of these parameters will be topic of Sec. IV-D. The ANN was implemented in Keras and Tensorflow and was trained with a learning rate of 0.001, a cross-entropy-loss-function, the adam optimizer, and a batch size of 1500 for 2000 epochs.

Similar to our ANN model, the SNN's architecture embodies one hidden layer of neurons. In this work, we use the simple Leaky-Integrate-and-Fire (LIF) spiking neuron model [43]. Since networks of spiking neurons are not differentiable, we train the network with the rate approximation of the LIF neuron using back-propagation for 200 epochs with a batch size of 500, a drop-out rate of 50 %, the cross-entropy-loss function and the RMSprop optimizer with a learning rate of 0.001. We implemented our SNN in the Nengo simulator [44] and its extension Nengo-DL [8] integrating deep learning approaches from Tensorflow and replace the rate approximation with the actual LIF spiking neuron model during inference. Empirical evaluations similar to those of Table III showed that the SNN achieved its best performance with 1000 neurons in the hidden layer.

3) Evaluated approaches: The most related approach from the literature is the LSTM from [4] that directly operates on the raw sensor data (that is pre-processed as described in Sec. IV-A.1, but is still in the raw form of multivariate time series data). Fig. 5 provides an overview

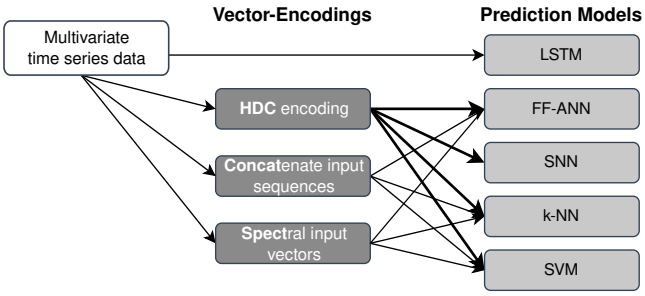

Fig. 5. This is an overview of the different evaluated combinations of vector-encodings with prediction models.

of all evaluated methods, which are different combinations of vector encodings and prediction models.

Encodings: We compare our HDC encoding approach with two alternative approaches to generate vector representations from multivariate time series data: *Concatenate Input Sequences* contain the raw input data from [4] but all sensor sequences are concatenated together (similar to time series processing in [45]) – hence, the outcome is a $64 \times 9 = 576$ dimensional vector; *Spectral feature vectors* are built with a discrete Fourier-transformation for each sequence and sensor-type, and the resulting magnitude of each frequency is concatenated to a final spectral feature vector.

Prediction models: Each of these vector encodings can be combined with different prediction models. Besides the *FF-ANN* and *SNN* described in Sec. III-C, we evaluate the following baseline models: *SVM* is a Support Vector Machine classifier. We use the standard Matlab implementation with an error-correcting output code for multi-class classification. *k-NN* is a simple k nearest neighbour classifier using a stored lookup table of all training sample data that outputs the most frequently occurring class of the k neighbours (similar to [2]).

4) Metric: We use a standard evaluation procedure based on ground-truth information (labels) about the corresponding class to a given sequence. The evaluation metric is weighted F1-Score for multiple classes computed by Python Scikit-learn's report function.

B. Driving style classification results

The main result of this paper is presented in Table I that shows driving style classification results on the UAH dataset. To account for randomness within ANNs, we repeat each experiment 10 times and report means and standard deviations. We observe that the combination of HDC with the simple ANN prediction model achieves the best results. The results are almost preserved when using a smaller number of HDC dimensions (reduction from 2048 to 1024) which also leads to significantly smaller number of parameters in the ANN (reduction from 82,083 to 41,123, cf. Table IV).

Note, that the performance of the SNN with the same number of neurons (40) in the hidden layer drops to 0.85 after conversion to spikes compared to the rate-based network. By increasing the number of neurons in the hidden layer to 1000 neurons, we are able to reach a comparable performance of 0.92, hence the SNN requires a much

TABLE I

DRIVING STYLE CLASSIFICATION RESULTS OF ALL MODELS (F_1 SCORE ON THE UAH TEST SET). BEST RESULTS ARE IN BOLD FONT. ALL LEARNED MODELS WERE RUN 10 TIMES - RESULTS ARE MEAN AND STANDARD DEVIATION.

vector encod.	prediction model	HDC dim.	UAH-DriveSet [3] Secondary	Motorway	Full
HDC (ours)	ANN (ours)	2048	**0.99** ±0.00	**0.94** ±0.00	**0.94** ±0.00
		1024	0.98	0.90	0.90
		576	0.97 ±0.00	0.90 ±0.00	0.89 ±0.00
	SNN (ours)	2048	**0.99** ±0.01	**0.94** ±0.00	0.92 ±0.02
		1024	0.93	0.88	0.91
	SVM	576	0.84	0.72	0.71
	kNN	576	0.93	0.91	0.90
(none)	LSTM	[4]	0.91 ±0.05	0.82 ±0.03	0.88 ±0.04
Concat	ANN		0.77 ±0.01	0.62 ±0.01	0.61 ±0.01
	SVM		0.51	0.35	0.30
	kNN		0.89	0.82	0.84
Spect	ANN		0.67 ±0.04	0.67 ±0.02	0.64 ±0.03
	SVM		0.70	0.71	0.61
	kNN		0.88	0.75	0.81

larger number of hidden neurons to achieve similar results. However, we expect that our SNN will provide significant improvements in terms of energy-efficiency on dedicated neuromorphic hardware such as Intel's Loihi system [46], where one chip is able to process up to $130,000$ spiking neurons with significantly lower power consumption than traditional computing hardware [9]

The Concat and Spect vector encodings each create a 576 dimensional vector. For a fair comparison, we also report the performance of the HDC approach when using the same number of dimensions in combination with the SVM and kNN prediction models. The results demonstrate the general potential of HDC encodings of multivariate time series data beyond the combination with neural networks.

The results from Table I were obtained using the exact same training and evaluation procedure as in [4]. In addition to the random train-test split from [4], we also evaluated a 3-fold cross-validation with three independent blocks of data. These three blocks contain the consecutive ordered driver records (one block contains at least one complete driver record). During cross-validation, we used two blocks for training and one for testing. Using these blocks ensures that there are previously unseen drivers in the test data. Results are reported in Table II. Although the HDC encoded vectors with the ANN achieves better classification results than the LSTM network, the overall results show that this is a considerably more challenging task than the random split from [4] since all models show a significant decrease in performance compared to the random splits from Table I. This indicates that the recorded sessions are either relatively dissimilar to each other or the number of training samples is too small to generalize to completely unseen sessions, e.g., an unknown driver.

C. Data efficiency

To evaluate the required amount of training data, we analyzed the data efficiency of the LSTM [4] model and our proposed HDC approach. We varied the training volume

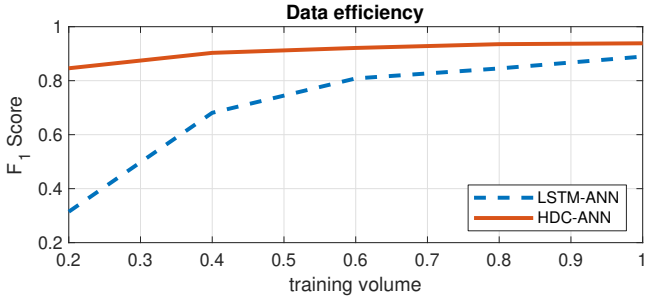

Fig. 6. Data efficiency of the original LSTM-ANN and the HDC-ANN (with 2048 dimensions, scaling of 6 and 40 neurons in the ANN hidden layer).

TABLE II

F_1-SCORES WITH 3-FOLD-CROSS VALIDATION ON THE UAH-DRIVESET [3].

vector encod.	prediction model	HDC Dim	Split 1	Split 2	Split 3
HDC (ours)	ANN	2048	**0.46** ±0.01	**0.54** ±0.01	**0.41** ±0.01
	SNN	2048	**0.46** ±0.01	**0.54** ±0.05	0.40 ±0.02
	kNN	576	0.45	0.45	0.41
(none)	LSTM	[4]	0.43 ±0.02	0.48 ±0.04	**0.41** ±0.03
Spect	kNN		0.42	0.41	0.40

between $20\,\%$ and $100\,\%$ of the full training set, the test set is unaltered. The results in Fig.6 show that the HDC-ANN is considerably more data efficient than the LSTM. This means that learning the internal time representation in this recurrent LSTM requires more training examples than generating the temporal structure through systematic application of HDC (that does not require training) and then using a simple feed-forward ANN.

D. Parameter Evaluation

The previous experiments used the default parameters as described in Sec. IV-A.2. Table III shows the influence on the classification performance when varying the number of dimensions, the scaling parameter in the fractional binding, and the number of neurons in the hidden layer of the ANN. The results indicate that the scale parameter of the fractional binding is the only parameter which is potentially hard to choose. Fortunately, the performance degrades smoothly when moving away from the sweet-spot of the default value 6. For the two other parameters (numbers of dimensions and neurons), increasing their values tends to improve the performance. However, this also changes the number of weights in the ANN as provided in Table IV. For the default setup of 2,048 dimensions and 40 neurons the number of weights is comparable to the LSTM [4].

E. Complexity and runtime

Table V shows runtime measures of the original LSTM model from [4] and the HDC models using a Nvidia GTX 1080Ti GPU. For the HDC approaches, we separately report runtime of the prediction model and of the HDC encoding during training process. In particular the training of the HDC model is significantly faster than that of the LSTM (more than factor 30). During inference, we build the HDC encoding and prediction model into one tensorflow graph to

TABLE III

RESULTS OF OUR HDC APPROACH WITH ANN PREDICTION MODEL AND DIFFERENT PARAMETER RANGES. DATA SET IS THE UAH-DRIVESET [3]. BEST RESULTS ARE HIGHLIGHTED IN BOLD.

	# Dimensions = 512					**# Dimensions = 1024**					**# Dimensions = 2048**				
# hidden neurons	20	40	60	80	100	20	40	60	80	100	20	40	60	80	100
scale = 2	0.83	0.84	0.85	0.85	0.86	0.84	0.87	0.87	0.88	0.88	0.88	0.9	0.9	0.9	0.9
scale = 4	0.85	0.88	0.88	0.88	0.88	0.89	0.91	0.9	0.91	0.91	0.91	0.93	0.93	0.93	0.92
scale = 6	0.86	0.89	0.89	0.89	0.89	0.9	0.91	0.93	0.92	0.93	0.93	**0.94**	**0.94**	**0.94**	**0.94**
scale = 8	0.85	0.87	0.88	0.88	0.88	0.9	0.91	0.92	0.93	0.92	0.93	0.93	**0.94**	**0.94**	**0.94**

TABLE IV

NUMBER OF TRAINABLE PARAMETERS FOR OUR HDC MODEL. ORIGINAL LSTM MODEL HAS 81,703 TRAINABLE PARAMETER.

# hidden Dim	20	40	60	80	100
# Dim. = 512	10,323	20,643	30,963	41,283	51,603
# Dim. = 1024	20,563	41,123	61,683	82,243	102,803
# Dim. = 2048	41,043	82,083	123,123	164,163	205,203

TABLE V

COMPUTING TIME ON PC WITH NVIDIA GEFORCE GTX 1080 TI ON THE UAH-DRIVESET. SECOND TERM IN HDC TRAINING REPRESENTS THE ENCODING TIME FOR THE CONTEXT VECTOR. INFERENCE TIMES FOR SNN ARE IN PARENTHESES SINCE THEY ARE COMPUTED BY SIMULATION ON A CONVENTIONAL GPU. $^{(1)}$ AND $^{(2)}$ MEANS DIFFERENT INFERENCE METHODS – SEE TEXT.

Model	Training on Train set [s]	Inference on Test set [s]	Overall [s]	Inference one sequence [ms]
LSTM [4]	2,166	0.455	2,166.455	$0.145^{(1)}$
	2,166	0.039	2,166.039	$0.013^{(2)}$
HDC-ANN	69 + 0.276	0.202	69.478	$0.065^{(1)}$
	69 + 0.202	0.092	69.294	$0.029^{(2)}$
HDC-SNN	16 + 0.276	(1.279)	(17.555)	$(0.408)^{(2)}$

be as fast as possible. Dependent on the setup, we come to two qualitatively different conclusions: (1) if we need to create the tensorflow session for each inference set, the HDC approach is twice as fast as the LSTM model, and (2) if we create the tensorflow session once and run it for each inference, the LSTM model is twice as fast as the HDC. The relevance of these two insights depends on a possible application: if a GPU is only dedicated to this specific classification task, the second strategy can be applied. If a GPU is used for multiple tasks, the first one becomes relevant and the HDC is faster than the LSTM. A straight forward way to further decrease the HDC-ANN runtime for strategy (2) is to use a smaller number of dimensions since this linearly scales the runtime complexity of the HDC encoding (the encoding runtime can be halved by using 1024-dimensional vectors at a small loss in accuracy, see Table I).

The SNN prediction model in combination with the HDC vectors requires less training time than the ANN (due to the smaller number of training epochs), but requires more time for inference. This is due to the spiking behaviour of the neurons: we simulate the neurons for some time to allow the network to converge to its result, which is why we put the measured inference times in Table V in parentheses. However, we expect the network to run much faster and more energy-efficient when being deployed on dedicated neuromorphic hardware [9].

V. CONCLUSIONS

This paper proposed a novel approach for driving style classification based on multivariate time series. We demonstrated that a recurrent neural network (LSTM) can be replaced by a feed-forward network through representing the temporal context within high-dimensional vectors based on HDC. The HDC approach not only achieved similar or slightly better classification results, but also significantly reduced training time and increased data efficiency. In addition, the proposed HDC model allows implementation in neuromorphic hardware based on SNNs, which promises high energy efficiency compared to traditional computer-hardware. We also observed that the standard data split on the UAH dataset used in [4] leads to significantly better results for all evaluated approaches compared to a split that ensures data of unknown drivers during test. Therefore, future work should include a more extensive evaluation on more diverse and larger datasets, including variations of sensor information and non-uniform sample rates. Testing an actual practical value of the compatibility of the HDC-SNN approach with actual neuromorphic hardware is also left for future work. We applied the proposed HDC encoding of multivariate time series data to driving style classification. However, in principle, this type of HDC encoding could potentially be applied to many other types of sensor streams and tasks. Particularly promising are applications where the input sensor data is already high dimensional, e.g., images (very likely pre-processed by a CNN, similar to the HDC localization approach in [23]).

REFERENCES

[1] G. A. M. Meiring and H. C. Myburgh, "A Review of Intelligent Driving Style Analysis Systems and Related Artificial Intelligence Algorithms," *Sensors*, vol. 15, no. 12, pp. 30 653–30 682, 2015.

[2] V. Vaitkus, P. Lengvenis, and G. Žylius, "Driving style classification using long-term accelerometer information," *2014 19th International Conference on Methods and Models in Automation and Robotics, MMAR 2014*, pp. 641–644, 2014.

[3] E. Romera, L. M. Bergasa, and R. Arroyo, "Need data for driver behaviour analysis? presenting the public UAH-DriveSet." in *ITSC*. IEEE, 2016, pp. 387–392.

[4] K. Saleh, M. Hossny, and S. Nahavandi, "Driving behavior classification based on sensor data fusion using LSTM recurrent neural networks," in *International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017.

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

[6] K. Schlegel, P. Neubert, and P. Protzel, "A comparison of Vector Symbolic Architectures," *Computing Research Repository (CoRR)*, vol. abs/2001.11797, 2020. [Online]. Available: https://arxiv.org/abs/2001.11797

[7] E. Hunsberger and C. Eliasmith, "Training Spiking Deep Networks for Neuromorphic Hardware," *Computing Research Repository (CoRR)*, vol. abs/1611.05141, 2016. [Online]. Available: http://arxiv.org/abs/1611.05141

[8] D. Rasmussen, "NengoDL: Combining Deep Learning and Neuromorphic Modelling Methods," *Neuroinformatics*, Apr 2019. [Online]. Available: https://doi.org/10.1007/s12021-019-09424-z

[9] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith, "Benchmarking Keyword Spotting Efficiency on Neuromorphic Hardware," *arXiv e-prints*, p. arXiv:1812.01739, 2018.

[10] C. Marina Martinez, M. Heucke, F. Y. Wang, B. Gao, and D. Cao, "Driving Style Recognition for Intelligent Vehicle Control and Advanced Driver Assistance: A Survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 666–676, 2018.

[11] V. Manzoni, A. Corti, P. De Luca, and S. M. Savaresi, "Driving style estimation via inertial measurements," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 777–782, 2010.

[12] M. M. Bejani and M. Ghatee, "Convolutional Neural Network with Adaptive Regularization to Classify Driving Styles on Smartphones," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 2, pp. 543–552, 2020.

[13] ——, "A context aware system for driving style evaluation by an ensemble learning on smartphone sensors data," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 303–320, apr 2018.

[14] T. K. Chan, C. S. Chin, H. Chen, and X. Zhong, "A comprehensive review of driver behavior analysis utilizing smartphones." *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 10, pp. 4444–4475, 2020.

[15] Y. Zhang, J. Li, Y. Guo, C. Xu, J. Bao, and Y. Song, "Vehicle Driving Behavior Recognition Based on Multi-View Convolutional Neural Network with Joint Data Augmentation," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4223–4234, 2019.

[16] M. Van Ly, S. Martin, and M. M. Trivedi, "Driver classification and driving style recognition using inertial sensors," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 1040–1045, 2013.

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

[18] P. Kanerva, "Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors." *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, 2009.

[19] T. A. Plate, "Distributed representations and nested compositional structure," Ph.D. dissertation, University of Toronto, Toronto, Ont., Canada, Canada, 1994.

[20] R. W. Gayler, "Vector Symbolic Architectures answer Jackendoff's challenges for cognitive neuroscience," in *Int. Conf. on Cognitive Science*, 2003.

[21] B. Cheung, A. Terekhov, Y. Chen, P. Agrawal, and B. A. Olshausen, "Superposition of many models into one." in *NeurIPS*, 2019.

[22] D. Widdows and T. Cohen, "Reasoning with Vectors: A Continuous Model for Fast Robust Inference," *Logic journal of the IGPL / Interest Group in Pure and Applied Logics*, no. 2, p. 141–173, 2015.

[23] P. Neubert, S. Schubert, and P. Protzel, "An introduction to hyperdimensional computing for robotics." *Künstliche Intell.*, vol. 33, no. 4, pp. 319–330, 2019.

[24] D. Kleyko, E. Osipov, N. Papakonstantinou, V. Vyatkin, and A. Mousavi, "Fault detection in the hyperspace: Towards intelligent automation systems," in *International Conference on Industrial Informatics (INDIN)*, 2015.

[25] D. A. Rachkovskij and S. V. Slipchenko, "Similarity-based retrieval with structure-sensitive sparse binary distributed representations," *Computational Intelligence*, vol. 28, no. 1, pp. 106–129, 2012.

[26] D. Kleyko, E. Osipov, R. W. Gayler, A. I. Khan, and A. G. Dyer, "Imitation of honey bees' concept learning processes using Vector Symbolic Architectures," *Biologically Inspired Cognitive Architectures*, vol. 14, pp. 57 – 72, 2015.

[27] I. Danihelka, G. Wayne, B. Uria, N. Kalchbrenner, and A. Graves, "Associative long short-term memory," *CoRR*, vol. abs/1602.03032, 2016. [Online]. Available: http://arxiv.org/abs/1602.03032

[28] D. Kleyko, A. Rahimi, D. A. Rachkovskij, E. Osipov, and J. M. Rabaey, "Classification and Recall With Binary Hyperdimensional Computing: Tradeoffs in Choice of Density and Mapping Characteristics," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 12, pp. 5880–5898, 2018.

[29] E. Osipov, D. Kleyko, and A. Legalov, "Associative synthesis of finite state automata model of a controlled object with hyperdimensional computing," in *Conference of the IEEE Industrial Electronics Society (IECON)*, 2017.

[30] P. Neubert and S. Schubert, "Hyperdimensional computing as a framework for systematic aggregation of image descriptors," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[31] P. Neubert, S. Schubert, K. Schlegel, and P. Protzel, "Vector semantic representations as descriptors for visual place recognition," in *Proc. of Robotics: Science and Systems (RSS)*, 2021.

[32] D. Hallac, S. Bhooshan, M. Chen, K. Abida, R. Sosic, and J. Leskovec, "Drive2Vec: Multiscale State-Space Embedding of Vehicular Sensor Data," in *21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3233–3238.

[33] J. Camacho-Collados and M. T. Pilehvar, "From word to sense embeddings: A survey on vector representations of meaning." *J. Artif. Intell. Res.*, vol. 63, pp. 743–788, 2018.

[34] B. Komer, T. C. Stewart, A. R. Voelker, and C. Eliasmith, "A neural representation of continuous space using fractional binding," in *41st Annual Meeting of the Cognitive Science Society*. Montreal, QC: Cognitive Science Society, 2019.

[35] F. Mirus, T. C. Stewart, and J. Conradt, "Towards cognitive automotive environment modelling: reasoning based on vector representations," in *26th European Symposium on Artificial Neural Networks, ESANN 2018, Bruges, Belgium*, 2018, pp. 55–60.

[36] F. Mirus, P. Blouw, T. C. Stewart, and J. Conradt, "An investigation of vehicle behavior prediction using a vector power representation to encode spatial positions of multiple objects and neural networks," *Frontiers in Neurorobotics*, vol. 13, p. 84, oct 2019.

[37] F. Mirus, T. C. Stewart, and J. Conradt, "The importance of balanced data sets: Analyzing a vehicle trajectory prediction model based on neural networks and distributed representations," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul 2020, pp. 1–8.

[38] ——, "Detection of abnormal driving situations using distributed representations and unsupervised learning," in *28th European Symposium on Artificial Neural Networks, ESANN 2020, Bruges, Belgium*, 2020.

[39] C. Eliasmith, "How to build a brain: from function to implementation." *Synthese*, vol. 159, no. 3, pp. 373–388, 2007.

[40] E. P. Frady, D. Kleyko, and F. T. Sommer, "A theory of sequence indexing and working memory in recurrent neural networks." *Neural Comput.*, vol. 30, no. 6, 2018.

[41] S. Schubert, P. Neubert, and P. Protzel, "Unsupervised Learning Methods for Visual Place Recognition in Discretely and Continuously Changing Environments," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2020, pp. 4372–4378.

[42] W. Maass, "Networks of Spiking Neurons: The Third Generation of Neural Network Models," *Neural Networks*, vol. 14, no. 4, pp. 1659–1671, 1997.

[43] E. M. Izhikevich, "Which Model to Use for Cortical Spiking Neurons?" *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1063–1070, 2004.

[44] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. C. Stewart, D. Rasmussen, X. Choo, A. R. Voelker, and C. Eliasmith, "Nengo: A Python tool for building large-scale functional brain models," *Frontiers in Neuroinformatics*, vol. 7, no. 48, 2014.

[45] J. Langner, J. Bach, L. Ries, S. Otten, M. Holzäpfel, and E. Sax, "Estimating the Uniqueness of Test Scenarios derived from Recorded Real-World-Driving-Data using Autoencoders," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1860–1866.

[46] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. H. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.