Benchmarking online sequence-to-sequence and character-based handwriting recognition from IMU-enhanced pens

Felix Ott · David Rügamer · Lucas Heublein · Tim Hamann · Jens Barth · Bernd Bischl · Christopher Mutschler

Received: 31 March 2022 / Revised: 2 June 2022 / Accepted: 1 September 2022 / Published online: 22 September 2022
© The Author(s) 2022

Abstract
Handwriting is one of the most frequently occurring patterns in everyday life and with it comes challenging applications such as handwriting recognition, writer identification and signature verification. In contrast to offline HWR that only uses spatial information (i.e., images), online HWR uses richer spatio-temporal information (i.e., trajectory data or inertial data). While there exist many offline HWR datasets, there are only little data available for the development of OnHWR methods on paper as it requires hardware-integrated pens. This paper presents data and benchmark models for real-time sequence-to-sequence learning and single character-based recognition. Our data are recorded by a sensor-enhanced ballpoint pen, yielding sensor data streams from triaxial accelerometers, a gyroscope, a magnetometer and a force sensor at 100 Hz. We propose a variety of datasets including equations and words for both the writer-dependent and writer-independent tasks. Our datasets allow a comparison between classical OnHWR on tablets and on paper with sensor-enhanced pens. We provide an evaluation benchmark for seq2seq and single character-based HWR using recurrent and temporal convolutional networks and transformers combined with a connectionist temporal classification (CTC) loss and cross-entropy (CE) losses. Our convolutional network combined with BiLSTMs outperforms transformer-based architectures, is on par with InceptionTime for sequence-based classification tasks and yields better results compared to 28 state-of-the-art techniques. Time-series augmentation methods improve the sequence-based task, and we show that CE variants can improve the single classification task. Our implementations together with the large benchmark of state-of-the-art techniques of novel OnHWR datasets serve as a baseline for future research in the area of OnHWR on paper.

Keywords Online handwriting recognition · Sequence-based and character datasets · Time-series data · Sensor-enhanced pen · Writer-(in)dependent · Architectures

1 Introduction

Handwriting provides language information based on structured symbols and is used for communication or documentation of speech. HWR refers to the digitalization of written text and can be categorized into offline and online HWR. Research on offline HWR systems is very advanced and has almost reached a human-level performance, but cannot be applied for real-time recognition applications (as they induce
an unacceptable delay) [23] as the written text has first to be
digitalized. Optical character recognition (OCR), one of the
dominant approaches in offline HWR, deals with the analysis
of the visual representation of handwriting only. Its appli-
cation and accuracy are limited as it cannot make use of
temporal information such as writing direction and speed, or
the pressure applied to the paper [65,69].

In contrast, online HWR typically works on different types
of spatio-temporal signals such as the positions of the pen tip
(in 2D), its temporal context or the movement on the writing
surface. These handwriting signals can then, e.g., be partitioned
into (indexed) strokes [96]. Compared to offline HWR, OnHWR
has its own merits, e.g., the difficult segmentation of
cursive written sequences into single characters. Many
highly relevant handwriting problems in everyday life require
both an informative representation of the writing and well-
working classification algorithms [35]. Examples include the
verification of signatures, the identification of writers, or the
recognition of handwriting.

The representation of written text crucially depends on the
way it has been recorded. Many recording systems make use of
a stylus pen (a touch pen with a sensible magnetic mesh
tip) together with a touch screen surface [2]. Systems for
writing on paper are only prototypical, such as the ones used
in [11,79,100] or the GyroPen [17] that provides a pen-like
interaction from standard built-in sensors in modern smartphones.
An advanced system for recording online HWR data was
proposed by Ott et al. [65] who use a sensor-enhanced
ballpoint pen that is extended with inertial measurement units
(IMUs). The hand movement and velocity patterns with such
a pen are different to air-writing [108]. In this paper, we
propose a novel dataset collection of equations and words
recorded with an IMU-enhanced pen. Using this pen allows
an online representation of the accelerations, orientations and
the pressure applied to the pen. Writing styles can thereby be
characterized by an information-rich multivariate time-series
(MTS). These datasets lay the foundation for HWR from
deps on whether the pen tip) together with a touch screen surface [2]. Systems for
writing on paper are only prototypical, such as the ones used
in [11,79,100] or the GyroPen [17] that provides a pen-like
interaction from standard built-in sensors in modern smartphones.
An advanced system for recording online HWR data was
proposed by Ott et al. [65] who use a sensor-enhanced
ballpoint pen that is extended with inertial measurement units
(IMUs). The hand movement and velocity patterns with such
a pen are different to air-writing [108]. In this paper, we
propose a novel dataset collection of equations and words
recorded with an IMU-enhanced pen. Using this pen allows
an online representation of the accelerations, orientations and
the pressure applied to the pen. Writing styles can thereby be
characterized by an information-rich multivariate time-series
(MTS). These datasets lay the foundation for HWR from
pens with integrated sensors [17,45,62–65,79,100,104], a so
far unsolved challenge in machine learning.

For machine learning tasks derived from online handwriting
data, we distinguish between single-label prediction
tasks (i.e., classifying characters, digits and symbols) and
tasks to predict sequences of labels (i.e., words, sentences
and equations). We here focus on the online seq2seq prediction
task for writer-dependent (WD) and writer-independent
(WI) classification, but also consider the single-label classi-
fication task for writer-dependent (WD) and writer-independent
and equations). We here focus on the online seq2seq predic-
tion task for both the writer-dependent and the writer-
independent case. Table 1 gives an overview of available
online datasets. For the single character prediction task,
the Kuchibue [56,57], MRG-OHTC [53], CASIA [98] and
OnHWR-chars [65] datasets are available. While the OnHWR-
chars dataset is rather small, we provide single character-
based datasets from a larger database. For our sequence-
based method (i.e., a technique that predicts a whole sequence
of characters), the IRONOFF [95], ICROW [78], IAM-
OnDB [51], LMCA [42], ADAB [1], IBM-UB [84] and
VNONDB [58,59] word and sentence datasets can be used.

The commonly used IAM-OnDB [51] and VNONDB [59]
datasets only include online trajectory data written on a tablet.
However, writing on even and smooth surfaces influences
the writing style of the user [28]. To circumvent this disad-
vantage, we initially recorded a small character-only dataset
| Dataset          | Content                               | Device              | Writers | Statistics                                      |
|------------------|----------------------------------------|---------------------|---------|-------------------------------------------------|
| Kuchibue [56,57] | Japanese characters                    | Tablet              | 120     | 10, 154 × 120 char. patterns                   |
| MRG-OHTC [53]    | Tibetan characters                     | Tablet              | 130     | 910 character classes                           |
| CASIA [98]       | Chinese characters                     | Anoto pen on paper  | 1020    | 3.5 m characters                                |
| OnHW-chars [65]  | English characters                     | Sensor pen          | 119     | 31275 characters, 52 classes                    |
| UNIPEN [33]      | Sentences, words, characters           | Pen-based computer  | –       | > 12,000 chars. per writer                      |
| CROHME [55]      | Mathematical expressions               | White-board, tablet | > 100   | 9507 expressions                                |
| IRONOFF [95]     | French words, chars., digits           | Trajectory, images  | –       | 50,000 words, 32,000 chars.                    |
| Sequence         |                                        |                     |         |                                                 |
| ICROW [78]       | Dutch, Irish, Italian words            | –                   | 67      | 13,119 words                                    |
| IAM-OnDB [51]    | English sentences                      | Whiteboard          | 197     | 82,272 words                                    |
| LMCA [42]        | Arabic words, chars., digits           | Tablet              | 55      | 30 k digits, 100 k chars., 500 w.               |
| ADAB [1]         | Arabic words                           | Tablet              | 170     | 20,000+ words                                   |
| IBM_UB_1 [84]    | English words                          | Notepad             | 43      | 6654 pages                                      |
| VNONDB [58,59]   | Vietnamese words, lines, paragr.       | Tablet              | 200     | 110,746 words                                   |
| OnHW-equations   | Equations written on paper             | Sensor pen          | 55      | 10,720 equations, 15 classes                    |
| Ours             |                                        |                     |         |                                                 |
| OnHW-words500    | Repeated 500 words on paper            | Sensor pen          | 53      | 25,218 words, 59 classes                        |
| OnHW-wordsRandom | Random words written on paper          | Sensor pen          | 54      | 14,645 words, 59 classes                        |
| OnHW-wordsTraj   | Words written on a tablet              | Sensor pen on tablet| 2       | 16,752 words, 52 classes                        |
| OnHW-symbols     | Numbers, symbols on paper              | Sensor pen          | 27      | 2326 characters, 15 classes                     |
with a sensor-enhanced pen on usual paper in previous work [65]. In this paper we make use of this novel pen and record sequence-based samples for a comparison and evaluation benchmark with the trajectory-based IAM-OnDB (line level) and VNONDB-words datasets. Hence, our datasets allow a broad research on sequence-based classification from sensor-enhanced pens and allow the connection between classical OnHW recognition on tablets with sensor-enhanced pens.

2.2 Methods

While hidden Markov models (HMMs) [3,8,18,19,22] have initially been applied to offline HWR, more recently, deep learning models became predominant, including convolutional neural networks (CNNs) [109], temporal convolutional networks (TCNs) [82,83], recurrent neural networks (RNNs) [20,31,68,70,85,105] including long short-term memory (LSTMs), bidirectional LSTMs (BiLSTMs) [12,91] and multidimensional RNNs [32,97]. More recently, attention models further improved the classification accuracy of RNNs [10], but did not outperform previous approaches for OnHW. Despite transformers [94] and its variants [13,36,38,44,90,101] got very popular for NLP [75] and image processing, these have so far only been applied to offline HWR [38]. The transformer by [66] is based on a language model and is used for Chinese text recognition. Similarly, variational autoencoders (VAEs), RNNs [29] and generative adversarial networks (GANs) [26] have been successfully applied for synthetic offline handwriting generation, but not for the online case so far. For the time-series classification task, standard convolutional architectures [25,34,72,103,113], spatio-temporal methods [6,15,21,39,40] and variants [24,87,89,99] as well as transformers [110] have been employed. In [65], we evaluated machine learning techniques, while in this paper we provide a broad evaluation benchmark on classical and novel time-series classification methods previously mentioned. While many approaches predict one class after the other, [14,54] predicted sequences similar to our approach. This is necessary to construct a suitable loss function described in the following. See Appendix 1 for a more detailed overview of related work.

**Loss functions** For sequence prediction the connectionist temporal classification (CTC) [30,31,43] loss combined with beam search [77] has extensively been used. The Edit distance (ED) [48] quantifies how dissimilar two strings are to one another by counting the minimum number of operations required to transform one string into the other. The ED allows deletion, insertion and substitution. However, the ED is a discrete function that is known to be hard to optimize. Ofitserov et al. [60] proposed a soft ED, which is a smooth approximation of ED that is differentiable. Seni et al. [80] used the ED for HWR. We use the CTC loss for sequence prediction (see Sect. 4).

3 Datasets and evaluation methodology

Our datasets are a collection of existing and newly generated online handwriting recordings. Section 3.1 first describes our recording setup to create novel and information-rich datasets. Section 3.2 gives details about the properties of our different OnHW datasets and compares them to existing datasets. Section 3.3 proposes a set of evaluation metrics.

3.1 Recording setup

Our datasets are recorded with a sensor-enhanced pen developed by STABILO International GmbH that contains two accelerometers at the front and the back (3 axes each), one gyroscope (3 axes), one magnetometer (3 axes) and one force sensor at 100 Hz (see Fig. 2). The data recordings contain 14 measurements provided by the sensors: four sensor data signals (each in x, y and z direction), the force with which the pen tip touches the surface, and the timestep at which the tablet receives the data from the pen. Figure 1 shows an exemplary sensor signal from a written equation. Using the force sensor the sensor data allow to separate strokes well as the writer lifts the pen between every character (this is not possible for cursive writing, e.g., for words). In total, we let 131 adult writers participate in our data collection. For more information on the sensor pen and data acquisition, see Appendices 2 and 3.

3.2 Datasets

We propose a large set of four different sequence-based datasets (see the first four entries in Table 2): the OnHW-words500 dataset was part of the UbiComp 2021 challenge and is written by 55 writers and consists of 10 number classes and 5 operator classes (+, −, ·, ÷, =). The dataset consists of a total of 10,713 samples. While in the OnHW-words500 dataset only the same 500 words per each writer appear, in the OnHW-wordsRandom dataset every sample is randomly chosen from a large German and English word list. This allows the comparison of indirectly learning a lexicon of 500 words or, alternatively, completely lexicon-free learning. The OnHW-wordsRandom dataset (14,641 samples) is smaller than the OnHW-words500 dataset (25,218 samples), but contains longer words with a maximal length of 27 labels (19 labels for OnHW-words500). The train/validation split for the OnHW-words500 dataset is based on words for the
Benchmarking online sequence-to-sequence and character-based handwriting recognition from IMU...

**Fig. 1** Exemplary sensor data for the x-, y- and z-axis of the equation: 20583:70

(a) Force.  (b) Front acc.  (c) Rear acc.  (d) Gyroscope.  (e) Magnetometer.

**Fig. 2** Sensor-enhanced pen

Table 2 Overview of our recordings from right-handed writers and state-of-the-art online handwriting datasets for writer-dependent (WD) and writer-independent (WI) tasks

| Dataset              | Number Writers | Number Classes | Maximal Length | Number samples | Total WD | WI | Total Chars |
|----------------------|----------------|----------------|----------------|----------------|----------|----|-------------|
| OnHW-equations       | 55             | 15             | 15             | 10,713         | 8595     | 2118| 8610        |
| OnHW-words500(R)     | 53             | 59             | 19             | 25,218         | 20,176   | 5042| 19,918      |
| OnHW-wordsRandom     | 54             | 59             | 27             | 14,641         | 11,744   | 2897| 11,716      |
| OnHW-wordsTraj       | 2              | 59             | 10             | 16,752         | 13,250   | 3502| –           |
| OnHW-symbols         | 27             | 15             | Single         | 2326           | 1853     | 473 | 1715        |
| ICROW [78]           | 67             | 53             | 15             | 13,119         | 10,500   | 2619| 10,524      |
| IAM-OnDB [51]        | 197            | 81             | 64             | 10,773         | 8702     | 2071| 8624        |
| VNOnDB-words [59]    | 201            | 147            | 11             | 110,746        | 88,677   | 22,069| 88,486      |
| OnHW-chars [65]      | 119            | 52             | Single         | 31,275         | 23,059   | 8216| 23,059      |

Table 3 Overview of our datasets from left-handed writers for writer-dependent (WD) and writer-independent (WI) tasks

| Dataset              | Number Writers | Maximal Length | Number Samples | Total WD | WI | Total Chars |
|----------------------|----------------|----------------|----------------|----------|----|-------------|
| OnHW-equations-L     | 4              | 15             | 843            | 677      | 166| 543         |
| OnHW-words500-L      | 2              | 19             | 1000           | 800      | 200| 500         |
| OnHW-wordsRandom-L   | 2              | 26             | 996            | 798      | 198| 497         |
| OnHW-symbols-L       | 4              | Single         | 361            | 289      | 72 | 271         |
| OnHW-chars-L [65]    | 9              | Single         | 2270           | 1816     | 454| –           |

For WD tasks a 80/20 train/validation split is used; for WI a dataset-specific split is used

**Fig. 3** Exemplary online samples of our OnHW-wordsTraj dataset (left: tablet, middle: camera) and the IAM-OnDB [51] (line level) dataset (right)
WD task such that the same 400 words per writer are in the train set and the same 100 words per writer are in the validation set. For the WI task, the split is done by writer such that all 500 words of a single writer are either in the train or validation set. As it is more likely to overfit on the same words, the WD task of OnHW-words500 is more challenging compared to the OnHW-wordsRandom dataset. The OnHW-words500R dataset is a random split of OnHW-words500.

Additionally, we record the OnHW-wordsTraj dataset that consists of four different data sources. We replace the ink refill with a Wacom EMR module and record online trajectories at 30 Hz on a Samsung Galaxy Tab S4 tablet along with the sensor data. Four cameras pointed on the pen to record the movement of the pen tip at 60 Hz. We manually label the pixels of 100 random images of the recorded videos in the classes “pen”, “pen tip” and “background” and train U-Net [76] to predict the pen tip pixels from all images. We manually label the pixels of 100 random images of the recorded videos in the classes “pen”, “pen tip” and “background” and train U-Net [76] to predict the pen tip pixels from all images.

Datasets for single character classification For the OnHW-equations dataset, it is possible to split the sensor sequence based on the force sensor as the pen is lifted between every single character. This approach provides another useful dataset for a single character classification task. We set split constraints for long tip lifts and recursively split these sequences by assigning a possible number of strokes per character. This results in a total of 39,643 single characters. Furthermore, we recorded the OnHW-symbols dataset with the same labels (numbers 0 to 9 and operators +, −, \cdot, \div, =), written by 27 writers and a total of 2326 single characters. Figure 5 compares the distribution of sample numbers for the OnHW-chars [65] (characters) and OnHW-symbols as well as split OnHW-equations (numbers, symbols) datasets. While the samples are equally distributed for small and capital characters (~600 per character), the numbers and symbols are unevenly distributed for the split OnHW-equations dataset (similar to Fig. 4b).

### 3.3 Evaluation metrics

We define a set of task-specific seq2seq and single character-based evaluation metrics that are commonly used in the community. Metrics for seq2seq evaluation are the character error rate (CER) and word error rate (WER) that are based on the Edit distance (ED). The ED is the minimum number of substitutions \( S \), insertions \( I \) and deletions \( D \) required to change the sequences \( f \) into \( g \) with lengths \( r \) and \( n \), respectively. The ED is defined by

\[
ED_{i,j} = \begin{cases} 
ED_{i-1,j-1} & \text{for } f_i = g_j \\
\min \left( \begin{array}{ll}
ED_{i-1,j} + D(f_i) & \text{for } f_i \neq g_j \\
ED_{i,j-1} + I(g_j) & \text{for } f_i = g_j
\end{array} \right) 
\end{cases}
\]

for \( 1 \leq i \leq r, 1 \leq j \leq n \), \( ED_{0,0} = \sum_{k=1}^r D(f_k) \) for \( 1 \leq i \leq r \), and \( ED_{0,j} = \sum_{k=1}^r I(g_k) \) for \( 1 \leq j \leq n \) [16]. We define the CER as \( S_w + I_w + D_w \) as the ED, the sum of character substitutions \( S_w \), insertions \( I_w \) and deletions \( D_w \), divided by the total number of characters in the set \( N_c \). Similarly, the WER as \( S_w + I_w + D_w \) is computed with word operations \( S_w \), \( I_w \) and \( D_w \) and number of words in the set \( N_w \) [38]. For single character evaluation, we use the character recognition rate (CRR) that is the number of correctly classified characters divided by the total number of characters in the test set.

### 4 Benchmark methods

This section formally defines the seq2seq classification task and our loss functions. We propose our architecture for HWR from IMU-enhanced pens and describe our data augmentation techniques.

**Sequence-based classification task** An MTS \( U = \{u_1, \ldots, u_m\} \in \mathbb{R}^{m \times l} \) is an ordered sequence of \( l \in \mathbb{N} \) streams with \( u_i = (u_{i,1}, \ldots, u_{i,l}), i \in \{1, \ldots, m\} \), where \( m \) is the length...
of the time-series that is variable and \( l \) is the number of dimensions. Each MTS is associated with \( v \), a sequence of \( L \) class labels from a pre-defined label set \( \Omega \) with \( K \) classes. For our classification task, \( v \in \Omega^L \) describes words and equations. The training set is a subset of the array \( U = \{ U_1, \ldots, U_n \} \in \mathbb{R}^{n \times m \times l} \), where \( n \) is the number of time-series, and the corresponding labels \( V = \{ v_1, \ldots, v_n \} \in \Omega^{n \times L} \). The aim of the MTS classification task is to predict an unknown class label for a given MTS. We train the classifier using the loss \( L_{CTC}(U, V) \) [30].

Character-based classification task In contrast to the sequence-based classification task, corresponding labels \( V \) for the character-based classification task are of length \( L = 1 \). We define \( p(i|u) \) to be the predicted probability for the \( i \)th class and \( q(i|u) \) to be the true class distribution. We train the classifier using the cross-entropy loss and variants against overconfidence and class imbalance [50,67,73,88,102,112].

Sequence-based loss The CTC loss is a solution to avoid pre-segmentation of the training samples. The key idea of CTC is to transform the network outputs into a conditional probability distribution over label sequences. An intermediate label representation allows repetitions of labels and occurrences of blank labels to identify no output label. Hence, the network with the CTC loss has a softmax output layer with one more unit than there are labels. These outputs define the probabilities of all possible ways to align all label sequences with the input sequence. [30]

Character-based losses We use the categorical cross-entropy (CCE) loss defined by

\[
\mathcal{L}_{CCE}(U, V) = -\frac{1}{K} \sum_{i=1}^{K} q(i|u) \log p(i|u) \tag{2}
\]

for model training. Samples with softmax outputs that are less congruent with provided labels are implicitly weighted more than confident sample predictions (more emphasis is put on difficult samples with CCE). Hence, more emphasis is put on difficult samples, which can cause overfitting to noisy labels [102,112]. To account for this imbalance, we modify the CCE loss such that it down-weights the loss assigned to well-classified examples. We use the Focal loss (FL) [50] defined by

\[
\mathcal{L}_{FL}(U, V, \alpha, \gamma) = -\frac{1}{K} \sum_{i=1}^{K} \alpha (1 - p(i|u))^\gamma q(i|u) \log p(i|u), \tag{3}
\]

with class balance factor \( \alpha \in [0, 1] \), and the modulating factor \((1 - p(i|u))^\gamma\) with focusing parameter \( \gamma \geq 0 \). As alternative, we apply label smoothing (LSR) [67] that prevents overconfidence by applying a confidence penalty through a regularization term, yielding

\[
\mathcal{L}_{LSR}(U, V, \beta) = -\frac{1}{K} \sum_{i=1}^{K} \log p(i|u) - \beta H(p(i|u))
\]
Fig. 6 Overview of our CNN architecture in combination with a (Bi)LSTM or a TCN

\[ L_{SBS}(U, V, \beta) = - \frac{1}{K} \sum_{i=1}^{K} \left[ \beta q(i|u) + (1 - \beta) p(i|u) \right] \log p(i|u), \]  

(5)

for predicted class probabilities \( p \) with weighting parameter \( \beta \), while the hard bootstrapping loss (HBS)

\[ L_{HBS}(U, V, \beta) = - \frac{1}{K} \sum_{i=1}^{K} \left[ \beta q(i|u) + (1 - \beta) z_i \right] \log p(i|u) \]  

(6)

uses the maximum a posteriori (MAP) estimation of \( p \) given \( u \), with \( z_i := 1[i = \arg \max q_l, l = 1, \ldots, K] \). MAP treats every sample equally for a higher robustness against noisy labels. This can lead to longer training times to reach convergence and can make optimization more difficult [112]. The generalized cross-entropy (GCE) [112] loss

\[ L_{GCE}(U, V, \alpha) = - \frac{1}{K} \sum_{i=1}^{K} \frac{1 - p(i|u)\alpha}{\alpha} \]  

(7)

with \( \alpha \in (0, 1] \) uses a negative Box-Cox transformation to combine benefits of the MAP and the CCE. The symmetric cross-entropy (SCE) [102] is

\[ L_{SCE}(U, V, \alpha, \beta) = \alpha L_{CCE}(U, V) + \beta L_{RCE}(U, V) \]  

(8)

based on the reverse cross-entropy (RCE) loss

\[ L_{RCE}(U, V) = - \frac{1}{K} \sum_{i=1}^{K} p(i|u) \log q(i|u), \]  

(9)

aims for a more effective and robust learning, where \( \alpha \) mitigates the overfitting of CCE and \( \beta \) allows for flexible exploration of the RCE. Furthermore, we make use of the joint optimization (JO) [88], which overcomes the noisy labels problem by learning network parameters and labels jointly. The loss is defined by

\[ L_{JO}(\Theta, \nu|U, \alpha, \beta) = L_{CCE}(\Theta, \nu|U) + \alpha L_{p}(\Theta|U) + \beta L_{e}(\Theta|U) \]  

(10)

with regularization losses \( L_p \) and \( L_e \), and network parameters \( \Theta \).

Architectures We propose two different architectures for seq2seq sensor signal classification. For the first method (see Fig. 6), a convolution block consisting of 1D convolutions (200 filter, kernel size 4), max pooling (pool size 2), batch normalization and dropout (with rate 0.2) layers is used. One TCN layer of 100 units, one LSTM layer of 100 units or two BiLSTM layers, each with 60 units, follow to extract the temporal context [74]. While we use \( \tanh \) activations for BiLSTM layers, we choose ReLU for the TCN and LSTM layers. A dense layer with 100 units with the CTC loss predicts a sequence of class labels.
labels. Second, we implement an attention-based network (see Fig. 7) that consists of an encoder with batch normalization, 1D convolutional and (Bi)LSTM layers. These maps the input sequence $U \in \mathbb{R}^{m \times l}$ to a sequence of continuous representations $z$. A transformer transforms $z$ using $n_{\text{head}}$ stacked multi-head self-attention $\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, \ldots, head_{n_{\text{head}}})W^O$ with $W^O \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$. The attention consists of point-wise, fully connected time-distributed layers followed by a scaled dot product layer and layer normalization [5] with $d_{\text{model}}$ output dimension [94]. $head_i = \text{Attention}(QW^Q_i, KW^K_i, VW^V_i)$, where $W^Q_i, W^K_i \in \mathbb{R}^{d_{\text{model}} \times d_k}$ and $W^V_i \in \mathbb{R}^{d_{\text{model}} \times d_k}$. The attention can be described as mapping a set of key-value pairs of dimension $d_k$ and a query of dimension $d_k$ to an output, and is computed by $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$. The matrices $Q, K$ and $V$ are a set of queries, keys and values.

**Data augmentation** As the size of the datasets is limited, data augmentation is a critical pre-processing step for networks to prevent overfitting and improve generalization. However, it is not obvious how to carry out label-preserving augmentation in some domains, i.e., scaling of acceleration signals [93]. We apply the following different data augmentation methods for wearable sensor data on each sensor channel at 50% probability. **Time warping** perturbs the temporal location by smoothly distorting the time intervals between samples that, e.g., simulates different sampling rates through time shifts of the connection between device and tablet. **Scaling** changes the magnitude of the data in a window by multiplying by a random scalar $\sigma = \pm 0.1$ that augments drifts in the signals. **Shifting** adds a random value $\alpha = \pm 20$ to the force data and $\alpha = \pm 20$ to the other sensor data. While **jittering** is a way of simulating additive sensor noise by adding a multiple $\sigma = \pm 0.1$ of the standard deviation to all sensor channels, **magnitude warping** changes the magnitude by convolving the data window with a smooth curve varying around $[0.7, 1.3]$ (only for the accelerometer data). For time and magnitude warping, the data are augmented by Bézier curves in the interval $[1 - \sigma, 1 + \sigma]$ that are generated based on 10 random points. As one sample is represented by a sequence of characters and a sample cannot be split into sub-sequences, applying cropping and permutation augmentation is not possible. Figure 8 zooms into the augmented sensor data of the x-axis signal from Fig. 1.

### 5 Experiments

This section provides evaluation results for the seq2seq (Sect. 5.1) and the single character-based classification task (Sect. 5.2), and evaluates left-handed datasets (Sect. 5.3). We propose a writer-dependent evaluation in Sect. 5.4.

**Hardware and training setup** For all experiments we use Nvidia Tesla V100-SXM2 GPUs with 32 GB VRAM equipped with Core Xeon CPUs and 192 GB RAM. We use the Adam optimizer with a learning rate of $10^{-4}$. We run each experiment for 1000 epochs with a batch size of 50 (unless stated differently) and report results for the best epoch. We split each dataset into five approx. 80/20 train/validation splits and report the mean and standard deviation of the WER and CER. We use our OnHW-equations, OnHW-words500(R), OnHW-wordsRandom and OnHW-wordsTraj as well as the IAM-OnDB [51] and VNonDB-words [59] datasets for the sequence-based classification task, and the OnHW-symbols, split OnHW-equations and OnHW-chars [65] datasets for the single character-based classification task. Each model is trained from scratch for every dataset. We make use of the time-series classification toolbox tsai [61] that contains a variety of state-of-the-art techniques [6,15,24,25,34,39,72,87,89,99,103,110,113].

#### 5.1 Seq2seq task evaluation

**Method and architecture evaluation** We first evaluate our CNN and attention-based models for the seq2seq classification task. A summary of results is given in Table 4. For all datasets our CNN+BiLSTM model significantly outperforms the CNN+LSTM and CNN+TCN models. The attention-based model performs poorly on large datasets (OnHW-equations, words500(R), wordsRandom), but yields better

---

**Fig. 8** Data augmentation methods of the original sensor sample (black)
Table 4 Evaluation results (WER, CER) in % (mean and standard deviation) for our OnHW-equations, OnHW-words500(R), OnHW-wordsRandom and OnHW-wordsTraj writer-dependent (WD) and writer-independent (WI) datasets, and the publicly available IAM-OnDB [51] (line level) and VNOnDB-words [59] datasets

| Method                  | Metric | OnHW-equations | OnHW-words500(R) | OnHW-wordsRandom | OnHW-wordsRandom |
|-------------------------|--------|----------------|------------------|------------------|------------------|
|                         |        | WD  WER | CER  | WI  WER | CER  | WD  WER | CER  | WI  WER | CER  | WD  WER | CER  | WI  WER | CER  |
| CNN+LSTM                | Mean   | 22.96 | 3.50 | 69.22 | 18.11 | 80.70 | 28.41 | 93.30 | 48.24 | 76.80 | 23.73 | 82.29 | 17.90 | 93.90 | 46.92 |
|                         | STD    | 1.83  | 0.38 | 7.91  | 5.20  | 3.32 | 2.50  | 1.13 | 4.95  | 0.34 | 0.23  | 8.49  | 1.66  | 6.00  | 2.88  |
| CNN+BiLSTM              | Mean   | 13.19 | 1.78 | 55.25 | 12.98 | 51.95 | 17.16 | 60.91 | 27.80 | 18.77 | 5.20 | 41.27 | 7.87 | 84.52 | 35.22 |
|                         | STD    | 0.52  | 0.13 | 10.56 | 5.23  | 12.72 | 4.98  | 5.16 | 5.97  | 0.87 | 0.31 | 1.18  | 0.35 | 7.53  | 5.07  |
| CNN+TCN                 | Mean   | 28.57 | 4.29 | 82.06 | 23.95 | 63.51 | 21.07 | 90.54 | 49.53 | 62.61 | 19.13 | 83.16 | 19.26 | 96.46 | 51.42 |
|                         | STD    | 1.16  | 0.23 | 6.14  | 4.44  | 11.81 | 4.37  | 5.56 | 7.93  | 12.03 | 3.90 | 7.82 | 2.42 | 3.14  | 3.73  |
| Attention-based model    | Mean   | 73.69 | 16.45 | 87.48 | 27.45 | 88.34 | 45.70 | 83.53 | 42.42 | 78.53 | 35.05 | 96.33 | 42.14 | 98.39 | 52.23 |
|                         | STD    | 2.55  | 1.04 | 2.19  | 2.33  | 1.74  | 1.46  | 2.42 | 5.21  | 2.12 | 1.96 | 1.73 | 0.32 | 0.37  | 3.00  |
| InceptionTime [25]      | Mean   | 20.72 | 2.92 | 60.24 | 14.71 | 41.92 | 12.08 | 76.84 | 35.07 | 40.18 | 11.39 | 63.04 | 12.81 | 89.18 | 39.59 |
| (32, 6)                 | STD    | 0.58  | 0.18 | 8.81  | 4.75  | 2.38 | 0.64 | 2.92 | 5.63 | 0.62 | 0.24 | 0.99 | 0.21 | 7.87  | 3.72  |
| InceptionTime [25]      | Mean   | 19.48 | 2.72 | 60.90 | 14.29 | 53.34 | 16.24 | 78.22 | 36.85 | 47.52 | 13.87 | 65.68 | 13.63 | 89.84 | 41.81 |
| (32, 6) +BiLSTM        | STD    | 0.29  | 0.13 | 7.87  | 4.61  | 4.34 | 0.71 | 3.53 | 6.53 | 2.02 | 0.83 | 1.31 | 0.36 | 8.17  | 3.38  |
| InceptionTime [25]      | Mean   | 12.94 | 1.77 | 52.40 | 12.23 | 37.12 | 12.96 | 62.09 | 26.36 | 21.34 | 5.34 | 42.88 | 7.19 | 84.14 | 32.35 |
| (96, 11)                | STD    | 0.33  | 0.12 | 8.09  | 4.71  | 2.11 | 0.55 | 5.66 | 2.21 | 0.56 | 0.20 | 1.27 | 0.25 | 8.13  | 3.75  |
| InceptionTime [25]      | Mean   | 12.06 | 1.65 | 49.92 | 11.28 | 43.22 | 13.07 | 61.62 | 26.08 | 21.18 | 5.35 | 39.14 | 6.39 | 85.42 | 33.31 |
| (96, 11) +BiLSTM        | STD    | 0.32  | 0.10 | 7.78  | 4.20  | 2.93 | 0.79 | 5.39 | 6.27 | 0.84 | 0.26 | 0.83 | 0.13 | 7.32  | 4.32  |
| XceptionTime [72]       | Mean   | 38.66 | 5.67 | 71.06 | 17.52 | 49.10 | 15.07 | 78.54 | 36.80 | 45.84 | 13.81 | 69.20 | 15.60 | 89.74 | 41.34 |
| (144)                   | STD    | 0.80  | 0.20 | 5.70  | 4.56 | 2.79 | 0.57 | 3.55 | 6.14 | 0.48 | 0.14 | 0.55 | 0.21 | 8.05  | 3.25  |
| XceptionTime [72]       | Mean   | 38.40 | 5.71 | 70.56 | 17.47 | 51.62 | 16.24 | 80.00 | 38.06 | 46.44 | 14.26 | 71.74 | 16.77 | 90.92 | 44.43 |
| (144) +BiLSTM          | STD    | 1.14  | 0.21 | 5.07  | 4.32 | 4.00 | 1.37 | 2.96 | 5.55 | 0.45 | 0.11 | 0.72 | 0.31 | 7.91  | 3.59  |
| ResNet [103]            | Mean   | 39.36 | 5.78 | 87.10 | 27.56 | 90.30 | 44.23 | 95.90 | 58.61 | 77.02 | 27.64 | 92.50 | 27.37 | 93.00 | 59.52 |
| (144)                   | STD    | 2.44  | 0.61 | 4.77  | 4.10 | 7.29 | 13.13 | 0.95 | 3.35 | 4.28 | 3.38 | 0.53 | 0.38 | 8.29  | 4.95  |
| ResNet [103]            | Mean   | 37.50 | 5.50 | 84.02 | 25.84 | 79.54 | 28.19 | 96.66 | 59.76 | 79.04 | 28.16 | 91.36 | 25.31 | 92.84 | 57.57 |
| (144) +BiLSTM          | STD    | 3.10  | 0.59 | 9.33  | 5.97 | 3.11 | 1.51 | 0.34 | 2.58 | 0.48 | 0.37 | 0.84 | 0.97 | 8.36  | 4.46  |
| ResCNN [113]            | Mean   | 81.92 | 18.20 | 98.50 | 45.59 | 94.42 | 48.01 | 98.92 | 70.68 | 92.32 | 44.81 | 98.68 | 41.78 | 93.14 | 68.26 |
| (144)                   | STD    | 1.29  | 0.79 | 0.87  | 4.60 | 1.57 | 2.04 | 0.16 | 2.08 | 1.89 | 2.96 | 0.17 | 0.74 | 8.40  | 5.82  |
| Method                  | Metric | OnHW-equations | OnHW-words500(R) | OnHW-wordsRandom |
|-------------------------|--------|----------------|------------------|------------------|
|                         | WD     | WI             | WD               | WI               | Random           | WD               | WI             | CER   | WD               | WI             | CER   |
| ResCNN [113]            | Mean   | 87.66          | 23.24            | 99.54            | 51.77            | 94.56            | 48.59           | 98.86 | 70.06            | 93.80          | 45.59 |
|                         | STD    | 2.33           | 1.77             | 0.43             | 4.67             | 1.41             | 1.77            | 0.33  | 1.57             | 0.49           | 1.84  |
| LSTM-[103]              | Mean   | 91.62          | 24.66            | 99.46            | 53.84            | 96.82            | 54.89           | 99.34 | 75.46            | 96.74          | 54.58 |
|                         | STD    | 0.92           | 1.04             | 0.37             | 0.73             | 0.66             | 0.88            | 0.14  | 2.80             | 0.14           | 0.55  |
| GRU-FCN [39]            | Mean   | 90.82          | 24.47            | 99.44            | 52.49            | 96.18            | 52.53           | 99.48 | 76.94            | 95.82          | 51.50 |
|                         | STD    | 1.40           | 1.44             | 0.40             | 3.96             | 1.06             | 1.52            | 0.07  | 1.80             | 0.51           | 1.33  |
| MLSTM-FCN [40]          | Mean   | 89.12          | 23.03            | 99.32            | 52.01            | 96.78            | 55.11           | 99.46 | 76.05            | 96.66          | 54.32 |
|                         | STD    | 1.53           | 1.08             | 0.53             | 3.93             | 0.93             | 1.55            | 0.10  | 1.40             | 0.51           | 1.43  |
| CNN+LSTM                | Mean   | 87.18          | 21.75            | 99.28            | 48.82            | 98.46            | 70.02           | 99.30 | 77.03            | 97.66          | 63.19 |
|                         | STD    | 1.67           | 0.96             | 0.35             | 3.68             | 2.08             | 10.30           | 0.11  | 1.70             | 1.89           | 10.21 |
| CNN+TCN                 | Mean   | 88.64          | 22.50            | 99.34            | 50.56            | 96.80            | 55.22           | 99.4  | 74.34            | 96.16          | 53.02 |
|                         | STD    | 0.99           | 0.90             | 0.60             | 4.43             | 0.89             | 2.05            | 0.11  | 2.21             | 0.64           | 1.26  |
| Attention-based model    | Mean   | 60.50          | 14.93            | 57.00            | 8.95             | 61.10            | 10.66           | 83.14 | 11.23            | 84.56          | 12.96 |
|                         | STD    | 3.03           | 0.95             | 8.74             | 2.50             | 1.30             | 0.65            | 1.66  | 1.35             | 13.77          | 7.57  |
| InceptionTime [25]      | Mean   | 64.00          | 16.10            | 67.47            | 11.40            | 69.40            | 23.94           | 87.07 | 12.97            | 87.18          | 14.32 |
|                         | STD    | 7.12           | 4.80             | 16.61            | 27.28            | 3.00             | 1.49            | 2.91  | 1.74             | 8.43           | 3.57  |
| InceptionTime [25]      | Mean   | 60.99          | 17.21            | 74.80            | 16.74            | 33.50            | 5.78            | –     | –                | –              | –    |
|                         | STD    | 2.09           | 0.74             | 4.45             | 1.01             | –                | –              | –     | –                | –              | –    |
| (96, 11)                | Mean   | 59.30          | 51.91            | 34.64            | 2.70             | 12.32            | 2.14            | 73.50 | 8.72             | 78.36          | 10.99 |
|                         | STD    | 1.74           | 0.47             | 1.86             | 0.56             | 2.71             | 0.82            | 3.35  | 1.47             | 1.61           | 0.61  |
| InceptionTime [25]      | Mean   | 99.75          | 75.76            | 16.35            | 2.56             | 11.34            | 2.00            | 71.46 | 8.23             | 75.14          | 9.91  |
|                         | STD    | 2.23           | 0.53             | 1.55             | 0.47             | 1.87             | 0.42            | 2.2   | 1.04             | 5.25           | 2.12  |

**Table 4 continued**

Best results are in bold
1Averaged results over two writers.
2Only one train/validation split
Comparison to state-of-the-art techniques For comparison, we train nine different well-established time-series classification architectures on our OnHW datasets and InceptionTime [25] on the tablet datasets. For these methods we interpolate and zero pad the time-series to 800 timesteps to obtain a fixed sequence length. We use linear spline interpolation. In total, 800 timesteps lead to a low CER (see Fig. 9), while above 800 timesteps the training time significantly increases. As these methods are introduced for classifying single labels (not sequences of labels), we replace the last linear layer with a max pooling layer (of kernel size 4), a dropout layer (40%) and an 1D convolutional layer (kernel size 1 and channels are the number of class labels). Similar to our approaches, we further add two BiLSTM layers each of size 60. InceptionTime is an ensemble of CNNs inspired by Inception-v4. As its default parameters \(nf\) of 32 and \(depth\) of 6) lead to inferior performance compared to our methods, we perform a large hyperparameter search for \(depth\) (between 3 and 12) and \(nf\) (16, 32, 64, 96 and 128) with and without BiLSTMs for the WD and WI tasks (see Fig. 10). On the WD dataset, a higher \(nf\) and greater \(depth\) leads to a lower CER. For the WI task, the model tends to overfit on specific writers for larger models, and hence, the error rates are constant for \(nf\) between 64 and 128, while the CER still decreases for a greater \(depth\). For \(nf\) of 32 and \(depth\) of 11, InceptionTime+BiLSTM can marginally outperform our CNN+BiLSTM model on the OnHW-equations dataset (1.65% CER WD and 11.28% CER WI) and is notably better on the OnHW-words500 (WD) dataset (12.96% CER) without the two additional BiLSTM layers, but is on par with our CNN+BiLSTM model on the WI task (26.08% CER) and yields marginally higher error rates on the random splits. Results further suggest that the performance strongly depends on the network size. XceptionTime [72] consists of depthwise separable convolutions and adaptive average pooling to capture both temporal and spatial contents. We search for the hyperparameter \(hf\) (see Fig. 11) and set \(nf = 144\). The small FCN [103] model yields high error rates, but ResNet [103] (based on FCN) enables the exploitation of class activation maps to find contributing regions in the raw data and improves FCN. ResCNN [113] integrates residual networks with CNNs. We set also \(nf = 144\) for ResCNN and ResNet (see Fig. 12), which perform similar, but cannot outperform XceptionTime on our datasets. While additional BiLSTM layers improve the results of InceptionTime, the error rates for XceptionTime, ResNet and ResCNN decrease with additional BiLSTM layers. The univariate models LSTM-FCN [39] and GRU-FCN [21] as well as the multivariate models MLSTM-FCN [40] and MGRU-FCN [40] that augment the fully convolutional block with a squeeze-and-excitation block improve the FCN results, but are not complex enough to outperform other architectures on our datasets. In general, word beam search [77] did not improve results and even leads to degraded performance. See Appendix 7 for more evaluation details and a comparison to state-of-the-art techniques.

Influence of data augmentation We train the CNN+ LSTM model on the OnHW-equations dataset with the augmentation techniques described in Sect. 4. Results are given in Table 5. The baseline WER of 22.96% (WD) can be improved with all augmentation techniques, while the WI error of 69.21% is only affected by time warping and jittering. The most notable improvement is given by time warping with 20.90% for the WD task and 64.10% for the WI task. Interpolation to 1,000 timesteps did not improve the accuracy, and normalization to \([-1, 1]\) deteriorates training performance. Figure 13 shows augmentation results and combinations of these for InceptionTime on the OnHW-
Fig. 10. Hyperparameter search of depth and $nf$ for InceptionTime [25] with and without BiLSTM on the OnHW-equations datasets averaged over 5 splits.

Fig. 11. Hyperparameter search of $nf$ for XceptionTime [72] with and without BiLSTM on the OnHW-equations datasets averaged over 5 splits.

Fig. 12. Hyperparameter search of $nf$ for ResNet [103] and ResCNN [113] with and without BiLSTM on the OnHW-equations datasets averaged over 5 splits.

Table 5. Evaluation results (WER, CER) in % (mean and standard deviation over fivefold splits) for different augmentation techniques and sensor choices for the OnHW-equations dataset.

| Augmentation Technique | Sensors | WD | WER Mean | STD | CER Mean | STD | WI | WER Mean | STD | CER Mean | STD |
|------------------------|---------|---|----------|------|----------|------|---|----------|------|----------|------|
| None                   | All     | WD | 22.96    | 1.83 | 3.50     | 0.38 | WI | 69.21    | 7.91 | 18.11    | 5.20 |
| Scaling (S)            | All     | WD | 22.70    | 0.40 | 3.43     | 0.22 | WI | 69.70    | 7.90 | 18.80    | 5.84 |
| Time Warping (TW)      | All     | WD | 20.90    | 0.83 | 3.18     | 0.27 | WI | 64.10    | 5.51 | 15.26    | 2.27 |
| Jittering (J)          | All     | WD | 22.87    | 0.75 | 3.47     | 0.33 | WI | 68.14    | 10.03| 18.68    | 7.18 |
| Magnitude Warping (MW) | All     | WD | 22.88    | 1.21 | 3.53     | 0.29 | WI | 76.80    | 8.35 | 18.47    | 5.21 |
| Shifting (SH)          | All     | WD | 22.40    | 1.12 | 3.43     | 0.24 | WI | 69.81    | 7.59 | 18.80    | 4.88 |
| Interpolation          | All     | WD | 25.04    | 0.92 | 3.96     | 0.32 | WI | 70.50    | 8.30 | 19.42    | 5.96 |
| Normalization          | All     | WD | 55.26    | 2.04 | 7.97     | 0.51 | WI | 82.48    | 8.74 | 22.71    | 5.04 |
| None                   | w/o Magnetometer | WD | 22.60    | 1.51 | 3.44     | 0.36 | WI | 63.48    | 8.32 | 16.07    | 4.73 |
| None                   | w/o Front Accelerometer | WD | 21.36    | 0.60 | 3.28     | 0.29 | WI | 70.24    | 8.25 | 19.55    | 5.52 |
| None                   | w/o Rear Accelerometer | WD | 23.20    | 0.86 | 3.57     | 0.26 | WI | 68.30    | 8.14 | 16.64    | 5.40 |
| None                   | w/o Mag., w/o Front Acc. | WD | 22.46    | 1.55 | 3.41     | 0.38 | WI | 69.12    | 8.40 | 17.31    | 4.02 |

**Bold** are baseline improvements.
Influence of sensor dropping  We train the OnHW-equations dataset and drop data from single sensors, e.g., the front or rear accelerometer or the magnetometer data, in order to evaluate the influence of each sensor, see Table 5. Only dropping the front accelerometer (WD) and the rear accelerometer (WI) decreases the WER and CER, which could also be attributed to the smaller dataset size while leaving the architecture unchanged. Without magnetometer the WER improves for the WI task as the magnetic field changes with the recording location, but keeps constant for the same writer. Dropping the force sensor leads to a significant higher classification error as the force sensor provides information that allows a segmentation of strokes.

5.2 Single character task evaluation

Method and architecture evaluation  We use our OnHW-symbols and split OnHW-equations datasets, the combination of both (samples randomly shuffled) and the OnHW-chars [65] dataset, and interpolate the single characters to the recording location, but keeps constant for the same writer. For all methods we use the categorical CE loss for training and the CRR for evaluation. Network parameter choices are described in Appendix 6. The results are summarized in Table 6. We also compare to state-of-the-art results provided in [65]. While GRU [15] yields very low accuracies for all datasets, standard LSTM units (2 and 3 stacked layers), BiLSTM units and TCNs can increase the CRR. Further, FCN [40] and the spatio-temporal variants RNN-FCN [40], LSTM-FCN [39] and GRU-FCN [21] as well as the multivariate variants MRNN-FCN, MLSTM-FCN [40] and MGRU-FCN [40] yield better results. MLSTM-FCN [40] with a standard or attention-based LSTM and with/without a squeeze-and-excitation (SE) block achieves high accuracies, but cannot improve over state-of-the-art results achieved by [65].

Due to minor and inconsistent changes in performance, it is not possible to make a statement about the importance of the SE block and the attention-based LSTM. The networks based on CNNs, i.e., ResCNN [113], ResNet [103], XResNet [34], InceptionTime [25] and XceptionTime [72], can partly outperform the FCN variants. For XResNet, a smaller depth of the network is preferable, while for InceptionTime the greater depth and larger nf generally yields better results. We train TapNet [111], an attentional prototypical network for semi-supervised learning, that achieves the lowest accuracies. We propose a benchmark for the transformer variants [13,36,44,90,101] (for details, see Appendix 6). The performance improves for all transformer variants compared to TapNet, but are notably lower than those of the convolutional and spatio-temporal methods. TST [110] with Gaussian encoding is on par with the convolutional techniques on the WD datasets. While our CNN+BiLSTM outperforms all methods on all OnHW-chars [65] datasets, it is not notably different from results achieved by the CNN+LSTM and CNN+TCN architectures, which in turn achieve the best results on the OnHW-symbols and split OnHW-equations datasets as well as on the combined cases.

Loss functions evaluation  We train the CNN+BiLSTM architecture for all single-based datasets with the CCE loss as baseline and the seven variants described in Sect. 4. For FL, we search for the optimal hyperparameters for the OnHW-chars combined dataset and for the other methods for the OnHW-symbols dataset (see Appendix 6). We set $\alpha = 0.75$ and $\gamma = 8$. From the hyperparameter searches and literature recommendation, we set $\beta = 0.1$ for LSR, $\beta = 0.95$ for SBS, $\beta = 0.8$ for HBS, and $\alpha = 0.95$ for GCE. For the SCE loss, we set $\alpha = 0.5$ and $\beta = 0.5$ for the weighting of the CCE and RCE losses, respectively. Similar, the regularization terms of the JO loss are weighted by $\alpha = 1.2$ and $\beta = 0.8$. Table 7 gives an overview of the results for all loss functions for all single character-based datasets. The FL improves the CRR results of the symbols and equations datasets (WI) in comparison with the baseline, but yields worse results for the other datasets. As characters in the OnHW-chars dataset are equally distributed, the FL does not have any benefit on training performance. LSR prevents over-confidence and increases the accuracy for all datasets. LSR also achieves the highest accuracy of all losses for eight of the 12 datasets. As there are many samples that are written similarly, the model is overconfident for such samples by integrating a confidence penalty. Similar to FL, the SBS and HBS losses can only marginally improve results for symbols...
Table 6  Recognition rates (CRR) in % for the symbols, split equations and characters WD and WI datasets

| Method                       | OnHW-symbols\(^1\) | OnHW-equations\(^{1,2}\) | OnHW-sym.\(^1\) + equations\(^{1,2}\) | OnHW-chars\(^1\) | OnHW-chars\(^2\) |
|------------------------------|--------------------|--------------------------|----------------------------------------|----------------|----------------|
| CNN+LSTM                     | 96.44              | 80.00                    | 95.43                                  | 82.42          | 95.65          |
| CNN+BiLSTM                   | 96.20              | 79.51                    | 95.70                                  | 83.88          | 95.50          |
| CNN+TCN                      | 94.21              | 76.83                    | 96.70                                  | 84.91          | 95.48          |
| LSTM (2 layers)              | 81.18              | 62.85                    | 91.05                                  | 74.11          | 90.64          |
| LSTM (3 layers)              | 83.51              | 64.48                    | 92.08                                  | 75.77          | 91.52          |
| BiLSTM (2 layers)            | 83.30              | 63.01                    | 91.39                                  | 73.43          | 91.48          |
| BiLSTM (3 layers)            | 83.09              | 59.74                    | 92.46                                  | 76.60          | 91.93          |
| GRU [15]                     | 47.57              | 33.22                    | 70.80                                  | 40.53          | 64.26          |
| TCN [6]                      | 85.41              | 70.21                    | 91.64                                  | 77.44          | 92.02          |
| FCN [103]                    | 92.18              | 74.63                    | 94.03                                  | 81.46          | 94.22          |
| RNN-FCN [40]                 | 93.23              | 74.63                    | 94.24                                  | 81.56          | 94.59          |
| LSTM-FCN [39]               | 92.39              | 73.32                    | 93.95                                  | 81.24          | 94.33          |
| GRU-FCN [21]                 | 92.39              | 73.32                    | 93.95                                  | 81.24          | 94.33          |
| MRNN-FCN                     | 92.60              | 74.30                    | 94.24                                  | 81.30          | 94.36          |
| MLSTM-SE                     | 89.22              | 70.38                    | 93.78                                  | 82.49          | 94.04          |
| FCN [40] SE, Att. LSTM       | 89.43              | 70.07                    | 93.92                                  | 80.56          | 93.59          |
| LSTM                         | 87.74              | 71.85                    | 94.12                                  | 80.13          | 90.14          |
| Att                          | 88.37              | 70.54                    | 93.95                                  | 81.18          | 94.14          |
| MGRU-FCN [40]                | 92.60              | 74.30                    | 94.21                                  | 81.28          | 94.43          |
| ResCNN (64) [113]            | 92.23              | 77.41                    | 94.58                                  | 80.95          | 94.55          |
| ResNet (64) [103]            | 94.50              | 76.76                    | 94.68                                  | 83.45          | 94.74          |
| XResNet (18) [34]            | 93.45              | 74.14                    | 94.80                                  | 81.51          | 94.73          |
| XResNet (34) [34]            | 93.45              | 74.63                    | 94.64                                  | 81.77          | 94.74          |
| XResNet (50) [34]            | 93.66              | 74.47                    | 94.63                                  | 81.74          | 94.83          |
| XResNet (101) [34]           | 92.60              | 75.29                    | 93.64                                  | 80.95          | 93.48          |
| XResNet (152) [34]           | 92.18              | 73.16                    | 93.47                                  | 80.00          | 92.58          |
| XceptionTime (16) [72]       | 91.54              | 72.34                    | 94.03                                  | 82.24          | 93.95          |
| InceptionTime (32,6) [25]    | 91.33              | 76.10                    | 94.05                                  | 81.30          | 93.88          |
| InceptionTime (47,9) [25]    | 92.60              | 75.94                    | 94.49                                  | 83.42          | 94.20          |
| InceptionTime (62,9) [25]    | 91.97              | 78.07                    | 94.83                                  | 81.57          | 95.01          |
| InceptionTime (64,12) [25]   | 91.97              | 76.92                    | 94.87                                  | 84.35          | 95.06          |
| MultiIncep.Time (32,6) [25]  | 91.12              | 75.29                    | 93.91                                  | 80.57          | 93.61          |
| MiniRocket [87]              | 69.77              | 58.76                    | 75.91                                  | 45.34          | 75.58          |
| OmniScaleCNN [89]            | 84.78              | 68.09                    | 91.76                                  | 75.46          | 92.23          |
| XEM [24]                     | 85.84              | 67.10                    | 92.13                                  | 77.04          | 91.42          |
| TapNet [111]                 | 67.02              | 48.12                    | 66.38                                  | OOM            | 65.96          |
| mWDN [99]                    | 88.58              | 67.43                    | 92.37                                  | 77.30          | 92.02          |
| Perceiver [36]               | 67.40              | 48.10                    | 89.60                                  | 58.10          | 89.30          |
| Sinkhorn [90]                | 61.10              | 50.90                    | 76.80                                  | 66.40          | 97.30          |
| Performer [13]               | 55.40              | 47.80                    | 76.10                                  | 68.30          | 74.90          |
| Reformer [44]                | 56.90              | 47.80                    | 75.80                                  | 70.10          | 75.40          |
| Linformer [101]              | 53.90              | 42.90                    | 75.20                                  | 67.40          | 74.90          |
| TST [110] (Gaussian)         | 91.12              | 71.85                    | 93.07                                  | 80.40          | 93.16          |
| MultiTST [110]               | 87.53              | 71.19                    | 92.36                                  | 78.82          | 91.96          |
| TSiT [110]                   | 84.99              | 68.09                    | 93.30                                  | 78.98          | 92.91          |

\(^1\) lower \(WD\), \(WI\), upper combined

\(^2\) lower \(WD\), upper combined

\(^{1,2}\) lower \(WD\), upper combined
Table 6 continued

| Method | OnHW-symbols | OnHW-equations | OnHW-sym. + equations | OnHW-chars [65] |
|--------|--------------|----------------|-----------------------|-----------------|
|        | WD           | WI             | WD                   | WI              | Lower | Upper | Combined |
| CNN (from [65]) | – – – – – – – – | 84.62 | 76.85 | 89.89 | 83.01 | 70.50 | 64.01 |
| LSTM (from [65]) | – – – – – – – – | 79.83 | 73.03 | 88.68 | 81.91 | 67.83 | 60.29 |
| CNN+LSTM (from [65]) | – – – – – – – – | 82.64 | 74.25 | 88.55 | 82.96 | 69.42 | 64.13 |
| BiLSTM (from [65]) | – – – – – – – – | 82.43 | 75.72 | 89.15 | 81.09 | 69.37 | 63.38 |

11-fold cross-validation split; samples interpolated to 79 timesteps. 2Split into single symbols and numbers.
35-fold cross-validation split; samples interpolated to 64 timesteps. Underlined: State-of-the-art results. Bold: Best results. SE: squeeze-and-excitation. At.: attentional LSTM

Table 7 Recognition rates (CRR) in % for the symbols, split equations and characters WD and WI datasets for the CNN+BiLSTM architecture trained with different loss functions

| Loss Function | OnHW-symbols | OnHW-equations | OnHW-sym. + equations | OnHW-chars [65] |
|---------------|--------------|----------------|-----------------------|-----------------|
|               | WD           | WI             | WD                   | WI              | Lower | Upper | Combined |
| Categorical CE (CCE) | 96.20 | 79.51 | 95.57 | 83.88 | 95.50 | 84.55 | 89.66 | 80.00 | 92.58 | 85.64 | 78.98 | 68.44 |
| Focal loss (FL) [50] | 95.78 | 79.67 | 95.42 | 84.53 | 95.25 | 85.20 | 88.56 | 78.88 | 91.91 | 85.62 | 77.48 | 68.15 |
| Label smoothing (LSR) [67] | 96.22 | 81.83 | 95.86 | 87.09 | 95.74 | 86.52 | 89.74 | 80.96 | 92.72 | 86.13 | 79.09 | 69.43 |
| Boot soft (SBS) [73] | 96.00 | 79.00 | 95.70 | 84.87 | 95.65 | 85.91 | 89.08 | 79.76 | 92.12 | 85.79 | 78.19 | 68.47 |
| Boot hard (HBS) [73] | 96.22 | 79.17 | 95.63 | 85.27 | 95.60 | 87.11 | 89.20 | 80.00 | 92.29 | 85.82 | 78.28 | 68.41 |
| Generalized CE (GCE) [112] | 96.44 | 80.83 | 95.81 | 86.46 | 95.64 | 86.69 | 88.18 | 79.34 | 91.51 | 85.49 | 76.91 | 67.76 |
| Symmetric CE (SCE) [102] | 96.44 | 81.00 | 95.76 | 85.15 | 95.58 | 85.43 | 89.24 | 79.90 | 92.09 | 85.84 | 78.11 | 68.65 |
| Joint optimization (JO) [88] | 97.33 | 82.17 | 95.67 | 85.40 | 95.60 | 85.87 | 89.71 | 80.14 | 92.65 | 86.56 | 79.07 | 69.26 |

11-fold cross-validation split; samples interpolated to 79 timesteps. 2Split into single symbols and numbers.
35-fold cross-validation split; samples interpolated to 64 timesteps. Bold: Best results.

And split equations datasets, and even decrease performances for the character datasets. HBS is slightly better than SBS. The GCE loss decreases the classification accuracy for the OnHW-characters datasets, while it achieves the second best CRR of all losses for the split OnHW-equations WD (95.81%) and WI (86.46%) datasets. Yet, the GCE loss often results in NaN loss (see Fig. 25, Appendix 7), and hence, is non-robust for our datasets. The improvement for the SCE loss is less significant than other losses and even decreases for the OnHW-characters dataset. JO leads to an improvement for all OnHW-characters datasets. JO further outperforms all losses for the WI upper task and achieves marginally lower accuracies than the LSR loss for the lower and combined datasets. LSR also achieves the highest accuracies on the OnHW-symbols WD (97.33%) and WI (82.17%) datasets. In summary, all loss variants can improve results of the CCE loss for the OnHW-symbols, split OnHW-equations and combined datasets as these are not equally distributed. LSR, SCE and JO can most significantly outperform other techniques. For more details of accuracy plots, see Appendix 7, Fig. 25.

5.3 Left-handed Writers datasets evaluation

For the left-handed writers datasets, we use the pre-trained weights from the right-handed datasets and train the CNN+BiLSTM architecture for 500 epochs. Table 8 summarizes all results for the sequence-based classification task (left) and the single character-based classification task (right). The motion dynamics of right- and left-handed writers is very different, especially with different rotations, and hence, also the sensor data are different [45]. The models can still make use of the pre-trained weights, and fine tuning leads to 1.24% CER for the OnHW-equations-L dataset for the WD task, and 15.32% CER for the OnHW-words500-L dataset, which is better than for the right-handed task. For the OnHW-wordsRandom-L dataset, the CER (5.40%) increases, while the WER (32.73%) decreases. Consistently, the results for the WI task decrease as the model overfits to specific writers due to the small amount of different left-handed writers in the training set. For single-based datasets, the fine tuning leads to a high WD classification accuracy of
Table 8 Evaluation results (WER, CER) in % (mean and standard deviation) for our left-handed OnHW-equations-L, OnHW-words500-L and OnHW-wordsRandom-L datasets (left), and recognition results (CRR) in % for our left-handed OnHW-symbols-L, split OnHW-equations-L and OnHW-chars-L datasets (right) for the CNN+BiLSTM architecture

| Dataset (CNN+BiLSTM architecture) | WD | WI | WD | WI | Dataset (CNN+BiLSTM architecture) | WD | WI |
|-----------------------------------|----|----|----|----|-----------------------------------|----|----|
| OnHW-equations-L                  | 8.56 | 1.59 | 1.24 | 0.25 | 95.73 | 3.13 | 32.16 | 5.16 |
| OnHW-words500-L                   | 47.90 | 17.25 | 15.32 | 6.03 | 97.90 | 1.10 | 81.43 | 11.66 |
| OnHW-wordsRandom-L                | 32.73 | 3.43 | 5.40 | 1.15 | 99.70 | 0.30 | 72.27 | 15.55 |
| OnHW-symbols-L                    | 92.00 | 54.00 |
| OnHW-equations-L, split            | 92.02 | 51.50 |
| OnHW-chars-L                      | 94.70 | – |
| [65] Upper                         | 91.90 | – |
| Combined                           | 82.80 | – |

1 1-fold cross-validation split; samples interpolated to 79 timesteps
2 Split into single symbols and numbers
3 5-fold cross-validation split; samples interpolated to 64 timesteps
Writer-dependent (WD) and writer-independent (WI) classification tasks

Fig. 14 Evaluation of the ED dependent on the normalized sample lengths for the OnHW-equations dataset

(a) Mismatches. (b) Insertions. (c) Deletions.

Fig. 15 Evaluation of the ED dependent on the normalized sample lengths for the OnHW-wordsTraj dataset

(a) Mismatches. (b) Insertions. (c) Deletions.

5.4 Edit distance and writer analysis

Evaluation of sample length-dependent edit distance We show the sample length-dependent counts of wrong predictions, i.e., mismatches, insertions and deletions, for the OnHW-equations (see Fig. 14) and OnHW-wordsTraj (see Fig. 15) datasets. For the OnHW-equations dataset, a high appearance of mismatches and insertions appears at the starting and end characters, while deletions emerge more even over the whole equations. The first character of words is significantly often mismatched or has to be inserted or deleted for the OnHW-wordsTraj dataset. This shows the unequal distribution of samples for the words datasets (see Fig. 4c), while the equations dataset is very equally distributed (see Fig. 4b).

Writer-dependent evaluation Figure 16 shows the writer-dependent evaluation of the OnHW-equations dataset. The CER of many samples of several writers, e.g., ID 0, 2-4, 24-35, 42-44, and 49-53, is 0%. The CER increases only for a small number of samples. The range of the CER increases...
for writer IDs 1, 5-7, 22, 23, and 36-39. Hence, the writing style and with that the sensor data is different and out-of-distribution in the dataset.

6 Discussion and summary

6.1 Social impact, applications and limitations

Handwriting is important in different fields, in particular graphomotoric. The visual feedback provided by the pen, for instance, helps young students and children to learn a new language. Hence, research for HWR is very advanced. However, state-of-the-art methods to recognize handwriting (a) require to write on a special device, which might adversely affect the writing style, (b) require to take images of the handwritten text, or (c) are based on premature technical systems, i.e., the sensor pen is only a prototype [17]. The publicly available sensor pen developed by STABILO International GmbH has previously been used by [46, 65] and allows an easier data collection than previous techniques. The research for collecting devices which do not influence the handwriting style is becoming increasingly important and with it also the social impact of resulting datasets. The aim of our dataset is to support the learning of students in schools or self-paced learning from home without additional effort [4, 106]. A well-known bottleneck for many machine learning algorithms is their requirement for large amounts of data samples without under-represented data patterns. For our HWR application, a large variety of different writing styles (cursive or printed characters, left- or right-handed and beginner or advanced writers), pen rotations and writing surfaces (especially different vibrations of the paper) are necessary. We provide an evaluation benchmark for right- and left-handed datasets. As motion dynamics between right- and left-handed writers are very different, extracting mutual information is a challenging task [45, 63]. The ratio between both groups approximately fits the real-world distribution, i.e., the under-representation of left-handed writers (10.6%). Only adults without any selection participated at data recording as the handwriting style of students changes quickly with the age [7].

6.2 Experimental results

We performed several benchmarks and come to the following conclusions: (1) For the seq2seq classification task, we evaluated several methods based on CNNs in combination with RNNs on inertial-based datasets written on paper and on tablet and evaluated state-of-the-art trajectory-based datasets. Depending on the dataset size, our CNN+BiLSTM model is on par with the InceptionTime+BiLSTM architecture. A search of architecture hyperparameters is important to achieve a generalized model for a real-world application. Our transformer-based architecture could not outperform simpler convolutional models. (2) Sensor data augmentation leads to a better generalized training. (3) For the single classification task, our simple CNN+LSTM, BiLSTM, TCN] can outperform state-of-the-art techniques. (4) Cross-entropy variants (i.e., label smoothing) improve results that are dependent on the dataset (i.e., label noise and class balance). (5) Writer-independent classification of (under-represented) left-handed writers is very challenging that is interesting for future research.

6.3 Collection consent and personal information

While recording the datasets, we collected the consent of all participants. We only collected the raw data from the sensor-enhanced pen, and for statistics the age and gender of the participant and their handedness. The handedness is necessary because the pen is differently rotated between left- and right-handed writers. The recording localization was Germany. An ID is assigned to every participant such that the dataset is fully pseudonymized. The ID is necessary for the WD and WI evaluation.
6.4 Conclusion and future research

We proposed several equations and words OnHWR datasets for a seq2seq classification task, as well as one symbol dataset for the single character classification task based on a novel sensor-enhanced pen. By utilizing (Bi)LSTM and TCN models combined with CNNs and different transformer models, we proposed a broad evaluation benchmark for lexicon-free classification. Various augmentation techniques showed notable improvement in classification accuracy. Our detailed evaluation of the WD and WI tasks sets important challenges for future research and provides a benchmark foundation for novel methodological advancements. For example, semi-supervised learning and few-shot learning such as prototypical networks could improve the classification accuracy of under-represented writers. Exploiting offline datasets for pre-training or the use of lexicon and language models might further allow the model to better learn the task.

Acknowledgements We sincerely thank all participants taking part in the data recordings and acknowledge the work of various researchers from the STABILIO International GmbH, Kinemic GmbH, Fachdidaktik Deutsch Primarstufe (DID) of the Saarland University, Machine Learning and Data Analytics Lab of the Friedrich-Alexander University (FAU) and Fraunhofer Institute for Integrated Circuits (IIS) for their help with the data collection.

Funding Open Access funding enabled and organized by Projekt DEAL. This work was supported by the Federal Ministry of Education and Research (BMBF) of Germany by Grant No. 01IS18036A (David Rügamer) and by the research program Human-Computer-Interaction through the project “Schreibtrainer,” Grant No. 16SV8228, as well as by the Bavarian Ministry for Economic Affairs, Infrastructure, Transport and Technology through the Center for Analytics-Data-Applications (ADA-Center) within the framework of “BAYERN DIGITAL II.”

Data availability Data and materials will be publicly available upon publication that includes the OnHW-chars, OnHW-symbols, split OnHW-equations, OnHW-equations, OnHW-wordsTraj and OnHW-words500(R) datasets. We include left-handed and right-handed as well as writer-dependent (WD) and writer-independent (WI) splits. www.iis.fraunhofer.de/de/ff/lv/dataanalytics/anwproj/schreibtrainer/onhw-dataset.html

Code Availability Will be publicly available.

Declarations

Ethics approval see Sect. 6.1

Consent to participate see Sect. 6.3

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

Appendices

In this appendix, we will give a general overview of related work in Sect. 1. We propose more details about the sensor pen in Sect. 2 and present the data acquisition and format in Sect. 3. While Sect. 4 shows additional samples, Sect. 5 proposes more detailed statistics of the datasets. We state the chosen transformer parameters in Sect. 6. Section 7 concludes with more evaluation details.

General overview of related work

Temporal convolutional networks (TCNs) TCNs consist of CNNs as encoders to extract spatio-temporal information for low-level feature computation and a classifier that captures high-level temporal information using a recurrent network. TCNs can take a series of any length and output it with the same length. They perform well in prediction tasks with time-series data. Yan et al. [107] TCNs have been used for the HWR task in [82, 83].

RNNs Wigington et al. [105] proposed a CNN-LSTM model for text detection, segmentation and recognition. The performance of RNNs can be improved using dropout [68]. Carbune et al. [12] highly improve classification accuracies by a stack of bidirectional LSTMs [31]. Tian et al. [91] combined BiLSTMs in the word encoder with word interattention for a multi-task document classification approach. Multi-dimensional RNNs as the MDLSTM-RNNs [32] scan the input in the four possible directions, where LSTM cell inner states and output are computed from previous positions in the vertical and horizontal directions. Voigtlaender et al. [97] processed the input in a diagonal-wise fashion to enable GPU-based training and explored deeper and wider MDLSTMs architectures for HWR. Bluche [10] transformed the 2D representation into a sequence of predictions to enable end-to-end processing of paragraphs. However, these architectures are computationally expensive and extract features visually similar to CNNs; hence, 2D long-term dependencies may not be essential [70]. Dutta et al. [20] integrated a spatial transformer network into their RCNN method.

Transformers They aim for handling long-range dependencies with ease relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution. Vaswani et
al. [94] showed their transformer architecture consisting of a decoder, encoder and multi-head attention to be superior in quality while being more parallelizable and requiring significantly less time to train. Kang et al. [38] introduced a novel method for offline HWR that bypasses any recurrence and uses multi-head self-attention layers at visual and textual stages. As transformer-based models scale quadratically with the sequence length due to their self-attention, the Longformer introduced an attention mechanism that scales linearly and was applied to process documents of thousands of tokens. The Performer [13] that estimates (softmax) full-rank attention transformers also use only linear complexity. The Perceiver [36] scales to high-dimensional inputs such as audio, videos, images and point-clouds by using cross-attentional principles before using a stack of transformers in the latent space.

Additional information of the sensor pen

The DigiPen by STABILO International GmbH is a sensor-enhanced ballpoint pen with internal data processing capabilities. A Bluetooth module enables live streaming of the integrated sensor at 100 Hz to a connected device. The DigiPen development kit is also publicly available. The pen has an ergonomic soft-touch grip zone, such that the writing feels comfortable and is as normal writing on paper. The pen’s overall length is 167 mm, its diameter is 15 mm, and its weighs 25 g. The pen is equipped with a front accelerometer (STM LSM6DSL), a rear accelerometer (Freescale MMA8451IQ), a gyroscope (STM LSM6DSL), a magnetometer (ALPS HSCD6008A) and a force sensor (ALPS HSFPAR003A). The front and rear accelerometers are differently oriented. The accelerometers were adjusted to a range of ±2g with a resolution of 16 bit of the front and 14 bit for the rear accelerometer. The gyroscope has a range of ±1000°/s (16 bit), and the magnetometer has a range of 2.4 mT (14 bit). The measurement range of the force sensor is between 0 and 5.32 N (12 bit).

Data acquisition and format

STABILO International GmbH provides a recording app to obtain the sensor data that are publicly available. Through this setup we also recorded the ground truth labels. The data were recorded over a period of 1.5 years. To achieve equally distributed datasets, we apply the following constraints. The writer has to write on a normal, white paper padded by five additional sheets, and has to sit on a chair in front of a table. The logo of the pen needs to face upwards. Users are allowed to write in a cursive or printed style. The way of holding the pen and the size of handwriting was not constrained. Prior to recording the gyroscope and magnetometer biases and the magnetometer scaling has to be determined by calibrating the pen. We do not use the calibration data, but publish the calibration files along the datasets for possible future research. For more information, see [65].

The data format is given as following. For each dataset we will publish the raw data that consist of the calibration file, a labels file with start and end timestep, and a data file with the corresponding 13 channels for each timestep. Additionally, we already preprocess the data and upload pickle (.pkl) files. For each dataset and each of the five cross-validation splits, we generated a train and validation file with the sensor data, the corresponding label and the writer IDs.

Exemplary sensor data

Figures 17 and 18 show the sensor data of the 13 channels for an exemplary equation and words written on normal paper and on tablet. The accelerometer data are given in m/s², the gyroscope data in °/s, the magnetometer data in mT and the force sensor in N. The equation sample consists of 567 timesteps, while the word sample on paper consists of 217 timesteps and on tablet of 402 timesteps. It can be shown that for all three samples the single strokes can be clearly separated through the force sensor (see Fig. 17d). By comparing the accelerometer and gyroscope data of a selected word written on normal paper (see Fig. 18a and b) with the word written on tablet (see Fig. 18c and d), we can see that the surface of the paper introduces higher sensor noise than the surface of the tablet.

Statistics of the datasets

The characteristics of a dataset influences the behavior of a deep learning model. If the deployed context does not match the evaluation datasets, a model is unlikely to perform well. Hence, we will propose more detailed statistics of our proposed datasets, in this section, while we already compared our datasets with state-of-the-art datasets in Sect. 3.2. Table 9 proposes average and deviation timesteps and number of strokes for each sample length of the OnHW-equations, OnHW-words500 and OnHW-wordsRandom datasets. The number of timesteps per sample is significantly larger for the OnHW-equations dataset than for the words datasets. We can conclude that writing numbers and symbols requires more time as words are mostly written in cursive font, while equations are written in printed font. Hence, the deviation of timesteps is also larger for equations. The deviation in timestep lengths is important as the data have to be split by the CTC loss, and a larger deviation leads to more split varieties. Additionally, the number of strokes per sample is an significant feature for the classification task, which can be learned
Fig. 17 Exemplary sensor data of one sample of the OnHW-equations dataset

Fig. 18 Exemplary accelerometer and gyroscope data of one sample of the OnHW-words500 (top) and OnHW-wordsTraj (bottom) datasets

by the model from the force sensor data. The average number of strokes is clearly larger (about one to three strokes) for the OnHW-equations dataset than the words datasets, while the deviation of strokes is less. We can state from that number and symbols require many strokes for printed writing, while cursive writing of words leads to less number of strokes with a use-specific writing style. Hence, training a model for a writer-independent classification task is more difficult. To split the OnHW-equations dataset into single symbols and numbers, we use the following split constraints, where the possible number of strokes per character is $0 [1], 1 [1], 2 [1], 3 [1], 4 [1,2], 5 [2], 6 [1], 7 [1,2], 8 [1], 9 [1], + [2], - [1], \cdot [1], : [2]$ and $= [2]$.

Figure 19 gives an overview of writers contributed to the sequence-based datasets. For the OnHW-equations dataset most participants wrote about 180 equations, while for the OnHW-words500 and OnHW-wordsRandom each writer contributed about 500 words. This leads to a equally balanced dataset and a proper writer-independent evaluation.

Not only the diversity of the samples per participant is important, but also the diversity of the sensor data. In particular, out-of-distribution sensor data from one writer can decrease classification accuracy. Figure 20 gives an overview of the mean distribution per writer for the x-axis of the sensors. For the force data, the writers with IDs 7 and 17 have many outliers, while the writers with IDs 12, 37 and 42 press...
Table 9

| Dataset          | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 |
|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|
| OnHW-equations   | $L_A$ | – | – | – | 228 | 332 | 420 | 482 | 559 | 642 | 703 | 777 | 855 | 906 |
|                  | $L_D$ | – | – | – | 37  | 105 | 158 | 167 | 192 | 207 | 211 | 230 | 235 | 260 |
|                  | $S_A$ | – | – | – | 6.21 | 7.90 | 9.28 | 10.37 | 11.76 | 13.05 | 14.21 | 15.57 | 17.03 | 18.28 |
|                  | $S_D$ | – | – | – | 2.21 | 1.67 | 1.88 | 2.32 | 2.61 | 2.69 | 2.86 | 2.93 | 3.76 | 3.00 |
| OnHW-words500    | $L_A$ | 87 | 120 | 150 | 187 | 223 | 272 | 320 | 349 | 393 | 425 | 486 | 502 | 603 |
|                  | $L_D$ | 258 | 36 | 35 | 51 | 63 | 72 | 95 | 86 | 90 | 106 | 118 | 140 |
|                  | $S_A$ | 2.46 | 3.25 | 3.84 | 4.57 | 5.32 | 6.64 | 7.40 | 8.32 | 8.88 | 9.55 | 10.36 | 11.81 | 14.75 |
|                  | $S_D$ | 1.00 | 1.30 | 1.41 | 1.76 | 2.33 | 2.64 | 2.95 | 2.94 | 3.30 | 3.44 | 4.26 | 4.68 |
| OnHW-wordsRandom | $L_A$ | 111 | 167 | 201 | 240 | 290 | 340 | 397 | 438 | 493 | 538 | 598 | 648 | 703 |
|                  | $L_D$ | 41  | 80  | 86 | 94  | 112 | 128 | 158 | 165 | 179 | 194 | 209 | 222 | 234 |
|                  | $S_A$ | 2.83 | 3.89 | 4.64 | 5.25 | 5.95 | 6.82 | 7.69 | 8.50 | 9.23 | 10.08 | 11.11 | 11.74 | 13.01 |
|                  | $S_D$ | 1.07 | 1.41 | 1.72 | 1.80 | 2.11 | 2.34 | 2.65 | 2.96 | 3.12 | 3.42 | 3.56 | 3.82 | 4.04 |

Fig. 19 Overview of writer IDs contributed to the OnHW-equations, OnHW-words500, and OnHW-wordsRandom datasets.

the pen tip strongly to the paper. While the front accelerometer data are very diverse between $-10^3$ and $10^3$ (e.g., writer 14, 25 and 45 with many outliers, against writer 16, 19 and 37 with consistent sensor data), the movement of the rear accelerometer is slower between $-4 \cdot 10^3$ and $3 \cdot 10^3$, as the pen tip typically moves faster than the rear accelerometer. The gyroscope distribution per writer draws conclusion of the writing style. These findings lead to the conclusion that the writer-dependent problem is an easier classification task than the writer-independent problem.

Transformer parameters and hyperparameter searches

Transformer parameters

This section describes the transformer parameters. For our attention-based model, we search for the optimal parameters $d_{\text{model}} = [150, 300]$, $d_k = [32, 64]$, $d_v = [32, 64]$, the number of multi-head attentions $n_{\text{head}} = [3, 4, 5]$ and a convolutional factor $c_{\text{fac}} = [4, 6, 8, 10, 16]$, while the network consists of the 1D convolutions ($c_{\text{fac}}, 2 \cdot c_{\text{fac}}, 4 \cdot c_{\text{fac}}, 8 \cdot c_{\text{fac}}$).
and the (Bi)LSTM layers ($4 \cdot c_{\text{fac}}, 2 \cdot c_{\text{fac}}$). We train only 500 epochs, as each training takes 4h. We choose $d_{\text{model}} = 150$, $d_k = 32, d_v = 64, c_{\text{fac}} = 6$, with BiLSTM and time distribution for follow-up training. The number of heads $n_{\text{heads}}$ is 3.

We apply the Transformer variants Perceiver [36], Sinkhorn Transformer [90], Performer [13], Reformer [44] and Linformer [101] to the single character classification task with the following parameters. We choose non-reversible transformers without a language model or a lexicon. The input is the inertial MTS. We evaluated different combinations of last layers for all variants, i.e., with and without 1D convolution or 1D max pooling. The best results yielded a permutation with a 1D max pooling of kernel size 5 and stride 5, in combination with a linear layer of size $(\text{indim}, n_{\text{classes}})$. For the Perceiver [36], we set $\text{crossheads} = 1, \text{numfreq} = 4, \text{depth} = 2, \text{numlatents} = 64, \text{latentdim,heads} = 128, \text{maxfreq} = 10$ and $\text{latentheads} = 4$. We set $\text{attndrop}$ and $\text{ffdrop}$ to 0.2. $n_{\text{classes}}$ depends on the dataset and is for the OnHW-symbols and OnHW-equations dataset 15, and for the OnHW-chars dataset 26 for the lower and upper datasets and 52 for the combined dataset. We choose the parameters of the Sinkhorn Transformer [90] $\text{dim} = 1, 024, \text{heads} = 8, \text{depth} = 12, \text{dimhead} = 6$ and $\text{bucketsize} = 20$. For the Performer [13], we choose $\text{dim} = 512, \text{depth} = 1, \text{heads} = 5, \text{dimhead} = 4$ and $\text{causal} = \text{True}$. The parameters of the Reformer [44] are $\text{dim} = 128, \text{heads} = 8, \text{bucketsize} = 20, \text{dimhead} = 6, \text{depth} = 12, \text{ishdrop} = 0.1$ and $\text{causal} = \text{True}$.

We set for the Linformer Transformer [101] the parameters $\text{dim} = 512, \text{seqlen} = 79$ for split OnHW-equations and OnHW-symbols and $\text{seqlen} = 64$ for OnHW-chars [65], $\text{depth} = 12, \text{heads} = 5, \text{sharekv} = \text{True}$ and $k = 256$. For the Sinkhorn and Reformer Transformers, the sequence length has to be divisible by the bucket size. For Performer and Linformer Transformers, the input dimension has to be divisible by the number of heads, and hence, we exclude the magnetometer data.
Hyperparameter search

We search for the Focal loss \([50]\) for the class balance factor \(\alpha \in [0, 1]\) and \(\gamma \geq 0\) in the modulating factor \((1 - p_i)\gamma\). We use the combined OnHW-chars (WI) dataset. Figure 21 shows the hyperparameter search for \(\alpha\) and \(\gamma\) with Optuna\(^4\). The objective value is the character recognition rate. The optimal parameters are \(\alpha = 0.75\) and a large \(\gamma = 8\). Note that the search space is in a small range between 71\% and 73\%. We use these parameters, for follow-up trainings.

Detailed evaluation

Evaluation of the accuracy per Label

Figure 22 shows confusion matrices for sequence-based classification tasks for the accuracy of predicted single class labels regarding the ground truth class labels in \%. For the OnHW-equations dataset (see Fig. 22a), the accuracies per labels are between 96.6\% and 99.3\%. While the ground truth ‘0’ is confused with ‘6’ and ‘9’ because of the similar round shape of these numbers, the ‘-’ is misclassified with the ‘.’ as both symbols are short samples, and the ‘:’ is misclassified with a single dot ‘·’. From analyzing the confusion matrix of the OnHW-wordsTraj (see Fig. 22b) dataset, we see two significant patterns. First, small letters are highly accurate starting from 80\% (see the second part of the diagonal), while only ‘j’ is misclassified with ‘W’ and ‘s’. Second, capital letters are highly incorrect (see the first part of the diagonal). Letters as ‘C’, ‘P’ and ‘T’ are indistinguishable from other letters, while ‘Q’ and ‘O’ are interchanged. The reason is the under-representation of capital letters in the dataset (see Fig. 4c), as capital letters only appear at the starting letter of a word in German. By plotting the confusion matrix for the OnHW-wordsRandom dataset, the mismatches for capital letters improve compared to the OnHW-wordsTraj dataset.

\(^4\) Optuna: https://optuna.org/
dataset, but is still significantly higher compared with small letters. The vowel mutations (‘ä’, ‘ö’, ‘ü’, ‘Ä’, ‘Ö’, ‘Ü’) are also highly under-represented, and the classification accuracy highly decreases. Figure 23 separates the mismatches, deletions and insertions per labels (see Eq. (1)). The number on top of the box plot indicates the amount of occurrences in the validation test. For the OnHW-equations dataset (see Fig. 23a), the number ‘0’ does not have to be inserted, and number ‘4’ does not have to be deleted. It is significant that the symbols with less timesteps ‘‘, ‘·’ and ‘:’ are often mismatched and missed, while only ‘−’ has to be deleted. Numbers are distinguishable more easily.

For the OnHW-wordsTraj (see Fig. 23b), again, the CER of capital letters is considerably higher than small letters. While some letters, i.e., ‘C’, ‘Q’, ‘T’, ‘U’ and ‘V’, are only mismatched, other edit errors appear for ‘A’, ‘I’, ‘J’ and ‘Z’. A dataset with more capital letters could mitigate these errors.

### Evaluation of sample length-dependent edit distance

We show the sample length-dependent counts of wrong predictions, i.e., mismatches, insertions and deletions, for the OnHW-equations (see Fig. 14) and OnHW-wordsTraj (see Fig. 15) datasets. For the OnHW-equations dataset, a high appearance of mismatches and insertions appears at the starting and ending character, while deletions emerge more evenly over the whole equations. As previously shown, the first character of words is significantly often mismatched or has to be inserted or deleted for the OnHW-wordsTraj dataset. This shows the unequal distribution of samples for the words datasets (see Fig. 4c), while the equations dataset is very equally distributed (see Fig. 4b).

### Evaluation results of state-of-the-art techniques for online HWR

This section summarizes state-of-the-art results for the IAM-OnDB [51], VNOnDB [59] and IBM-UB-1 [84] datasets (see Table 10). Graves et al. [31] (2008) started to improve the classification accuracy by proposing an alternative approach based on a RNN specifically designed for sequence labeling tasks where data contain long-range interdependencies and that is hard to segment. Liwicki et al. [52] introduced recognizers based on hidden Markov models and BiLSTMs, and on different set of features from online and offline data. Frinken et al. [27] showed that a deep BiLSTM neural network outperforms the standard BiLSTM model by combining ReLU activation with BiLSTM layers, but get a high WER of 24.99% and a CER of 12.26% on the IAM-OnDB dataset. Keysers et al. [41] used a training for feature combination, a trainable segmentation technique, unified time- and position-based input interpretation and a cascade of pruning strategies. The method achieves a WER of 26.7% and a CER of 8.80% with a BiLSTM, and up to 10.4% WER and 4.30% CER with a segmentation approach. The system is used in several Google products such as for translation. Carbune et al. [12] used bi-directional recurrent layers in combination with a softmax layer and the CTC loss. Their approach supports 102 languages. Hence, the architecture is based on a language model. The system combines methods from sequence recognition with a new input encoding using Bézier curves. This technique achieves the currently best results for the IAM-OnDB and IBM_UB_1 datasets. Feature functions (FF) introduce prior knowledge about the underlying language into the system. This method was used for the ICFHR2018 competition on Vietnamese online handwritten text recognition using VNOnDB. Along with this challenge, results from GoogleTask2, IVTOVTask2

---

**Table 10 State-of-the-art evaluation results in % for the online IAM-OnDB [51], VNOnDB-words [59] and IBM-UB-1 [84] datasets**

| Method           | IAM-OnDB [51] | VNOnDB [59] | IBM-UB-1 [84] |
|------------------|---------------|-------------|---------------|
|                  | WER  | CER | WER | CER | WER | CER |
| BiLSTM [12]      | 6.50 | 2.50 | 12.20 | 6.10 | 15.10 | 4.10 |
| curve, w/o FF    | 18.60 | 5.90 | – | – | 25.10 | 6.00 |
| curve, w/ FF     | 10.60 | 4.00 | – | – | 15.10 | 4.10 |
| BiLSTM [27]      | 24.99 | 12.26 | – | – | – | – |
| BiLSTM [31]      | 20.30 | 11.50 | – | – | – | – |
| LSTM [52]        | 18.93 | – | – | – | – | – |
| combination²     | 13.84 | – | – | – | – | – |
| BiLSTM [41]      | 26.70 | 8.80 | – | – | 22.20 | 6.70 |
| Seg-and-Dec [41]² | 10.40 | 4.30 | – | – | – | – |
| GoogleTask2³     | – | – | 19.00 | 6.86 | – | – |
| IVTOVTask2⁴      | – | – | 14.11 | 3.24 | – | – |
| MyScriptTask2_1⁴ | – | – | 2.02 | 1.02 | – | – |
| MyScriptTask2_2⁴ | – | – | 1.57 | 4.02 | – | – |

1 Feature functions (FF) ² Open training set ³ VieTreeBank (VTB) corpus ⁴ Results available: here
and MyScriptTask2 are available, where MyScriptTask2_2 achieves the lowest WER of 1.57% on the VNOnDB dataset. This method uses a segmentation component with a feed-forward network along with BiLSTMs and the CTC loss. The IVTOVTask2 system also uses BiLSTM layers with the CTC loss similar to our approach. Unfortunately, public code is not available for these approaches. A direct comparison of these results with our results is not possible, as we used a fivefold cross-validation of the IAM-OnDB [51] and VNOnDB-words [59] datasets, different to the train/test splits used for public results. But for our task, we can differentiate between WD and WI classification tasks. With our best model CNN+BiLSTM we achieve a CER of 6.94% (WD) and 9.11% (WI) for the IAM-OnDB dataset that is better than the BiLSTM approaches by [27,31,41], but worse than the BiLSTM by [12] and the method by [41]. On the VNOnDB-words dataset, our CER of 6.71% (WD) and the WER of 15.54% (WD) is lower than GoogleTask2, but higher than [12] and MyScriptTask2.

Error and accuracy plots

Figure 24 shows an overview of error plots for all sequence-based datasets. While the training losses converge very fast (see Fig. 24a and e) and the models slightly overfit (see Fig. 24b and f), the WER (see Fig. 24c and g) and the CER (see Fig. 24d and h) are continuously decreasing. We propose the validation accuracies (CRR) while training in Fig. 25 for single-based datasets for the eight training losses. The generalized cross-entropy (GCE) is often not robust, see Fig. 25c to j. The difference between loss function of the WD symbols and equations datasets is small (see Fig. 25a and c), but gets more important for the WI tasks (see Fig. 25b and d). From the OnHW-chars [65] dataset, we can conclude that symmetric cross-entropy (SCE), label smoothing (LSR) and joint optimization (JO) can improve the baseline categorical cross-entropy (CCE) loss. The Focal loss (FL) [50] converges slower, and boot soft (SBS) and boot hard (HBS) are similar to CCE.

Training times

Table 11 compares training times of all methods used for our benchmark for the lower OnHW-chars [65] (WD) and our OnHW-equations (WD) datasets. For all trainings, we used Nvidia Tesla V100-SXM2 GPUs with 32 GB VRAM. TapNet [111] has the fastest training time of 3.6s, but we trained 3000 epochs for convergence. While we train our CNN+LSTM model for 1000 epochs with 8.0s each, the MLSTM-FCN [40] trains slower, but converges faster (only 200 epochs). The training times per epoch of the transformer variants [13,36,44,90,101] are significantly higher (between 15.7 and 24.5s), but the convergence is also significantly faster of less than 100 epochs. The Linformer [101] (8.8s) is as fast as our CNN+LSTM model (8.47s). For seq2seq classification tasks, our attention-based model is the fastest with 27.5s. The CNN+TCN model requires 43.5s and the CNN+LSTM model 62s, and can emphasize the advantage of attention-based models. The CNN+BiLSTM model achieves the lowest error rates, but trains clearly slower with 131s. In conclusion, transformers train faster than classical methods, but our classical CNN and RNN models achieve the highest accuracies. Modules trained with the tsai toolbox have lower training times: InceptionTime (2.44s), XceptionTime (3.53s), ResCNN (2.2s) and ResNet (2.9s). The small model FCN is very fast at training (1.5s) that increases for added temporal units such as LSTM-FCN (6.8s) and MLSTM-FCN (7.4s). The training of InceptionTime increases from 2.0s for depth
Fig. 25 Overview of validation accuracies evaluated every $10^{th}$ training epoch for the WD and WI OnHW-symbols, split OnHW-equations and OnHW-chars [65] datasets

Table 11 Comparison of training times per epoch in seconds ($s$)

| Method                  | OnHW-chars | OnHW-equations |
|-------------------------|------------|----------------|
| CNN+LSTM                | 8.0        | 62             |
| CNN+BiLSTM              | 19.7       | 131            |
| CNN+TCN                 | 7.3        | 43.5           |
| Attention-based model   | –          | 27.5           |
| Perceiver [36]          | 17.1       | –              |
| Sinkhorn [90]           | 16.1       | –              |
| Performer [13]          | 15.7       | –              |
| Reformer [44]           | 24.5       | –              |
| Linformer [101]         | 8.8        | –              |
| TapNet [111]            | 3.6        | –              |
| MLSTM-FCN [40]          | 12.0       | –              |

3 and nf 16 up to 37.6s for depth 12 and nf 128. Added BiLSTM layers up to double the training times.

References

1. Abed, H.E., Kherallah, M., Märgner, V., Alimi, A.M.: On-line Arabic handwriting recognition competition: ADAB database and participating systems. IJDAR 4, 15–23 (2010). https://doi.org/10.1109/ICDAR.2011.289
2. Alimoglu, F., Alpaydin, E.: Combining multiple representations and classifiers for pen-based handwritten digit recognition. In: ICDAR, vol. 2. Ulm, Germany (1997). https://doi.org/10.1109/ICDAR.1997.620583
3. Almazán, J., Gordo, A., Fornés, A., Valveny, E.: Word spotting and recognition with embedded attributes. TPAMI 36(12), 2552–2566 (2014). https://doi.org/10.1109/TPAMI.2014.2339814
4. Alonso, M.A.P.: Metacognition and sensorimotor components underlying the process of handwriting and keyboarding and their impact on learning. An analysis from the perspective of embodied psychology. Procedia Soc. Behav. Sci. 176, 263–269 (2015). https://doi.org/10.1016/j.sbspro.2015.01.470
5. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. In: arXiv:1607.06450 (2016)
6. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. In: arXiv:1803.01271 (2018)
7. Barrett, P., Davies, F., Zhang, Y., Barrett, L.: The impact of classroom design on pupils’ learning: final results of a holistic Multi-level analysis. Build. Environ. 89, 118–133 (2015). https://doi.org/10.1016/j.buildenv.2015.02.013
8. Bertolami, R., Bunke, H.: Hidden Markov model-based ensemble methods for offline handwritten text line recognition. Pattern Recogn. 41(11), 3452–3460 (2008). https://doi.org/10.1016/j. patrec.2008.04.003

9. Bluche, T.: Deep neural networks for large vocabulary handwritten text recognition. Dissertation (2015)

10. Bluche, T.: Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. In: NIPS, pp. 838–846. Barcelona (2016)

11. Bu, Y., Xie, L., Ying, Y., Ning, C.W.J., Cao, J., Lu, S.: Handwriting-assistant: reconstructing continuous strokes with millimeter-level accuracy via attachable inertial sensors. IMWUT 5(4), 1–25 (2021). https://doi.org/10.1145/3494956

12. Carbune, V., Gonnet, P., Deselaers, T., Rowley, H.A., Daryin, A., Calvo, M., Wang, L.L., Keysers, D., Feuz, S., Gervais, P.: Fast Multi-language LSTM-based online handwriting recognition. IJADAR 23, 89–102 (2020). https://doi.org/10.1007/s11227-020-00350-4

13. Chormanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., Weller, A.: Rethinking Attention with Performers. In: ICLR (2021)

14. Chowdhury, A., Vig, L.: An efficient end-to-end neural model for handwritten text recognition. In: BMVC (2018)

15. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. In: arXiv:1412.3555 (2014)

16. Damerau, F.J.: A technique for computer detection and correction of spelling errors. Commun. ACM 7(3), 171–176 (1964). https://doi.org/10.1145/363958.363994

17. Deselaers, T., Keysers, D., Hosang, J., Rowley, H.A.: GyroPen: gyroscope for pen-input with mobile phones. THMS 45(2), 263–271 (2015). https://doi.org/10.1109/THMS.2014.2365723

18. Doetsch, P., Kozierski, M., Ney, H.: Fast and robust training of recurrent neural networks for offline handwriting recognition. In: ICFHR, pp. 279–284 (2014). https://doi.org/10.1109/ICFHR.2014.54

19. Dreuw, P., Doetsch, P., Pfahl, C., Ney, H.: Hierarchical hybrid MLP/HMM or rather MLP Fatures for a discriminatively trained Gaussian HMM: A comparison for offline handwriting recognition. In: ICIP, pp. 3541–3544 (2011). https://doi.org/10.1109/ICIP.2011.6116480

20. Dutta, K., Krishnan, P., Mathew, M., Jawahar, C.V.: Improving CNN-RNN hybrid networks for handwriting recognition. In: ICFHR, pp. 80–85 (2018). https://doi.org/10.1109/ICFHR.2018.00023

21. Elsayed, N., Maida, A.S., Bayoumi, M.: Deep gated recurrent and convolutional network hybrid model for univariate time series classification. In: arXiv:1812.07683 (2018)

22. España-Boquera, S., Castro-Bleda, M.J., Gorbe-Moya, J., Zamora-Martinez, F.: Improving Offline handwritten text recognition with hybrid HMM/ANN models. TPAMI 33(4), 767–779 (2010). https://doi.org/10.1109/TPAMI.2010.141

23. Fahmy, M.M.M.: Online signature verification and handwriting classification. ASEJ 1(1), 59–70 (2010). https://doi.org/10.1016/ j.asej.2010.09.007

24. Fauvel, K., Elisa Fromont, Masson, V., Faverdin, P., Termier, A.: XEM: An explainable ensemble method for multivariate time series classification. In: arXiv:2005.03645 (2020)

25. Fawaz, H.I., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D.F., Weberf, J., Webb, G.I., Idoumghar, L., Muller, P.A., Petitjean, F.: InceptionTime: finding AlexNet for Time series classification. In: arXiv:1909.04939 (2019)

26. Fogel, S., Averbuch-Elor, H., Cohen, S., Mazor, S., Litman, R.: ScraibleGAN: semi-supervised varying length handwritten text generation. In: CVPR, pp. 4324–4333 (2020). https://doi.org/10.1109/CVPR42600.2020.00438

27. Frinken, V., Uchida, S.: Deep BLSTM neural networks for unconstrained continuous handwritten text recognition. In: ICDAR, pp. 911–915 (2015). https://doi.org/10.1109/ICDAR.2015.7338894

28. Gerth, S., Klassert, A., Dolk, T., Fleissner, M., Fischer, M.H., Notbusch, G., Festman, J.: Is handwriting performance affected by the writing surface? Comparison of preschoolers’, Second Graders’, and adults’ Writing Performance on a Tablet vs Paper. Front. Psychol. (2016). https://doi.org/10.3389/fpsyg.2016.01308

29. Graves, A.: Generating sequences with recurrent neural networks. In: arXiv:1308.0850 (2014)

30. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: ICML, pp. 369–376. Pittsburgh, PA (2006). https://doi.org/10.1109/IC03844.1143891

31. Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., Schmidhuber, J.: A novel connectionist system for unconstrained handwriting recognition. TPAMI 35(5), 855–868 (2009). https://doi.org/10.1109/TPAMI.2008.137

32. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. In: NIPS, pp. 545–552 (2008)

33. Guyon, I., Schomaker, L., Plamondon, R., Liberman, M., Janet, S.: UNIPEN project of on-line data exchange and recognizer benchmarks. In: ICPR, vol. 3 (1994). https://doi.org/10.1109/ICPR.1994.576870

34. He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., Li, M.: Bag of tricks for image classification with convolutional neural networks. In: CVPR, pp. 558–567. Long Beach, CA (2019). https://doi.org/10.1109/CVPR.2019.00065

35. Hussain, R., Raza, A., Siddiqi, I., Khurshid, K., Djiedjé, C.: A comprehensive survey of handwritten document benchmarks: Structure, usage and evaluation. J. Image Video Process. (2015). https://doi.org/10.1186/s13640-015-0102-5

36. Jaegle, A., Gimeno, F., Brock, A., Zisserman, A., Vinyals, O., Carreira, J.: Perceiver: general perception with iterative attention. In: ICML (2021)

37. Kaita, B., Balakrishnan, V.: An integrated semi-automated framework for domain-based polarity words extraction from an unannotated non-English corpus. J. Supercomput. 76, 9772–9799 (2020). https://doi.org/10.1007/s10917-020-03222-0

38. Kang, L., Riba, P., Rusinol, M., Forés, A., Villegas, M.: Pay attention to what you read: non-recurrent handwritten text-line recognition. In: arXiv:2005.13044 (2020)

39. Karim, F., Majumdar, S., Darabi, H., Chen, S.: LSTM fully convolutional networks for time series classification. In: arXiv:1709.05206 (2017)

40. Karim, F., Majumdar, S., Darabi, H., Harford, S.: Multivariate LSTM-FCNs for time series classification. Neural Netw. 116, 237–245 (2019). https://doi.org/10.1016/j.neunet.2019.04.014

41. Keysers, D., Deselaers, T., Rowley, H.A., Wang, L.L., Carbone, V.: Multi-language online handwriting recognition. TPAMI 36(6), 1180–1194 (2017). https://doi.org/10.1109/TPAMI.2016.2572693

42. Kherallah, M., Elbaati, A., Abed, H.E., Alimi, A.M.: The On/Off (LMCA) Dual Arabic handwriting database. In: ICFHR (2008)

43. Kim, S., Hori, T., Watanabe, S.: Joint CTC-attention based end-to-end speech recognition using multi-task learning. In: arXiv:1609.06773 (2017)

44. Kitaev, N., Kaiser, L., Levskaya, A.: Reformer: the efficient transformer. In: ICLR (2020)

45. Klavš, A., Lorenz, S.M., Lauer-Schmaltz, M.W., Rügamer, D., Bischl, B., Mutschler, C., Ott, F.: Uncertainty-aware evaluation of time-series classification for online handwriting recognition with domain shift. In: IJCAI-ECAI Workshop on Spatio-Temporal
Benchmarking online sequence-to-sequence and character-based handwriting recognition from IMU...
80. Seni, G., Kripäsvard, N., Srihari, R.K.: Generalizing edit distance to incorporate domain information: handwritten text recognition as a case study. Pattern Recogn. 29(3), 405–414 (1996). https://doi.org/10.1016/0031-3203(95)00102-6

81. Seni, G., Srihari, R.K., Nasrabadi, N.: Large vocabulary recognition of on-line handwritten cursive words. TPAMI 18(7), 757–762 (1996). https://doi.org/10.1109/34.506798

82. Sharma, A., Ambati, R., Jayagopi, D.B.: Towards faster offline handwriting recognition using temporal convolutional networks. In: NCVRPIG, pp. 344–354 (2020). https://doi.org/10.1109/ACOMP.2019.00015

83. Sharma, A., Jayagopi, D.B.: Towards efficient unconstrained handwriting recognition using dilated temporal convolutional network. Expert Syst. Appl. (2021). https://doi.org/10.1016/j.eswa.2020.114004

84. Shivram, A., Ramaiah, C., Setlur, S., Govindaraju, V.: IBM_UB_1: a dual mode unconstrained english handwriting dataset. In: ICDAR, pp. 13–17 (2013). https://doi.org/10.1109/ICDAR.2013.12

85. Sudholt, S., Fink, G.A.: Attribute CNNs for word spotting in handwritten documents. IJDAIR 21, 199–218 (2018). https://doi.org/10.1007/s10032-018-0295-0

86. Synnaeve, G., Xu, Q., Kahn, J., Likhomanenko, T., Grave, E., Pratap, V., Srinan, A., Liptchinsky, V., Collobert, R.: End-to-End ASR: from supervised to semi-supervised learning with modern architectures. In: ICML Workshop. Vienna, Austria (2020)

87. Tan, C.W., Dempster, A., Bergmeir, C., Webb, G.I.: MultiRocket: multiple pooling operators and transformations for fast and effective time series classification. In: arXiv:2102.00457 (2021)

88. Tanaka, D., Ikami, D., Yamasaki, T., Aizawa, K.: Joint optimization framework for learning with noisy labels. In: CVPR, pp. 5552–5560. Salt Lake City, UT (2018). https://doi.org/10.1109/CVPR.2018.00582

89. Tang, W., Long, G., Liu, L., Zhou, T., Jiang, J., Blumenstein, M.: Rethinking 1D-CNN for time series classification: a stronger baseline. In: arXiv:2002.10061 (2020)

90. Tay, Y., Bahri, D., Yang, L., Metzler, D., Juan, D.C.: Sparse Sinkhorn attention. In: arXiv:2002.11296 (2020)

91. Tian, B., Zhang, Y., Wang, J., Xing, C.: Hierarchical inter-attention network for document classification with multi-task learning. In: IJCAI, pp. 3569–3575 (2019). https://doi.org/10.24963/ijcai.2019/495

92. Uhang, J., Du, J., Yang, Y., Song, Y.Z., Dai, L.: SRD: a tree structure based decoder for online handwritten mathematical expression recognition. Trans. Multimed. 23, 2471–2480 (2020)

93. Um, T.T., Pfister, F.M.J., Pichler, D., Endo, S., Lang, M., Hirche, S., Fietzek, U., Kulic, D.: Data augmentation of wearable sensor data for Parkinson's disease monitoring using convolutional neural networks. In: ICI, pp. 216–220. Glasgow, UK (2017). https://doi.org/10.1145/3136755.3136817

94. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.: Attention is all you need. In: NIPS, pp. 5998–6008. Long Beach, CA (2017)

95. Viard-Gaudin, C., Lallican, P.M., Binter, P., Knerr, S.: The IRESTE On/Off (IRONOFF) dual handwriting database. In: ICDAR, pp. 455–458 (1999). https://doi.org/10.1109/ICDAR.1999.791823

96. Vinciarelli, A., Perrone, M.P.: Combining online and offline handwriting recognition. In: ICDAR, pp. 844–848. Edinburgh, UK (2003). https://doi.org/10.1109/ICDAR.2003.1227781

97. Voigtlaender, P., Doetsch, P., Ney, H.: Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In: ICIFHR, pp. 228–233 (2016). https://doi.org/10.1109/ICFHR.2016.0052

98. Wang, D.H., Liu, C.L., Zhou, X.D.: An approach for real-time recognition of online Chinese handwritten sentences. Pattern Recogn. 45(10), 3661–3675 (2012). https://doi.org/10.1016/j.patcog.2012.04.020

99. Wang, J., Wang, Z., Li, J., Wu, J.: A transformer-based framework for multivariate time series representation learning. In: SIGKDD, pp. 2347–2446 (2018). https://doi.org/10.1145/3219819.3220060

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.