Auxiliary Cross-Modal Common Representation Learning with Triplet Loss Functions for Online Handwriting Recognition

Anonymous authors
Paper under double-blind review

Abstract

Common representation learning (CRL) learns a shared embedding between two or more modalities to improve in a given task over using only one of the modalities. CRL from different data types such as images and time-series data (e.g., audio or text data) requires a deep metric learning loss that minimizes the distance between the modality embeddings. In this paper, we propose to use the triplet loss, which uses positive and negative identities to create sample pairs with different labels, for CRL between image and time-series modalities. By adapting the triplet loss for CRL, higher accuracy in the main (time-series classification) task can be achieved by exploiting additional information of the auxiliary (image classification) task. Our experiments on synthetic data and handwriting recognition data from sensor-enhanced pens show an improved classification accuracy, faster convergence, and a better generalizability.

1 Introduction

Cross-modal retrieval such as common representation learning (CRL) for learning across two or more modalities (i.e., image, audio, text and 3D data) has attracted a lot of attention recently. It can be applied in a wide range of applications such as multimedia management and identification. Extracting information from several modalities and adapting the domain with cross-modal learning allows to use information in all domains. CRL, however, remains challenging due to the heterogeneity gap (inconsistent representation forms of different modalities).

A limitation of CRL is that most approaches require the availability of all modalities at inference time. However, in many applications certain data sources are only available during training by means of elaborate laboratory setups. For instance, consider a human pose estimation task that uses inertial sensors together with RGB videos during training. A camera setup might not be available at inference time due to bad lighting conditions or other application-specific restrictions. This requires a model that allows inference on the main modality only, while auxiliary modalities may only be used to improve the training process (as they are not available at inference time).

Learning using privileged information is one approach in the literature that describes and tackles this problem. During training it is assumed that in addition to X*, the privileged information, is available which is, however, not present in the inference stage. For CRL, we need a deep metric learning (DML) technique that aims to transform training samples into feature embeddings that are close for samples that belong to the same class and far apart for samples from different classes. As DML requires no model update (simply fine-tuning for training samples of new classes), DML is an interesting approach for continual learning. Typical DML methods use simple distances (e.g., Euclidean distance) but also highly complex distances (e.g., canonical correlation analysis and maximum mean discrepancy). While CRL learns representations from all modalities, single-modal learning commonly uses pair-wise learning. The triplet loss selects a positive and negative triplet pair for a corresponding anchor and forces the positive pair distance to be smaller than the negative pair distance.
Figure 1: **Method overview:** Common representation learning between image and time-series data using the triplet loss based on DML functions to improve the time-series classification task.

While research of triplet selection for single-modal classification is very advanced [Do et al. (2019)], pair-wise selection for CRL is investigated for specific applications only [Zhen et al. (2015); Lee et al. (2020); Zhang & Zheng (2020)]. One exemplary application for cross-modal learning is handwriting recognition (HWR). HWR can be categorized into offline and online HWR. Offline HWR such as optical character recognition (OCR) deals with the analysis of the visual representation of handwriting only, but cannot be applied for real-time recognition applications [Fahmy (2010)]. Online HWR works on different types of spatio-temporal signals and can make use of temporal information such as writing speed and direction [Plamondon & Srihari (2000)]. Many recording systems make use of a stylus pen together with a touch screen surface [Alimoğlu & Alpaydín (1997)]. Systems for writing on paper are only prototypical [Chen et al. (2021); Schrapel et al. (2018); Wang et al. (2013); Deselaers et al. (2015)] and cannot be applied for real-world applications. A novel sensor-enhanced pen based on inertial measurement units (IMUs) enables new applications for writing on normal paper. This pen has previously been used for single character [Ott et al. (2020, 2022a)] and sequence [Ott et al. (2022c)] classification. However, the accuracy of previous online HWR methods is limited due to the limited size of datasets as recording of larger amounts of data is time consuming. A possible solution is to combine datasets of different modalities using common representation learning to increase the generalizability. In this work, we combine offline HWR from generated images (i.e., OCR) and online HWR from sensor-enhanced pens by learning a common representation between both modalities. The aim is to integrate information of OCR, i.e., typeface, cursive or printed writing, and font thickness, into the online HWR task, i.e., writing speed and direction [Vinciarelli & Perrone (2003)].

**Our Contribution.** Models that use rich data (e.g., images) usually outperform those that use a less rich modality (e.g., time-series). We therefore propose to train a shared representation using the triplet loss between pairs of image and time-series data to learn a common representation between both modality embeddings (cf. Figure 1). This allows to improve the accuracy for single-modal inference in the main task. We prove the efficacy of our DML-based triplet loss for CRL both with simulated data and in a real-world application. More specifically, our proposed CRL technique 1) improves the multivariate time-series (MTS) classification accuracy and convergence, 2) results in a small MTS-only network independent from the image modality while allowing for fast inference, and 3) has better generalizability and adaptability [Huang et al. (2020)]. Code and datasets are available upon publication.[1]

The paper is organized as follows. Section 2 discusses related work followed by the mathematical foundation of our method in Section 3. The experimental setup is described in Section 4 and the results are discussed in Section 5.

## 2 Related Work

In this section, we discuss related work, in particular approaches for learning a common representation from different modalities (in Section 2.1) and DML (in Section 2.2) to minimize the distance between feature embeddings.

### 2.1 Cross-Modal Representation Learning

For traditional methods that learn a common representation, a cross-modal similarity for the retrieval can be calculated with linear projections as basic models [Rasiwasia et al. (2010)]. However, cross-modal correlation is highly complex,
and hence, recent methods are based on a modal-sharing network to jointly transfer non-linear knowledge from a single modality to all modalities \cite{wei2016cross}. \cite{huang2020cross} use a cross-modal network between different modalities (image to video, text, audio and 3D models) and a single-modal network (shared features between images of source and target domains). They use two convolutional layers (similar to our proposed architecture) that allows the model to adapt more trainable parameters. However, while their auxiliary network uses the same modality, our auxiliary network is based on another modality. \cite{lee2020learning} learn a common embedding between video frames and audio signals with graph clusters, but at inference both modalities must be available. \cite{sarafianos2019multimodal} proposed an image-text modality adversarial matching approach that learns modality-invariant feature representations, but their projection loss is used for learning discriminative image-text embeddings only. \cite{hafner2022learning} propose a model for single-modal inference. However, they use image and depth modalities for person re-identification without a time-series component, which makes the problem considerably different. \cite{lim2019end} handled multisensory modalities for 3D models only.

2.2 Deep Metric Learning

Networks trained for the classification task can produce useful feature embeddings with efficient runtime complexity $O(NC)$ per epoch, where $N$ is the number of training samples and $C$ the number of classes. The classical cross-entropy (CE) loss, however, is not useful for DML as it ignores how close each point is to its class centroid (or how far apart from other class centroids). The pairwise contrastive loss \cite{chopra2005learning} minimizes the distance between feature embedding pairs of the same class and maximizes the distance between feature embedding pairs of different classes dependent on a margin parameter. The issue is that the optimization of positive pairs is independent from negative pairs, but the optimization should force the distance between positive pairs to be smaller than negative pairs \cite{do2019three}. The triplet loss \cite{yoshida2019three} addresses this by defining an anchor and a positive as well as a negative point, and forces the positive pair distance to be smaller than the negative pair distance by a certain margin. The runtime complexity of the triplet loss is $O(N^3/C)$, and can be computationally challenging for large training sets. Hence, several works exist to reduce this complexity such as hard or semi-hard triplet mining \cite{schroff2015facenet}, or smart triplet mining \cite{harwood2017smart}. Often, data is evolving over time, and hence, \cite{semedo2020time} proposed a formulation of the triplet loss where the traditional static margin is superseded by a temporally adaptive maximum margin function. While \cite{zeng2017triplet,li2021online} combine the triplet loss with the CE loss, \cite{guo2019using} use a triplet selection with $L_2$-normalization for language modeling, but considered all negative pairs for triplet selection with fixed similarity intensity parameter. For our experiments, we use a triplet loss with a dynamic margin together with a novel word level triplet selection. The DeepTripletNN \cite{zeng2020deep} also uses the triplet loss on embeddings between an anchor from audio data and positive and negative samples from visual data, and the cosine similarity for the final representation comparison. CrossATNet \cite{chaudhuri2020cross}, another triplet loss-based method which uses single class labels, defines class sketch instances as anchor, the same class image instance as positive sample and a different class image instance as negative sample. While the previous methods are based on a triplet selection method using single label classification, we are – to the best of our knowledge – the first to propose the triplet loss for sequence-based classification (i.e., words).

Most of the related work uses the Euclidean metric as distance loss, although the triplet loss can be defined based on any other (sub-)differentiable distance metric. \cite{wan2021offline} proposed a method for offline signature verification based on a dual triplet loss that uses the Euclidean space to project an input image to an embedding function. While \cite{rantsch2016offline} use the Euclidean metric to learn the distance between feature embeddings, \cite{zeng2017triplet} use the Cosine similarity. \cite{hermans2017defense} state that using the non-squared Euclidean distance is more stable, while the squared distance made the optimization more prone to collapsing. Recent methods extend the canonical correlation analysis (CCA) \cite{ranjan2015unsupervised} that learns linear projection matrices by maximizing pairwise correlation of cross-modal data. To share information between the same modality (i.e., images), typically the maximum mean discrepancy (MMD) \cite{long2015 Há} is minimized.

3 Methodology

We define the problem of common representation learning and present DML loss functions in Section 3.1. In Section 3.2 we propose the triplet loss for cross-modal learning.
3.1 Common Representation Learning

A multivariate time-series (MTS) \( \mathbf{U} = \{ \mathbf{u}_1, \ldots, \mathbf{u}_m \} \in \mathbb{R}^{m \times l} \) is an ordered sequence of \( l \in \mathbb{N} \) streams with \( \mathbf{u}_i = (u_{i,1}, \ldots, u_{i,l}), i \in \{1, \ldots, m\} \), where \( m \in \mathbb{N} \) is the length of the time-series. The MTS training set is a subset of the array \( \mathcal{U} = \{ \mathbf{U}_1, \ldots, \mathbf{U}_{n_U} \} \subseteq \mathbb{R}^{n_U \times m \times l} \), where \( n_U \) is the number of time-series. Let \( \mathbf{X} \in \mathbb{R}^{n \times p} \) with entries \( x_{i,j} \in [0, 255] \) represent an image from the image training set. The image training set is a subset of the array \( \mathcal{X} = \{ \mathbf{X}_1, \ldots, \mathbf{X}_{n_X} \} \subseteq \mathbb{R}^{n_X \times n \times p} \), where \( n_X \) is the number of time-series. The aim of joint MTS and image classification tasks is to predict an unknown class label \( v \in \Omega \) for single class prediction or \( v \in \Omega \) for sequence prediction for a given MTS or image (see also Section 4.2). The time-series samples denote the main training data, while the image samples represent the privileged information that is not used for inference. In addition to good prediction performance, the goal is to learn representative embeddings \( f_c(\mathbf{U}) \) and \( f_c(\mathbf{X}) \in \mathbb{R}^{q \times w} \) to map MTS and image data into a feature space \( \mathbb{R}^{q \times w} \), where \( f_c \) is the output of the convolutional layer(s) \( c \in \mathbb{N} \) of the latent representation.

We force the embedding to live on the \( q \times w \)-dimensional hypersphere by using a Softmax attention, i.e., \( \| f_c(\mathbf{U}) \|_2 = 1 \) and \( \| f_c(\mathbf{X}) \|_2 = 1 \forall c \) (see (Weinberger et al., 2005)). In order to obtain a small distance between the embeddings \( f_c(\mathbf{U}) \) and \( f_c(\mathbf{X}) \), we minimize DML functions \( \mathcal{L}_{\text{DML}}(f_c(\mathbf{X}), f_c(\mathbf{U})) \). Well-known DML metrics are the distance-based mean squared error (MSE) \( \mathcal{L}_{\text{MSE}} \), the spatio-temporal cosine similarity (CS) \( \mathcal{L}_{\text{CS}} \), the Pearson correlation (PC) \( \mathcal{L}_{\text{PC}} \), or the distribution-based Kullback-Leibler (KL) divergence \( \mathcal{L}_{\text{KL}} \). In our experiments, we additionally evaluate the kernelized maximum mean discrepancy (kMMD) \( \mathcal{L}_{\text{MMD}} \), Bray Curtis (BC) \( \mathcal{L}_{\text{BC}} \), and Poisson \( \mathcal{L}_{\text{PO}} \) losses. We study their performance in Section 5. A combination of classification and CRL losses can be realized by dynamic weight averaging (Liu et al., 2019) as a multi-task learning approach that performs dynamic task weighting over time (see Appendix A.1).

3.2 Triplet Loss

While the training with the previous loss functions uses inputs where the image and MTS have the same label, pairs with similar but different labels can improve the training process. This can be achieved using the triplet loss (Schroff et al., 2015) which enforces a margin between pairs of image and MTS data with the same identity to all other different identities. As a consequence, the convolutional output for one and the same label lives on a manifold, while still enforcing the distance and thus discriminability to other identities. We therefore seek to ensure that the embedding of the MTS \( \mathbf{U}^p \) (anchor) of a specific label is closer to the embedding of the image \( \mathbf{X}^p \) (positive) of the same label than it is to the embedding of any image \( \mathbf{X}^n \) (negative) of another label (see Figure 2). Thus, we want the following inequality to hold for all training samples \( (f_c(\mathbf{U}^p), f_c(\mathbf{X}^p), f_c(\mathbf{X}^n)) \in \Phi \):

\[
\mathcal{L}_{\text{DML}}(f_c(\mathbf{U}^p), f_c(\mathbf{X}^p)) + \alpha < \mathcal{L}_{\text{DML}}(f_c(\mathbf{U}^p), f_c(\mathbf{X}^n)),
\]

where \( \mathcal{L}_{\text{DML}}(f_c(\mathbf{X}), f_c(\mathbf{U})) \) is a DML loss, \( \alpha \) is a margin between positive and negative pairs, and \( \Phi \) is the set of all possible triplets in the training set. Based on (1), we can formulate a differentiable loss function that we can use for optimization:

\[
\mathcal{L}_{\text{triplet}}(\mathbf{U}^p, \mathbf{X}^p, \mathbf{X}^n) = \sum_{i=1}^{N} \max \left[ \mathcal{L}_{\text{DML}}(f_c(\mathbf{U}^p), f_c(\mathbf{X}^p)) - \mathcal{L}_{\text{DML}}(f_c(\mathbf{U}^p), f_c(\mathbf{X}^n)) + \alpha, 0 \right],
\]

where \( c \in \mathbb{N} \). Selecting negative samples that are too close to the anchor (in relation to the positive sample) can cause slow training convergence. Hence, triplet selection must be handled carefully and application-specific (Do et al., 2019). We choose negative samples based on the class distance (single labels) and on the Edit distance (sequence labels), see Section 4.2.

\(^2\)To have a larger number of trainable parameters in the latent representation with a greater depth, we evaluate one and two stacked convolutional layers, each trained with a shared loss \( \mathcal{L}_{\text{triplet}} \).
4 Experiments

We now demonstrate the efficacy of our proposal. In Section 4.1 we generate sinusoidal time-series with introduced noise (main task) and compute the corresponding Gramian angular summation field (GASF) with different noise parameters (auxiliary task), see Figure 3a. In Section 4.2 we combine online (inertial sensor signals, main task) and offline data (visual representations, auxiliary task) for handwriting recognition (HWR) with sensor-enhanced pens. This task is particularly challenging due to different data representations based on images and MTS data. For both applications, our approach allows to only use the main modality (MTS) for inference. We further analyze and evaluate different DML functions to minimize the distance between the learned embeddings.

4.1 Cross-Modal Learning on Synthetic Data

We first investigate the influence of the triplet loss for cross-modal learning between synthetic time-series and image-based data. For this, we generate signal data of 1,000 timesteps with different frequencies for 10 classes (see Figure 3a) and add noise from a continuous uniform distribution $U(a, b)$ for $a = 0$ and $b = 0.3$. We use a recurrent CNN with the CE loss to classify these signals. From each signal without noise, we generate a GASF Wang & Oates (2015). For classes with high frequencies, this results in a fine-grained pattern, and for low frequencies in a coarse-grained pattern. We generate GASFs with different added noise between $b = 0$ (Figure 3b) and $b = 1.95$ (Figure 3c). A small CNN classifies these images with the CE loss. To combine both networks, we train each signal-image pair with the triplet loss. As the frequency of the sinusoidal signal is closer for more similar class labels, the distance in the manifold embedding should also be closer. For each batch, we select negative sample pairs for samples with the class label $CL = 1 + \left\lfloor \frac{\max_e - e - 1}{25} \right\rfloor$ as lower bound for current epoch $e$ and maximum epoch $\max_e$. We set the margin $\alpha$ in the triplet loss separately for each batch such that $\alpha = \beta \cdot (CL_p - CL_n)$ depends on the positive $CL_p$ and negative $CL_n$ class labels of the batch and is in the range $[1, 5]$ with $\beta = 0.1$. The batch size is 100 and $\max_e = 100$. Appendix A.2 provides further details. This combination of the CE loss with the triplet loss can lead to a mutual improvement of the utilization of the classification task and embedding learning.

4.2 Cross-Modal Learning for HWR

Method Overview. Figure 4 gives a method overview. The main task is online HWR to classify words written with a sensor-enhanced pen and represented by MTS of the different pen sensors. To improve the classification task with a better generalizability, the auxiliary network performs offline HWR based on an image input. We pre-train ScrabbleGAN Fogel et al. (2020) on the IAM-OffDB Liwicki & Bunke (2005) dataset and for all MTS word labels generate the corresponding image as the positive MTS-image pair. Each MTS and each image 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. The MTS training set is a subset of the array $U$ with labels $\mathcal{V}_U = \{v_1, \ldots, v_{nu}\} \in \Omega^{nu \times L}$. The image training set is a subset of the array $X$ and the corresponding labels $\mathcal{V}_X = \{v_1, \ldots, v_{nx}\} \in \Omega^{nx \times L}$. Offline HWR techniques are based on Inception, ResNet34, or GTR Yousef et al. (2018) modules. The online method is improved by sharing layers with a common representation by minimizing the distance of the feature embedding of the convolutional layers $c \in \{1, 2\}$ (integrated in both networks) with a shared loss $L_{\text{shared}, c}$. We set the embedding size $\mathbb{R}^{q \times w}$ to $400 \times 200$. Both networks are trained with the connectionist temporal classification (CTC) Graves.
Datasets for Online HWR. We make use of two word datasets proposed in Ott et al. (2022c). These datasets are recorded with a sensor-enhanced pen that uses two accelerometers (3 axes each), one gyroscope (3 axes), one magnetometer (3 axes), and one force sensor at 100 Hz. One sample of size $m \times l$ represents an MTS of a written word of $m$ timesteps from $l = 13$ sensor channels. One word is a sequence of small or capital characters (52 classes) or with mutated vowels (59 classes). The OnHW-words500 dataset contains 25,218 samples where each of the 53 writers contributed the same 500 words. The OnHW-wordsRandom dataset contains 14,641 randomly selected words from 54 writers. For both datasets, 80/20 train/validation splits are available for writer-(in)dependent (WD/WI) tasks. We transform (zero padding, interpolation) all samples to 800 timesteps.

Image Generation for Offline HWR. In order to couple the online MTS data with offline image data, we use a generative adversarial network (GAN) to generate arbitrarily many images. ScrabbleGAN [Fogel et al., 2020] is a state-of-the-art semi-supervised approach that consists of a generator $G$ that generates images of words with arbitrary length.
Methods for Offline HWR. OrigamiNet [Yousef & Bishop (2020)] is a state-of-the-art multi-line recognition method using only unsegmented image and text pairs. An overview of offline HWR methods is given in Appendix A.3. Similar to OrigamiNet, our offline method is based on different encoder architectures with one or two additional 1D convolutional layers (each with filter size 200, Softmax activation [Zeng et al. (2017)]) with 20% dropout for the latent representation, and a common representation decoder with BiLSTMs. For the encoder, we make use of Inception modules from GoogLeNet, the ResNet34 architecture, and re-implement the newly proposed gated, fully convolutional method gated text recognizer (GTR) [Yousef et al. (2018)]. See Appendix A.4 for detailed information on the architectures. We train the networks on the generated OffHW-German dataset for 10 epochs, and fine-tune on the OffHW-wordsRandom datasets for 15 epochs. For comparison with state-of-the-art techniques, we train OrigamiNet and compare with IAM-OffDB. For OrigamiNet, we apply interline spacing reduction via seam carving [Avidan & Shamir (2007)], resizing the images to 50% height, and random projective (rotating and resizing lines) and random elastic transform [Wigington et al. (2017)]. We augment the OffHW-German dataset with random width resizing and apply no augmentation for the OffHW-words500, wordsRandom datasets for fine-tuning.

Offline/Online Common Representation Learning. Our architecture for online HWR is based on [Ott et al. (2022c)]. The encoder extracts features of the inertial data and consists of three convolutional layers (each with filter size 400, ReLU activation) and one convolutional layer (filter size 200, ReLU activation), a max pooling, batch normalization and a 20% dropout layer. As for the offline architecture, the network then learns a latent representation with one or two convolutional layers (each with filter size 200, Softmax activation) with 20% dropout and the same CRL decoder. The output of the convolutional layers of the latent representation are minimized with the $L_{\text{shared},c}$ loss. The layers of the common representation are fine-tuned based on the pre-trained weights of the offline technique. Here, two BiLSTM layers with 60 units each and ReLU activation extract the temporal context of the feature embedding. As for the baseline classifier, we train for 1,000 epochs. For evaluation, the main MTS network is independent of the image auxiliary network by using only the weights of the main network.

Triplet Selection. To ensure (fast) convergence, it is crucial to select triplets that violate the constraint from Equation 1. Typically, it is infeasible to compute the loss for all triplet pairs or this leads to poor training performance as poorly chosen pairs dominate hard ones. This requires an elaborate triplet selection [Do et al. (2019)]. We use the Edit distance (ED) to define the identity and select triplets. The ED is the minimum number of substitutions $S$, insertions $I$ and deletions $D$ required to change the sequences $h = (h_1, \ldots, h_r)$ into $g = (g_1, \ldots, g_t)$ with length $r$ and $t$, respectively.
5 Experimental Results

5.1 Evaluation of Synthetic Data

We train the time-series (TS) model 18 times with noise \( b = 0.3 \), and the combined model with the triplet loss for all 40 noise combinations \( \{ b \in \{0, \ldots, 1.95\} \} \) with different DML functions. Figure 7 shows the validation accuracy averaged over all trainings as well as the combined cases separately for noise \( b < 0.2 \) and noise \( 0.2 \leq b < 2.0 \) (for the \( L_{CS} \) loss). The accuracy of the models that use only images and in combination with MTS during inference reach an accuracy of 99.7% (which can be seen as an unreachable upper bound for the TS-only models). The triplet loss improves the final TS baseline accuracy from 92.5% to 95.36% (averaged over all combinations) while combining TS and image data leads to a faster convergence. Conceptually similar to Long et al. (2015), we use the \( L_{kMMD} \) loss which yields 95.83% accuracy. The \( L_{PC} \) (96.03%), \( L_{KL} \) (96.22%), \( L_{MSE} \) (96.25%), \( L_{BC} \) (96.62%), and \( L_{PO} \) (96.76%) loss functions can further improve the accuracy. We conclude that the triplet loss can be successfully used for cross-modal learning by utilizing negative identities.

5.2 Evaluation of HWR

Evaluation Metrics. A metric for sequence evaluation is the character error rate (CER) defined as \( \text{CER} = \frac{S_c + I_c + D_c}{N_c} \) as the Edit distance (the sum of character substitutions \( S_c \), insertions \( I_c \) and deletions \( D_c \)) divided by the total number of characters in the set \( N_c \). Similarly, the word error rate (WER) is defined as \( \text{WER} = \frac{S_w + I_w + D_w}{N_w} \) computed with word operations \( S_w, I_w \) and \( D_w \), and number of words in the set \( N_w \).

Evaluation of Offline HWR Methods. All our models yield low error rates on the generated OffHW-German dataset. Our approach with GTR blocks outperforms (0.24% to 0.44% CER) the models with Inception (1.27% CER) and ResNet (1.24% CER). OrigamiNet achieves the lowest error rates of 1.50% WER and 0.11% CER. Four GTR blocks yield the best results at a significantly lower training time compared to six or eight blocks. We fine-tune the model with four GTR blocks for one and two convolutional layers and achieve notably low error rates between 0.22% to 0.76% CER, and between 0.85% to 2.95% WER on the OffHW-[words500, wordsRandom] datasets. For more results, see Appendix A.5.

Evaluation of CRL Feature Embeddings. Table 1 shows the feature embeddings for image \( f_2(X_i) \) and MTS data \( f_2(U_i) \) of the positive sample Export and the two negative samples Expert (ED = 1) and Import (ED = 2) based on four DML loss functions. The pattern of characters are similar as the words differ only in the fourth letter.
Table 1: Feature embeddings $f_c(X_i)$ and $f_c(U_i)$ of exemplary image $X_i$ and MTS $U_i$ data of the convolutional layer $c = \text{conv}_2$ for different deep metric learning functions for positive pairs ($ED = 0$) and negative pairs ($ED > 0$) trained with the triplet loss. The feature embeddings are similar in the red box (character $x$) or blue box (character $p$) for $f_2(X_i)$, or the last pixels (character $t$) of $f_2(U_i)$ for $L_{PC}$ marked green.

<table>
<thead>
<tr>
<th>ED</th>
<th>Label</th>
<th>Image $U_i$</th>
<th>$f_2(X_i)$</th>
<th>$f_2(U_i)$: $L_{MSE}$</th>
<th>$f_2(U_i)$: $L_{CS}$</th>
<th>$f_2(U_i)$: $L_{PC}$</th>
<th>$f_2(U_i)$: $L_{KL}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Export</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="feature_embedding1.png" alt="Feature Embedding" /></td>
<td><img src="feature_embedding2.png" alt="Feature Embedding" /></td>
<td><img src="feature_embedding3.png" alt="Feature Embedding" /></td>
<td><img src="feature_embedding4.png" alt="Feature Embedding" /></td>
<td><img src="feature_embedding5.png" alt="Feature Embedding" /></td>
</tr>
<tr>
<td>1</td>
<td>Expert</td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="feature_embedding1.png" alt="Feature Embedding" /></td>
<td><img src="feature_embedding2.png" alt="Feature Embedding" /></td>
<td><img src="feature_embedding3.png" alt="Feature Embedding" /></td>
<td><img src="feature_embedding4.png" alt="Feature Embedding" /></td>
<td><img src="feature_embedding5.png" alt="Feature Embedding" /></td>
</tr>
<tr>
<td>2</td>
<td>Import</td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="feature_embedding1.png" alt="Feature Embedding" /></td>
<td><img src="feature_embedding2.png" alt="Feature Embedding" /></td>
<td><img src="feature_embedding3.png" alt="Feature Embedding" /></td>
<td><img src="feature_embedding4.png" alt="Feature Embedding" /></td>
<td><img src="feature_embedding5.png" alt="Feature Embedding" /></td>
</tr>
<tr>
<td>3</td>
<td>Vorort</td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="feature_embedding1.png" alt="Feature Embedding" /></td>
<td><img src="feature_embedding2.png" alt="Feature Embedbing" /></td>
<td><img src="feature_embedding3.png" alt="Feature Embedding" /></td>
<td><img src="feature_embedding4.png" alt="Feature Embedding" /></td>
<td><img src="feature_embedding5.png" alt="Feature Embedding" /></td>
</tr>
</tbody>
</table>

(a) Feature embedding of IMU samples for the single modality network.  
(b) Feature embeddings of IMU and image samples for the cross-modal network.

Figure 8: Plot of $400 \times 200$ feature embeddings of image and IMU modalities with t-SNE.

In contrast, Import has a different feature embedding as the replacement of E with I and x with m leads to a higher feature distance in the embedding hypersphere. Note that image and MTS data can vary in length for $ED > 0$. Figure 8 shows the feature embeddings of the output of the convolutional layers ($c = 1$) processed with t-SNE by van der Maaten & Hinton (2008). Figure 8a visualizes the MTS embeddings $f_1(U_i)$ of the single modal network, and Figure 8b visualizes the MTS and image embeddings, $f_1(U_i)$ and $f_1(X_i)$ respectively, in a cross-modal setup. While the embedding of the single modal network is unstructured, the embeddings of the cross-modal network are structured (distance of samples visualizes the Edit distance between words).

Evaluation of Cross-Modal CRL. Table 2 gives an overview of CRL (for $c = 1$). The first row are baseline results by Ott et al. (2022c): 13.04% CER on OnHW-words500 (WD) and 6.75% CER on OnHW-wordsRandom (WD) with mutated vowels (MV). Compared to various time-series classification techniques, their benchmark results showed superior performance of CNN+BiLSTMs on these OnHW recognition tasks. In general, the word error rate (WER) can vary for a similar character error rate (CER). The reason is that a change of one character of a correctly classified
Table 2: Evaluation results (WER and CER in %) average over five splits of the baseline MTS-only technique and our cross-modal learning technique for the inertial-based OnHW datasets [Ott et al. (2022c)] with and without mutated vowels (MV) for one convolutional layer $c = 1$.

<table>
<thead>
<tr>
<th>Method</th>
<th>OnHW-words500</th>
<th></th>
<th>OnHW-wordsRandom</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WD</td>
<td>WI</td>
<td>WD</td>
<td>WI</td>
</tr>
<tr>
<td>$\mathcal{L}_{\text{CTC}}$, w/ MV</td>
<td>42.81</td>
<td>13.04</td>
<td>60.47</td>
<td>28.30</td>
</tr>
<tr>
<td></td>
<td>42.77</td>
<td>13.44</td>
<td>59.82</td>
<td>28.54</td>
</tr>
<tr>
<td>$\mathcal{L}_{\text{MSE}}$</td>
<td>40.76</td>
<td>12.71</td>
<td>55.54</td>
<td>25.97</td>
</tr>
<tr>
<td>$\mathcal{L}_{\text{CS}}$</td>
<td>38.62</td>
<td>11.55</td>
<td>56.37</td>
<td>25.90</td>
</tr>
<tr>
<td>$\mathcal{L}_{\text{PC}}$</td>
<td>39.09</td>
<td>11.69</td>
<td>57.90</td>
<td>27.23</td>
</tr>
<tr>
<td>$\mathcal{L}_{\text{KL}}$</td>
<td>38.36</td>
<td>11.28</td>
<td>60.23</td>
<td>27.99</td>
</tr>
<tr>
<td>$\mathcal{L}<em>{\text{trpl,1}}(\mathcal{L}</em>{\text{MSE}})$</td>
<td>42.95</td>
<td>14.13</td>
<td>56.48</td>
<td>26.66</td>
</tr>
<tr>
<td>$\mathcal{L}<em>{\text{trpl,1}}(\mathcal{L}</em>{\text{CS}})$</td>
<td>38.01</td>
<td>11.29</td>
<td>58.50</td>
<td>27.10</td>
</tr>
<tr>
<td>$\mathcal{L}<em>{\text{trpl,1}}(\mathcal{L}</em>{\text{PC}})$</td>
<td>40.43</td>
<td>12.41</td>
<td>58.20</td>
<td>27.48</td>
</tr>
<tr>
<td>$\mathcal{L}<em>{\text{trpl,1}}(\mathcal{L}</em>{\text{KL}})$</td>
<td><strong>37.55</strong></td>
<td><strong>11.21</strong></td>
<td>63.52</td>
<td>30.52</td>
</tr>
</tbody>
</table>

A word leads to a large change in the WER, while the change of the CER is marginal. We compare to results without MV as ScrabbleGAN is pretrained on IAM-OfDB that does not contain MV, and hence, such words cannot be generated. Here, the error rates are slightly higher for both datasets. As expected, cross-modal learning improves the baseline results up to 11.28% CER on the OnHW-words500 WD dataset and up to 7.01% CER on the OnHW-wordsRandom WD dataset. With triplet loss, $\mathcal{L}_{\text{CS}}$ outperforms other metrics on the OnHW-wordsRandom dataset, but is inconsistent on the OnHW-words500 dataset. The importance of the triplet loss is more significant for one convolutional layer ($c = 1$) than for two convolutional layers ($c = 2$), see Appendix A.5. Further, training with kMMD (implemented as in [Long et al., 2015]) does not yield reasonable results. We assume that this metric cannot make use of the important time component in the HWR application.

6 Conclusion

We evaluated DML-based triplet loss functions for CRL between image and time-series modalities with class label specific triplet selection. On synthetic data as well as on different HWR datasets, our method yields notable accuracy improvements for the main time-series classification task and can be decoupled from the auxiliary image classification task at inference time. Our cross-modal triplet selection further yields a faster training convergence with better generalization on the main task.

Broader Impact Statement

While research for offline handwriting recognition (HWR) is very advanced (an overview is proposed in the appendix), research for online HWR from sensor-enhanced only emerged in 2019. Hence, the methodological research currently does not meet the requirements for real-world applications. Handwriting is still important in different fields, in particular graphomotoric. The visual feedback provided by the pen helps young students to learn a new language. A well-known bottleneck for many machine learning algorithms is their requirement for large amounts of datasets, while data recording of handwriting data is time consuming. This paper extends the online HWR dataset with generated images from offline handwriting, and closes the gap between offline and online HWR by using offline HWR as auxiliary task by learning with privileged information. One downside of training the offline architecture (consisting of GTR blocks) is its long training time. But as this model is not required at inference time, processing the time-series is still fast. The common representation between both modalities (image and time-series) is achieved by using the triplet loss and a sample selection depending on the Edit distance. This approach is important in future applications of sequence-based classification as the triplet loss may also evolve for language processing applications as strong as in typically applied fields such as image recognition. Ethical statement about collection consent and personal information: For data recording, the consent of all participants was collected. The datasets only contain the raw data from the sensor-enhanced pen, and for statistics the age and gender of the participants and their handedness. The dataset is
fully pseudonymized by assigning an ID to every participant. The dataset does not contain any offensive content. The approach proposed in this paper, in particular used for the application of online handwriting recognition from sensor-enhanced pens, does not (1) facilitate injury to living beings, (2) raise safety or security concerns due to the anonymity of the data, (3) raise human rights concerns, (4) have a detrimental effect on people’s livelihood, (5) develop harmful forms of surveillance as the data is pseudonymized, (6) damage the environment, and (7) deceive people in ways that cause harm.

References


Appendices

We present the multi-task learning technique in Section A.1, and show more details on learning with the triplet loss on synthetically generated signal and image data in Section A.2. We present an overview for offline handwriting recognition (HWR) in Section A.3 and propose more details of our architectures in Section A.4. Section A.5 presents results of representation learning for HWR.

A.1 Multi-Task Learning (MTL)

We simultaneously train the $L_{CTC}$ loss for sequence classification combined with one or two shared losses $L_{shared,1}$ and $L_{shared,2}$ for common representation learning (CRL). As both losses are in different ranges, the naive weighting

$$L_{total} = \sum_{i=1}^{T} \omega_i L_i,$$

with pre-specified, constant weights $\omega_i = 1, \forall i \in \{1, \ldots, |T|\}$ can harm the training process. Hence, we apply dynamic weight average (DWA) [Liu et al. 2019] as an MTL approach that performs dynamic task weighting over time (i.e., after each batch).

A.2 Training Synthetic Data with the Triplet Loss

Signal and Image Generation. We combine the networks for both, signal and image classification, to improve the classification accuracy over each single-modal network. The aim is to show that the triplet loss can be used for such a cross-modal setting in the field of common representation learning. Hence, we generate synthetic data where the image data contains information of the signal data. We generate signal data $x$ with $x_{i,k} = \sin(0.05 \cdot t_i k)$ for all $t_i \in \{1, \ldots, 1,000\}$ where $t_i$ is the timestep of the signal. The frequency of the signal is dependent on the class label $k$. We generate signal data for 10 classes (see Figure 9a). We add noise from a continuous uniform distribution $U(a,b)$ for $a = 0$ and $b = 0.3$ (see Figure 9b), and add time and magnitude warping (see Figure 9c). We generate a signal-image pair such that the image is based on the signal data. We make use of the Gramian angular field (GAF) that transforms time-series into images. The time-series is defined as $x = (x_1, \ldots, x_n)$ for $n = 1,000$. The GAF creates a matrix of temporal correlations for each $(x_i, x_j)$ by rescaling the time-series in the range $[p, q]$ with $-1 \leq p < q \leq 1$ by

$$\hat{x}_i = p + (q-p) \cdot \frac{x_i - \min(x)}{\max(x) - \min(x)}, \forall i \in \{1, \ldots, n\},$$

and computes the cosine of the sum of the angles for the Gramian angular summation field (GASF) [Wang & Oates 2015] by

$$\text{GASF}_{i,j} = \cos(\phi_i + \phi_j), \forall i, j \in \{1, \ldots, n\},$$

with $\phi_i = \arccos(\hat{x}_i), \forall i \in \{1, \ldots, n\}$, being the polar coordinates. We generate image datasets based on signal data with different noise parameters ($b \in \{0.0, \ldots, 1.95\}$) to show the influence of the image data on the classification accuracy. Figure 10 exemplarily shows the GASF plots for the noise parameters $b = \{0, 0.5, 1.0, 1.5, 1.95\}$. We present the GASF for the classes 0, 5 and 9 to show the dependency of the frequency of the signal data on the GASF.

![Figure 9: Plot of the 1D signal data for 10 classes.](image)
Models. We use the following models for classification. Our encoder for time-series classification consists of a 1D convolutional layer (filter size 50, kernel 4), a max pooling layer (pool size 4), batch normalization, and a dropout layer (20%). The image encoder consists of a layer normalization and 2D convolutional layer (filter size 200), and batch normalization with ELU activation. It follows a 1D convolutional layer (filter size 200, kernel 4), max pooling (pool size 2), batch normalization, and 20% dropout. For both models, it follows a common representation, i.e., an LSTM with 10 units, a Dense layer with 20 units, a batch normalization layer, and a Dense layer of 10 units (for 10 sinusoidal classes). These layers are shared between both models.

A.3 Overview of Offline HWR Methods

In the following, we give a detailed overview of offline HWR methods to select a suitable lexicon and language model free method. There is no recent paper summarizing the work for offline HWR. For an overview of offline and online HWR datasets, see [Plamondon & Srihari (2000), Hussain et al. (2015)]. Table 3 presents related work. Methods for offline HWR range from hidden markov models (HMMs) to deep learning techniques that became predominant such as convolutional neural networks (CNNs), temporal convolutional networks (TCNs) and recurrent neural networks (RNNs). RNN techniques are well explored including long short-term memorys (LSTMs), bidirectional LSTMs (BiLSTMs), and multidimensional RNNs (MDRNN, MDLSTM). Recent methods are generative adversarial networks (GANs) and Transformers. We note the use of a language model (LM) and its size.

HMMs. Methods based on HMMs from last decades are [Bertolami & Bunke (2018), Dreuw et al. (2011), Li et al. (2014); Pastor-Pellicer et al. (2015)]. Recently, España-Boquera et al. (2011) proposed HMM+ANN, a HMM modeled with Markov chains in combination with a multilayer perceptron (MLP) to estimate the emission probabilities. Kozielski et al. (2013) presented Tandem GHMM that uses moment-based image normalization, writer adaptation and discriminative feature extraction with an 3-gram open-vocabulary of size 50k with an LSTM for recognition. Doetsch et al. (2014) proposed an LSTM unit that controls the shape of the squashing function in gating units decoded in a hybrid HMM. This approach yields the best results based on HMMs.
Table 3: Evaluation results (WER and CER in %) of different methods on the IAM-OffDB Liwicki & Bunke (2005) and RIMES Grosicki & El-Abed (2011) datasets. We state information about the method and the size of the language model (LM). LN = layer normalization. P = paragraph or full text level. L = line level. W = word level. The table is sorted by year.

<table>
<thead>
<tr>
<th>Method</th>
<th>Information</th>
<th>LM size k</th>
<th>Level</th>
<th>IAM-OffDB WER</th>
<th>CER</th>
<th>RIMES WER</th>
<th>CER</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMM</td>
<td>Markov chain with MLP</td>
<td>w (50)</td>
<td>×</td>
<td>13.30</td>
<td>5.10</td>
<td>13.70</td>
<td>4.60</td>
</tr>
<tr>
<td>TandemGHMM</td>
<td>GhMM and LSTM, writer adaptation</td>
<td>w (50)</td>
<td>×</td>
<td>12.20</td>
<td>4.70</td>
<td>12.90</td>
<td>4.30</td>
</tr>
<tr>
<td>LSTM-HMM</td>
<td>Combination of LSTM with HMM</td>
<td>w/o</td>
<td>×</td>
<td>27.50</td>
<td>8.30</td>
<td>17.70</td>
<td>4.00</td>
</tr>
<tr>
<td>Multi-</td>
<td>Combined MDLSTM with CTC</td>
<td>w/o</td>
<td>×</td>
<td>16.60</td>
<td>6.50</td>
<td>12.30</td>
<td>3.00</td>
</tr>
<tr>
<td>2DLSTM</td>
<td>LSTM-HMM</td>
<td>w/o</td>
<td>×</td>
<td>24.50</td>
<td>7.90</td>
<td>12.60</td>
<td>2.90</td>
</tr>
<tr>
<td>MDLSTM-RNN</td>
<td>GPU-based, diagonal MDLSTM</td>
<td>w/o</td>
<td>×</td>
<td>9.30</td>
<td>3.50</td>
<td>9.60</td>
<td>2.80</td>
</tr>
<tr>
<td>Voigtlaender et al. (2016)</td>
<td>LSTM</td>
<td>w/ (50)</td>
<td>×</td>
<td>10.50</td>
<td>3.60</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SepMLDSTM</td>
<td>LSTM</td>
<td>w/o</td>
<td>×</td>
<td>11.10</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Castro et al. (2018)</td>
<td>LSTM</td>
<td>w/o</td>
<td>×</td>
<td>7.50</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BiLSTM</td>
<td>CNN+BiLSTM</td>
<td>w/o</td>
<td>×</td>
<td>18.20</td>
<td>25.90</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Graves &amp; Schmidhuber (2008)</td>
<td>CNN+BiLSTM</td>
<td>w/ (50)</td>
<td>×</td>
<td>10.50</td>
<td>3.30</td>
<td>10.90</td>
<td>3.50</td>
</tr>
<tr>
<td>Graves et al. (2009)</td>
<td>CNN+BiLSTM</td>
<td>w/o</td>
<td>×</td>
<td>18.40</td>
<td>5.80</td>
<td>9.60</td>
<td>2.30</td>
</tr>
<tr>
<td>HMM+RNN</td>
<td>CNN+BiLSTM-MC (128 x W)</td>
<td>w/o</td>
<td>×</td>
<td>12.20</td>
<td>4.40</td>
<td>9.00</td>
<td>2.50</td>
</tr>
<tr>
<td>Michael et al. (2019)</td>
<td>CNN connected branches, CCA</td>
<td>w/o</td>
<td>×</td>
<td>6.45</td>
<td>3.44</td>
<td>3.90</td>
<td>1.90</td>
</tr>
<tr>
<td>Poznan et al. (2018)</td>
<td>CNN+CTC (32 x W)</td>
<td>w/o</td>
<td>×</td>
<td>6.50</td>
<td>3.44</td>
<td>3.90</td>
<td>1.90</td>
</tr>
<tr>
<td>Cosgrove et al. (2017)</td>
<td>CNN+CTC (64 x W)</td>
<td>w/o</td>
<td>×</td>
<td>8.50</td>
<td>2.52</td>
<td>1.86</td>
<td>0.65</td>
</tr>
<tr>
<td>Channel &amp; Delahaye (2019)</td>
<td>CNN+BiLSTM</td>
<td>w/o</td>
<td>×</td>
<td>17.82</td>
<td>5.70</td>
<td>9.60</td>
<td>2.30</td>
</tr>
<tr>
<td>Spigel et al. (2019)</td>
<td>CNN+BiLSTM-MC (128 x W)</td>
<td>w/o</td>
<td>×</td>
<td>22.90</td>
<td>13.10</td>
<td>15.80</td>
<td>9.70</td>
</tr>
<tr>
<td>Yousef et al. (2018)</td>
<td>CNN+BiLSTM</td>
<td>w/o</td>
<td>×</td>
<td>21.00</td>
<td>11.40</td>
<td>13.50</td>
<td>7.30</td>
</tr>
<tr>
<td>GTR</td>
<td>CNN+BiLSTM</td>
<td>w/o</td>
<td>×</td>
<td>16.70</td>
<td>8.10</td>
<td>9.60</td>
<td>3.50</td>
</tr>
<tr>
<td>Yousef &amp; Bishop (2020)</td>
<td>CNN+BiLSTM</td>
<td>w/o</td>
<td>×</td>
<td>35.10</td>
<td>10.80</td>
<td>28.50</td>
<td>6.80</td>
</tr>
<tr>
<td>GTR-8</td>
<td>CNN+BiLSTM</td>
<td>w/o</td>
<td>×</td>
<td>35.10</td>
<td>10.80</td>
<td>28.50</td>
<td>6.80</td>
</tr>
<tr>
<td>Kang et al. (2020)</td>
<td>CNN, Transformer decoder</td>
<td>w/o</td>
<td>×</td>
<td>15.45</td>
<td>4.67</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FPHR</td>
<td>CNN encoder, Transformer decoder</td>
<td>w/o</td>
<td>×</td>
<td>14.55</td>
<td>4.67</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>smith &amp; Karayev (2021)</td>
<td>CNN encoder, Transformer decoder</td>
<td>w/o</td>
<td>×</td>
<td>14.55</td>
<td>4.67</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Messina &amp; Kermorvant (2014)</td>
<td>CNN encoder, Transformer decoder</td>
<td>w/o</td>
<td>×</td>
<td>14.55</td>
<td>4.67</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Under review as submission to TMLR
The gated text recognizer (GTR) is a module that decouples the alignment operation from using historical decoding results based on visual features. This module has been used for OrigamiNet with the CTC loss. Thus, the GTR module can handle arbitrary input sizes and can recognize strings with arbitrary length. The fully convolutional network without recurrent connections is trained on a specific author with learned parameters for spacing, pressure and line thickness. Alonso et al. (2019) used a HWGAN method to synthesize online data based on RNNs. The technique is used in handwriting text generation (HTG), which is a relatively new field. The first approach by Graves (2014) was a method for HWR based on two classification modules by minimizing the CTC and negative log likelihood losses. While the MDLSTM by Bluche et al. (2017) contains covert and overt attention without prior segmentation, the Castro et al. (2018) integrated MDLSTMs within a hybrid HMM. However, these architectures come with quite an expensive computational cost. Furthermore, they extract features visually similar to those of convolutional layers. Puigcerver (2017) and End2End Krishnan et al. (2018) jointly learns text and image embeddings based on LSTMs.

The 2DLSTM approach by Graves & Schmidhuber (2008) combines multidimensional LSTMs (MDLSTMs) with the CTC loss. The MDLSTM-RNN approach Bluche (2016) works at paragraph level by replacing the collapse layer by a recurrent version. A neural network performs implicit line segmentation by computing attention weights on the image representation. Voigtlaender et al. (2016) proposed an efficient GPU-based implementation of MDLSTMs by processing the input in a diagonal-wise fashion. SepMDLSTM Chen et al. (2017) is a multi-task learning method for script identification and HWR based on two classification modules by minimizing the CTC and negative log likelihood losses. While the MDLSTM by Bluche et al. (2017) contains covert and overt attention without prior segmentation, the Castro et al. (2018) integrated MDLSTMs within a hybrid HMM. However, these architectures come with quite an expensive computational cost. Furthermore, they extract features visually similar to those of convolutional layers. Puigcerver (2017) and End2End Krishnan et al. (2018) jointly learns text and image embeddings based on LSTMs.

RNNs: MDLSTMs. The 2DLSTM approach by Graves & Schmidhuber (2008) combines multidimensional LSTMs (MDLSTMs) with the CTC loss. The MDLSTM-RNN approach Bluche (2016) works at paragraph level by replacing the collapse layer by a recurrent version. A neural network performs implicit line segmentation by computing attention weights on the image representation. Voigtlaender et al. (2016) proposed an efficient GPU-based implementation of MDLSTMs by processing the input in a diagonal-wise fashion. SepMDLSTM Chen et al. (2017) is a multi-task learning method for script identification and HWR based on two classification modules by minimizing the CTC and negative log likelihood losses. While the MDLSTM by Bluche et al. (2017) contains covert and overt attention without prior segmentation, the Castro et al. (2018) integrated MDLSTMs within a hybrid HMM. However, these architectures come with quite an expensive computational cost. Furthermore, they extract features visually similar to those of convolutional layers. Puigcerver (2017) and End2End Krishnan et al. (2018) jointly learns text and image embeddings based on LSTMs.

RNNs: LSTMs and BiLSTMs. RNNs for HWR marked an important milestone reaching impressive recognition accuracies. Sequential architectures are perfect to fit text lines due to the probability distributions over sequences of characters and due to the inherent temporal aspect of text. Kang et al. (2020) and Graves et al. (2009) introduced the BiLSTM layer in combination with the CTC loss. Pham et al. (2014) showed that the performance of LSTMs can be greatly improved using dropout. Voigtlaender et al. (2015) investigated sequence-discriminative training of LSTMs using the maximum mutual information (MMI) criterion. While Bluche (2015) utilized a RNN with a HMM and a language model, Menasri et al. (2012) combined a RNN with a sliding window Gaussian HMM. GCRNN Bluche & Messina (2017) combines a convolutional encoder (aiming generic and multilingual features) and a BiLSTM decoder predicting character sequences. Also, Puigcerver (2017) proposed a CNN+BiLSTM architecture (CNN-1DLSTM-CTC) that uses the CTC loss. The start, follow, read (SFR) Wigington et al. (2018) model jointly learns text detection and segmentation. Dutta et al. (2018) used synthetic data for pre-training and image normalization for slant correction. The methods by Chowdhury & Vgi (2018), Sueiras et al. (2018), Ingle et al. (2019), Michael et al. (2019) make use of BiLSTMs. While Carbonell et al. (2019) uses a feature pyramid network (FPN), the adversarial feature deformation module (AFDM) Bhunia et al. (2019) learns ways to elastically warp features in a scalable manner. Further methods that combine CNNs with RNNs are Liang et al. (2017); Sudholt & Fink (2018); Xiao & Cho (2016), while BiLSTMs are utilized in Carbune et al. (2020); Tian et al. (2019).

TCNs. TCNs use dilated causal convolutions and have been applied to air-writing recognition by Bastas et al. (2020). As RNNs are slow to train, Sharma et al. (2020) presented a faster system which is based on text line images and TCNs with the CTC loss. This approach achieves 9.6% CER on the IAM-OffDB dataset. Sharma & Jayagopi (2021) combined 2D convolutions with 1D dilated non-causal convolutions that offers a high parallelism with a smaller number of parameters. They analyzed re-scaling factors and data augmentation, and achieved comparable results for the IAM-OffDB and RIMES datasets.

CNNs. Poznanski & Wolf (2016) utilized a CNN with multiple fully connected branches to estimate its n-gram frequency profile (set of n-grams contained in the word). With canonical correlation analysis (CCA), the estimated profile can be matched to the true profiles of all words in a large dictionary. As most attention methods suffer from an alignment problem, Wang et al. (2020) proposed a decoupled attention network (DAN) that has a convolutional alignment module that decouples the alignment operation from using historical decoding results based on visual features. The gated text recognizer (GTR) Yousef et al. (2018) aims to automate the feature extraction from raw input signal with minimum required domain knowledge. The fully convolutional network without recurrent connections is trained with the CTC loss. Thus, the GTR module can handle arbitrary input sizes and can recognize strings with arbitrary length. This module has been used for OrigamiNet Yousef & Bishop (2020) that is a segmentation-free multi-line or full page recognition system. OrigamiNet yields state-of-the-art results on the IAM-OffDB dataset, and shows an improved performance of GTR over VGG and ResNet26. Hence, we use the GTR module as our visual feature encoder for offline HWR (see Section 4.4).

GANs. Handwriting text generation (HTG) is a relatively new field. The first approach by Graves (2014) was a method to synthesize online data based on RNNs. The technique HwGAN by Ji & Chen (2020) extends this method by adding a discriminator D. DeepWriting Aksan et al. (2016) is a GAN that is capable of disentangling style from content and thus making digital ink editable. Haines et al. (2016) proposed a method to generate handwriting based on a specific author with learned parameters for spacing, pressure and line thickness. Alonso et al. (2019) used a
Under review as submission to TMLR

BiLSTM to get an embedding of the word to be rendered, and added an auxiliary network as recognizer \( R \). The model is trained with a combination of an adversarial loss and the CTC loss. ScrabbleGAN by [Pogel et al. (2020)] is a semi-supervised approach that can generate arbitrarily many images of words with arbitrary length from a generator \( G \) to augment handwriting data and uses a discriminator \( D \) and recognizer \( R \). The paper proposes results for original data, with random affine augmentation, using synthetic images and refinement.

Transformers. RNNs prevent parallelization due to their sequential pipelines. [Kang et al. (2020)] introduced a non-recurrent model by the use of Transformer models by using multi-head self-attention layers at the textual and visual stages. Their method is unconstrained to any pre-defined vocabulary. For the feature encoder, they used modified ResNet50 models. The full page HTR (FPHR) method by [Singh & Karayev (2021)] uses a CNN as encoder and a Transformer as decoder with positional encoding.

A.4 Details on Architectures for Offline HWR

In this section, we give details about the integration of Inception [Szegedy et al. (2015)], ResNet [He et al. (2016)] and GTR [Yousef et al. (2018)] modules into the offline HWR system. All three architectures are based on publicly available implementations, but we changed or adapted the first layer for the image input and the last layer for a proper input for our latent representation module.

Inception. Figure 11 gives an overview of the integration of the Inception module. The Inception module is part of the well-known GoogLeNet architecture. The main idea is to consider how an optimal local sparse structure can be approximated by readily available dense components. As the merging of pooling layer outputs with convolutional layer outputs would lead to an inevitable increase in the number of output and would lead to computational blow up, we apply the Inception module with dimensionality reduction to our offline HWR approach [Szegedy et al. (2015)]. The input image is of size \( H \times W \). What follows is the Inception (3a), Inception (3b), a max pooling layer (\( 3 \times 3 \)) and Inception (4a). We add three 1D convolutional layers to get an output dimensionality of 400 × 200 as input for the latent representation.

ResNet34. Figure 12 gives an overview of the integration of the ResNet34 architecture. Instead of learning unreference functions, [He et al. (2016)] reformulated the layers as learning residual functions with reference to the layer inputs. This residual network is easier to optimize and can gain accuracy from considerably increased depth. The ResNet block let the layers fit a residual mapping denoted as \( \mathcal{H}(x) \) with identity \( x \), and fits the mapping \( \mathcal{F}(x) := \mathcal{H}(x) - x \). The original mapping is recast into \( \mathcal{F}(x) + x \). We reshape the output of ResNet34, add a 1D convolutional layer, and reshape the output for the latent representation.

GTR. Figure 13 gives an overview of the integration of the gated text recognizer (GTR) [Yousef et al. (2018)] module that is a fully convolutional network that uses batch normalization (BN) and layer normalization (LN) to regularize the training process and increase convergence speed. The module uses batch renormalization [Ioffe (2017)] on all BN layers. Depthwise separable convolutions reduce the number of parameters at the same/better classification performance.
Under review as submission to TMLR

Input image \((H \times W)\)

Conv2D (64, 7)
Batch Norm., ReLU
ZeroPadding2D (3\times3)
ZeroPadding2D (1\times1)
MaxPooling2D (3)
Conv2D (64, 7 \times 7, /2)
Conv2D (64, 3 \times 3)
Conv2D (64, 3 \times 3)
Conv2D (64, 3 \times 3)
Conv2D (64, 3 \times 3)
Conv2D (64, 3 \times 3)
Conv2D (64, 3 \times 3)
Conv2D (64, 3 \times 3)
Conv2D (64, 3 \times 3)
Conv2D (64, 3 \times 3)
Conv2D (128, 3 \times 3)
Conv2D (128, 3 \times 3)
Conv2D (128, 3 \times 3)
Conv2D (128, 3 \times 3)
Conv2D (128, 3 \times 3)
Conv2D (128, 3 \times 3)
Conv2D (128, 3 \times 3)
Conv2D (128, 3 \times 3)
Conv2D (128, 3 \times 3)
Conv2D (128, 3 \times 3)
Conv2D (128, 3 \times 3)
Conv2D (128, 3 \times 3)
Conv2D (128, 3 \times 3)
Conv2D (128, 3 \times 3)
Conv2D (128, 3 \times 3)
Conv2D (256, 3 \times 3)
Conv2D (256, 3 \times 3)
Conv2D (256, 3 \times 3)
Conv2D (256, 3 \times 3)
Conv2D (256, 3 \times 3)
Conv2D (256, 3 \times 3)
Conv2D (256, 3 \times 3)
Conv2D (256, 3 \times 3)
Conv2D (256, 3 \times 3)
Conv2D (256, 3 \times 3)
Conv2D (256, 3 \times 3)
Conv2D (256, 3 \times 3)
Conv2D (256, 3 \times 3)
Conv2D (256, 3 \times 3)
Conv2D (256, 3 \times 3)
Conv2D (512, 3 \times 3)
Conv2D (512, 3 \times 3)
Conv2D (512, 3 \times 3)
Conv2D (512, 3 \times 3)
Conv2D (512, 3 \times 3)
Conv2D (512, 3 \times 3)
Conv2D (512, 3 \times 3)
Conv2D (512, 3 \times 3)
Conv2D (512, 3 \times 3)
Conv2D (512, 3 \times 3)
Conv2D (512, 3 \times 3)
Conv2D (512, 3 \times 3)
Conv2D (512, 3 \times 3)
Conv2D (512, 3 \times 3)
Conv2D (512, 3 \times 3)
Reshape (208, 128)
Conv1D (200, 5)
Reshape (400, 104)
Latent Representation

Figure 12: Offline HWR method based on the ResNet34 architecture \cite{He2016ResNet}.  

GTR uses spatial dropout instead of regular unstructured dropout for better regularization. After the input image of size \(H \times W\) that is normalized follows a convolutional layer with Softmax normalization, a \(13 \times 13\) filter, and dropout (40%). It follows a stack of 2, 4, 6 or 8 gate blocks that models the input sequence. Similar as \cite{Yousef2018}, we add a dropout of 20% after the last GTR block. Lastly, we add a 2D convolutional layer of 200, a BN layer and a LN layer that is the input for our latent representation.

A.5 Detailed HWR Evaluation

Offline HWR Results. Table \ref{table:offline_hwr} shows offline HWR results on our generated OffHW-German dataset and on the IAM-OffDB \cite{Liwicki2005} dataset. ScrabbleGAN \cite{Fogel2020} yields an WER of 23.61% on the
Table 4: Evaluation results (WER and CER in %) for the generated dataset with ScrabbleGAN [Fogel et al., 2020] OffHW-German and the IAM-OffDB [Liwicki & Bunke, 2005] dataset. We propose writer-dependent (WD) and writer-independent (WI) results.

<table>
<thead>
<tr>
<th>Method</th>
<th>OffHW-German</th>
<th>IAM-OffDB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WER</td>
<td>CER</td>
</tr>
<tr>
<td>ScrabbleGAN</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Fogel et al. (2020)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OrigamiNet (12 × GTR)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Yousef &amp; Bishop (2020)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OrigamiNet (ours, 4 × GTR)</td>
<td>1.50</td>
<td>0.11</td>
</tr>
<tr>
<td>Inception</td>
<td>12.54</td>
<td>1.17</td>
</tr>
<tr>
<td>ResNet</td>
<td>13.05</td>
<td>1.24</td>
</tr>
<tr>
<td>GTR (2 blocks), 1 conv. layer</td>
<td>4.34</td>
<td>0.39</td>
</tr>
<tr>
<td>GTR (2 blocks), 2 conv. layer</td>
<td>5.02</td>
<td>0.44</td>
</tr>
<tr>
<td>GTR (4 blocks), 1 conv. layer</td>
<td>3.35</td>
<td>0.34</td>
</tr>
<tr>
<td>GTR (4 blocks), 2 conv. layer</td>
<td>2.52</td>
<td>0.24</td>
</tr>
<tr>
<td>GTR (6 blocks)</td>
<td>2.85</td>
<td>0.26</td>
</tr>
<tr>
<td>GTR (8 blocks)</td>
<td>4.22</td>
<td>0.38</td>
</tr>
</tbody>
</table>

Table 5: Evaluation results (WER and CER in %) for the generated OffHW-words500 and OffHW-wordsRandom datasets for one and two convolutional layers (c). We propose writer-dependent (WD) and writer-independent (WI) results.

<table>
<thead>
<tr>
<th>Method (4 × GTR)</th>
<th>OffHW-words500</th>
<th>OffHW-wordsRandom</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WD WER</td>
<td>CER</td>
</tr>
<tr>
<td>c = 1</td>
<td>2.94</td>
<td>0.76</td>
</tr>
<tr>
<td>c = 2</td>
<td>2.51</td>
<td>0.69</td>
</tr>
</tbody>
</table>

IAM dataset, while OrigamiNet [Yousef & Bishop, 2020] achieves an CER of 4.70% with 12 GTR modules. As the training takes more than one day for one epoch on the large OffHW-German dataset, we train OrigamiNet with four GTR modules, and achieve 0.11% CER on the generated dataset and 15.67% on the IAM dataset, which is higher than the model with 12 GTR modules. While the paper did not propose WER results, OrigamiNet yields only an WER of 90.40%. With our own implementation of four GTR modules and one convolutional layer for the common representation, our model achieves similar results. While GTR modules yield slightly lower CERs on the OffHW-German dataset than our architectures with Inception and ResNet modules, the WERs are significantly higher. Fine-tuning the architecture with four GTR modules and one (c = 1) or two (c = 2) convolutional layers on the OffHW-words500 and OffHW-wordsRandom datasets, yields better results for c = 2 than for c = 1 (see Table 5). While results for OffHW-wordsRandom are similar for writer-dependent (WD) and writer-independent (WI), WI results of the OffHW-words500 dataset are lower than WD results, as words with the same label appear in the training and test dataset. We use the weights of the fine-tuning as initial weights of the image model for the common representation learning.

**Online HWR Results.** Table 2 gives an overview of CRL results based on two convolutional layers (c = 2) for the common representation. Our CNN+BiLSTM contains three additional convolutional layers and outperforms the smaller CNN+BiLSTM by Ott et al. (2022) on the WD classification tasks. Without triplet loss, \( L_{PC} \) yields the best results on the OnHW-wordsRandom dataset. The triplet loss partly marginally decreases results and partly improves results on the OnHW-words500 dataset. In conclusion, two convolutional layers for the common representation decreases results, while here the triplet loss does not have a positive impact.
Table 6: Evaluation results (WER and CER in %, averaged over five splits) of the baseline MTS-only technique and our cross-modal techniques for the inertial-based OnHW datasets Ott et al. (2022) with and without mutated vowels (MV) for two convolutional layers $c = 2$. We propose writer-(in)dependent (WD/WI) results.

<table>
<thead>
<tr>
<th>Method</th>
<th>OnHW-words500</th>
<th></th>
<th></th>
<th>OnHW-wordsRandom</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WD</td>
<td>WI</td>
<td>WD</td>
<td>WI</td>
<td>WD</td>
<td>WI</td>
</tr>
<tr>
<td>Small CNN+BiLSTM, $L_{CTC}$, w/ MV</td>
<td>51.95</td>
<td>17.16</td>
<td>60.91</td>
<td>27.80</td>
<td>41.27</td>
<td>7.87</td>
</tr>
<tr>
<td>CNN+BiLSTM (ours), $L_{CTC}$, w/ MV</td>
<td>42.81</td>
<td>13.04</td>
<td>60.47</td>
<td>28.30</td>
<td>37.13</td>
<td>6.75</td>
</tr>
<tr>
<td>CNN+BiLSTM (ours), $L_{CTC}$, w/o MV</td>
<td>42.77</td>
<td>13.44</td>
<td>59.82</td>
<td>28.54</td>
<td>41.52</td>
<td>7.81</td>
</tr>
<tr>
<td>$L_{MSE}$</td>
<td>39.79</td>
<td>12.14</td>
<td>60.35</td>
<td>28.48</td>
<td>39.98</td>
<td>7.79</td>
</tr>
<tr>
<td>$L_{CS}$</td>
<td>43.40</td>
<td>13.70</td>
<td>59.31</td>
<td>27.99</td>
<td>40.31</td>
<td>7.68</td>
</tr>
<tr>
<td>$L_{PC}$</td>
<td>38.90</td>
<td>11.60</td>
<td>60.77</td>
<td>28.45</td>
<td>39.93</td>
<td>7.60</td>
</tr>
<tr>
<td>$L_{KL}$</td>
<td>37.25</td>
<td>11.29</td>
<td>65.10</td>
<td>31.26</td>
<td>41.81</td>
<td>8.22</td>
</tr>
<tr>
<td>$L_{trpl,2}(L_{MSE})$</td>
<td>41.16</td>
<td>12.71</td>
<td>58.65</td>
<td>28.19</td>
<td>41.16</td>
<td>8.03</td>
</tr>
<tr>
<td>$L_{trpl,2}(L_{CS})$</td>
<td>42.74</td>
<td>13.43</td>
<td>58.13</td>
<td>27.62</td>
<td>41.49</td>
<td>8.18</td>
</tr>
<tr>
<td>$L_{trpl,2}(L_{PC})$</td>
<td>39.94</td>
<td>12.17</td>
<td>62.76</td>
<td>30.68</td>
<td>41.58</td>
<td>8.18</td>
</tr>
<tr>
<td>$L_{trpl,2}(L_{KL})$</td>
<td>38.34</td>
<td>11.77</td>
<td>67.08</td>
<td>33.84</td>
<td>39.93</td>
<td>7.60</td>
</tr>
</tbody>
</table>

Table 7: Evaluation results (WER and CER in %, averaged over five splits) of the baseline MTS-only technique and our cross-modal techniques for the inertial-based OnHW datasets Ott et al. (2022) with and without mutated vowels (MV) for two convolutional layers $c = 1$. We propose writer-(in)dependent (WD/WI) results.

<table>
<thead>
<tr>
<th>Method</th>
<th>OnHW-words500-L</th>
<th></th>
<th></th>
<th>OnHW-wordsRandom-L</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WD</td>
<td>WI</td>
<td>WD</td>
<td>WI</td>
<td>WD</td>
<td>WI</td>
</tr>
<tr>
<td>InceptionTime, $L_{CTC}$, w/ MV</td>
<td>49.70</td>
<td>14.02</td>
<td>100.00</td>
<td>96.06</td>
<td>48.10</td>
<td>8.63</td>
</tr>
<tr>
<td>CNN+BiLSTM, $L_{CTC}$, w/ MV</td>
<td>14.20</td>
<td>3.30</td>
<td>94.40</td>
<td>71.41</td>
<td>30.20</td>
<td>4.86</td>
</tr>
<tr>
<td>CNN+BiLSTM, $L_{CTC}$, w/o MV</td>
<td>12.94</td>
<td>3.33</td>
<td>89.07</td>
<td>62.07</td>
<td>30.89</td>
<td>5.26</td>
</tr>
<tr>
<td>$L_{MSE}$</td>
<td>11.62</td>
<td>2.77</td>
<td>90.65</td>
<td>67.90</td>
<td>30.53</td>
<td>4.93</td>
</tr>
<tr>
<td>$L_{CS}$</td>
<td>14.92</td>
<td>3.53</td>
<td>94.14</td>
<td>65.10</td>
<td>29.06</td>
<td>4.87</td>
</tr>
<tr>
<td>$L_{PC}$</td>
<td>12.29</td>
<td>3.04</td>
<td>91.33</td>
<td>60.89</td>
<td>29.61</td>
<td>4.91</td>
</tr>
<tr>
<td>$L_{KL}$</td>
<td>13.07</td>
<td>2.57</td>
<td>93.02</td>
<td>66.64</td>
<td>29.61</td>
<td>4.91</td>
</tr>
<tr>
<td>$L_{trpl,2}(L_{MSE})$</td>
<td>11.97</td>
<td>2.83</td>
<td>84.34</td>
<td>57.84</td>
<td>27.19</td>
<td>4.79</td>
</tr>
<tr>
<td>$L_{trpl,2}(L_{CS})$</td>
<td>11.65</td>
<td>2.63</td>
<td>94.70</td>
<td>67.69</td>
<td>28.39</td>
<td>4.62</td>
</tr>
<tr>
<td>$L_{trpl,2}(L_{PC})$</td>
<td>13.02</td>
<td>2.94</td>
<td>89.86</td>
<td>60.26</td>
<td>30.22</td>
<td>4.81</td>
</tr>
<tr>
<td>$L_{trpl,2}(L_{KL})$</td>
<td>13.55</td>
<td>3.22</td>
<td>97.86</td>
<td>76.54</td>
<td>28.14</td>
<td>4.71</td>
</tr>
</tbody>
</table>

Transfer Learning on Left-Handed Writers. To adapt the model to left-handed writers that are typically under-represented in the real-world, we make use of the left-handed datasets OnHW-words500-L and OnHW-wordsRandom-L proposed by Ott et al. (2022). These datasets contain recordings of two writers with 1,000 samples, respectively 996. As baseline we pre-train the MTS-only model on the right-handed datasets and post-train the left-handed datasets for 500 epochs (see the first two rows of Table 7). As these datasets are rather small, the models can overfit on these specific writers and achieve a very low CER of 3.33% on the OnHW-words500-L datasets and 5.26% CER on the OnHW-wordsRandom-L dataset without mutated vowels (MV) for the writer-dependent (WD) tasks, but the models cannot generalize on the writer-independent (WI) tasks: 62.07% CER on the OnHW-words500-L dataset and 81.15% CER on the OnHW-wordsRandom-L dataset. Hence, we focus on the WD tasks. For comparison, we use the state-of-the-art time-series classification technique InceptionTime Fawaz et al. (2019) with $depth = 11$ and $nf = 96$ (without pre-training). Our CNN+BiLSTM clearly outperforms InceptionTime. We use the weights of the pre-training with the offline handwriting datasets and again post-train on the left-handed datasets with $c = 1$. Using the weights of the cross-modal learning without the triplet loss, can decrease the error rates up to 2.57% CER with $L_{PC}$, respectively 4.47% CER with $L_{KL}$. Using the triplet loss $L_{trpl,2}(L_{CS})$, we can further decrease the CER to 2.63% for the OnHW-words500-L dataset. In conclusion, due to the use of the weights of the cross-modal setup, the model can adapt faster to new writers and generalize better to unseen words due to the triplet loss.
References


Manuel Carbonell, Joan Mas, Mauricio Villegas, Alicia Fornés, and Josep Lladós. End-to-End Handwritten Text Detection and Transcription in Full Pages. In Intl. Conf. on Document Analysis and Recognition Workshops (ICDARW), Sydney, Australia, September 2019.


Patrick Doetsch, Michal Kozielski, and Hermann Ney. Fast and Robust Training of Recurrent Neural Networks for Offline Handwriting Recognition. In Intl. Conf. on Frontiers in Handwriting Recognition (ICFHR), Hersonissos, Greece, September 2014.


