A. Appendix

We present criteria for loss functions and an overview of metrics in Section A.1. We propose more details about the architectures in Sec. A.2, and present more results in Sec. A.3. Sec. A.4 describes a dataset-specific feature embedding analysis.

A.1. Loss Functions

In the following, we state all loss functions that we used for our methodology. We describe the cross-entropy loss for the Multivariate Time Series Classification (MTSC) task. Next, we present criteria for the trajectory regression task. Finally, we propose distance-based, spatio-temporal and distribution-based loss functions.

MTSC Task: Cross-entropy Loss. For the MTSC task, the cross-entropy loss [25] is defined by

\[ L_{CE}(U, Y) = -\frac{1}{n} \sum_{i=1}^{n} v_i \log \hat{v}_i, \]  

(2)

where the Multivariate Time Series (MTS) \( U = \{ u_1, \ldots, u_m \} \in \mathbb{R}^{m \times l} \) is an ordered sequence of \( m \in \mathbb{N} \) streams with \( u_i = (u_{i,1}, \ldots, u_{i,l}), i \in \{1, \ldots, m\} \). \( m \) is the length of the time series and \( l \) is the number of dimensions. Each MTS is associated with a class label \( v \in \Omega \) from a pre-defined label set \( \Omega \). 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 \( \mathcal{V} = \{ v_1, \ldots, v_n \} \in \Omega^n \) [60]. The MTSC task is to predict an unknown class label \( \hat{v} \) for a given MTS.

Trajectory Regression Task: Criteria. For the trajectory regression task, it is given a ground truth time series \( \mathcal{Y} = \{ y_1, \ldots, y_m \} \in \mathbb{R}^{m \times d} \). The goal is to predict a time series \( \mathcal{X} = \{ x_1, \ldots, x_n \} \in \mathbb{R}^{n \times d} \), such that \( \mathcal{X} \) is closely aligned with \( \mathcal{Y} \). In the following, we consider \( r_i = y_i - x_i \) be the residual between \( \mathcal{X} \) and \( \mathcal{Y} \). We consider a (differentiable) substitution-cost function \( L : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}_+ \) to learn the trajectory regression task. All metrics to be used in a neural network have to obey the following criteria, where \( \mathcal{X}, \mathcal{Y}, \mathcal{Z} \in \mathbb{R}^d \) [34]:

\[ L(\mathcal{X}, \mathcal{Y}) \geq 0 \text{ (non-negativity)} \quad \text{(I)} \]
\[ L(\mathcal{X}, \mathcal{Y}) = L(\mathcal{Y}, \mathcal{X}) \text{ (symmetry)} \quad \text{(II)} \]
\[ L(\mathcal{X}, \mathcal{Y}) \leq L(\mathcal{X}, \mathcal{Z}) + L(\mathcal{Z}, \mathcal{Y}) \text{ (triangle inequ.)} \quad \text{(III)} \]
\[ L(\mathcal{X}, \mathcal{Y}) = 0 \Leftrightarrow \mathcal{X} = \mathcal{Y} \text{ (ident. of indiscernibles)} \quad \text{(IVa)} \]

It is difficult to make accurate predictions about the injectivity as floating points operations and approximation errors lead to a distance of zero for slightly different inputs.

Hence, Eq. (IVa) can be formulated as a pseudometric with a relaxed identity of indiscernibles where \( \mathcal{X} \in \mathbb{R}^d \):

\[ L(\mathcal{X}, \mathcal{X}) = 0 \quad \text{(IVb)} \]

Trajectory Regression Task: Distance-based Loss Functions. We consider the Mean Squared Error

\[ L_{MSE}(\mathcal{X}, \mathcal{Y}) = \frac{1}{n} \| \mathcal{Y} - \mathcal{X} \|_2^2 = \frac{1}{n} \sum_{i=1}^{n} r_i^2 \quad \text{(3)} \]

with \( L_2 \)-norm \( \| \cdot \|_2 \) (see Fig. 10a). The derivative of the \( L_{MSE} \) loss is \( \frac{\partial}{\partial \mathcal{X}} L_{MSE}(\mathcal{X}, \mathcal{Y}) = -\frac{2}{n} \sum_{i=1}^{n} r_i \). The Mean Absolute Error (MAE)

\[ L_{MAE}(\mathcal{X}, \mathcal{Y}) = \frac{1}{n} \| \mathcal{Y} - \mathcal{X} \|_1 = \frac{1}{n} \sum_{i=1}^{n} |r_i| \quad \text{(4)} \]

with the \( L_1 \)-norm \( \| \cdot \|_1 \). Its derivative is \( \frac{\partial}{\partial \mathcal{X}} L_{MAE}(\mathcal{X}, \mathcal{Y}) = 1/n \sum_{i=1}^{n} \text{sign}(r_i) \). The Huber loss [30]

\[ L_{H}(\mathcal{X}, \mathcal{Y}, \delta_H) = \sum_{i: |r_i| \leq \delta_H} \frac{1}{2} r_i^2 + \sum_{i: |r_i| > \delta_H} \delta_H |r_i| - \frac{1}{2} \delta_H^2 \quad \text{(5)} \]

is less sensitive to outliers, but depends on the hyperparameter \( \delta_H \) (see Fig. 10b). The derivative of the Huber loss is

\[ \frac{\partial}{\partial \mathcal{X}} L_{H}(\mathcal{X}, \mathcal{Y}, \delta_H) = -\sum_{i: |r_i| \leq \delta_H} r_i - \sum_{i: |r_i| > \delta_H} \delta_H \text{sign}(r_i). \quad \text{(6)} \]
Similar, the Andrew’s Sine loss [8] is
\[
L_{AS}(\mathcal{X}, \mathcal{Y}, \delta_{AS}) = \sum_{|i|:|r_i|\leq 1} 4\sin\left(\frac{r_i}{\delta_{AS}}\right)^2 + \sum_{|i|:|r_i|> 1} 1,
\]
with hyperparameter \(\delta_{AS}\) (see Fig. 10c) with the derivative
\[
\frac{\partial}{\partial \mathcal{X}} L_{AS}(\mathcal{X}, \mathcal{Y}, \delta_{AS}) = \sum_{|i|:|r_i|\leq 1} \frac{2}{\delta_{AS}} \sin\left(\frac{r_i}{\delta_{AS}}\right).
\]

Trajectory Regression Task: Spatio-temporal Loss Functions. We define the Cosine Similarity by
\[
L_{CS}(\mathcal{X}, \mathcal{Y}) = 1 - \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}.
\]
The Cosine Similarity is a proper metric as it satisfies the requirements \(L_{CS}(\mathcal{X}, \mathcal{Y}) \geq 0\) (I), \(L_{CS}(\mathcal{X}, \mathcal{Y}) = L_{CS}(\mathcal{Y}, \mathcal{X})\) (II), and \(L_{CS}(\mathcal{X}, \mathcal{X}) = 0\) (IVb). Under certain conditions the triangle inequality (III) is not fulfilled [36]. This loss function is a measure of similarity between two non-zero vectors of an inner product space, but is not invariant to shifts. The Pearson Correlation loss [47] \(L_{PC}(\mathcal{X}, \mathcal{Y}) = L_{CS}(\mathcal{X} - \mathcal{X}, \mathcal{Y} - \mathcal{Y})\), in contrast, is invariant to shifts. This means, when \(\mathcal{X}\) is transformed by \(a + b\mathcal{X}\) and \(\mathcal{Y}\) is transformed by \(c + d\mathcal{Y}\), where \(a, b, c\) and \(d\) are constants \((b, d > 0)\), the Pearson Correlation coefficient is invariant in location and scale in the two variables. The Pearson Correlation loss is defined by
\[
L_{PC}(\mathcal{X}, \mathcal{Y}) = 1 - \frac{s_{xy}}{s_x \cdot s_y} = 1 - \frac{(\mathbf{x} - \mathbf{x}) \cdot (\mathbf{y} - \mathbf{y})}{\|\mathbf{x} - \mathbf{x}\|_2 \|\mathbf{y} - \mathbf{y}\|_2}.
\]
with the sample mean \(\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i\). Analogously for \(\bar{y}\). The covariance is \(s_{xy} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x}) \cdot (y_i - \bar{y})\), and the variance of the features is \(s_x^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2\), analogously for \(s_y^2\). The partial derivative of the Pearson Correlation [54] regarding \(\mathbf{x}\) is
\[
\frac{\partial}{\partial \mathbf{x}} L_{PC}(\mathcal{X}, \mathcal{Y}) = \frac{(\mathbf{y} - \mathbf{y}) - s_{xy} \cdot (\mathbf{x} - \mathbf{x})}{s_x \cdot s_y}.
\]

Further alternative distance-based metrics are, e.g., the LogCosh, the Quantile [45], the Tukey’s Biweight [7], the Hampel [8] and the Geman McClure metric [6] (see Fig. 10d). For more information, see [37] for distance-based metrics and [52] for spatio-temporal metrics.

Trajectory Regression Task: Distribution-based Loss Functions. We use the distribution-based loss function, i.e., the Wasserstein distance [23], defining a distance between two probability distributions on a given metric space \(\mathcal{M}\) and representing the cost \(\delta\) of an optimal mass transportation problem. Optimal transport can be used to compare probability measures in metric spaces. There exists some \(\mathcal{X}_0\) in \(\mathcal{M}\) such that the Wasserstein space of order \(p\) is defined as
\[
P_p(\mathcal{M}) := \{\mu \in P(\mathcal{M}) : \int_{\mathcal{M}} \delta(\mathcal{X}, \mathcal{X}_0)^p d\mathcal{X} < \infty\}.
\]
The \(p^\text{th}\) Wasserstein distance between two probability measures \(\mu\) and \(\nu\) is defined as
\[
W_p(\mu, \nu) := \left( \int_{\mathcal{M} \times \mathcal{M}} \delta(\mathcal{X}, \mathcal{Y})^p d\gamma(\mathcal{X}, \mathcal{Y}) \right)^{\frac{1}{p}} = \inf \left\{ \left[ \mathbb{E}[d(X, Y)^p] \right]^{\frac{1}{p}},\ \text{law}(X) = \mu,\ \text{law}(Y) = \nu \right\},
\]
with the collection of all probability measures \(\Gamma(\mu, \nu)\) on \(\mathcal{M} \times \mathcal{M}\) and \(p \in [1, \infty)\). \(\mathbb{E}[X]\) denotes the expected value of a random variable \(X\). We consider the classical case where the metric is the Euclidean metric in space \(\mathbb{R}^d \subset \mathcal{M}\), and hence, \(\delta(\mathcal{X}, \mathcal{Y}) = ||\mathcal{X} - \mathcal{Y}||\). For all subsets \(P \subset \mathbb{R}^d\), we have \(\gamma(P \times \mathbb{R}^d) = \mu(P)\) and \(\gamma(\mathbb{R}^d \times P) = \nu(P)\). [48] The \(W_1\) distance is also called the Kantorovich-Rubinstein distance. The Wasserstein distance satisfies the criteria (I) to (IVa): It holds the non-negativity criteria \(W_p(\mu, \nu) \geq 0\) (I), and the symmetry criteria \(W_p(\mu, \nu) = W_p(\nu, \mu)\) (II). Assume that \(W_p(\mu, \mu) = 0\), then there exists a transference plan that is concentrated on the diagonal, and it holds \(\mathcal{X} = \mathcal{Y}\) (IVa). Furthermore, let \(\mu_1, \mu_2\) and \(\mu_3\) be probability measures on \(\mathcal{M} \times \mathcal{M}\), and \((T_1, T_2)\), respectively \((Q_2, Q_3)\), be an optimal coupling of \((\mu_1, \mu_2)\), respectively \((\mu_2, \mu_3)\). There exist random variables \((T_1', T_2', T_3')\) with law \((T_1', T_2') = \text{law}(T_1, T_2)\) and law \((T_2', T_3') = \text{law}(Q_2, Q_3)\), such that
\[
W_p(\mu_1, \mu_3) \leq \left( \mathbb{E}[d(T_1', T_3')^p] \right)^{\frac{1}{p}} \leq \left( \mathbb{E}[d(T_1', T_2') + d(T_2', T_3')^p] \right)^{\frac{1}{p}} \leq \left( \mathbb{E}[d(T_1', T_2')^p] + \mathbb{E}[d(T_2', T_3')^p] \right)^{\frac{1}{p}} = W_p(\mu_1, \mu_2) + W_p(\mu_2, \mu_3),
\]
and the triangle inequality holds (III). The dual formula for the Kantorovich-Rubinstein distance is
\[
W_1(\mu, \nu) = \sup_{||\psi||_{L_p(\mathcal{M})} \leq 1} \left\{ \int_{\mathcal{M}} \psi d\mu \int_{\mathcal{M}} \psi d\nu \right\},
\]
for any \(\mu, \nu\) in the Wasserstein space \(P_1(\mathcal{M})\). The Wasserstein distance \(W_1\) of order 1 is the weakest of all, and hence, is easier to bound. The Wasserstein distance has the ability to capture weak convergence precisely and are rather strong as they take care of large distances in \(\mathcal{M} \times \mathcal{M}\). [57] For more information, see [43].

Summary. For the classification task, we use the \(L_{CE}\) loss function (2). For the regression task, we use a combination of the distance-based loss functions \(L_{MSE}\) (3), \(L_{MAE}\)
A.2. MTL Network Architectures

The general overview of the framework is given in Fig. 2 (Sec. 3). The input is for the IMU-based and the visual-based OnHW dataset a MTS that differs regarding its input size. For the IMU-based dataset, the input are the 13 channels of the accelerometers, gyroscope, magnetometer and force sensor. The number of timesteps depends on the sample length. For the visual dataset, the input is the two dimensional trajectory of the pen tip in camera coordinates. What follows, is a CNN trunk, a classification head, and a regression head. The classification head is used for the MTSC task by predicting a class label with the cross-entropy loss. The regression head is used for the trajectory regression task that predicts a MTS that represents the trajectory of the written character. The loss function for this task is a combination of distance-based, spatio-temporal, and distribution-based metrics.

The number of trainable parameters in the neural network trunk and in task-specific heads is important. We address the problem in the following. We construct for each dataset nine architectures with different split points. Architectures $A_0$ and $A_1$ are Single Task Learning (STL) CNNs for the MTSC task and the trajectory prediction task. Architectures $A_2$ to $A_8$ combine both tasks by MTL. All IMU-based architectures are given in Fig. 3. All visual-based architectures are given in Fig. 4, where we search for the optimal number of LSTM units and dropouts (see Fig. 11). We choose a combination of 500 and 100 LSTM units, and two dropout layers of 20%. An overview of all architectures and its number of trainable parameters is given in Table 5 and in Table 6. The number of total parameters increases for an early split point compared to late split points for both networks. A regression head with one dense layer of 200 neurons has 20,200 parameters, while a classification head with one dense layer of 83 neurons has 16,683 parameters. Fig. 12 compares the training loss of all architectures for the regression and classification tasks.

A.3. Evaluation Results

Hyperparameter Search for the Inertial-based Architectures. For the alternative distance-based metrics, i.e., Andrew’s Sine and Huber, we search for the hyperparameters $\delta_H$ and $\delta_{AS}$ in $\{0.1, 0.2, \ldots, 2.0, 2.5, \ldots, 5.0\}$ using a grid search (Fig. 14a). For $\mathcal{L}_{AS}$ we choose $\delta_{AS} = 0.3$, and for $\mathcal{L}_H$ we choose $\delta_H = 4.0$ for follow-up training. A large $\delta_H$ tends to weight outliers more, while a small outlier rejection has a higher standard deviation. We also search for the hyperparameter $p \in \{1, 2, 3, 4\}$ of the distribution-based loss $\mathcal{L}_{WAS_p}$ (Fig. 14a) and choose $p = 1$ for follow-up training.
Hyperparameter Search for Visual-based Architectures.
As for the inertial-based architectures, we search for the optimal hyperparameter of alternative distance-based metrics using a grid search (see Fig. 14b). For $L_{AS}$ we choose $\delta_{AS} = 2.5$, and for $L_{H}$ we choose $\delta_{H} = 2.0$ for follow-up training. We also search for the hyperparameter $p$ in the distribution-based loss $L_{WAS_p}$ and choose $p = 4$ for follow-up training as it achieves the highest classification accuracy for both the single $L_{WAS_4}$ loss and the combination of $L_{MSE}$ and $L_{WAS_4}$.

Camera-based reconstruction. In Fig. 13 we propose additional trajectory reconstruction results based on the visual dataset. We come to similar conclusions as for the inertial-based dataset. While the CNN trained with the $L_{MSE}$ loss combined with the $L_{PC}$ loss (Fig. 13d) predicts a smoother trajectory than the single distance-based loss functions (Fig. 13a), the $L_{MSE} + L_{WAS}$ loss allows a proper training (Fig. 13f).

Training Times. For all loss combinations, we present inference times (Table 7). While the visual-based CNN takes 5.3971s for each epoch (on average), the IMU-based CNN only takes 0.3022s. The differences between the loss functions are small for the IMU-based architectures. From the visual-based CNNs we can see that the spatio-temporal and distribution-based loss functions (4.4267s) are less compute-intensive compared to the distance-based loss functions (5.8488s). The marginally increased computing time for all loss combinations (5.9158s) is negligible.

<table>
<thead>
<tr>
<th>Loss function</th>
<th>IMU-based CNN</th>
<th>Visual-based CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_{MSE}$</td>
<td>0.3052 ± 0.0401</td>
<td>5.8961 ± 0.0533</td>
</tr>
<tr>
<td>$L_{H}$</td>
<td>0.2971 ± 0.0401</td>
<td>5.8154 ± 0.1067</td>
</tr>
<tr>
<td>$L_{AS}$</td>
<td>0.2993 ± 0.0399</td>
<td>5.8349 ± 0.1259</td>
</tr>
<tr>
<td>$L_{PC}$</td>
<td>0.3027 ± 0.0394</td>
<td>4.4017 ± 0.0924</td>
</tr>
<tr>
<td>$L_{CS}$</td>
<td>0.3001 ± 0.0395</td>
<td>4.4246 ± 0.0488</td>
</tr>
<tr>
<td>$L_{WAS_1}$</td>
<td>0.2994 ± 0.0395</td>
<td>4.4538 ± 0.0996</td>
</tr>
<tr>
<td>$L_{MSE} + L_{PC}$</td>
<td>0.3078 ± 0.0397</td>
<td>5.9077 ± 0.0610</td>
</tr>
<tr>
<td>$L_{MSE} + L_{CS}$</td>
<td>0.2984 ± 0.0387</td>
<td>5.8790 ± 0.1133</td>
</tr>
<tr>
<td>$L_{MSE} + L_{WAS_1}$</td>
<td>0.3096 ± 0.0396</td>
<td>5.9607 ± 0.1253</td>
</tr>
</tbody>
</table>
A.4. Dataset Feature Embedding Evaluation

Fig. 15 illustrates the challenges of the inertial dataset using a 300 dimensional feature embedding. The figure represents the feature embedding based on the t-SNE algorithm [56] (initial dimension of 301, perplexity of 30, an initial momentum of 0.5, and a final momentum of 0.8) and incorporates all 83 classes, i.e., capital and small characters, numbers and symbols. As it can be seen, the classes can be well separated. The main difficulty is to differentiate capital and small letters, e.g., 'V' and 'v', 'K' and 'k', and 'X' and 'x', as these differ only in the size and not in the number of strokes. Naturally, the embeddings of the letters 'O', 'o' and '0' are very close to each other. Furthermore, the classes 'G', '6', 'C', 'c' and '(' are challenging to distinguish. These are one of the most frequent errors of the networks for the MTSC task.