Surface Similarity Parameter: A New Machine Learning Loss Metric for Oscillatory Spatio-Temporal Data

Mathies Wedler[a,*], Merten Stender[a,], Marco Klein[a,], Svenja Ehlers[a,], Norbert Hoffmann[a,b,]

[a]*Hamburg University of Technology, Dynamics Group, Schlossmühlendamm 30, 21073 Hamburg, Germany*
[b]*Imperial College London, Department of Mechanical Engineering, 58 Prince's Gate, South Kensington, London SW7 1AY, United Kingdom*

Abstract

Supervised machine learning approaches require the formulation of a loss functional to be minimized in the training phase. Sequential data are ubiquitous across many fields of research, and are often treated with Euclidean distance-based loss functions that were designed for tabular data. For smooth oscillatory data, those conventional approaches lack the ability to penalize amplitude, frequency and phase prediction errors at the same time, and tend to be biased towards amplitude errors. We introduce the surface similarity parameter (SSP) as a novel loss function that is especially useful for training machine learning models on smooth oscillatory sequences. Our extensive experiments on chaotic spatio-temporal dynamical systems indicate that the SSP is beneficial for shaping gradients, thereby accelerating the training process, reducing the final prediction error, increasing weight initialization robustness, and implementing a stronger regularization effect compared to using classical loss functions. The results indicate the potential of the novel loss metric particularly for highly complex and chaotic data, such as data stemming from the nonlinear two-dimensional Kuramoto–Sivashinsky equation and the linear propagation of dispersive surface gravity waves in fluids.

Keywords: deep learning, loss function, error metric, similarity, nonlinear dynamics, spatio-temporal dynamics

1. Introduction

Given the fact that almost any natural process is dynamic and non-stationary, sequential data is ubiquitous across many scientific disciplines, such as weather forecasting, finance, natural language processing, acoustics and many more. Besides classical analytical and numerical techniques (Anderson, 1995; Wriggers, 2008), machine learning approaches have been adopted lately for various regression or classification tasks on sequential data given a continuous quantity $y(t)$ sampled at discrete instants $\mathbf{y}(t) = [y(t_1), \ldots y(t_N)]$ (Bagnall et al., 2016; Zhang et al., 2016; Shi & Yeung, 2018; Herzog et al., 2018; Vlachas et al., 2018; Fawaz et al., 2019; Legaard et al., 2021; Stender et al., 2021). Most data-driven approaches are rooted in a supervised optimization framework for a model's f parameters θ, essentially targeting at minimizing an error functional $J(\mathbf{y}, \hat{\mathbf{y}}(\theta))$ that describes the discrepancy between the model prediction $\hat{\mathbf{y}}$ and the ground truth $\mathbf{y}$ (LeCun et al., 2015; Singh et al., 2016). Some techniques utilize sequence featurization (Fulcher & Jones, 2014), i.e. turning sequential data into vectors of descriptive scalar values, while direct approaches perform the error computation on the raw sequences themselves (Hochreiter & Schmidhuber, 1997; Fawaz et al., 2019). There is a plethora of metrics for measuring the similarity of two (multivariate) sequences, such as generic time domain distance metrics or custom metrics tailored towards specific aspects of the given data (Mue, 2007; Fulcher et al., 2013; Jiang et al., 2019). The choice of a specific error metric as loss function

---

for the training of machine learning methods is of crucial importance: the loss function shapes the gradient (LeCun et al., 2015), thus the convergence behavior of the optimization, and potentially biased predictions.

Conventional Euclidean distance metrics such as the mean squared error (MSE) and mean absolute error (MAE) do not take the sequential data characteristic into account, nor smoothness or continuity of the predicted sequences. Instead, considering sequences as a set of values sampled from some distribution, the MSE and MAE aim at fitting the mean and median, respectively, of that distribution. Often times, a spectral decomposition of complex (spatio-) temporal dynamics is utilized to describe the data in a finite series of harmonics, each described by amplitude, frequency and phase (Giannakis et al., 2019). From this viewpoint, Euclidean distance metrics are biased towards amplitude errors, rendering them as sub-optimal loss functions for some machine learning tasks as our work will illustrate. We introduce the surface similarity parameter SSP (Perlin & Bustamante, 2014) as a novel loss metric, that is capable of combining amplitude, frequency and phase errors into a single scalar error metric. Thereby, the SSP can provide several benefits for machine learning applications on complex oscillatory sequential data.

Studying the highly chaotic dynamics of the a) one- and b) two-dimensional Kuramoto–Sivashinsky equation (KS-eq.), and the c) propagation of dispersive surface waves of a very broad spectrum, our work indicates that the SSP is a particularly beneficial loss function the more complex the sequences are, that is, the wider the underlying wave spectrum becomes (i.e. the larger the number of relevant harmonics in the spectral decomposition). Our numerical experiments indicate that the SSP enhances optimizer convergence through effective gradient shaping, increases robustness against broad-banded spectra and weight initialization, and introduces strong regularization effects. Further, the benefits of the SSP loss function are shown to be rooted in an error saturation effect, that is effectively bounding the degree to which two sequences can be dissimilar, thereby forming sharper global minima in a flattened error functional in the parameter optimization space.

This work is structured as follows: Section 2 introduces the surface similarity parameter, re-visits major considerations on machine learning loss functions and briefly introduces the experimental settings. Section 3 illustrates several in-depth studies comparing different loss metrics for learning a convolutional encoding-decoding flow map for the KS-eq. and a broad-banded dispersive wave spectrum.

2. Methods

We will introduce the SSP first, then compare against conventional Euclidean distance metrics, and finally highlight the property of an upward bounded loss surface of the proposed loss function. Subsequently, we briefly present the experimental settings for evaluation of the SSP as a machine learning loss function.

2.1. Surface similarity parameter

Perlin & Bustamante (2014) introduce the surface similarity parameter (SSP) as a quantitative comparison metric for two surfaces $\mathbf{y}_1, \mathbf{y}_2$ with spatial and/or temporal extent in one or more dimensions. In this work, we only consider surfaces with spatial extent, such that the SSP formulation can be written as

$$J_{\mathrm{SSP}}(\mathbf{y}_1, \mathbf{y}_2) = \frac{\sqrt{\int |F_{\mathbf{y}_1}(\mathbf{k}) - F_{\mathbf{y}_2}(\mathbf{k})|^2 d\mathbf{k}}}{\sqrt{\int |F_{\mathbf{y}_1}(\mathbf{k})|^2 d\mathbf{k}} + \sqrt{\int |F_{\mathbf{y}_2}(\mathbf{k})|^2 d\mathbf{k}}} \in [0, 1] \tag{1}$$

where $F_{\mathbf{y}}$ denotes the Fourier transform of the signal $\mathbf{y}(\mathbf{x}, t)$. $\mathbf{k}$ denotes the wave number vector $\mathbf{k} = \begin{bmatrix} k_x \end{bmatrix}$ for one-dimensional or $\mathbf{k} = \begin{bmatrix} k_x, k_y \end{bmatrix}^{\mathsf{T}}$ for two-dimensional spatial inputs, respectively. The SSP is based on the L^2 norm, originally denoted as a special case of the Sobolev norm by Perlin & Bustamante (2014), of the individual signals $||\mathbf{y}_i||^2 = \int |F_{\mathbf{y}_i}(\mathbf{k})|^2 d\mathbf{k}$ and their difference in space domain $||\mathbf{y}_1 - \mathbf{y}_2||^2 = \int |F_{\mathbf{y}_1}(\mathbf{k}) - F_{\mathbf{y}_2}(\mathbf{k})|^2 d\mathbf{k}$. By effectively measuring the similarity of two signals in Fourier space, the SSP inherently penalizes amplitude-, frequency- and phase errors uniformly in a single quantity[1]. If both signals are

---

[1]Similar to the A-weighting of sound pressure levels, the SSP may be tuned towards a non-balanced weighting of amplitude-, frequency- and phase errors by imposing a weighting function on the Fourier spectra $F_{\mathbf{y}_i}$

identical, the L^2 norm of their difference is 0, such that SSP = 0 denotes perfect agreement among the signals. With increasing difference between $\mathbf{y}_1$ and $\mathbf{y}_2$, the SSP approaches 1. In other words, the SSP is a normalized measure for the relative error between two signals, which greatly promotes interpretability. It is noteworthy that SSP = 1 (Perlin & Bustamante (2014) refer to SSP = 1 as *perfect disagreement*) is only achieved for two phase-inverted signals disregarding their respective amplitudes. While the SSP definition by Perlin & Bustamante (2014) utilizes the continuous Fourier transform, we compare discrete signals sampled at the same rate and thus use the discrete Fourier transform (DFT) computed by the fast Fourier transform algorithm (FFT) (Cooley & Tukey, 1965). By relying on the FFT algorithm, we impose two requirements on signals compared with SSP: While it is mandatory that the signals have enough grid points to include enough harmonic components in Fourier space to faithfully represent the original signals, the performance of the SSP calculation improves drastically when signals sampled at 2^n, $n \in \mathbb{N}$ grid points are compared.

The SSP is applied in the field of ocean engineering to quantify numeric ocean wave prediction (Klein et al., 2020) and -reconstruction methods (Desmars et al., 2021) as well as for ensuring consistent conditions in ship movement experiments (van Essen, 2021). Other applications include the quantification of denoised GPS signals (Sharie et al., 2020) and demodulated fringe pattern (Hernandez-Lopez et al., 2021). Despite occasional usage in a variety of fields, no publication reports an in-depth analysis on why the SSP is preferred over established Euclidean distance metrics in those fields. Furthermore, and to the knowledge of the authors, the SSP has not been used as a loss metric for machine learning-related research yet. The following paragraphs will highlight conceptual differences of the SSP and Euclidean error metrics, and point out aspects that render the SSP a preferable loss function for machine learning purposes.

2.2. Is there a limit to dissimilarity?

Focusing at the predicted model output and the ground truth for long sequential data, a loss function is all about a condensation of many pieces of information into a scalar and abstract deviation quantity. In simple words, the loss function tells *how similar and how dissimilar can two sequences be?* The former case is trivial to answer by almost any Euclidean distance metric. The latter, however, is nontrivial: what is the most *dissimilar* signal $y(t)$ to a given non-zero signal $x(t)$? Euclidean metrics would answer that question by increasing the amplitudes of $y(t)$ towards infinity. Alternatively, one could state that $y(t) = -x(t)$ is the most dissimilar signal to $x(t)$, and what about $y(t) = 0$? Looking at Euclidean distance metrics, we can observe that they are un-bounded, and hence over-sensitive to amplitude errors and under-sensitive to phase or frequency errors if we consider signals composed of harmonics. On the contrary, the SSP is a bounded quantity, which is most conveniently shown by calculating the SSP for amplitude scaling signals, i.e. $\mathbf{y}_1$ and $\mathbf{y}_2 = \kappa \cdot \mathbf{y}_1$, $\kappa \in \mathbb{R}$. For this case, Eq. (1) can be simplified to the C_0 continuous function

$$J_{\text{SSP}}(\mathbf{y}_1, \kappa \cdot \mathbf{y}_1) = \frac{|1 - \kappa|}{1 + |\kappa|} \begin{cases} \in [0, 1) & \forall \kappa \in \mathbb{R}^+ \\ = 1 & \forall \kappa \in \mathbb{R}_0^- \end{cases}, \quad \frac{\mathrm{d}^2 J_{\text{SSP}}}{\mathrm{d}\kappa^2} = \begin{cases} > 0 \text{ (convex)} & \forall \kappa \in [0, 1) \\ < 0 \text{ (concave)} & \forall \kappa > 1 \end{cases} \tag{2}$$

i.e. we observe the SSP running into an upper limit of SSP = 1 for $\kappa \in \mathbb{R}_0^-$, meaning that there is a bound to the dissimilarity of two signals when measured in SSP. When used as a ML loss function, this property will flatten the loss surface in remote regions of the parameter space. The loss function shapes the gradients seen by the ML optimizer and thus directly affects the training behavior such as convergence speed and final model accuracy. In general, gradient-based optimization methods like SGD (Robbins & Monro, 1951; Kiefer & Wolfowitz, 1952) benefit from large gradients as the step size in parameter space is directly linked to the gradient of the loss surface. More advanced optimization schemes like AdaGrad (Duchi et al., 2011), AdaDelta (Zeiler, 2012), RMSprop (Tieleman & Hinton, 2018) or Adam (Kingma & Ba, 2014) introduce momentum, i.e. information about past gradients. This allows the optimizer to *accelerate* under certain circumstances. Kingma & Ba (2014) state that Adam's step size in parameter space reaches an upper bound "when a gradient has been zero at all timesteps except at the current timestep", i.e. for plateaus in the loss surface. Purely gradient-based optimizers would be trapped for $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbf{0}$, the Adam optimizer on the other hand even benefits from small gradients as shown in Appendix B. When training a ML model, the initial weights $\boldsymbol{\theta}_0$ are sampled from a certain distribution (we refer to Narkhede et al.

(2021) for a comprehensive overview on weight initialization methods), i.e. the optimization is started at a random location in parameter space. When working with signals composed of harmonics, the prediction of the randomly initialized model will almost certainly include amplitude-, phase- and frequency errors at the same time. In consequence, the optimizer will never encounter a perfectly flat plateau in the loss surface when using the SSP but rather very small gradients $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \mathbf{0}$.

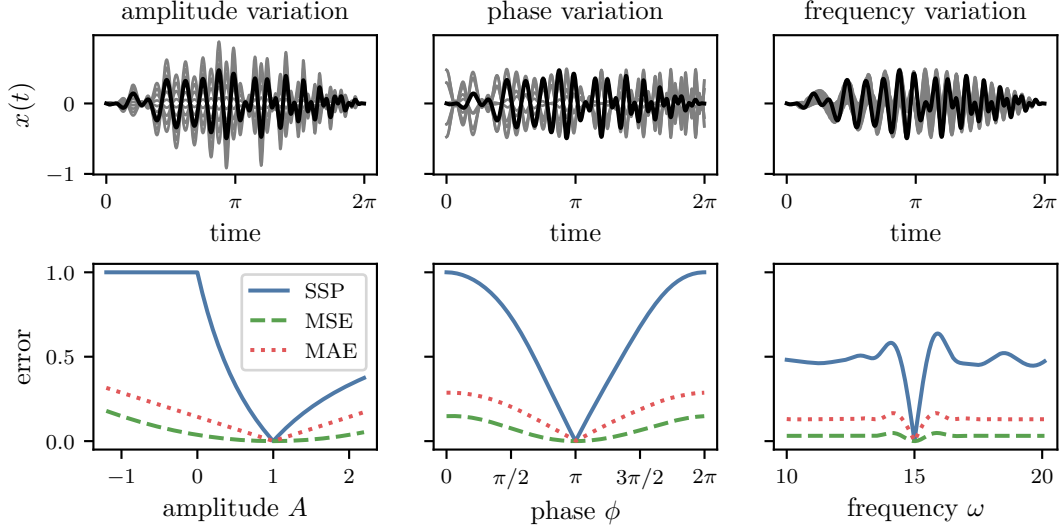*2.3. Comparison against Euclidean distance metrics*

Figure 1: Evaluation of three different distance metrics under amplitude, phase and frequency variation of the sample signal $x(t) = A \cdot (0.5 \sin(0.5t + \phi) \cdot (0.5 \cos((2 + 3t)t) + 0.5 \cdot \cos(\omega t))$ (gray lines) compared against the baseline signal (solid black line in top panel) defined by $A = 1$, $\phi = \pi$, $\omega = 15$

Figure 1 displays the MSE and MAE metrics in comparison to the SSP for a synthetic test signal composed of three arbitrary harmonic components including a variable phase term ϕ, a variable frequency term ω as well as an amplitude scaling envelope term A. Looking at the frequency variation about the reference configuration, there is no qualitative difference among the metrics, particularly not in the gradients of the metrics in vicinity of the optimum. In contrast, this observation does not hold for the amplitude variation. Per definition, the MAE exhibits a constant gradient and the MSE shows a linearly decreasing gradient, i.e. becoming very flat, towards the optimum. The SSP is convex to the left, and concave to the right of the optimum according to Eq. (2). While the Euclidean metrics increase monotonously for positive and negative deviations from the reference amplitude, the SSP exhibits the previously discussed upper limit of SSP = 1 for zero and negative amplitudes, as well as for larger positives amplitudes. There exists a large and perfectly flat error functional for negative amplitudes A, hence forcing the optimizer to take large steps and escape that desert of zero gradients. Once the optimizer approaches the optimum neighborhood, suddenly gradients become large towards the minimum. Similarly, if the true solution is approached from the far right, i.e. positive A values, the error becomes steeper as the minimum is approached. This is not the case for the convex MSE, which becomes flat towards the minimum.

When training a ML model on oscillatory data, the prediction most certainly includes amplitude-, phase- and frequency errors at the same time. Starting from some remote region in the model's parameter space, the loss function is minimized by iterative weight updates. Since Euclidean distance metrics are over-sensitive to amplitude errors, MSE (and MAE) will first fit the mean (and median) amplitudes of $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$, while phase- and frequency errors play a minor role compared to amplitude. The SSP on the other hand is stricter as it only decreases once a weight-update tackles amplitude-, phase- and frequency errors at the same time. This property will most likely translate into less local minima in the SSP loss surface compared to MSE and

MAE. In other words, the SSP is more robust against weight initialization effects as it squashes the error functional and thus local minima in remote regions. One may even conclude that the strictness of the SSP leads to less non-convex regions in the loss landscape.

2.4. Experimental settings

For the numerical experiments, we evaluate the performance of a fully convolutional encoder-decoder model learning a flow map, i.e. taking a number of $n_\tau = 8$ states at preceding time steps and making a prediction about the system state at the next time step. As illustrated in Figure 2, the model consists of an encoding path with three stages and a symmetric decoding path. Each convolutional layer in the encoding path is followed by downsampling along the spatial dimension via Max Pooling (upsampling via transposed convolutions in the decoder path), such that we extract features on different scales. While the spatial dimension of the internal input representation is bisected, the amount of convolutional filters is doubled per stage (divided in halve in the decoding path). We apply a final convolutional layer with a filter size of 1 to output $\hat{\mathbf{y}} \in \mathbb{R}^{1024}$. The Adam optimizer (Kingma & Ba, 2014) is employed with learning rate $\alpha = 1e-3$, 1st order exponential decay $\beta_1 = 0.9$, 2nd order exponential decay $\beta_2 = 0.999$ and $\varepsilon = 1e-7$. For the SGD optimizer (Robbins & Monro, 1951; Kiefer & Wolfowitz, 1952), a learning rate $\alpha = 1e-2$ is set with no momentum.
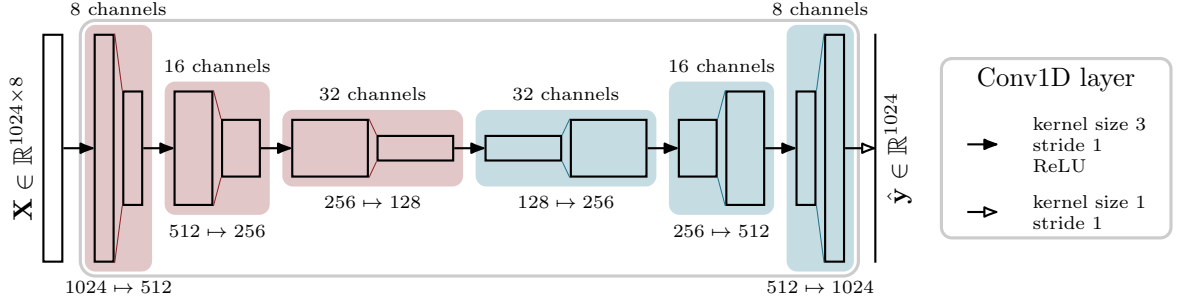

Figure 2: Fully convolutional network with encoder-decoder structure for multivariate sequential data. The model is build from three encoder- (red) and decoder blocks (blue), each of which applies a convolution and a pooling operation. The number of channels is stated above, the pooling operation is stated below each block. By alternating convolution and up (or down-) sampling, the model is able to extract features on different scales. Overall, the model maps an input $\mathbf{X} \in \mathbb{R}^{1024 \times 8}$ to an output $\hat{\mathbf{y}} \in \mathbb{R}^{1024}$. For the two-dimensional evaluation cases, the respective two-dimensional instances of the layers depicted here are used, e.g. Conv1D is replaced by Conv2D

As typical test cases, we study the chaotic spatio-temporal dynamics of the (a) one- and (b) two-dimensional Kuramoto–Sivashinsky equation (KS-eq.) (C.1) and the (c) time evolution of dispersive surface waves. Figure 3 shows one qualitative sample of each test case. Cases (a) and (b) have been studied as benchmark cases in many recent works on data-driven forecasting methods (Pathak et al., 2017, 2018; Raissi & Karniadakis, 2018; Vlachas et al., 2020). Importantly, we note that our aim is *not* to propose an optimal neural architecture for future state forecasting, but to study the training behavior of a given model architecture under different loss functions. The training behavior of the given neural architecture is studied under different choices of the loss function $J \in \mathbb{M} = \{\text{SSP}, \text{MSE}, \text{MAE}, \text{RMSE}, \text{Huber}\}$, RMSE: Root Mean Squared Error, Huber: Huber loss (Huber, 1964) with $\delta = 1$. All metrics in the set $\mathbb{M} \setminus \{J\}$ are used as error metrics to observe the training behavior in their respective quantity, allowing us to evaluate a model trained on loss function J in terms of all metrics in $\mathbb{M}$. To cross-validate our results and analyze their variability, each loss function $J \in \mathbb{M}$ is used to train the model on 50 pseudo random data splits: The set of all preprocessed samples ($\mathbf{X}_i \in \mathbb{R}^{1024 \times 8}, \mathbf{y}_i \in \mathbb{R}^{1024}$) is deterministically shuffled using the training count $j \in \{1, 2, ..., 50\}$ as random seed and then split with ratio 70/30 into training- and validation set. The preprocessing routine is briefly described along the data generation in Appendix C.1. The smaller the variance in the validation set scores, the more robust a model is against weight initialization and training data distribution.
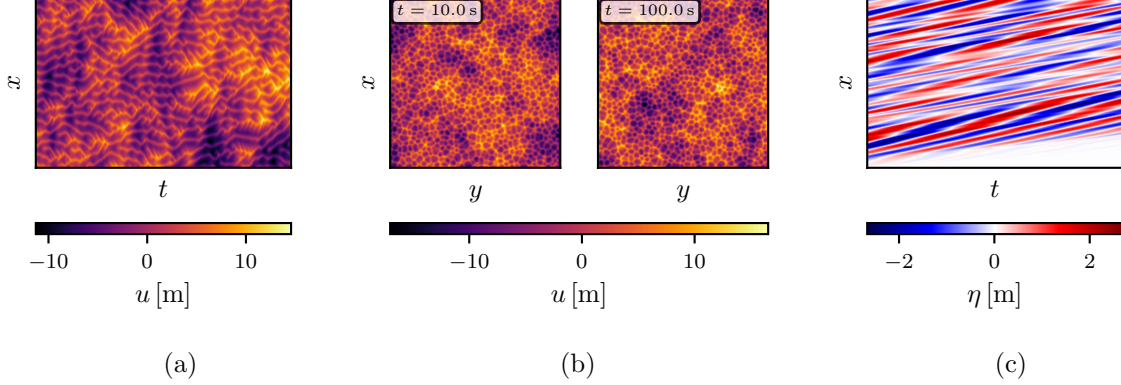
Figure 3: Qualitative samples for each numerical test case: (a) one-dimensional KS-eq., (b) two snapshots ($t = \{10, 100\}$ s) of the two-dimensional KS-eq. and (c) one-dimensional broad-banded surface waves

3. Results

We study the SSP as novel machine learning metric in regards of three central aspects: i) the property to shape gradients (convergence), ii) the regularization effect of the loss function (smoothness and final loss), and iii) compatibility with conventional optimizers (instability) such as Adam (Kingma & Ba, 2014) and SGD (Robbins & Monro, 1951; Kiefer & Wolfowitz, 1952). The SSP is chosen as the baseline metric for assessing the similarity between the ground truth $\mathbf{y}$ and prediction $\hat{\mathbf{y}}$. All validation and prediction errors presented hereafter hence are computed in terms of SSP, even if the individual models were trained towards a different loss function. We are well aware that this choice can seem biased or doubtful to the reader. For this reason, we will consult the well-established MSE metric for reference alongside the SSP as a validation set metric. Under outermost neutrality, we are convinced that the SSP is superior to Euclidean metrics for comparing two signals that are smooth and oscillatory following the reasoning presented before.

We introduce the convention of stating the model training loss function as a subscript of the evaluation metric, e.g. SSP_{MSE} indicates the validation set error measured in terms of SSP for a model trained with MSE loss function. As the loss function shapes the gradient used for training, a model trained on the MSE loss is, naively, expected to minimize the MSE validation loss best, while a model trained on the SSP loss is expected to arrive at the smallest SSP validation loss.

3.1. Chaotic dynamics in the 1D KS equation

The training behavior of machine learning models is usually evaluated using the loss curve, which provides information about the final loss value a model arrives at, the speed of convergence and the overall data fitting. We present and discuss exemplary validation loss curves for all loss functions in the following. Plotting the average loss curves over all 50 training runs would smooth out certain characteristics, hence Figure 4 shows a representative validation loss curve per loss function and evaluation metric. Particularly, loss curves are shown for training runs on the exact same data split. Model performance on the validation set is evaluated in terms of the SSP (Figure 4 (a)) and MSE metric (Figure 4 (b)) after each epoch. Box plots in Figure 4 depict summary statistics derived from all 50 training runs per loss function. For all loss functions and all model training runs there was no overfitting observable.

To assess the effect of the different loss functions on the training behavior, we first discuss the convergence speed. The horizontal box plots in Figure 4 depict the number of training epochs that was required to reach a certain loss threshold, e.g. SSP = 0.01 and MSE = 0.01, respectively. On average, models trained with SSP loss reached the threshold within 30 epochs, earliest at 17 and latest after 60 epochs. Models trained on RMSE loss show comparable convergence speed, although the mean number of epochs is 41. Huber loss and MSE loss show slower convergence with median of 53 and 64 epochs, respectively. The MAE loss shows a
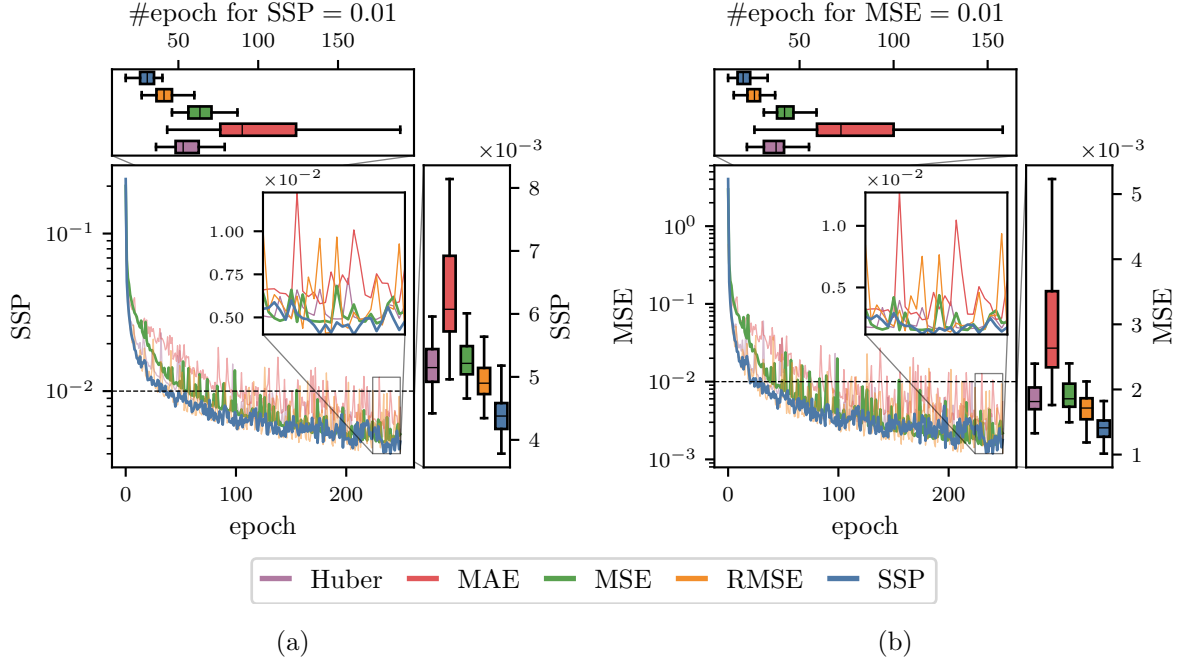
Figure 4: Model training on data of the 1D KS-eq. ($\nu = 0.25$, $\Delta t = 0.1\,\mathrm{s}$) evaluated per (a) SSP metric and (b) MSE metric. 50 training runs on random data splits are performed for each loss. The loss curves show the validation loss of one exemplary but representative training run. The vertical box-and-whisker plots show the distribution of the minimum validation loss over the last 25 epochs across all 50 training runs. The horizontal box-and-whisker plots above each loss curve show the distribution of the epoch count at which the loss falls below the threshold SSP = 0.01 (MSE = 0.01, indicated by the horizontal dashed line)

clearly different result with a very large variation and 90 epochs required on average to achieve SSP = 0.01. The variance in the model convergence behavior is minimal for the SSP and maximal for the MAE: The MAE shows the largest interquartile range (IQR) of 48 epochs. MSE and Huber both have an IQR of 15 epochs. RMSE (10 epochs) and SSP (9 epochs) have the lowest IQRs. As non-overlapping IQRs allow for statements that are statistically significant, this experiment indicates the superiority of the SSP loss function in terms of convergence speed. The qualitatively same picture is resembled when evaluating the validation set predictions in terms of MSE instead of SSP, see Figure 4 (b). Notably, models trained on SSP loss will minimize the MSE-rated prediction error faster than models trained on MSE loss. This observation will be confirmed in the following experiments on different data sets. Therefore, the SSP loss function is shown to extend the features taken into account by Euclidean loss functions, specifically the phase and frequency information, which was exemplified in Section 2.3.

Next, we focus on the final validation loss, i.e. the performance of several models trained for 250 epochs on the same data set but given a different loss function. The vertical box-and-whisker plots in Figure 4 shows the distribution of the minimum loss value within epochs 225 and 250 across all training runs per loss function. The models trained with SSP loss arrive at a median validation loss of $\mathrm{SSP_{SSP}} = 0.0044$, and the model trained on MSE loss arrives at $\mathrm{SSP_{MSE}} = 0.0052$. The Huber loss performs roughly the same as the MSE, the RMSE arrives at a slightly lower final loss value. Lastly, the MAE shows the weakest performance at a median of $\mathrm{SSP_{MAE}} = 0.0061$. Considering the variance among all training runs, the MAE has the largest IQR of 0.0012. The remaining loss functions exhibit similar IQRs at 0.0004 (SSP and RMSE) and 0.0005 (MSE and Huber). As the IQR of the SSP does not overlap with any of the other loss function's IQRs, one can state that the superiority of the SSP in terms of lowest final loss is statistically significant. Lastly, one can observe that the curves fluctuate for all loss functions. However, in direct comparison, the SSP loss function exhibits the smoothest loss characteristic. The loss curves allow to glance at the optimizer

paths towards a minimum: The less fluctuating loss curve for the SSP thus indicates that the optimizer has less trouble finding it's path downwards on the loss surface compared to the other loss functions.

To conclude, for the presented case, using the SSP loss function results in smoother loss curves which point towards better regularization properties than the competing loss functions. The SSP loss function not only speeds up the training but also results in better validation scores after a pre-defined number of training epochs. Training a model with SSP loss function results in lower MSE validation errors as when training a model with MSE loss. This behavior indicates the regularization property, i.e. shows that the SSP is penalizing small prediction errors more strongly than other loss functions.

3.2. Gradient shaping through SSP loss function

The training behavior discussed in the previous paragraph indicates several advantages of the SSP loss over other loss functions for the given prediction task. In order to understand some of those observations, and to come back to the discussion in Section 2.3, Figure 5 displays the loss hyperplane of a single model (1D KS-eq., $\nu = 0.25$, $\Delta t = 1.0$ s, SSP loss) at different stages of the training process, namely after epochs 1, 5, 10, and 500. For this analysis, two weights in the last layer, θ_1^* and θ_2^*, are varied while all other weights are kept frozen. The prediction of that model for a single data sample is performed for a wide variation of θ_1, θ_2 about the final weights $\theta_1^* = -1.07$, $\theta_2^* = 0.86$ reached after 500 epochs of training. For every choice of the weight values, the model prediction is measured in terms of SSP, MSE, and MAE metric. As a result, one can study the loss hyperplane and its gradients that the optimizer would see for that single data sample if only two weights of the model were trainable. Even though this situation does not reflect the actual training process, the error surfaces depict salient differences that can link to the observations made before on the training behavior of the SSP loss. The error isolines of the MAE are separated by a constant value, indicating a linear gradient towards the minimum. The MSE shows a different behavior, which resembles the shape of a flat bowl as gradients get smaller towards the optimum. The SSP error surface is not as circular, and exhibits large values particularly in the upper left parameter region. Gradients are very large in the anti-diagonal direction of the parameter space. The most important difference in the three metrics lies in the area surrounding the minimum at (θ_1^*, θ_2^*): while the MSE would allow for a significant change in the weights without crossing the lowest isoline, the SSP would increase very quickly under small variations of the weights. For example, reducing θ_1^* by 10% will increase the MSE by 0.3%, the MAE by 2.1%, and the SSP by 56.5%, in relation to the full range of the respective hyperplane, for the model after epoch 10. While the Euclidean metrics may accept a small variation in that weight, the SSP would respond with a severe increase of the loss value. Concluding, the local minima in the illustrated SSP error surface are very sharp and narrow, therefore the training process may converge faster and more precisely to the minimum, as the SSP is not as lenient to weight changes as the MAE and MSE losses are. While we were able to reproduce the preceding result by changing randomly picked sets of weights from the last layer, it is not feasible to illustrate the loss surface for all parameters. Here, we want to point towards the results for the actual training (i.e. all parameters are involved) presented in Section 3.1 to statistically support our hypothesis.

3.3. Effect of prediction task complexity

This section displays several experiments on the one- and two-dimensional KS-eq. for different values of ν, i.e. different levels of chaos in the physical problem. Further, the time step Δt is varied, which denotes the time difference between consecutive data samples used as input to the flow map model. Hence, for smaller ν and for larger Δt the learning task becomes more demanding, as more chaotic dynamics are to be predicted over longer time instants into the future. The role of small perturbations and inaccuracies is increased according to the exponential divergence driven by positive Lyapunov spectra. Table 1 displays validation scores for five KS-eq. cases of increasing complexity. For each case, models are trained on five different loss functions, and the validation set scores are evaluated in terms of the SSP and MSE metric. Cross-validation is performed against 50 realizations (10 for the 2D cases) of each experiment.

The one-dimensional KS-eq. case using $\nu = 1.0$, $\Delta t = 0.1$ s represents the seemingly simplest prediction task. Here, models trained on Huber loss outperform models trained on any other loss function after the same amount of training epochs. Huber loss reduces both validation metrics the most, and also across all
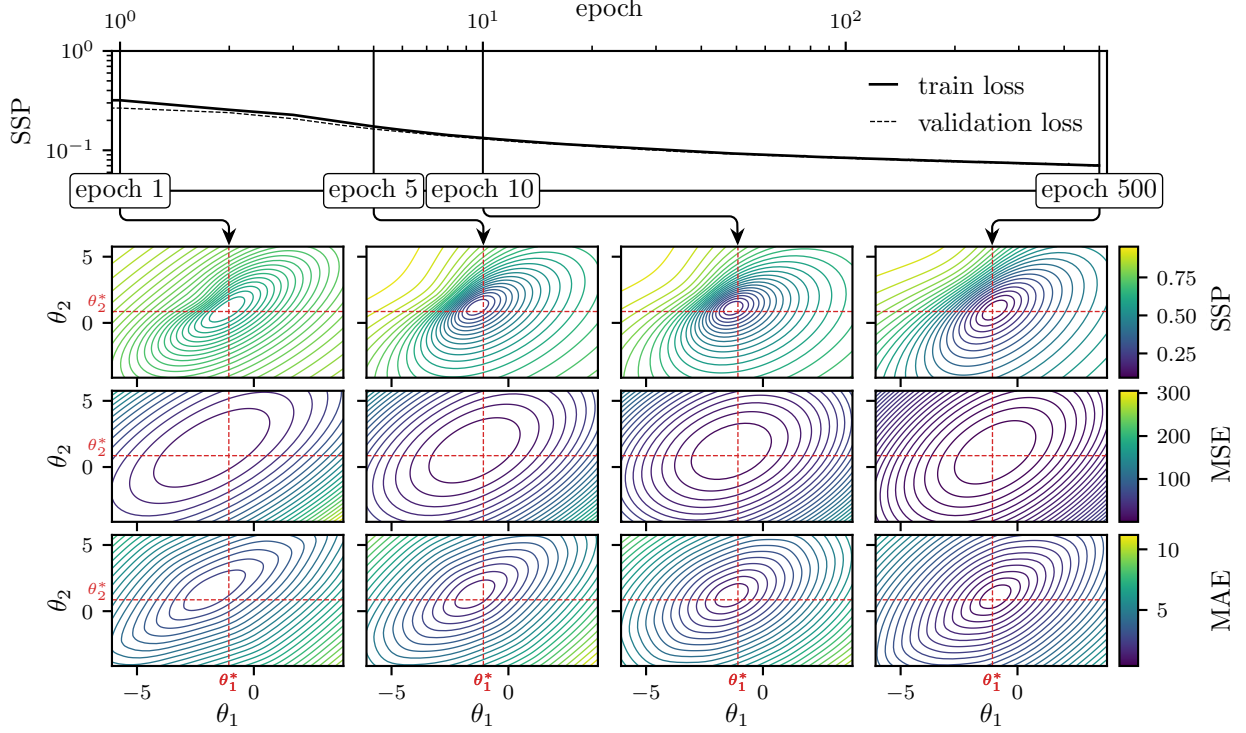
Figure 5: Visualization of loss hyperplanes for a single data sample by variation of two weights θ_1, θ_2 in the last layer of the model about the final weights θ_1^*, θ_2^* reached after 500 epochs. Loss hyperplanes are displayed for four stages of the training, and model predictions are evaluated in terms of SSP (top row), MSE (middle row) and MAE (bottom row) for the 1D KS-eq. ($\nu = 0.25$, $\Delta t = 1.0\,\text{s}$). The overall training process is depicted in the top panel

50 different models trained on different splits of the data set. The SSP loss achieves weaker results, while the MSE loss achieves only slightly weaker results than the Huber loss. Moreover, the range of validation scores in the cross validation study is minimal for the Huber loss models. With increasing prediction task complexity ($\nu = 0.25$, $\Delta t = \{0.1, 1.0\}\,\text{s}$), we observe an expected degradation in overall prediction quality for all models. However, both experiments show a similar result: models trained on the SSP loss achieve the best validation scores, even in terms of MSE. That is, if a model is trained using SSP loss, it outperforms a model trained on MSE, even if the validation score is measured in terms of MSE. Moreover, the standard deviation of validation scores across all 50 models trained on SSP loss is the smallest for both experiments and both validation metrics across all loss functions. For the two-dimensional KS-eq. cases, a diverse observation is made: using a time increment of $\Delta t = 0.1\,\text{s}$, no performance degradation is visible, and even an increase in prediction accuracy can be read from the SSP metric when compared to the one-dimensional case studies. However, as the time increment is increased, the prediction task is rendered much more demanding as with larger time steps, the effect of nonlinearities will become more prominent such that the next system state cannot be estimated by linear extrapolation from the current system state. As a result, considerable performance degradation is observed for $\Delta t = 1.0\,\text{s}$ throughout all loss functions. Still, the SSP loss function manages to minimize the loss in terms of both SSP and MSE metric best. The remaining loss functions perform on par with each other, showing only minor deviations of $\Delta_{\text{SSP}} \leq 1.3 \times 10^{-3}$ or $\Delta_{\text{MSE}} \leq 4.8 \times 10^{-2}$ between results.

The experiments conducted on the Kuramoto-Sivashinsky equation indicate that the SSP is especially well-suited for more complex prediction tasks on sequential data with an oscillatory character. While the Huber loss performs best on the least complex problem, the models trained with SSP loss achieve better validation scores for the remaining problems. The 2D implementation of the SSP manages to pull ahead

9

Table 1: Validation scores for all case studies, sorted by the estimated complexity of the prediction task from top (low complexity) to bottom (high complexity). For a given model and the same data set, training is performed for 250 epochs (100 for KS-2D cases) using different loss functions. In each case, the model prediction quality on the validation set is measured by metrics SSP and MSE. 50 training runs (10 for KS-2D) are performed for each set-up on different data splits, and mean ± standard deviation of validation scores are reported. Bold font and green backgrounds highlight the best score per metric and case

case	loss	metric SSP	metric MSE
KS-1D ($\nu = 1.0,\ \Delta t = 0.1\,\mathrm{s}$)	SSP	0.0034 ± 0.0008	0.0006 ± 0.0004
	MSE	0.0023 ± 0.0003	0.0002 ± 0.0001
	MAE	0.0045 ± 0.0012	0.0016 ± 0.0030
	RMSE	0.0030 ± 0.0006	0.0005 ± 0.0004
	Huber	$\mathbf{0.0022 \pm 0.0002}$	$\mathbf{0.0002 \pm 0.0000}$
KS-1D ($\nu = 0.25,\ \Delta t = 0.1\,\mathrm{s}$)	**SSP**	$\mathbf{0.0044 \pm 0.0003}$	$\mathbf{0.0014 \pm 0.0002}$
	MSE	0.0053 ± 0.0004	0.0020 ± 0.0004
	MAE	0.0063 ± 0.0010	0.0034 ± 0.0021
	RMSE	0.0049 ± 0.0004	0.0018 ± 0.0006
	Huber	0.0052 ± 0.0004	0.0018 ± 0.0003
KS-1D ($\nu = 0.25,\ \Delta t = 1.0\,\mathrm{s}$)	**SSP**	$\mathbf{0.0755 \pm 0.0020}$	$\mathbf{0.4554 \pm 0.0210}$
	MSE	0.0800 ± 0.0023	0.4882 ± 0.0262
	MAE	0.0812 ± 0.0021	0.5039 ± 0.0253
	RMSE	0.0781 ± 0.0023	0.4728 ± 0.0247
	Huber	0.0806 ± 0.0023	0.4958 ± 0.0243
KS-2D ($\nu = 0.25,\ \Delta t = 0.1\,\mathrm{s}$)	**SSP**	$\mathbf{0.0074 \pm 0.0003}$	$\mathbf{0.0043 \pm 0.0004}$
	MSE	0.0090 ± 0.0004	0.0061 ± 0.0006
	MAE	0.0091 ± 0.0005	0.0065 ± 0.0008
	RMSE	0.0090 ± 0.0011	0.0064 ± 0.0018
	Huber	0.0089 ± 0.0008	0.0064 ± 0.0019
KS-2D ($\nu = 0.25,\ \Delta t = 1.0\,\mathrm{s}$)	**SSP**	$\mathbf{0.2336 \pm 0.0044}$	$\mathbf{4.2762 \pm 0.0833}$
	MSE	0.2483 ± 0.0033	4.2998 ± 0.0619
	MAE	0.2473 ± 0.0025	4.3478 ± 0.0347
	RMSE	0.2484 ± 0.0035	4.3055 ± 0.0848
	Huber	0.2470 ± 0.0028	4.3220 ± 0.0743
dispersive waves ($\Delta t = 1.3\,\mathrm{s}$)	**SSP**	$\mathbf{0.0162 \pm 0.0015}$	$\mathbf{0.0005 \pm 0.0001}$
	MSE	0.0210 ± 0.0018	0.0008 ± 0.0001
	MAE	0.0242 ± 0.0016	0.0010 ± 0.0001
	RMSE	0.0190 ± 0.0016	0.0007 ± 0.0001
	Huber	0.0209 ± 0.0017	0.0008 ± 0.0001

from the competition. Most importantly, the SSP loss function manages to reduce the loss in terms of MSE better than the MSE itself for all cases except the least complex one. While the presented results are achieved using the Adam optimizer (Kingma & Ba, 2014), we ran a subset of the experiments listed in Table 1 using the SGD optimizer (Robbins & Monro, 1951; Kiefer & Wolfowitz, 1952). The loss curves for the SSP showed a very smooth convergence compared to the remaining loss functions, however training turned out to be unstable in some cases across all loss functions. Overall, the results on SGD, while seeming less stable, confirm the observations presented here.

3.4. Dispersive wave propagation

The third evaluation case deals with one-dimensional dispersive surface waves governed by the linear wave equation Eq. (C.3). Compared to the one- and two-dimensional nonlinear KS-eq., the surface waves are characterized by a very broad wave spectrum, rendering this prediction case conceptually different to the KS-eq. experiments. The dispersion relation lets long waves travel faster than short waves, hence creating complex spatio-temporal patterns. Details on the data generation are presented in Appendix C.2. In analogy to the 1D KS-eq., we cross validate our results against 50 training runs on independent and random data splits, and each model is trained for 250 epochs. Figure 6 (a) shows three exemplary model trainings using MAE, MSE and SSP as loss function. Here, we can observe that the loss curves for MAE and MSE overlap for the most part while the model with SSP loss undershoots both MAE and MSE for all epochs. The validation scores (see Table 1) indicate that models with SSP loss outperform models using different loss functions on the dispersive wave data. On average, a mean validation set score of $SSP_{SSP} = 0.0162$ is achieved. The models trained on RMSE performs second best ($SSP_{RMSE} = 0.0190$), MSE and Huber loss perform equally well at a mean $SSP \approx 0.0210$ and lastly the MAE performs the weakest ($SSP_{MAE} = 0.0242$).

To quantify the convergence speed of each loss function, we define a threshold at $SSP = 0.03$, indicated by the dashed line in Figure 6 (a). On average over all 50 training runs, models with MAE loss reach the threshold within 142 epochs, and models with MSE take 130 epochs. Models trained on RMSE and Huber loss take 108 and 120 epochs, respectively. The models trained with SSP loss converge considerably faster and reach the threshold after 78 epochs. Training on Huber and SSP loss shows the smallest variability among the loss functions with an IQR of 16 and 17 epochs, respectively. MSE and RMSE model trainings both have an IQR of 24 epochs while the MAE loss exhibits the largest variability with an IQR of 34 epochs. Variability is directly linked with the sensitivity of the training process against weight initialization and data distribution in the training sets. The low training variability in combination with the highest convergence speed makes the SSP loss function superior to the competition in this experiment. Again, the difference in convergence speed can be stated statistically significant based on the non-overlapping IQRs.

To shed more light onto the training process, we use the model to predict one validation sample after each weight update within the first 10 epochs of training (using a batch size of 2048, we obtain 81 weight updates per epoch). The results discussed hereafter are representative for a wide range of sample signals that were analyzed. In addition, both RMSE and Huber, being Euclidean distance-based metrics, exhibit the same characteristics as portrayed by MSE and MAE and are therefore not presented explicitly. Figures 6 (c) and (d) show the spectra of the predictions after 2 and 250 epochs of training, respectively. After 2 epochs, the models trained on MSE and MAE achieve validation scores around $SSP \approx 0.17$ while the model trained on SSP achieves a slightly lower validation score of $SSP_{SSP} = 0.1584$. The spectra indicate that this marginal lead is due to the fact that the model trained with SSP already attempts to fit higher frequency components while the spectra associated with MAE and MSE remain mostly flat for $k > 0.2\,\mathrm{m}^{-1}$. After 250 epochs all models fit the ground truth spectrum well over all frequencies with a slight advantage of the SSP over MSE and MAE. Figure 6 (b) shows the integral of the Fourier transform of each prediction compared against the ground truth integral value. As the integral represents a measure of overall energy contained in the predicted wave signal, this value is mostly related with amplitude errors. While the models trained with MAE and MSE loss functions slowly approach the ground truth value, the model trained with SSP exhibits an overshooting behavior in the very first training epoch. Hereafter, the SSP model quickly approaches the ground truth. These observations link well with the analytical discussion of the SSP metric and the conceptual analysis presented in Section 2.3. First, as conventional Euclidean error metrics are amplitude-biased, the overall energy in the predictions is very small during the first training epochs. To reduce the loss value, the optimizer reduces the overall amplitudes first, before energy is slowly increased to match the ground truth. On the contrary, the SSP is not amplitude-biased, such that also a frequency or phase error can increase the loss value. The above mentioned characteristic of the SSP to shape higher frequency components during early training results from this property. Second, the convergence speed towards the ground truth value is linked to the gradients in vicinity of a loss minimum: the MSE gradient becomes smaller as the minimum is approached, such that convergence becomes slower over the number of training epochs. Even for excessive training, the MSE exhibits a non-vanishing difference to the ground truth, which is fundamentally rooted in the bowl-type shape of the MSE. The constant gradient of the MAE results
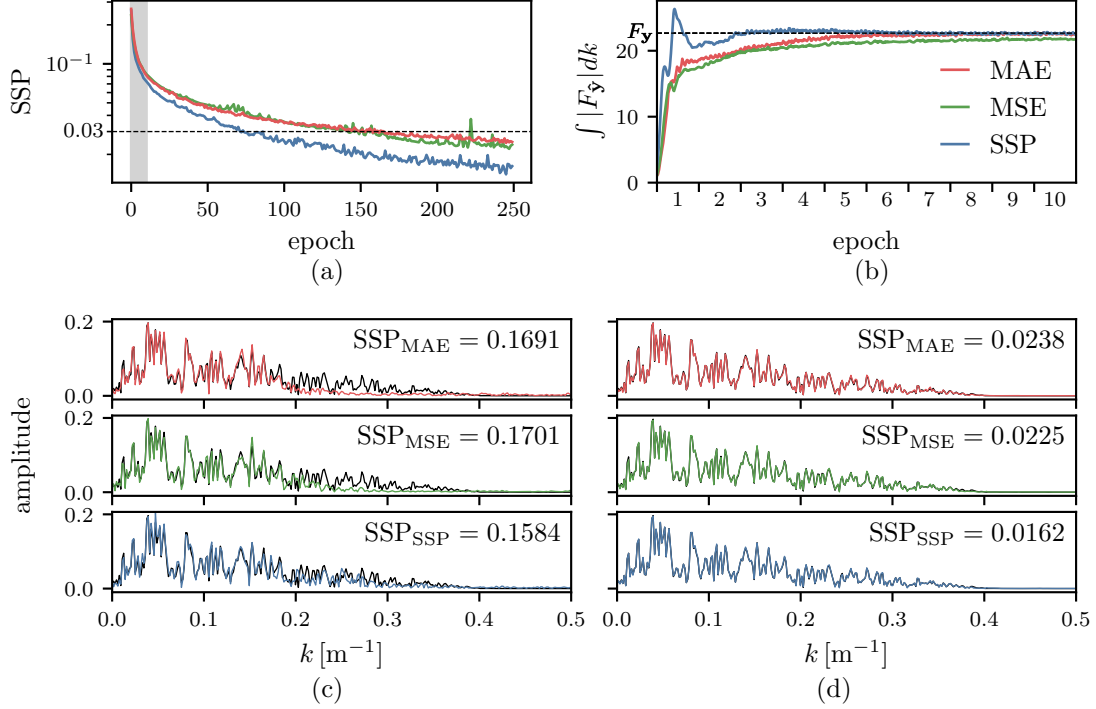
Figure 6: Model training on the 1D dispersive wave data evaluated in terms of SSP metric. The loss curves (a) show the validation loss for models trained with SSP, MSE and MAE loss functions. The gray area marks the first ten epochs, where we obtain predictions for one validation sample after each weight update. (b) shows the integral of the Fourier spectra of the respective prediction after each weight update. The integral of the ground truth is indicated by the dashed line. Panels (c) and (d) show the Fourier spectra of the predictions using the model trained with MAE, MSE and SSP loss functions (from top to bottom) after 2 (c) and 250 (d) training epochs, respectively. The Fourier spectrum of each prediction is evaluated against the ground truth spectrum shown in black

in a steady approach of the minimum. Conceptually different, the gradients of the SSP are very sharp in vicinity of a minimum, hence increasing convergence speed. Predictions far from the optimum enter the plateau in the loss surface that exists in sub-optimal regions of the parameter space. Overall, the SSP is the most unforgiving in cases of small errors in amplitude, phase, or frequency, hence enhancing the convergence behavior of the neural network training. This supremacy becomes especially evident for the broad-banded wave spectrum, while it is already observable for the chaotic KS-eq. case studies. Thus, we conclude that the SSP is a particularly promising alternative to conventional loss functions when training machine learning models on sequential data with complex spectra.

4. Conclusions

This work proposes the surface similarity parameter (SSP) as a novel machine learning loss function that is designed for oscillatory sequences arising across many fields of research related to physics and natural processes. Our experiments cover the chaotic dynamics in the one- and two-dimensional Kuramoto-Sivashinsky equation and dispersive surface gravity waves in fluids.

Our research investigates the effect of the loss function on the training behavior of a machine learning flow map, i.e. a fully convolutional neural network that advances historic state observations one time increment into the future. The experiments indicate that the proposed SSP loss function is superior to classical Euclidean loss functions, such as MSE and MAE, in terms of a) convergence rate, b) robustness against weight initialization effects, and c) final model prediction accuracy. Compared to conventional

Euclidean loss functions, the SSP loss function is a bounded error metric that exhibits large gradients in vicinity of local minima. This regularization property allows to faster train more precise models. While Euclidean distance-based metrics turn out to be amplitude-biased for oscillatory data, the SSP metric is taking amplitude, phase and frequency information explicitly into account, and is therefore much stricter even for small deviations between prediction and ground truth sequences. The superiority of the SSP is especially evident the more complex the spectral composition of the sequential data is, i.e. the wider the spectrum of harmonic components and the more irregular the amplitude distribution becomes. The novel SSP loss function is therefore highly promising for regression on oscillatory sequential data.

Appendix A. Conventional loss functions

Table A.1 lists an overview on all conventional Euclidean loss functions used in the course of this work, accompanied by a short description of the interpretation.

Table A.1: Overview on common regression loss functions for sequence prediction tasks

loss	definition	comment
mean absolute error	$\text{MAE} = \frac{1}{N} \sum_i^N |y_i - \hat{y}_i|$	arithmetic mean of absolute errors, also denoted $L1$ loss. Estimates the median of the error distribution function, therefore robust against outliers. But: constant and large (!) gradients also for small errors, requires adaptive learning rates for better convergence. Gradients are not continuous (not differentiable at 0.)
mean squared error	$\text{MSE} = \frac{1}{N} \sum_i^N (y_i - \hat{y}_i)^2$	arithmetic mean of squared errors, also denoted $L2$ loss. Estimates the mean of the error distribution function, put emphasis on large values. Large gradients at large errors, small gradients at small errors. Helps training neural networks with constant learning rate
root mean squared error	$\text{RMSE} = \sqrt{\frac{1}{N} \sum_i^N (y_i - \hat{y}_i)^2}$	arithmetic mean of squared errors, also denoted $L2$ loss. Same scale as MAE. Estimates the mean of the error distribution function, put emphasis on large values.
Huber loss	$\mathcal{L}_\delta = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{for } |y - \hat{y}| \leq \delta, \\ \delta(|y - \hat{y}| - \frac{1}{2}\delta) & \text{else} \end{cases}$	Combination of MSE (small errors) and MAE (large errors) with small and large gradients, respectively, and differentiable at 0.

Appendix B. On the optimizer step size

The optimizer used for ML model training has the crucial task of finding a local minimum in the loss surface. Purely gradient-based optimization schemes like SGD (Robbins & Monro, 1951; Kiefer & Wolfowitz, 1952) rely on large gradients, as the optimizer's step size is directly linked to the gradient, $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)$ with model parameters $\boldsymbol{\theta}$, learning rate α and iteration count t. Adaptive momentum estimation (Adam) Kingma & Ba (2014) injects 1st and 2nd order momentum, $\hat{\boldsymbol{m}}_t (\beta_1, \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}))$ and $\hat{\boldsymbol{v}}_t (\beta_2, \nabla_{\boldsymbol{\theta}}^2 J(\boldsymbol{\theta}))$ with 1st and 2nd order exponential decay β_1 and β_2, into the weight update scheme, $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} - \varepsilon}$. Thus, the step size in parameter space does not only rely on the current but also on past gradients. Kingma & Ba (2014) state that the step size of Adam has an upper bound "when a gradient has been zero at all time steps except the current one." To evaluate the effect of the loss surface on the step size of the Adam optimizer ($\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 1\text{e}{-8}$), we introduce a one dimensional toy-example using
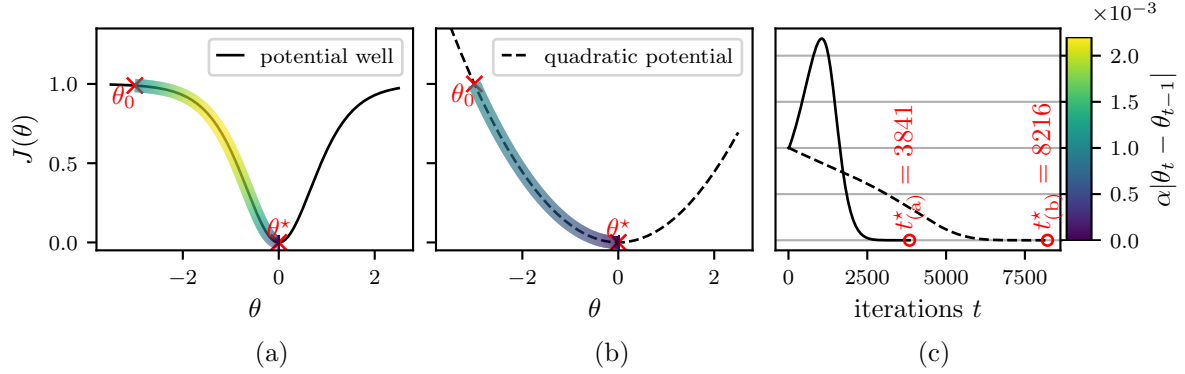
Figure B.1: Convergence speed of Adam optimization scheme on (a) potential well and (b) quadratic potential. Both potential functions have their minimum at $\theta^\star = 0$, the initial weight is $\theta_0 = -3$, $J_{(a)}(\theta_0) = 0.99$, $J_{(b)}(\theta_0) = 1$. (c) shows the optimizer step size $\Delta\theta = \alpha|\theta_t - \theta_{t-1}|$ over the iteration count t. The step sizes are also superimposed onto the potential functions as multicolored lines. The Adam optimization takes $t^\star_{(a)} = 3841$ iterations on the potential well (a) and $t^\star_{(b)} = 8216$ iterations on the quadratic potential (b)

two potentials

$$J_{(a)}(\theta) = 1 - \cosh^{-2}(\theta) \in [0, 1], \qquad J_{(b)}(\theta) = \left(\frac{\theta}{\theta_0}\right)^2 \in \mathbb{R}^+, \quad \theta_0 = -3 \qquad \text{(B.1)}$$

of which $J_{(a)}$ is characterized by a sharp minimum and thus qualitatively resembles the loss surface of SSP, while $J_{(b)}$ qualitatively resembles the bowl-shaped loss surface of the MSE. $J_{(b)}$ is normalized with the initial weight θ_0, such that $J_{(b)}(\theta_0) = 1$. Both potentials have their minimum at $\theta^\star = 0$. We define the initial weight $\theta_0 = -3$, such that

$$\left(J_{(a)}(\theta_0) \approx J_{(b)}(\theta_0)\right) \wedge \left(\left|\nabla_\theta J_{(a)}(\theta_0)\right| \ll \left|\nabla_\theta J_{(b)}(\theta_0)\right|\right). \qquad \text{(B.2)}$$

Each optimizer is applied to the potentials $J_{(a)}$ and $J_{(b)}$ and step sizes $\alpha|\theta_t - \theta_{t-1}|$ over the iteration count t are logged. $t^\star$ resembles the iteration count it takes for the algorithm to converge to $|\theta - \theta^\star| < 1e{-}8$. While the step size for the quadratic potential is steadily descending over t, the step size for the potential well increases to more than double its initial value and then falls off quickly. Figure B.1 (a) indicates that the maximum step size is reached shortly after the well begins, i.e. the initial plateau in $J_{(a)}$ causes the optimizer to ramp up the step size, while the convex shape of $J_{(b)}$ leads to a steadily decreasing step size. Convergence is faster for the potential well than for the quadratic potential, which were chosen to resemble the SSP and MSE loss surfaces, respectively.

We study SGD ($\alpha = 0.01$, no momentum) under identical initial conditions. Since SGD is purely gradient-based, the step sizes can be directly derived from the potential shape. Therefore, the initial step size in $J_{(b)}$ is significantly larger than in $J_{(a)}$. However, as θ converges towards the optimum $\theta^\star$, $\nabla_\theta J_{(b)} \to 0$, leading to vanishing step sizes for the quadratic potential. For $J_{(a)}$, being first concave and then convex, SGD starts with a very small step size which is then greatly increased and reaches its maximum in the inflection point of $J_{(a)}$ (the maximum of $\nabla_\theta J_{(a)}$). SGD converges after 3539 iterations for the potential well, and after 8774 iterations for the quadratic potential.

Appendix C. Data generation

Appendix C.1. Kuramoto-Sivashinsky equation

The Kuramoto-Sivashinsky equation has been studied in several recent works (Smaoui & Al-Enezi, 2004; Pathak et al., 2017, 2018; Raissi & Karniadakis, 2018; Vlachas et al., 2018; Bar-Sinai et al., 2019) related

15

to machine learning as an illustrative example for complex spatio-temporal dynamics. For our case studies, we use the KS-eq.

$$u_t + \Delta u + \nu \Delta^2 u + \frac{1}{2}|\nabla u|^2 \tag{C.1}$$

augmented with a viscosity term $\nu \in \{0.25, 1\}$ that controls the amount of chaos in the system (Smaoui & Al-Enezi, 2004; Vlachas et al., 2018). We initialize both our 1D and 2D KS systems with the superposition of 10 waves

$$u(x) = \sum_{i=1}^{10} A_i \sin(2\pi \frac{i}{L}x + \varphi_i), \quad u(x,y) = \sum_{i=1}^{10} A_i \sin(2\pi \frac{i}{L}\left(x \cos(\phi_i) + y \sin(\phi_i)\right) + \varphi_i) \tag{C.2}$$

with random amplitude $A_i \in [-1, 1]$ and random phase $\varphi_i \in [0, 2\pi]$, $\phi_i \in [-\pi/2, \pi/2]$. For the 1D KS equation, we use a domain x with length $L_x = 128\,\text{m}$ discretized into 1024 grid points. The 2D KS equation uses a square domain with the same set-up as the 1D case in x and y dimension. The definition of our initial conditions Eq. (C.2) ensure that the longest wave fits the domain exactly. All derivatives in Eq. (C.1) are calculated using the 4th order finite differences method. To ensure numeric stability, we apply a low pass filter so that only the first 80 (2D: 60) frequency components are taken into account after each time step. For time integration of the PDE we use the modified Euler method (Süli & Mayers, 2003; Burden & Faires, 2010). The system is simulated using $\Delta t_{\text{sim}} = 0.01\,\text{s}$ of which every 10th system state is saved to disk, resulting in a data time step size of $\Delta t = 0.1\,\text{s}$. The KS-eq. (C.1) is simulated 100 times (2D: 10 times) for 200 s for $\nu = \{0.25, 1.0\}$. The simulation data is preprocessed using a sliding window of size $n_\tau = 8$ along the time dimension with stride 1 and shift 1 for $\Delta t = 0.1\,\text{s}$ (shift 10 for $\Delta t = 1.0\,\text{s}$). For $\Delta t = 0.1\,\text{s}$ we obtain 1993, for $\Delta t = 1.0\,\text{s}$ we end up with 1921 input-output-pairs ($\mathbf{X} \in \mathbb{R}^{1024 \times 8}$, $\mathbf{y} \in \mathbb{R}^{1024}$) for each $\nu \in \{0.25, 1.0\}$ (2D: 1993 input-output pairs ($\mathbf{X} \in \mathbb{R}^{1024 \times 1024 \times 8}$, $\mathbf{y} \in \mathbb{R}^{1024 \times 1024}$) for $\Delta t = 0.1\,\text{s}$, 1921 for $\Delta t = 1.0\,\text{s}$).

Appendix C.2. Dispersive surface waves

The dispersive surface wave data is generated using linear wave theory, i.e. no interactions between waves are modeled. A superposition of $N = 651$ single harmonic waves forms the initial surface wave

$$\eta_0 = \eta(x, t = 0) = \sum_{i=1}^{N} A_i \cos(k_i x + \varphi_i) \tag{C.3}$$

with amplitude A_i, wave number k_i and random phase $\varphi_i \in [0, 2\pi]$. The amplitudes are defined by a normalized Fourier spectrum (Klein et al., 2021)

$$F(\omega_i) = \frac{27\,(\omega_i - \omega_{\text{min}}) \cdot (\omega_i - \omega_{\text{max}})^2}{4\,(\omega_{\text{max}} - \omega_{\text{min}})^3} \tag{C.4}$$

with $\omega_{\text{min}} = 0.3\,\text{rad}\,\text{s}^{-1}$ and $\omega_{\text{max}} = 2.0\,\text{rad}\,\text{s}^{-1}$ limiting the relevant frequency range. The wave number k is linked to the angular frequency ω via the linear dispersion relation

$$\omega(k) = \sqrt{g \cdot k \cdot \tanh(k \cdot d)} \tag{C.5}$$

with gravitational acceleration g and water depth $d = 500\,\text{m}$. In simple terms, the dispersion relation Eq. (C.5) defines the propagation velocity of a wave depending on its wave length. In consequence, long waves travel faster than short waves. To simulate the time evolution of the initial wave Eq. (C.3), the assumption of linear wave theory allows to simulate the time evolution of each single harmonic wave according to their respective dispersion relation and recreate the surface wave at any given time by superposition of all single harmonic waves. Since η_0 carries all information about the space domain (wave numbers $\mathbf{k} \in \mathbb{R}^N$ and initial phases $\boldsymbol{\varphi} \in \mathbb{R}^N$), we can simulate the time evolution of η_0 in the complex space by application of the Fourier transform

$$F_{\eta(t)} = F_{\eta_0} \exp(-i\boldsymbol{\omega}t) \tag{C.6}$$

and obtain the surface wave $\eta(t)$ by means of the inverse Fourier transform of $F_{\eta(t)}$. As this FFT-based simulation approach introduces periodic boundary conditions, the simulated wave exits the domain to the right and re-enters it on the left. To counteract this behavior, we append zeros to η_0 thus expanding the domain to the right. The initial wave now travels into the appended zeros instead of re-entering the domain. Overall, we generate 100 initial sea states with random phase φ_i and otherwise identical configuration. Every sea state is simulated for $250\,\text{s}$ with $\Delta t = 0.1\,\text{s}$. The data preprocessing for ML involves the application of a sliding window with stride 1 and shift 13, increasing the time step in the final data set to $\Delta t = 1.3\,\text{s}$. Using this approach, we obtain 2396 input-output-pairs ($\mathbf{X} \in \mathbb{R}^{1024 \times 8}$, $\mathbf{y} \in \mathbb{R}^{1024}$) from each simulation.

Acknowledgements

References

(2007). Dynamic time warping. In *Information Retrieval for Music and Motion* (pp. 69–84). Springer Berlin Heidelberg. URL: `https://doi.org/10.1007%2F978-3-540-74048-3_4`. doi:10.1007/978-3-540-74048-3_4.

Anderson, J. D. (1995). *Computational fluid dynamics : the basics with applications*. McGraw-Hill series in mechanical engineering. New York u.a.: McGraw-Hill. URL: `http://www.gbv.de/dms/hbz/toc/ht006712187.pdf`.

Bagnall, A., Lines, J., Bostrom, A., Large, J., & Keogh, E. (2016). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, *31*, 606–660. URL: `https://doi.org/10.1007%2Fs10618-016-0483-9`. doi:10.1007/s10618-016-0483-9.

Bar-Sinai, Y., Hoyer, S., Hickey, J., & Brenner, M. P. (2019). Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, *116*, 15344–15349. URL: `https://doi.org/10.1073/pnas.1814058116`. doi:10.1073/pnas.1814058116.

Burden, R. L., & Faires, J. D. (2010). *Numerical Analysis*. Clifton Park, NY: Cengage Learning.

Cooley, J., & Tukey, J. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, *19*, 297–301.

Desmars, N., Hartmann, M., Behrendt, J., Klein, M., & Hoffmann, N. (2021). Reconstruction of Ocean Surfaces From Randomly Distributed Measurements Using a Grid-Based Method. In *Volume 6: Ocean Engineering* (p. V006T06A059). Virtual, Online: American Society of Mechanical Engineers. URL: `https://asmedigitalcollection.asme.org/OMAE/proceedings/OMAE2021/85161/V006T06A059/1121470`. doi:10.1115/OMAE2021-62409.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*, 2121–2159.

van Essen, S. (2021). Influence of Wave Variability on Ship Response During Deterministically Repeated Seakeeping Tests at Forward Speed. In T. Okada, K. Suzuki, & Y. Kawamura (Eds.), *Practical Design of Ships and Other Floating Structures* Lecture Notes in Civil Engineering (pp. 899–925). Singapore: Springer. doi:10.1007/978-981-15-4624-2_54.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, *33*, 917–963. URL: `https://doi.org/10.1007%2Fs10618-019-00619-1`. doi:10.1007/s10618-019-00619-1.

Fulcher, B. D., & Jones, N. S. (2014). Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering*, *26*, 3026–3037. doi:10.1109/TKDE.2014.2316504.

Fulcher, B. D., Little, M. A., & Jones, N. S. (2013). Highly comparative time-series analysis: the empirical structure of time series and their methods. *Journal of The Royal Society Interface*, *10*, 20130048. URL: `https://doi.org/10.1098%2Frsif.2013.0048`. doi:10.1098/rsif.2013.0048.

Giannakis, D., Ourmazd, A., Slawinska, J., & Zhao, Z. (2019). Spatiotemporal pattern extraction by spectral analysis of vector-valued observables. *Journal of Nonlinear Science*, *29*, 2385–2445. URL: `https://doi.org/10.1007%2Fs00332-019-09548-1`. doi:10.1007/s00332-019-09548-1.

Hernandez-Lopez, F. J., Legarda-Sáenz, R., & Brito-Loeza, C. (2021). Parallel algorithm for fringe pattern demodulation. *Journal of Real-Time Image Processing*, *18*, 2441–2451. URL: `https://link.springer.com/10.1007/s11554-021-01129-4`. doi:10.1007/s11554-021-01129-4.

Herzog, S., Wörgötter, F., & Parlitz, U. (2018). Data-driven modeling and prediction of complex spatio-temporal dynamics in excitable media. *Front. Appl. Math. Stat.*, *4*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*, 1735–1780. URL: `https://doi.org/10.1162%2Fneco.1997.9.8.1735`. doi:10.1162/neco.1997.9.8.1735.

Huber, P. J. (1964). Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, *35*, 73–101. URL: `https://doi.org/10.1214%2Faoms%2F1177703732`. doi:10.1214/aoms/1177703732.

Jiang, G., Wang, W., & Zhang, W. (2019). A novel distance measure for time series: Maximum shifting correlation distance. *Pattern Recognition Letters*, *117*, 58–65. URL: `https://doi.org/10.1016%2Fj.patrec.2018.11.013`. doi:10.1016/j.patrec.2018.11.013.

Kiefer, J., & Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, *23*, 462–466. URL: `https://doi.org/10.1214/aoms/1177729392`. doi:10.1214/aoms/1177729392.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. `arXiv:arXiv:1412.6980`.

Klein, M., Dudek, M., Clauss, G. F., Ehlers, S., Behrendt, J., Hoffmann, N., & Onorato, M. (2020). On the deterministic prediction of water waves. *Fluids*, *5*. URL: `https://www.mdpi.com/2311-5521/5/1/9`. doi:10.3390/fluids5010009.

Klein, M., Hartmann, M., & von Bock und Polach, F. (2021). Note on the application of transient wave packets for wave–ice interaction experiments. *Water*, *13*. URL: `https://www.mdpi.com/2073-4441/13/12/1699`. doi:10.3390/w13121699.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436–444. URL: `https://doi.org/10.1038%2Fnature14539`. doi:10.1038/nature14539.

Legaard, C. M., Schranz, T., Schweiger, G., Drgoňa, J., Falay, B., Gomes, C., Iosifidis, A., Abkar, M., & Larsen, P. G. (2021). Constructing neural network-based models for simulating dynamical systems. `arXiv:arXiv:2111.01495`.

Narkhede, M. V., Bartakke, P. P., & Sutaone, M. S. (2021). A review on weight initialization strategies for neural networks. *Artificial Intelligence Review*, *55*, 291–322. URL: `https://doi.org/10.1007%2Fs10462-021-10033-z`. doi:`10.1007/s10462-021-10033-z`.

Pathak, J., Hunt, B., Girvan, M., Lu, Z., & Ott, E. (2018). Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical review letters*, *120*, 024102. doi:`10.1103/PhysRevLett.120.024102`.

Pathak, J., Lu, Z., Hunt, B. R., Girvan, M., & Ott, E. (2017). Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *27*, 121102. URL: `https://doi.org/10.1063/1.5010300`. doi:`10.1063/1.5010300`.

Perlin, M., & Bustamante, M. (2014). A robust quantitative comparison criterion of two signals based on the sobolev norm of their difference. *Journal of Engineering Mathematics*, *101*, 115–124.

Raissi, M., & Karniadakis, G. E. (2018). Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, *357*, 125–141. doi:`10.1016/j.jcp.2017.11.039`.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, *22*, 400–407. URL: `https://doi.org/10.1214/aoms/1177729586`. doi:`10.1214/aoms/1177729586`.

Sharie, M., Mosavi, M. R., & Rahemi, N. (2020). Determination of an appropriate mother wavelet for de-noising of weak GPS correlation signals based on similarity measurements. *Engineering Science and Technology, an International Journal*, *23*, 281–288. URL: `https://www.sciencedirect.com/science/article/pii/S2215098618322018`. doi:`https://doi.org/10.1016/j.jestch.2019.05.006`.

Shi, X., & Yeung, D.-Y. (2018). Machine learning for spatiotemporal sequence forecasting: A survey, . `arXiv:arXiv:1808.06865`.

Singh, A., Thakur, N., & Sharma, A. (2016). A review of supervised machine learning algorithms. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 1310–1315).

Smaoui, N., & Al-Enezi, S. (2004). Modelling the dynamics of nonlinear partial differential equations using neural networks. *Journal of Computational and Applied Mathematics*, *170*, 27–58. URL: `https://doi.org/10.1016/j.cam.2003.12.045`. doi:`10.1016/j.cam.2003.12.045`.

Stender, M., Tiedemann, M., Spieler, D., Schoepflin, D., Hoffmann, N., & Oberst, S. (2021). Deep learning for brake squeal: Brake noise detection, characterization and prediction. *Mechanical Systems and Signal Processing*, *149*, 107181. URL: `https://doi.org/10.1016%2Fj.ymssp.2020.107181`. doi:`10.1016/j.ymssp.2020.107181`.

Süli, E., & Mayers, D. F. (2003). *An Introduction to Numerical Analysis*. Cambridge: Cambridge University Press.

Tieleman, T., & Hinton, G. (2018). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 4, 26–31. URL: `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`.

Vlachas, P., Pathak, J., Hunt, B., Sapsis, T., Girvan, M., Ott, E., & Koumoutsakos, P. (2020). Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks*, *126*, 191–217. URL: `https://doi.org/10.1016/j.neunet.2020.02.016`. doi:`10.1016/j.neunet.2020.02.016`.

Vlachas, P. R., Byeon, W., Wan, Z. Y., Sapsis, T. P., & Koumoutsakos, P. (2018). Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *474*, 20170844. URL: `https://doi.org/10.1098/rspa.2017.0844`. doi:`10.1098/rspa.2017.0844`.

Wriggers, P. (2008). *Nonlinear Finite Element Methods*. Berlin, Heidelberg: Springer Berlin Heidelberg. URL: `https://doi.org/10.1007/978-3-540-71001-1` includes bibliographical references and index.

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. `arXiv:arXiv:1212.5701`.

Zhang, J., Zheng, Y., Qi, D., Li, R., & Yi, X. (2016). Dnn-based prediction model for spatio-temporal data. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* SIGSPACIAL '16. New York, NY, USA: Association for Computing Machinery. URL: `https://doi.org/10.1145/2996913.2997016`. doi:`10.1145/2996913.2997016`.