EIGENVALUE INITIALISATION AND REGULARISATION FOR KOOPMAN AUTOENCODERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Regularising the parameter matrices of neural networks is ubiquitous in training deep models. Typical regularisation approaches suggest initialising weights using small random values, and to penalise weights to promote sparsity. However, these widely used techniques may be less effective in certain scenarios. Here, we study the *Koopman autoencoder* model which includes an encoder, a Koopman operator layer, and a decoder. These models have been designed and dedicated to tackle physics-related problems with interpretable dynamics and an ability to incorporate physics-related constraints. However, the majority of existing work employs standard regularisation practices. In our work, we take a step toward augmenting Koopman autoencoders with initialisation and penalty schemes *tailored* for physics-related settings. Specifically, we propose the "eigeninit" initialisation scheme that samples initial Koopman operators from specific eigenvalue distributions. In addition, we suggest the "eigenloss" penalty scheme that penalises the eigenvalues of the Koopman operator during training. We demonstrate the utility of these schemes on two synthetic data sets: a driven pendulum and flow past a cylinder; and two real-world problems: ocean surface temperatures and cyclone wind fields. We find on these datasets that eigenloss and eigeninit improves the convergence rate by a factor of 2 to 5, and that they reduce the cumulative long-term prediction error by up to a factor of 2.5. Such a finding points to the utility of incorporating similar schemes as an inductive bias in other physics-related deep learning approaches.

1 INTRODUCTION

Modern neural networks are often overparameterised, i.e., their number of learnable parameters is significantly larger than the number of available training samples (Allen-Zhu et al., 2019a;b). To guide optimisation through this immense parameter space, and to potentially improve performance by avoiding overfitting, neural networks are trained with regularisation techniques (Goodfellow et al., 2016). The importance of regularisation has been shown in the theory and practice of deep learning. Prominent examples include the initialisation of parameter matrices (He et al., 2015; Hanin & Rolnick, 2018), and constraining the parameters' norm via loss penalties (Hinton, 1987; Krogh & Hertz, 1991). Initialising weights and penalising them with small random values and weight decay are arguably the most common regularisation techniques employed in training deep models with stochastic gradient descent algorithms. However, specific neural architectures, data domains, and learning problems may require different initialisation and penalty schemes. In this paper, we empirically study the effect of regularisation on physics-aware architectures.

The ground-breaking success of deep learning in solving complex tasks in vision and other domains has inspired the physics community to develop deep models suited to deal with real-world problems arising in the field (Willard et al., 2020; Karniadakis et al., 2021). In this context, we focus on dynamical systems analysed and processed using *Koopman-based* approaches (Takeishi et al., 2017; Lusch et al., 2018). Koopman theory (Koopman, 1931) proves that under certain assumptions, nonlinear and finite-dimensional systems can be transformed to a linear (albeit infinite-dimensional) representation via the Koopman operator. Using finite-dimensional approximations of this Koopman operator is advantageous as they facilitate the analysis and understanding of dynamical systems by utilising linear analysis tools. Despite the theoretical and practical advances that have significantly improved Koopman-based learning methods, the majority of existing models still apply regularisation practices designed for general neural networks. Our investigation aims to answer the research

question: can one exploit properties of Koopman operators to improve regularisation and to promote better overall performance?

The Koopman operator is a linear object with a complex spectrum, i.e., the associated eigenvalues are complex-valued, when eigendecomposition exists. Lusch et al. (2018) model approximate Koopman operators that admit a block diagonal structure, where each block learns a pair of complex-valued conjugate eigenvalues. Similarly, Pan & Duraisamy (2020) propose a skew-symmetric tridiagonal form to guarantee stable Koopman models. However, there has not been a systematic investigation that evaluates the effect of initialisation and penalty regularisation schemes on the behavior of Koopman-based neural networks. The overarching objective of this paper is to help bridge this gap. Key to our regularisation schemes is the observation that spectral properties of linear Koopman operators follow a *typical structure*, shared by many dynamical systems. Namely, Koopman eigenvalues of stable dynamical systems are constrained in the unit circle of the complex plane (Mauroy & Mezić, 2016).

Motivated by the theoretical observation that eigenvalues need to be within the unit circle, we propose two novel regularisation techniques: random eigenvalue generation from known distributions to initialise key weights in the network ("eigeninit"), and a loss penalty on the eigenvalues to limit their expansion ("eigenloss"). Importantly, our regularisation schemes can be incorporated with all Koopman-based approaches as a means to regularise the associated Koopman operator. We evaluate our approach on several challenging physics-related datasets, and in comparison to several state-of-the-art baselines. Our results indicate that our spectral schemes for initialisation and penalty improve the performance of Koopman-based networks, leading to faster and smoother convergence in the objective loss, as well as yielding models that generalise better in long-term prediction tests. We hope that our study and results will help to widen the scope of problems for which Koopman models are used.

2 Related Work

Regularisation of neural networks is a fundamental research topic in machine learning. Common regularisation techniques include dropout (Srivastava et al., 2014), batch normalisation (Ioffe & Szegedy, 2015), and data augmentation (Perez & Wang, 2017). In what follows, we mainly discuss parameter initialisation approaches and weight penalty methods.

Parameter initialisation. Proper initialisation of deep models is known to be crucial to their successful training (Sutskever et al., 2013). Standard initialisation schemes for $\tanh(\cdot)$ (Glorot & Bengio, 2010) and ReLU (He et al., 2015) activations sample small random values for the weight matrices of the network. These choices are backed by theoretical results showing that initial small scale weights generalise better (Woodworth et al., 2020), and random initialisation converges to local minimisers on differentiable losses (Lee et al., 2016). While guaranteed, convergence may be exponentially slow (Du et al., 2017). Other approaches promote information flow using initial orthogonal weight matrices (Saxe et al., 2014; Mishkin & Matas, 2016; Pennington et al., 2017). Koopman-based approaches (Pan & Duraisamy, 2020) suggest to initialise the Koopman operator using the dynamic mode decomposition (DMD) estimates (Schmid, 2010). Still, the general problem of weight initialisation in neural networks remains an active research topic in practice (Arpit et al., 2019), and theory (Hanin & Rolnick, 2018; Stöger & Soltanolkotabi, 2021).

Parameter loss penalties. Penalising the norms of weight matrices using L_1 and L_2 metrics is a common practice in machine learning, collectively termed weight decay (Hinton, 1987). More recently, Arjovsky et al. (2016) showed that recurrent neural networks can avoid the issue of exploding gradients if their hidden-to-hidden matrices are parameterised to be unitary. Similarly, Yoshida & Miyato (2017) introduce a regularisation scheme based on penalising the spectral norm of weight matrices. Greydanus et al. (2019) assume the underlying system is measure-preserving, and their network learns the Hamiltonian during training. Lusch et al. (2018) use block diagonal Koopman operators to support continuous spectra. To promote stability, Erichson et al. (2019) employ Lyapunov-based constraints, whereas Pan & Duraisamy (2020) guarantees that Koopman eigenvalues remain in the unit circle via tridiagonal Koopman operators. In contrast, a soft penalty on the forward and backward dynamics was introduced by Azencot et al. (2020), yielding stable Koopman systems and state of the art performance.

3 BACKGROUND

3.1 KOOPMAN OPERATOR THEORY

Suppose we have a discrete time dynamical system given by:

$$x_{k+1} = \varphi(x_k) \ , \ k \in \mathbb{N}_0 \ . \tag{1}$$

Let $f : S \to \mathbb{R}$ (where S is the system's state space) be a real-valued observable of the system. By real-valued observable we mean a function of the state of the dynamical system, where an observed data-variable is its range. The collection of all such observables forms a linear vector space. The Koopman operator K is a linear transformation on this functional space:

$$Kf(x) = f \circ \varphi(x) \ . \tag{2}$$

Essentially, the Koopman operator may be viewed as a *lifting* of the dynamics from the state space to the space of observables (Mezić, 2015), as shown in Fig. 1. Whilst the Koopman operator maps between function spaces and is thus infinite dimensional, it has been shown empirically that finite-dimensional approximations (U) are generally quite expressive, where U is usually found through dynamic mode decomposition or deep learning (Mauroy et al., 2020).

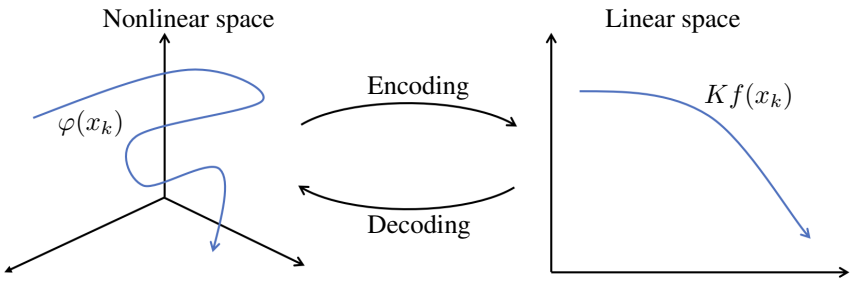


Figure 1: Illustration of the data transformation that maps the high-dimensional states x_k which evolve on a nonlinear trajectory via φ to a new space where the dynamics are linear and given by K.

3.2 KOOPMAN AUTOENCODERS

Pioneered by several groups (Takeishi et al., 2017; Lusch et al., 2018), the Koopman autoencoder (KAE) is a deep neural network designed to solve physics-related problems. Primarily, KAEs are based on an autoencoder architecture (Hinton & Zemel, 1993) which has a "bottleneck" structure including an encoder ψ which learns a low-dimensional representation of the input signal, and a decoder ω which recovers the original signal from the low-dimensional code. Koopman autoencoders extend these family of autoencoder models by introducing a Koopman operator module in-between ψ and ω whose purpose is to produce a finite-dimensional approximation of the Koopman operator. The Koopman module U is essentially a linear layer (with no bias) aiming at advancing latent codes forward in time. Put together, the KAE architecture can be described by the following equations,

$$y_k := \psi(x_k) \ , \tilde{x}_k := \omega(y_k) \ , \hat{y}_{k+1} := U y_k \ , \hat{x}_{k+1} := \omega(\hat{y}_{k+1}) \ , \tag{3}$$

where x_k is the input signal, $\tilde{x}_k$ is its reconstruction, and $\hat{x}_{k+1}$ is the reconstruction of the latent code obtained with the Koopman matrix U. The model is trained with a reconstruction loss $\text{MSE}(x_k, \tilde{x}_k)$, and a prediction loss $\text{MSE}(x_k, \hat{x}_k)$, where MSE is the mean squared error. Several existing methods adopt this general framework which we illustrate in Fig. 2, and they further generalize it to promote stability (Pan & Duraisamy, 2020; Azencot et al., 2020), to incorporate control (Han et al., 2021), and to other settings (Morton et al., 2018; Li et al., 2020; Yeung et al., 2019; Otto & Rowley, 2019; Iwata & Kawahara, 2020).

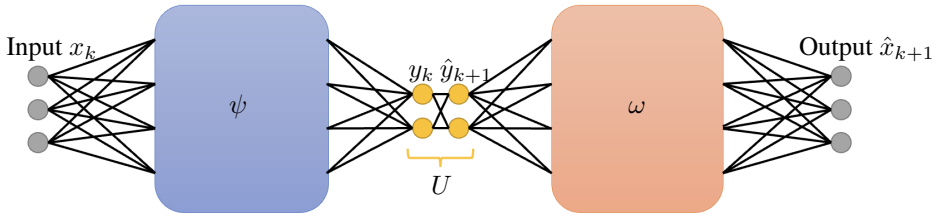


Figure 2: Diagram of Koopman autoencoder architecture.

3.3 Spectral Analysis of Koopman Operators

We briefly mentioned above that one of the main advantages to working with Koopman-based approaches, and KAE in particular, is the linearity of the Koopman matrix U. Specifically, we can harness tools from linear analysis and spectral approaches to study the underlying dynamics (Strogatz, 2018). For instance, the eigendecomposition of U to eigenvectors and eigenvalues is instrumental to a qualitative characterisation of the behavior of the system. We call (v_j, λ_j) an eigenvector-value pair of the Koopman matrix if they satisfy: $Uv_j = \lambda_j v_j$ where $\lambda_j \in \mathbb{C}$. The effect of the eigenvalues on the behavior of the system can be realized by considering the long-term evolution of inputs. Denote by V the eigenvectors in columns, and Λ the diagonal matrix with the eigenvalues along its main diagonal, then we can advance the latent code y_k by

$$\hat{y}_{k+l} = U^l y_k = V \Lambda^l V^{-1} y_k \ , \tag{4}$$

Thus, advancing y_k forward in time from time k to time $k + l$ amounts to simply multiplying y_k with powers of U. Eq. 4 also shows that every eigenvector v_j is scaled by its associated λ_j^l, and therefore, the modulus of $|\lambda_j|$ determines its long-term behavior. We identify three qualitatively different evolution profiles, determined by the modulus of eigenvalues: if $|\lambda_j| < 1$ then its associated eigenvector diminishes with time (i.e. $l \to \infty$), if $|\lambda_j| > 1$, then v_j explodes when $l \to \infty$, and if $|\lambda_j| = 1$ then the norm of v_j does not change, and thus it is stable for any l. In this way, the eigenvalues of the Koopman matrix encode the "memory" of an evolution rule. Finally, we recall a classical result relating the powers of a matrix and its long-term behavior (Stewart, 2001).

Theorem 1 *Let A be a square matrix. Then*

$$\lim_{l \to \infty} A^l = 0 \iff \rho(A) < 1 \ ,$$

where $\rho(A)$ is the spectral radius of A (the absolute value of the largest eigenvalue). On the other hand, if $\rho(A) > 1$, $\lim_{l \to \infty} |A^l| = \infty$, where $|\cdot|$ is any matrix norm.

Our regularisation schemes described in Sec. 4 are based on the above theorem and identification of dynamical modes—vanishing, exploding and stable.

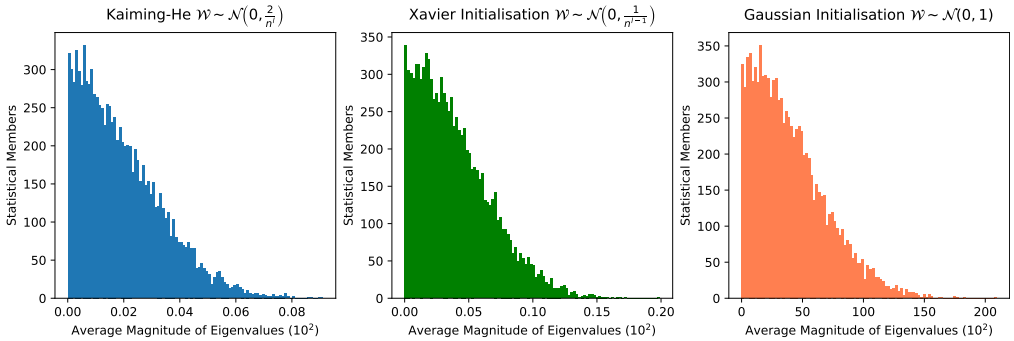


Figure 3: Eigenvalue distribution of various initialisation schemes for a 4×4 operator with six layers.

4 METHODS

Our discussion in Sec. 3 suggests that repeated applications of the approximated Koopman operator U will yield a system that converges to zero if the spectral radius of U is less than one. Similarly, long-term evolution of an initial input x_0 will diverge if $\rho(U) > 1$. This analysis motivates us to design initialisation and penalty schemes for which the spectral radius of U is $\rho(U) \leq 1$.

4.1 EIGENVALUE INITIALISATION OF LEARNABLE PARAMETERS

Initialising a deep neural network of any architecture is critical to its training and model performance. In what follows, we empirically evaluate the eigenvalue distribution of parameter matrices as attained by three of the most common initialisation schemes: Xavier (Glorot & Bengio, 2010), He (He et al., 2015), and a random Gaussian initialisation. All three sample from normal distributions with zero mean, a specified variance in the Gaussian case and a scaled variance in the other two. We find that the average magnitude of eigenvalues are on the order of 10^{-4} for Xavier and He, whilst they are on the order of 10^{-1} for a Gaussian initialisation (Fig. 3). Overall, these initialisations produce Koopman matrices whose eigenvalues span a distribution with an extremely low variance, and thus, such initialisations may negatively affect training in cases where the optimal Koopman matrix consists of eigenvalues with modulus $|\lambda_j| \approx 1$.

Using initial small random values for the parameter weights of deep neural networks is motivated by theoretical results which show that such initialisations converge to local minima on differentiable loss functions when solved with stochastic gradient descent (Lee et al., 2016). However, convergence may be extremely slow (Du et al., 2017). Can we improve convergence speed and model generalisation by developing initialisation schemes tailored to Koopman autoencoder architectures? Typical Koopman-based approaches utilise one of the three initialisation schemes above (see e.g., Azencot et al. (2020)); however, these choices are agnostic to the structure of the Koopman operator layer. Other techniques employ a DMD-based initialisation (see for instance, Pan & Duraisamy (2020)), which may be an unrealistic choice for highly-nonlinear and high-dimensional dynamical systems (Zhang et al., 2017). Inspired by the discussion in Sec. 3, we would like to exploit the unique role of the approximate Koopman operator U as an evolution matrix and its special structure.

Therefore, we choose to focus on eigenvalue-based initialisation rather than element-wise initialisation whose effect on the eigenvalues is indirect, and in practice leads to eigenvalues with low magnitude distribution. Namely, we propose initialisation schemes that directly control the eigenvalues of the Koopman matrix U and their distribution. To this end, we propose the following eigenvalue initialisation method (termed "eigeninit"): we start with an element-wise sampling from a Gaussian distribution to produce U_0. Then, we compute the eigendecomposition of $U_0 = V \Lambda V^{-1}$, where V is a matrix with the eigenvectors v_j organized in its columns, and Λ is a diagonal matrix with the eigenvalues λ_j along its main diagonal. We modify Λ by sampling from a pre-defined distribution D. In particular each eigenvalue λ_j, is replaced with $\tilde{\lambda}_j \sim D$. The initial Koopman matrix U is defined via $U := \text{Re}(V \tilde{\Lambda} V^{-1})$, where $\tilde{\Lambda}$ is a diagonal matrix with $\tilde{\lambda_j}$ along its main diagonal, and $\text{Re}(A)$ yields the real value component of each matrix element A. All other weights in the network are initialised according to the He initialisation as we utilise ReLU activation layers (He et al., 2015).

Taking into consideration the above discussion, a natural question arises: what distributions D should we sample from? Thm. 1 suggests one should focus on parameter matrices U initialised such that their spectral radius satisfies $\rho(U) \leq 1$. On the other hand, existing works on recurrent neural networks (Arjovsky et al., 2016) advocate the use of orthogonal weights for which $\rho(U) = 1$ as all eigenvalues are positioned on the unit circle. In what follows, we prefer the former option, i.e., U matrices with $\rho(U) \leq 1$ since orthogonal and unitary matrices are less expressive (Kerg et al., 2019). In particular, $\rho(U) \leq 1$ allows U to immediately capture any dissipative dynamics in a system. In practice, we sample the eigenvalues of U from one of the following distributions:

- $D(\text{gaussianEigen}) := \mathcal{N}(0, \sigma)$,
- $D(\text{doubleGaussianEigen}) := [\mathcal{N}(-1, \sigma) + \mathcal{N}(1, \sigma)] + i\,[\mathcal{N}(-1, \sigma) + \mathcal{N}(1, \sigma)]$,
- $D(\text{uniformEigen}) := \mathcal{U}(-\sigma, \sigma) + i\,\mathcal{U}(-\sigma, \sigma)$,
- $D(\text{unitPerturb}) := \mathcal{U}(-\epsilon + 1, \epsilon + 1) e^{i\,\mathcal{U}(0, 2\pi)}$,

where $\mathcal{N}(\mu, \sigma)$ is the normal distribution with mean μ and standard deviation σ, and $\mathcal{U}(a, b)$ is the uniform distribution on the segment $[a, b]$.

4.2 Eigenvalue Regularisation of Learnable Parameters

While proper initialisation of weights is crucial for a neural network to start optimisation from a successful trajectory that generalises well (Stöger & Soltanolkotabi, 2021), we note that without an additional mechanism that penalises non-preferable solutions, optimisation may become stuck on inferior local minima. In our setting, this means that the resulting Koopman matrix may settle on a spectrum for which many eigenvalues tend to zero, or worse, some eigenvalues may land outside the unit circle, and thus be associated with diverging eigenvectors. Thus, we are driven to penalise the eigenvalues of U to steer away from problematic distributions. Any penalty technique and moreover any regularisation method largely aims to reduce the variance in the model at the cost of increased bias. That is, we wish to sacrifice a certain level of performance on the training set in order to achieve better generalisability across different distributions of data.

Similarly to our initialisation scheme which directly affects the eigenvalues of U (Sec. 4.1), we propose a penalty method (termed "eigenloss") which constrains the spectrum of the approximate Koopman operator: we augment the reconstruction and prediction loss components of the Koopman autoencoder (Sec. 3) with a new and novel loss term ϵ_λ whose role is to penalise eigenvalues of U which are too distant from a certain value or distribution. While we could have used the same ansatz as in Sec. 4.1, promoting the spectral radius of U to satisfy $\rho(U) \leq 1$, we find that in practice, encouraging the eigenvalues to be close or on the unit circle yields better models in terms of convergence and generalisability features. Thus, we consider two penalisation schemes:

- unit circle mean absolute error: $\epsilon_\lambda(\text{MAE}) := \sum_j ||\lambda_j| - 1|_1^2$,
- unit circle mean squared error: $\epsilon_\lambda(\text{MSE}) := \sum_j ||\lambda_j| - 1|_2^2$,

where λ_j is an eigenvalue of U. That is, we require the modulus of the eigenvalues to be approximately one, under the metrics MAE or MSE. Importantly, backpropagation through the eigenvalues of a matrix is numerically stable if there are no repeating eigenvalues.

The choice of eigenvalue penalty imposes different soft biases on the Koopman approximation U. Based on the domain-knowledge of the relevant dynamical system, one can tune the above penalties better. For instance, if the underlying system is measure-preserving, we can weight ϵ_λ more strongly with respect to the reconstruction and prediction losses. In contrast, if the dynamics are dissipative, we can use a weaker weight.

5 Experiments

The empirical results for eigeninit and eigenloss are demonstrated in various experiments. Both were implemented in Pytorch (Paszke et al., 2019).

5.1 Datasets

Datasets were chosen as a combination of synthetic and real-world data, varying in complexity and including varying amounts of dissipation and measure preservation. Our initial dataset was a driven frictionless pendulum governed by:

$$\frac{\mathrm{d}^2 x}{\mathrm{d}t^2} + \omega_0^2 x = f_0 \sin(\omega t), \tag{5}$$

where ω_0 the characteristic frequency of the pendulum, and f_0 and ω the amplitude and frequency of the forcing. Here, we use $\omega_0 = 3.13$, $f_0 = 1$, and $\omega = 1$. To generate a dataset we produced solutions of Eq. 5 with a variety of initial conditions ($x \in [-\pi, \pi]$ and $\mathrm{d}x/\mathrm{d}t \in [-1, 1]$).

We also applied our techniques to cyclone wind fields, flow over a cylinder and sea surface temperatures. The cyclone prediction dataset was extracted from the ERA5 Re-Analysis dataset provided by ECMWF using the International Best-Tracks Archive for Climate Stewardship (IBTrACS). Prediction

Table 1: Minimum validation loss and final cumulative test prediction error for benchmark data sets. See bold for minimum.

			Validation Loss			
Dataset	None	Consistency	Init	Loss	Both	Consistency & Best
Pendulum	98.8	77.1	**80.8**	81.1	83.8	79.0
Fluid	0.33	**0.18**	0.23	0.20	0.22	0.33
Ocean ($\times$ 10)	0.28	0.28	0.21	0.29	**0.18**	0.29
Cyclone	17.7	15.7	17.0	16.0	16.2	**15.6**
			Cumulative Test Error			
Pendulum	27.0	16.6	**14.7**	27.0	17.6	18.6
Fluid	1.82	0.95	0.91	**0.81**	1.17	1.11
Ocean	3.16	2.77	1.51	2.80	**1.23**	1.41
Cyclone	186	201	168	**148**	178	174

sequences were sub-sampled from the u component of wind at the pressure level of $650\,\text{hPa}$ in a $20° \times 20°$ sliding window around the cyclone. The sea surface temperature benchmark was taken from Azencot et al. (2020). It is a subset of the NOAA OI SST V2 High Resolution Dataset over the gulf of Mexico with a spatial resolution of 100×100 spanning a time horizon of 1305 days. Finally, the fluids dataset modeled flow over a cylinder and was produced by Kutz et al. (2016). We took the u-component of the flow vector field.

5.2 Test and Validation Prediction Error with Datasets

One can see the application of different schemes (including their combination) and the influence of these on the prediction test loss across given prediction horizons and validation loss in Fig. 4. To choose the library members employed in prediction, we completed 5 training runs with all candidates on all data sets. The results of this testing can be seen in sections A and B of the appendix. Schemes with the best validation loss were used to generate Fig. 4. In addition, the state of the art consistency architecture presented by Azencot et al. (2020) is shown as an additional baseline and an architecture to which our schemes are applied. The particular schemes applied to the consistency architecture were chosen by a two-stage selection process: first employing the best dataset eigenloss or eigeninit and then taking the lead performer out of these. For a summary of results see Tab. 1. Of particular note is that all of our schemes outperform the standard KAE and do so by up to a factor of 2.5 in cumulative test error. Additionally, our initialisation schemes increased convergence rates by factors of 2 to 5 (see Appendix D for a summary). Regarding the consistency benchmark, our best candidate in each dataset outperforms it. In addition, for the two real-world datasets, augmentation of the consistent KAE with eigenloss or eigeninit improves its performance and rate of convergence, suggesting the potential for enhancing previous SOTA architectures with our techniques.

5.3 Paired t-tests

To show that the reduced loss driven by eigeninit and eigenloss was general across datasets, we used paired t-tests between methods on different datasets. These t-tests compared the significance of the difference in (1) validation loss over averaged training runs; (2) cumulative test error over a certain time-horizon; and (3) final validation loss at the end of training, between the selected method and the control. Each row in Tab. 2 shows the p-value for the difference between the listed method and the control (Gaussian elementwise initialisation with no eigenloss). For the difference in final validation loss, we use the adjusted t-test proposed by Nadeau & Bengio (1999), which corrects the variance estimate using the size of the training and validation sets. The p-values from these paired t-tests comparing final validation losses (at epoch 50, for 20 different seeds of models) to the control model are also shown in Tab. 2. All models with both eigeninit and eigenloss have significantly lower final validation loss than the control.

5.4 Investigating Eigenvalue Evolution

Although we have demonstrated the utility of our initialisation and regularisation schemes, we have yet to investigate the evolution of eigenvalues within the networks themselves. We can plot a heatmap

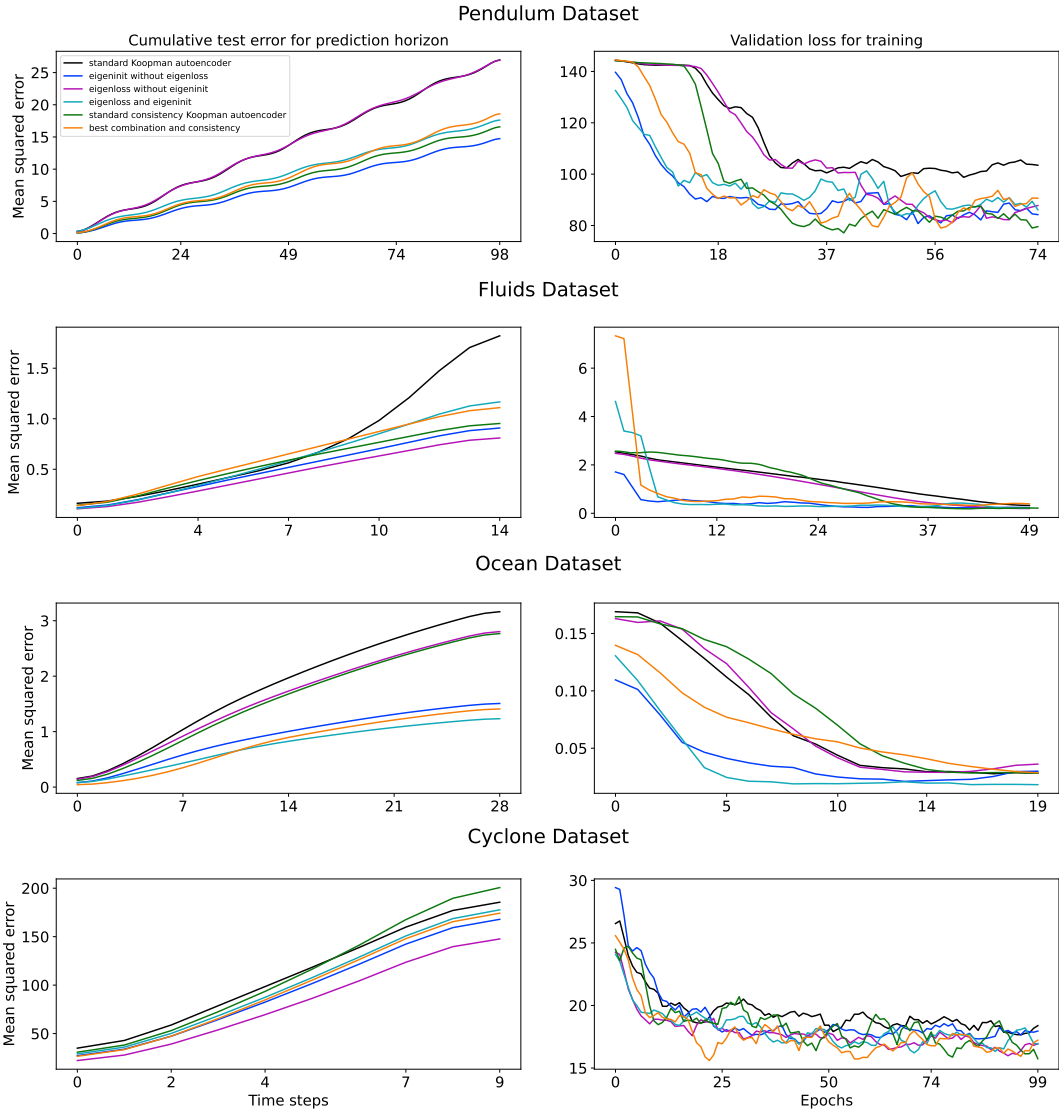


Figure 4: Averaged prediction horizon test loss and training validation loss for selected library members of eigeninit and eigenloss. Models are trained on the longest prediction horizon shown in the test loss graphs.

Table 2: Paired t-tests comparing our methods to control (Gaussian elementwise initialisation with no eigenloss). Each entry is the p-value for the difference between specified model and the control for the selected metric. Significant results ($\alpha = 0.05$) are shown in bold.

	Val. loss (all epochs)			**Test error**			**Final val. loss**		
Dataset	Init	Loss	Both	Init	Loss	Both	Init	Loss	Both
Ocean	**0.014**	0.617	**0.009**	**0.0**	**0.001**	**0.0**	**0.022**	0.302	**0.017**
Fluid	**0.0**	**0.0**	0.074	**0.041**	**0.028**	**0.042**	**0.0**	0.168	**0.0**
Cyclone	0.401	0.083	0.126	**0.0**	**0.0**	0.077	0.1	**0.02**	**0.0**
Pendulum	**0.0**	**0.049**	**0.002**	**0.0**	**0.486**	**0.0**	**0.04**	**0.001**	**0.001**

of eigenvalue magnitudes of the Koopman operator during training (Fig. 5). By doing so, we see that the control network has increasing eigenvalues even though there is no bias towards this and that using the unit circle MSE penalty promotes this tendency. Alternatively, with initialisation close to the unit circle, the eigenvalues tend towards the different eigenvalue classes mentioned in Sec. 3.3.

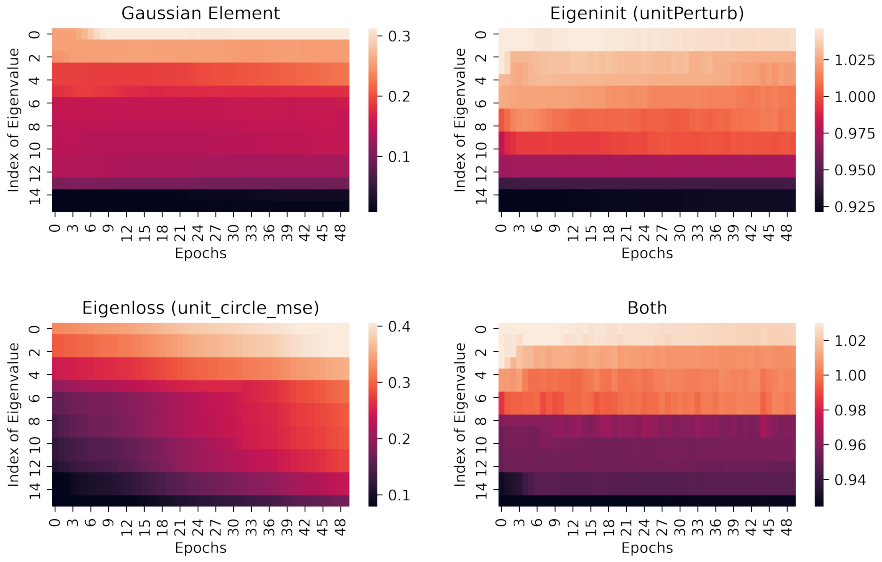


Figure 5: Change in eigenvalue magnitudes over time.

Interestingly, there is an evident asymmetric distribution of eigenvalues around one in the *Eigeninit* and *Both* plots, showing how the Koopman only requires a handful of eigenvalues with magnitude greater than one to account for noise. Other eigenvalues lie on or within the unit circle, representing conserved or dissipative aspects of the system.

6 Discussion

In this research we have asked whether an initial and ongoing bias on the eigenvalues of Koopman autoencoders would improve their accuracy and convergence. We tested this via empirical analysis on the field's existing benchmarks and through the introduction of a new one (cyclone wind fields). It seems clear that certain well developed biases do in fact have great utility for training these networks. In particular, our schemes outperform state of the art baseline models which do not use them. In addition, they can augment one such baseline introduced by Azencot et al. (2020), and improve its performance on real-world datasets.

These techniques are important because they marry together the two purposes of dynamical systems modelling: prediction and explanation. Whilst deep autoencoders have frequently been used for predicting complex dynamical systems, they are often criticised because they lack explainability. Koopman autoencoders are a front-running scheme to address this. However, inserting a Koopman approximation between the encoder and decoder introduces a bottleneck in the network and reduces its capacity for learning.

Because the linear dynamics are so useful in understanding the system, we want to be able to keep this interpretability whilst providing necessary forecasting capacity. The results above demonstrate that this is possible by exploiting the spectral properties of the Koopman approximation U. Specifically, we have shown that Koopman autoencoders using our eigeninit and eigenloss techniques *(i)* converge faster; *(ii)* reach a lower validation loss level; *(iii)* achieve better results across multiple prediction horizons than state of the art baseline networks; and *(iv)* can productively augment other inductive biases on the Koopman approximate.

These techniques will be of practical help for those using Koopman methods in dynamical systems. It is likely that the schemes presented here can be extended in many ways, particularly by devising better initialisation and regularisation candidates and tuning them to the nature of the problem. One such extension is presented in Appendix C, namely an algorithm for hyperparameter search.

REPRODUCIBILITY STATEMENT

In order to ensure reproducible results, we have provided an anonymised source code in *Supplemental materials* with a synthetic pendulum dataset, along with the scripts used for data processing and pipelining in this case. The Koopman autoencoder with all combinations of our techniques is included as a model, as well as other previous SOTA baselines for comparison. Further, the training script used is included. Using these models and data, training runs were performed several times and averaged. Additionally, other techniques (such as paired t-tests) were undertaken in Sec. 5, proving the significance of improvements over a large number of training runs. In particular, this showed significant improvements in several metrics, namely validation loss over entire training runs, cumulative prediction error on the test set, and final validation loss. Other datasets are available upon request from the referenced author. The full source code will be released upon conclusion of Open Review.

REFERENCES

Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. *Advances in neural information processing systems*, 32, 2019a.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pp. 242–252. PMLR, 2019b.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International conference on machine learning*, pp. 1120–1128. PMLR, 2016.

Devansh Arpit, Víctor Campos, and Yoshua Bengio. How to initialize your network? robust initialization for weightnorm & resnets. *Advances in Neural Information Processing Systems*, 32, 2019.

Omri Azencot, N Benjamin Erichson, Vanessa Lin, and Michael Mahoney. Forecasting sequential data using consistent Koopman autoencoders. In *International Conference on Machine Learning*, pp. 475–485. PMLR, 2020.

Simon S Du, Chi Jin, Jason D Lee, Michael I Jordan, Aarti Singh, and Barnabas Poczos. Gradient descent can take exponential time to escape saddle points. *Advances in neural information processing systems*, 30, 2017.

N Benjamin Erichson, Michael Muehlebach, and Michael W Mahoney. Physics-informed autoencoders for lyapunov-stable fluid flow prediction. *arXiv preprint arXiv:1905.10866*, 2019.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Minghao Han, Jacob Euler-Rolle, and Robert K Katzschmann. DeSKO: Stability-assured robust control with a deep stochastic Koopman operator. In *International Conference on Learning Representations*, 2021.

Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture. *Advances in Neural Information Processing Systems*, 31, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Geoffrey E Hinton. Learning translation invariant recognition in a massively parallel networks. In *International Conference on Parallel Architectures and Languages Europe*, pp. 1–13. Springer, 1987.

Geoffrey E Hinton and Richard Zemel. Autoencoders, minimum description length and Helmholtz free energy. *Advances in neural information processing systems*, 6, 1993.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Tomoharu Iwata and Yoshinobu Kawahara. Neural dynamic mode decomposition for end-to-end modeling of nonlinear dynamics. *arXiv preprint arXiv:2012.06191*, 2020.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Giancarlo Kerg, Kyle Goyette, Maximilian Puelma Touzel, Gauthier Gidel, Eugene Vorontsov, Yoshua Bengio, and Guillaume Lajoie. Non-normal recurrent neural network (nnrnn): learning long time dependencies while improving expressivity with transient dynamics. *Advances in neural information processing systems*, 32, 2019.

Bernard O Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

Anders Krogh and John Hertz. A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4, 1991.

J. Nathan Kutz, Steven L. Brunton, Bingni W. Brunton, and Joshua L. Proctor. *Dynamic Mode Decomposition: Data-driven Modeling of Complex Systems*. SIAM-Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2016. ISBN 1611974496.

Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *Conference on learning theory*, pp. 1246–1257. PMLR, 2016.

Yunzhu Li, Hao He, Jiajun Wu, Dina Katabi, and Antonio Torralba. Learning compositional Koopman operators for model-based control. In *International Conference on Learning Representations*, 2020.

Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):1–10, 2018.

Alexandre Mauroy and Igor Mezić. Global stability analysis using the eigenfunctions of the Koopman operator. *IEEE Transactions on Automatic Control*, 61(11):3356–3369, 2016.

Alexandre Mauroy, Yoshihiko Susuki, and Igor Mezić. Introduction to the Koopman operator in dynamical systems and control theory. In *The Koopman Operator in Systems and Control*, pp. 3–33. Springer, 2020.

Igor Mezić. On applications of the spectral theory of the Koopman operator in dynamical systems and control theory. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 7034–7041. IEEE, 2015.

Dmytro Mishkin and Jiri Matas. All you need is a good init. In *International Conference on Learning Representations*, 2016.

Jeremy Morton, Antony Jameson, Mykel J Kochenderfer, and Freddie Witherden. Deep dynamical modeling and control of unsteady fluid flows. *Advances in Neural Information Processing Systems*, 31, 2018.

Claude Nadeau and Yoshua Bengio. Inference for the generalization error. *Advances in neural information processing systems*, 12, 1999.

Samuel E Otto and Clarence W Rowley. Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1):558–593, 2019.

Shaowu Pan and Karthik Duraisamy. Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability. *SIAM Journal on Applied Dynamical Systems*, 19 (1):480–509, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. *Advances in neural information processing systems*, 30, 2017.

Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations*, 2014.

Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pp. 464–472. IEEE, 2017.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Gilbert W Stewart. *Matrix Algorithms: Volume II: Eigensystems*. SIAM, 2001.

Dominik Stöger and Mahdi Soltanolkotabi. Small random initialization is akin to spectral learning: Optimization and generalization guarantees for overparameterized low-rank matrix reconstruction. *Advances in Neural Information Processing Systems*, 34:23831–23843, 2021.

Steven H Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning koopman invariant subspaces for dynamic mode decomposition. *Advances in Neural Information Processing Systems*, 30, 2017.

Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919*, 1(1):1–34, 2020.

Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In *Conference on Learning Theory*, pp. 3635–3673. PMLR, 2020.

Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for Koopman operators of nonlinear dynamical systems. In *2019 American Control Conference (ACC)*, pp. 4832–4839. IEEE, 2019.

Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

Hao Zhang, Scott Dawson, Clarence W Rowley, Eric A Deem, and Louis N Cattafesta. Evaluating the accuracy of the dynamic mode decomposition. *arXiv preprint arXiv:1710.00745*, 2017.

A VALIDATION LOSS FOR VARYING SCHEMES ON NON-PENDULUM DATA SETS

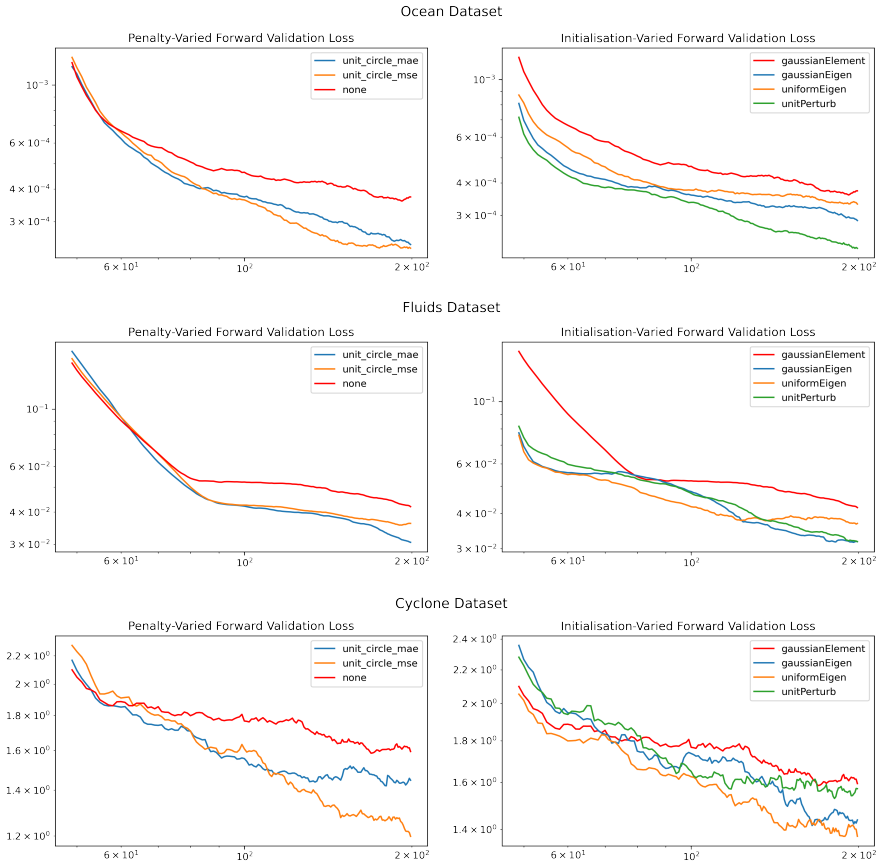


Figure 6: Average validation losses for data sets excluding the pendulum with different penalty and initialisation schemes

B VALIDATION LOSS FOR VARYING SCHEMES ON PENDULUM DATA SETS

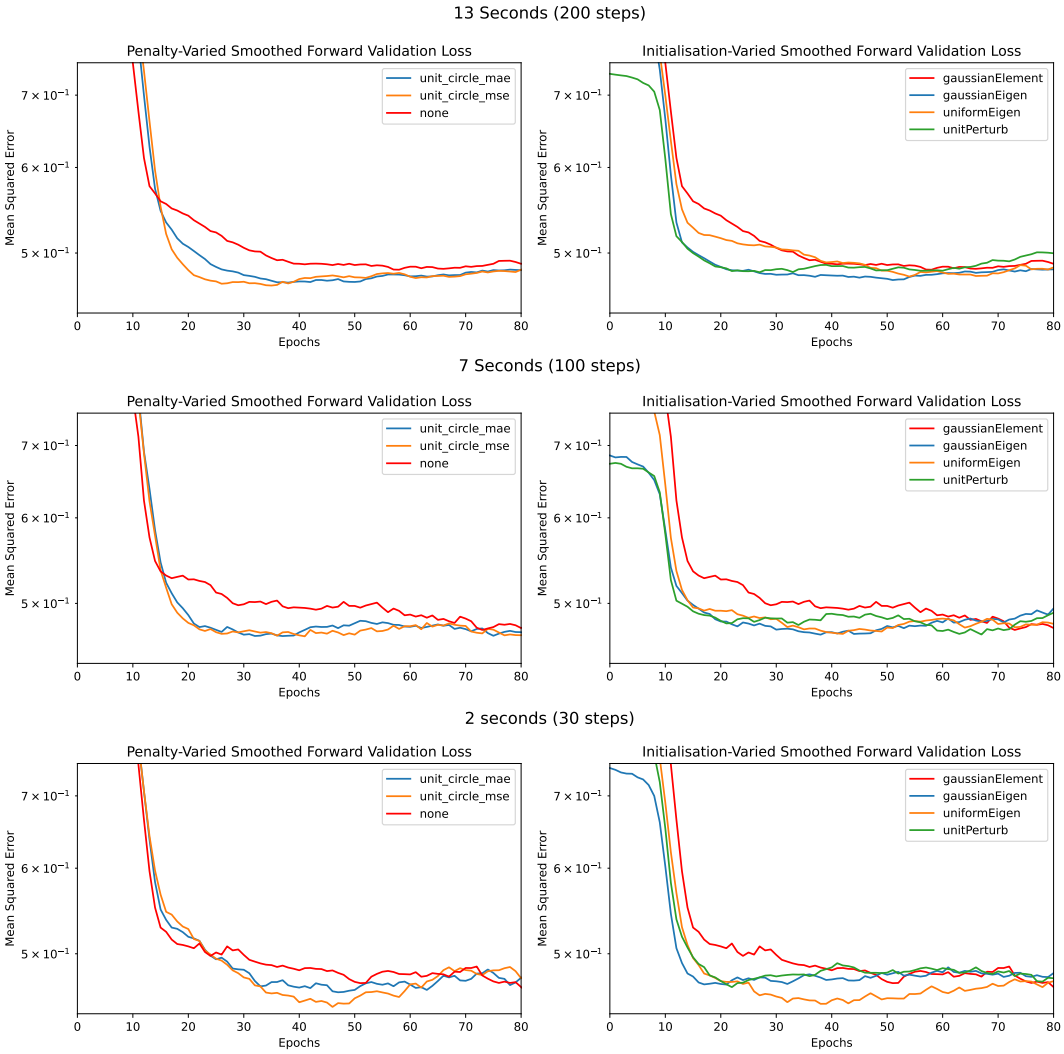
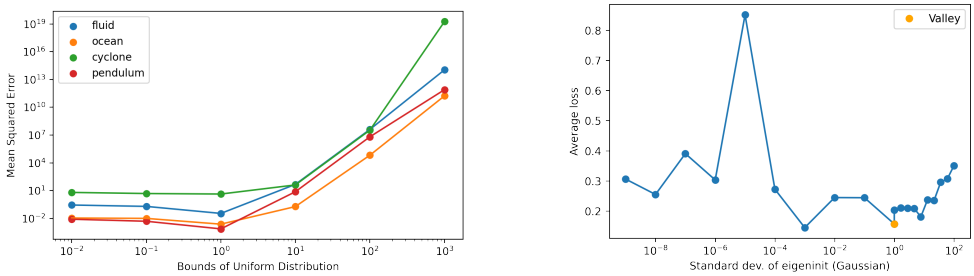


Figure 7: Average validation losses for pendulum data sets with different time horizons and penalty and initialisation schemes

C HYPERPARAMETER SEARCH



(a) Validation loss after 5 epochs

(b) Eigenvalue initialisation standard deviation finder

Figure 8: Hyperparameter search

One can see in Fig. 8a, the bounds of `gaussianEigen` have a large effect on performance after 5 epochs. To complete a hyper parameter search, we draw on empirical techniques used in deep learning practice. In particular, we adapt the learning rate finder described by Smith (2017). Here, we apply the same idea to the standard deviation of the Gaussian eigenvalue initialisation. We use a variety of different standard deviations for the distribution and calculate an appropriate point on the downward loss slope to use as the standard deviation for sampling. The benefit of this is that it is quick to run and may dramatically improve the model performance, particularly over shorter numbers of epochs. By comparing the performance of the valley algorithm in Fig. 8b to the validation loss in Fig. 8a, we can see that it appropriately identifies the minimum and thus might be a useful starting point for hyper parameter search.

D CONVERGENCE STATISTICS

The convergence of solutions also improves with eigeninit and eigenloss. One can see this in Tab. 3 where the approximate epoch of convergence is noted for different data sets and schemes. One can check the validity of this epoch by inspecting Fig. 4.

Table 3: Epoch of model convergence for benchmark data sets and best library candidates. See bold for minimum and *Best Scheme and Control* for the ratio of the best convergence to the control KAE.

Dataset	**Regular KAE**	**Eigeninit**	**Eigenloss**	**Both**	**Best scheme and control**
Ocean	18	13	13	**8**	2.25
Fluid	50	30	45	**10**	5.00
Cyclone	50	40	50	**20**	2.50
Pendulum	50	35	60	**25**	2.00