IIB Project Technical Milestone Report

Modelling dynamic natural phenomena with physics-informed generative models using Deep Learning and the statistical Finite Element Method

Andrew Wang, Christs' College
supervised by Prof. Mark Girolami, CSML.

January 2022

1 Summary

The aim of this project is to train physics informed generative models on observed data of dynamic natural phenomena (e.g. fluid flows) using deep learning. Given these observations, we want to be able to learn the physical states that evolve according to the system. For example, if we are given videos of fluid waves or flow or a chemical reaction, for example those in Figure 2, can we learn the sequences of velocity of chemical concentration?

To achieve this, we develop a model using recent advances in the statistical Finite Element Method (statFEM) [1] and the dynamical Variational Autoencoder methodology (DVAE) [3]. We can model latent states with a low-dimensional spatio-temporal process, and combine the knowledge of the physics with observed data using the statFEM framework. The VAE allows us to efficiently learn a mapping from the perhaps very high-dimensional visual data. The model admits the wide range of linear and non-linear PDEs available, such as the wave, heat and KdV equations. I show that the model can be easily trained to learn these mappings, and furthermore provides a feature extractor taught in an unsupervised manner. The resulting model lets us recover some knowledge about the latent spaces given the visual data. Further work is to apply this to real-life observations of possibly non-linear natural phenomena.

2 Introduction

The model is as follows. Firstly, we can posit or learn any continuous-time spatio-temporal low-dimensional process, described by a partial differential equation (PDE), that governs the state system, such as in the linear case:

$$\partial_t u + L_\Lambda u = f + \xi_\eta, \quad \xi_\eta \sim \mathcal{GP}(0, k_\eta(x, x') \cdot \delta(t - t')), \tag{1}$$

where $(u_t)_{t\geq 0}$ is called the hidden state process and $u := u(x), f := f(x), x \in \Omega \subset \mathbb{R}^d, t \in [0, T]$. The states are mapped at discrete time points $(t_n)_{n\geq 0}$ to "pseudo-observations" via

$$\mathbf{x}_n = \mathbf{C}\mathbf{u}_{t_n} + \mathbf{w}_n, \tag{2}$$

where $\mathbf{w}_n \sim \mathcal{N}(0, \mathbf{R})$. Then we can use statFEM to solve this equation over time whilst also incorporating observed data in a Bayesian manner, by discretising the PDE to obtain a discrete-time linear system. Secondly, these low-dimensional latent variables are mapped to the high-dimensional observed video pixels through a nonlinear observation model:

$$\mathbf{y}_n | \mathbf{x}_n \sim p_{\theta_y}(\mathbf{y}_n | \mathbf{x}_n), \tag{3}$$

where θ_y is neural network weights and the density p_{θ_y} is defined via a neural network. To learn the mappings in an unsupervised manner, we wrap the statFEM in a variational autoencoder using convolutional neural networks (CNN).

This way, we simply pass training examples of videos through the network and train everything end-to-end, including the neural networks and any parameters of the dynamical system. After we have trained the networks, we can obtain the low-dimensional dynamical system states by simply looking into the latent space of the VAE.

The key component that links the deep learning feature extractor and the FEM is the Kalman Filter (KF), which takes observations given by the data and returns state estimates according to the system dynamics.

The model is summarised in Figure 1. I implemented the model in Python and ran experiments using synthetic datasets, described in Sections 3.4 and 4 for results.

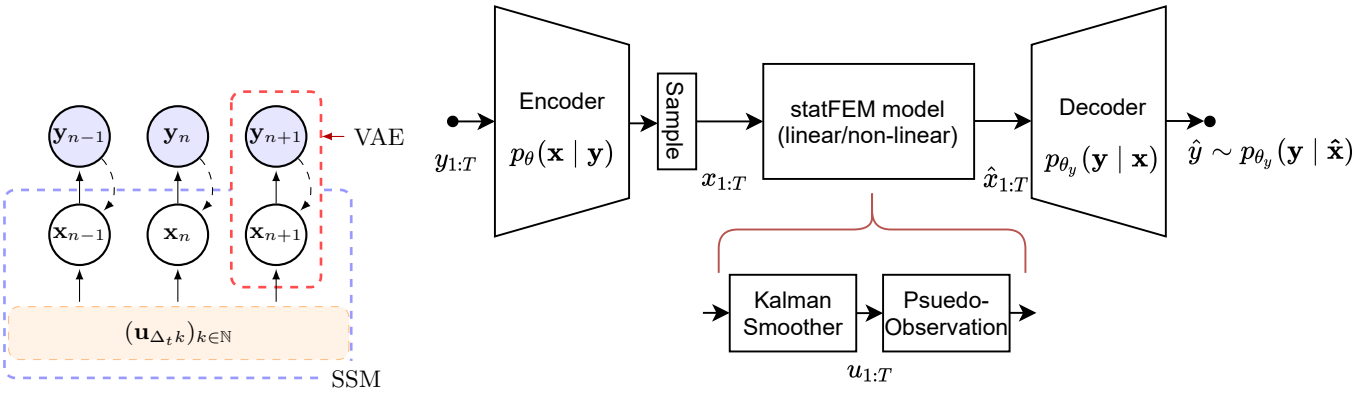

Figure 1: Two perspectives of our statFEM-VAE model. **Left**: probabilistic graph formulation. The discretised PDE evolves in the latent space and emits observations. Observations relate to data via nonlinear deep nets. **Right**: neural network formulation. Firstly, video frames are passed through a deep CNN "encoder" for feature extraction. A layer of stochasticity is introduced with the sampling as per the variational strategy. Then, the Kalman Filter estimates the latent state sequence of the video, according to dynamics obtained by solving the PDE with statFEM. Finally, the states are "observed" and passed through a transposed CNN "decoder" to obtain once again video frames. The whole model is trained end-to-end by comparing the original and reconstructed frames.

3 Methodology

We describe the mathematical developments behind each component of the overall method. Overall, we embed high-dimensional data into a low-dimensional space modelled by PDEs. The model is a generalisation of the Kalman Variational Autoencoders (KVAE) introduced by [2]. However, we replace their linear Gaussian state-space model (LGSSM) with a statistical PDE solver to be able to model a rich variety of dynamical processes, such as the linear wave equation, heat equation, Korteweg–De Vries (KdV) for water waves, viscid Burgers equation or even Navier-Stokes.

3.1 Variational Autoencoders

The variational autoencoder, introduced by [4], is a state-of-the-art model for data compression and image generation, and here we apply them to unsupervised image latent representation learning. Autoencoders chain together an encoding and a decoding CNN. These allow us to "bottleneck" the data and extract complex non-linear features to efficiently model the data. We can feed the model our images bit by bit, and we train the model globally by comparing the error between original and reconstruction and backpropagating this to update the NN weights. The VAE is a probabilistic formulation of this, where each network describes probability measures, see Figure 1. This allows us also to place a Gaussian prior $p_{\theta_y}(\mathbf{x})$ to regularise and ensure a meaningful latent space. However, the posterior $p_\theta(\mathbf{x} \mid \mathbf{y})$ is intractable so it is approximated as $q_\phi(\mathbf{x} \mid \mathbf{y})$. [4] show that training is still easy since the entire model is differentiable, and the model is trained to maximise the log marginal evidence $\log p_\theta(\mathbf{y})$ approximated by the variational lower bound (VLB), as follows:

$$\mathcal{L}(\mathbf{y};\theta,\phi) = \mathbb{E}_{q_\phi(\mathbf{y}|\mathbf{x})}\left[\log p_{\theta_y}(\mathbf{y} \mid \mathbf{x})\right] - D_{KL}\left(q_\phi(\mathbf{x} \mid \mathbf{y}) \| p_{\theta_y}(\mathbf{x})\right) \tag{4}$$

3.2 State-space modelling

The VAE deals with the non-linear feature extraction from the videos, and the SSM deals with modelling the dynamics. The linear case is described by [2], which, as a first order Markov model, can model or approximate a variety of dynamics:

$$\mathbf{u}_t = \mathbf{A}\mathbf{u}_{t-1} + \mathbf{Q}\epsilon_t \tag{5}$$
$$\mathbf{x}_t = \mathbf{C}\mathbf{u}_t + \mathbf{R}\eta_t \tag{6}$$

where $\mathbf{A}$ is the transition and $\mathbf{C}$ is the emission matrix, and $\mathbf{R}, \mathbf{Q}$ are set. The matrices are learnt when during end-to-end neural network training, allowing us to learn and embed familiar notions of gravity and Newton. One can also specify the matrices explicitly, and this is exactly equivalent to the case of statFEM with a linear PDE, described in Experiment 3.4.2; see Section 3.3.

In the linear case, we estimate the states $\mathbf{u}$ from $\mathbf{x}$ with classic forward-backward Kalman Filter (or Smoother):

$$\hat{u}_t = \arg\max p(u_t|x_{1:T}) = \mathcal{N}(\mu_\mathbf{u}, \mathbf{\Sigma}_\mathbf{u}), \tag{7}$$

In the non-linear case, our method uses the full statFEM theory derived in [1] to obtain filtering update equations according to a non-linear PDE, using extended/ensemble Kalman filtering.

3.3 statFEM

Following [1], we discretise the linear time-evolving PDE using the implicit Euler discretisation:

$$\mathbf{M}(\mathbf{u}_n - \mathbf{u}_{n-1}) + \Delta_t \mathbf{A}\mathbf{u}_n = \Delta_t \mathbf{b} + \mathbf{e}_{n-1}, \tag{8}$$

where the variables of interest are defined in [1, Section A.2], and the control $\mathbf{b} = 0$. The statFEM theory then solves the PDE to define a linear transition model of the form

$$\mathbf{u}_n|\mathbf{u}_{n-1}, \eta, \Lambda \sim \mathcal{N}\left(\mathbf{m}_n, \mathbf{\Sigma}_n\right), \tag{9}$$

with

$$\mathbf{m}_n = (\mathbf{M} + \Delta_t \mathbf{A})^{-1}(\mathbf{M}\mathbf{m}_{n-1|n-1} + \Delta_t \mathbf{b}) \tag{10}$$
$$\mathbf{\Sigma}_n = \Delta_t (\mathbf{M} + \Delta_t \mathbf{A})^{-1} \mathbf{G}(\eta)(\mathbf{M} + \Delta_t \mathbf{A})^{-\top} =: \mathbf{Q}, \tag{11}$$

where, from [5],

$$\mathbf{G}_{ii} = \beta^2 \sum_j \mathbf{M}_{ij}, \beta = 0.05 \tag{12}$$

The transition model above then amounts simply to the SSM transition matrix $\mathbf{A}_{SSM} = (\mathbf{M} + \Delta_t \mathbf{A})^{-1}\mathbf{M}$. We take the emission matrix as the identity $\mathbf{C}_{SSM} = \mathbf{I}$. The noise matrices follow similarly.

3.4 Experimental setup

I implemented the model and ran two experiments. All models were coded in Python using the PyTorch library. For both experiments, I trained for 80 epochs with the Adam optimizer with a learning rate of 8e−4 and batch size of 64, on CPU. I experimented with various loss functions, as follows:

1. VAE reconstruction loss.

(a) BCE loss $y\log(\hat{y}) + (1-y)\log(1-\hat{y})$ which is the Maximum Likelihood (ML) under a Bernoulli distributed i.e. when the data y is binary $\{0,1\}$ as used in [2].

(b) Itakura-Saito loss $\frac{y}{\hat{y}} - \log(\frac{y}{\hat{y}}) - 1$ which is the ML under a Gamma distribution, as used by [3] for speech data.

(c) Gaussian NLL loss, which boils down to MSE $(y-\hat{y})^2$ when variance is not learned (i.e. constant).

2. VAE regularisation loss, which regularises the learnt $\mathbf{x}$ distribution. Two methods were tested:

(a) KL Divergence of $q_\phi(\mathbf{x} \mid \mathbf{y})$ with the unit Gaussian as per the VLB:

$$D_{KL}(\mathcal{N}(\mu_x, \sigma_x^2)||\mathcal{N}(0,1)) = -0.5(1 + \log(\sigma_x^2) - \mu_x^2 - \sigma_x^2)$$

(b) Gaussian NLL loss $-(-0.5\log(\sigma_x^2) - \frac{(x-\mu_x)^2}{2\sigma_x^2})$.

3.4.1 Circular orbiting ball pseudo-data

In this experiment, the model is a 4D LGSSM, and the dataset consists of $n = 5000$ binary $32 \times 32 \times 20$ videos of a ball simulating a circular orbit, synthetically generated using the physics library Pymunk, and randomly initialised in space. The loss chosen is the Itakura-Saito loss even though the data should be Bernoulli, and the Gaussian NLL loss for regularisation, as only this combination gave the best reconstruction results.

Past experiments have included using different training hyperparameters, latent space sizes, loss functions, as well as different similar datasets including an non-linear elliptical orbit, or a bouncing ball as in [2].

3.4.2 Wave equation pseudo-data

Here the latent space is 32 dimensional, and we set the latent PDE to be the 1D wave equation:

$$u_t + cu_x = 0, \mathbf{u}(x,0) = \exp(-(x-0.5)^2/\ell) \tag{13}$$

over the domain $[0,3]$, with Dirichlet boundary conditions $\mathbf{u}(0,t) = 0, \mathbf{u}(3,t) = 0$, and where $c = 4, \ell = 0.1$ and $dt = 0.01, n_x = 32$ for the discretisation. At this stage, there is no stochasticity in Equation 13. The dataset consists of $n = 5000$ $32 \times 32 \times 20$ videos of the time-evolving wave with a further cosine non-linearity in the other spatial dimension (see Figure 2), generated using the analytical PDE solution. Because this PDE is linear, we can use the same LGSSM and the transition matrix given by statFEM. The loss chosen is the BCE loss and the KL regularisation, but this is subject to further experimentation.

Past experiments have included using different discretisation methods (explicit Euler, Crank-Nicholson), covariance hyperparameters, training hyperparameters, and latent space sizes.

4 Results

Below are the latest results obtained from the above experiments with hyperparameters, datasets and settings as per Section 3.4.

4.1 Wave pseudo-data

4.1.1 Reconstruction

After training, I assessed the model by visualising the data reconstruction, by passing a test sample through a forward pass of the model. In Figure 2, reconstruction is almost perfect for most of the frame sequence, suggesting that the VAE has learnt good feature extractors given the restrictions of the latent space bottleneck.

However, when I continued the reconstruction to where the wave data surpasses the end of the domain, I observed the reconstruction degenerating towards the end. This is naturally due to the PDE

integrator breaking down at the Dirichlet boundary conditions. By visualising the PDE solution one can see this happening in the latent space, see Figure 3. This will be solved by considering, perhaps, periodic boundary conditions with the Neumann condition. Another problem to solve is the FEM giving numerical dissipation in the PDE solution, resulting in the wave peak decreasing over time.

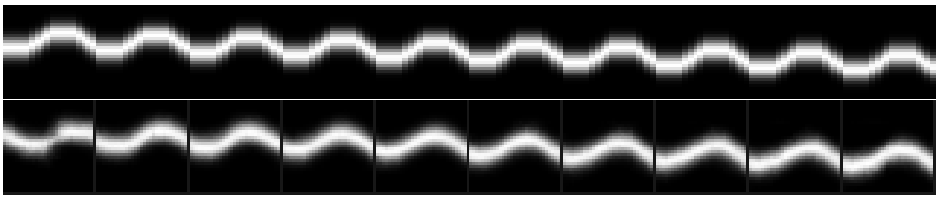

Figure 2: Wave data results. Above: every 2nd frame of one sample original video. Middle: every 2nd frame of the reconstruction of the above sample video. Below: total loss over epoch during training.

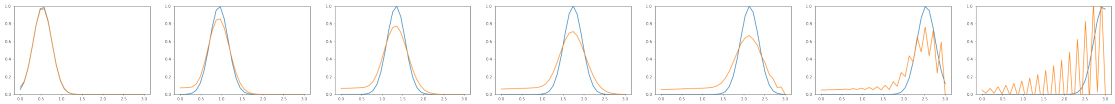

Figure 3: Left to right: every 8th frame from the PDE solution via finite elements (orange) versus the PDE analytical solution (blue), where the PDE is the linear time-evolving wave equation.

4.1.2 Latent space

Since the dataset is generated from the analytical PDE solution and we set the PDE in the latent space, the latent space variable $\mathbf{u}$ is expected to look like the original wave motion. The sequence $\mathbf{u}$ and are plotted in Figure 4. Although one can see the motion of a Gaussian wave-like peak across the frames, it is highly contaminated with noise. Current work is focussed on training the model such that these appear more similar to the inherent generating PDE.

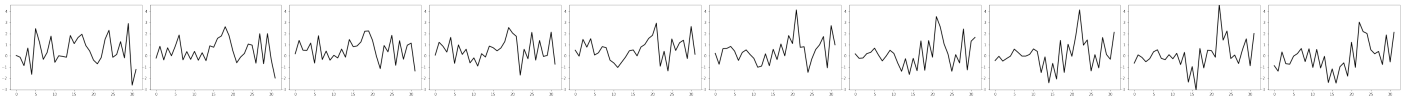

Figure 4: Wave data latent space. left to right: every 2nd frame from the latent space solution filtered from one sample training video.

4.1.3 Forward generation

Forward generation concerns setting an initial state, running the transition model, and observing the generated data. Because, in this experiment, filtered latent spaces did not accurately reflect the inherent generating PDE shape, forward generation from one initial state results in poor reconstruction results.

4.2 Circular orbiting ball data

4.2.1 Reconstruction and latent space

I observed very good results for the same experiment with the circular orbiting ball data, see Figure 5.

4.2.2 Forward generation

I observed very good results for forwards generation with the circular orbiting ball data, even long-term; see Figure 6.

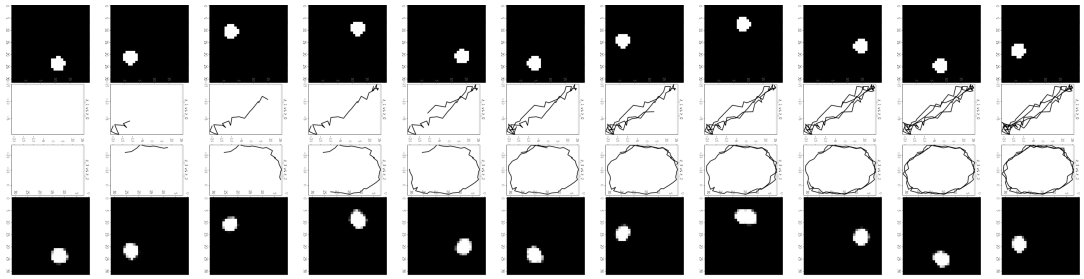

Figure 5: Circular orbiting ball results. Left to right, every 10th frame; top row: of one sample training video. 2nd row: of the 0th and 1st dimensions of the latent space solution filtered from one sample training video, plotted against each other. 3rd row: as above but the 2nd and 3rd dimensions. Bottom row: reconstruction.

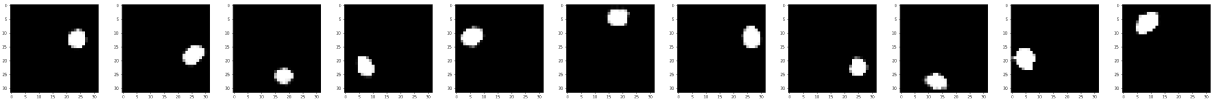

Figure 6: Circular orbiting ball long-term generation. Left to right, every 10th frame of video forward generated from one initial latent state.

5 Current work

My plan for current work is as follows:

1. **Short term.** Recovering interpretable low-noise wave-like statFEM latent space shapes, by playing with the learning. Testing gradient-based boundary conditions. Formalise mathematical construction. Ensure less numerical dissipation and instability in PDE solution and statFEM matrices, by ensuring stability of the discretisation and integration in statFEM.
2. **Before mid-term.** Testing other linear equations, such as heat.
3. **Before end of term.** Testing non-linear equations, such as KdV or Burgers', requiring the EKF or EnKF. Testing perhaps real video data showing advection, convection or diffusion. Perhaps integrating more advanced Computer Vision, such as image segmentation.
4. **Before end of holidays.** Prepare for NeurIPS submission. Full hyperparameter search, including formalising choice of loss function and training method.

References

[1] Connor Duffin, Edward Cripps, Thomas Stemler, and Mark Girolami. Statistical finite elements for misspecified models. *Proceedings of the National Academy of Sciences*, 118, 2021.

[2] Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. pages 3601–3610, 2017.

[3] Laurent Girin, Simon Leglaive, Xiaoyu Bie, Julien Diard, Thomas Hueber, and Xavier Alameda-Pineda. Dynamical variational autoencoders: A comprehensive review. 8 2020.

[4] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. 12 2013.

[5] Ömer Deniz Akyildiz, Connor Duffin, Sotirios Sabanis, and Mark Girolami. Statistical finite elements via langevin dynamics. 10 2021.