Time Series Forecasting for Mortality and Disease Incidence Data

Diogo Vieira Ferreira
diogo.v.ferreira@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2023

Abstract

In contemporary healthcare and mortality forecasting, conventional statistical models like the Auto-Regressive Integrated Moving Average model (ARIMA) and simple machine learning models, such as Long Short-Term Memory (LSTM) models, have long been the standard. However, this landscape sharply contrasts with the significant advancements in machine learning, particularly concerning state space models.The focal point of my thesis is to investigate the potential advantages of applying state-of-the-art machine learning models to a mortality forecasting task. To accomplish this, mortality data spanning 252 months was extracted from the United States' underlying mortality database, encompassing 57 different causes of death across five states. This data was used to formulate a mortality forecasting challenge, where models are presented with a 24-month historical context and tasked with forecasting the subsequent 6 months. Six state-of-the-art models were subjected to rigorous testing, which included the S4 and S4 diagonal models. In addition, the LSTM and Persistence model were utilised as baselines for comparison. The results revealed that, for short-term forecasts, the state-of-the-art models exhibited similar or marginally inferior performance compared to the baselines. However, as the forecasting horizon extended, these advanced models consistently outperformed the baselines. Moreover, it was observed variations in model performance based on the periodicity of the causes of death, with some models excelling at predicting periodic trends, such as the S4, and others proving more proficient at forecasting non-periodic patterns, exemplified by the S4 diagonal.
Keywords: Healthcare forecasting; Multivariate mortality forecasting; Mortality forecasting; Time-series forecasting; State Space Models; Machine Learning; United States underlying cause of death database;

1. Introduction

In recent times, a major breakthrough has occurred in the field of State Space Models (SSM), leading to remarkable outcomes, particularly in the Long Range Arena (LRA) [25]. This advancement can be attributed to the development of the S4 model [5], which integrates a distinctive parameterisation of the state space with an efficient algorithm for computing the state space kernel. As a versatile sequence-to-sequence model, the S4 model has demonstrated its adaptability across various domains, including image classification, language modeling, and univariate forecasting, consistently achieving competitive results in these diverse tasks. Building upon the success of the S4 model, several variants have emerged, including the S4D [6] and DSS [8] diagonal versions, and even a convolutional model inspired by the S4 principles, known as the structured global convolution model [16], among others. These models exhibit noteworthy competitiveness among themselves, with comparable performances within the LRA. This renders them intriguing candidates for applications that have yet to harness the potential of this development, particularly in healthcare-related forecasting, such as multivariate mortality forecasting.

Forecasting is pivotal in healthcare decision-making, equipping decision-makers with the ability to proactively prepare for future circumstances. It facilitates optimized resource allocation by providing insights into the prevalence of diseases or causes of death within specific periods. Mortality forecasting, in particular, holds vital importance in identifying patterns and trends in mortality rates, thereby guiding the development of effective public health policies and interventions. Moreover, monitoring changes in mortality rates over time is essential to evaluate the effectiveness of medical treatments and interventions. For actuaries involved in life and pension insurance, mortality and disease incidence data assume great significance as they underpin the financial viability of these businesses. Time-series forecasting in healthcare finds diverse applications, including predicting medicine expenditures [12], medical bookings [21], and disease incidence, such as forecasting COVID-19 infections [14]. However, these forecasts are commonly conducted using Recurrent Neural Networks, such as Long Short-Term Memory (LSTM) models, or simpler statistical models like the Auto-Regressive Integrated Moving Average (ARIMA) or its variations. Such models, although capable in their

respective applications, are far from the state of the art, the original LSTM [10] is from 1997 and the ARIMA model [1] was popularised by Box and Jenkins in 1970. On the other hand, the most recent models like the S4 have not yet seen their potential explored in more diverse areas, as forecasting in healthcare and mortality forecasting. This gap between the models currently in use in healthcare and state of the art model is the main driver behind this thesis.

1.1. Thesis Proposal and Research Questions

This thesis aims to compare the performance of novel state space models, including the aforementioned variants, with classical and presently employed models in the context of multivariate mortality forecasting. The goal is to generate a 6-month forecast based on a 2-year contextual framework, with a monthly granularity. The dataset utilised originates from the United States underlying cause of death database [2], focusing specifically on the most common Causes of Death (COD) in the five most populated states: California, Texas, Florida, New York, and Pennsylvania. The data has been meticulously extracted and processed to facilitate the training and testing of all models. Initially, the hyperparameters of each model will be fine-tuned to achieve optimal performance. Subsequently, the models will be evaluated using metrics such as Mean Squared error (MSE) and Mean Absolute Error (MAE) to assess their respective forecasting accuracy for 1, 3, and 6-month intervals. We will analyse the performance of each model according to the state of origin of the data and according to the causes of death, to see if there are differences in performance given this items. Additionally, We will explore the impact of the addition of extra input features, such as State features that encode the state of origin of a given training example. The state of the art models will be contrasted against an LSTM and a Persistence model that serve as baselines.

2. Background

This section will, firstly, present the baselines used in this work, the Persistence and the LSTM model along with the theoretical concepts behind machine learning. Secondly, it will analyse some of the related work done in the field of healthcare and mortality forecasting. Followed by a description of the latest machine learning models, with an emphasis on State Space Models, that were tested in this work, namely, the S4, S4D, DSS, LS4KB, LS4PB and SGconv.

2.1. Fundamental Concepts

This section is divided into two parts, the first introduces the baseline models, the Persistence and the LSTM models. The second introduces State Space Models as most of the models used in this work are of this kind.

2.1.1 Baselines

The first and the simplest baseline is the Persistence model. This model simply consists of forecasting the next time step using the value of the previous time step [13],

$$\hat{y}_{t+1} = y_t. \tag{1}$$

The second baseline is the LSTM model [10], which is an extension of typical Recurrent Neural Networks (RNN), which are designed to deal sequential input, like in time series analysis and speech recognition. Unlike feedforward neural networks, where a given input has always the same output regardless of the previous inputs, in RNNs the sequence by which inputs are fed will influence the output. In Classical RNNs this is done by the intermediary of a memory layer, m, whose goal is to keep track of the previous inputs, x. This interaction is governed by:

$$\begin{aligned} \mathbf{m}_t &= \mathrm{g}(W_M \mathbf{m}_{t-1} + W_{input} \mathbf{x}_t) \\ \mathbf{z}_t &= W_{output} \mathbf{m}_t, \end{aligned} \tag{2}$$

where applying the matrix W_m to the previous memory layer, m_{t-1} and summing this result with the next input, $\mathbf{x}_{t+1}$, times its own normal weight matrix, W_{input}, we obtain the next iteration of the memory layer, $\mathbf{m}_t$. This layer is then used to obtain the output, z_t [11].

The issue with this typical RNN structure is that, for long sequences, the memory layer will be saturated with information, because it never *forgets* an input. This causes the problem of vanishing/exploding gradients during back-propagation in training. The LSTM partially solves this issue by introducing more layers similar to the memory layer, but whose purpose is to learn what to forget and what to keep of a given sequence. It, therefore, consists of four layers or gates, the forget gate, **f**, the input gate, **i**, the output gate, **o**, and the $\tanh$ gate , $\tilde{\mathbf{c}}$. The information is stored in the cell state, **c**, that will be changed through element-wise summations and multiplications with the gates [19]. Finally, the output at every recursion is given by **h**, which is controlled by the output gate and the cell state, as shown in the next equations:

$$\begin{aligned} \mathbf{f}_t &= \sigma(W_{\mathbf{f}} \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_{\mathbf{f}}), \\ \mathbf{i}_t &= \sigma(W_{\mathbf{i}} \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_{\mathbf{i}}), \\ \tilde{\mathbf{c}}_t &= \tanh(W_{\mathbf{c}} \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_{\mathbf{c}}) \\ \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \\ \mathbf{o}_t &= \sigma(W_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_{\mathbf{o}}), \\ \mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t). \end{aligned} \tag{3}$$

These equations are displayed visually in Figure (1).

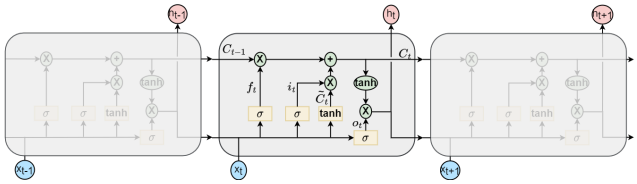

Figure 1: LSTM schematics. The divergence of lines represents vector duplication, the joining of represents concatenation, the green circles are element-wise operations, the red circles are the layers activations, the blue circles are the inputs of the layer and the yellow boxes represent the different gates and their activation function.

2.1.2 State-Space Models

State space models were originally developed to represent physical systems with a set of 1-D inputs $\mathsf{x}(t)$, outputs $\mathsf{y}(t)$ and state variables $\mathsf{u}(t)$ that were related via first order ordinary differential equations, according to:

$$\begin{aligned} \mathsf{u}'(t) &= A \cdot \mathsf{u}(t) + B \cdot \mathsf{x}(t), \\ \mathsf{y}(t) &= C \cdot \mathsf{u}(t) + D \cdot \mathsf{x}(t). \end{aligned} \tag{4}$$

Equation (4), defines a linear system with P inputs, Q outputs and N state variables. As such, matrix A, named state matrix, has dimensions $N \times N$, matrix B, named input matrix, has dimensions $N \times P$, matrix C, named output matrix, has dimensions $Q \times N$, and matrix D, named feedforward matrix, has dimensions $Q \times P$. The state matrix defines how to update the state space based on the current state space, the input matrix defines how to update the state space based on the current inputs, the output matrix defines how the state space influences the output, and the feedforward matrix bypasses the state space and defines how the current input influences current output. Recent research [7, 4, 27, 8] has applied SSMs in a machine learning context, particularly time series analysis, where the goal is to discover the optimal A, B, C and D matrices.

However, to apply SSMs to time series, we first need to accommodate this models for a discrete input. This is done by selecting a step size Δ such that, now the derivative in equation Equation (4) become the next iteration and $u(t)$ becomes the previous iteration:

$$\begin{aligned} u_k &= \bar{A} \cdot u_{k-1} + \bar{B} \cdot x_k, \\ y_k &= \bar{C} \cdot x_k + \bar{D} \cdot u_k. \end{aligned} \tag{5}$$

Note that the matrices also needed to be changed and, assuming a zero order hold:

$$\begin{aligned} \bar{A} &= e^{(A\Delta)}, \\ \bar{B} &= A^{-1}(\bar{A} - I)B, \\ \bar{C} &= C, \\ \bar{D} &= D. \end{aligned} \tag{6}$$

When A is not diagonal, computing its exponential is computationally demanding, and thus we often resort to approximations such as the bilinear approximation, where $e^{(A\Delta)} \approx (I - \Delta/2 \cdot A)^{-1}(I + \Delta/2 \cdot A)$ with this transformation the matrices are now as follow:

$$\begin{aligned} \bar{A} &= (I - \Delta/2 \cdot A)^{-1}(I + \Delta/2 \cdot A), \\ \bar{B} &= (I - \Delta/2 \cdot A)^{-1}\Delta B, \\ \bar{C} &= C, \\ \bar{D} &= D. \end{aligned} \tag{7}$$

An interesting property of state space models is their ability to be represented as a convolution as well, which facilitates training. This is done by unrolling the recursion in Equation (5), so that

$$\begin{aligned} y_k &= \overline{CA^kB} \cdot x_0 + \overline{CA^{k-1}B} \cdot x_1 + ... + \overline{CB} \cdot x_k, \\ y &= \bar{K} * \mathbf{x}, \\ \bar{K} &\in \mathbb{R}^L = (\overline{CB}, \overline{CA^1B}, ..., \overline{CA^{L-1}B}, \overline{CA^LB}). \end{aligned} \tag{8}$$

In the previous expression, $\bar{K}$ is the convolution kernel or filter, and $*$ denotes a convolution operation.

SSMs are sequence-to-sequence models, meaning the length of input is the same as the length of the output, which is not ideal for forecasting. To get around this we provide the model with the training sequence padded with a sequence the size of the intended forecast window. Then we train the model to output a sequence where the forecast is present in place of the padding.

2.2. Related work

This chapter takes a look at previous works surrounding forecasting in healthcare. First, it looks at current healthcare and mortality forecasting applications, the models that are being used and to what extent do they succeed. Secondly, it analyses recent work in the field of time series forecasting, so we can understand the most recent models, their advantages and disadvantages, and their potential applicability to healthcare forecasting.

2.2.1 Healthcare and Mortality Forecasting

In the realm of healthcare forecasting, we delve into several insightful studies exploring the effectiveness of various predictive models. Kaushik et al. [12] conducted a comparison of forecasting techniques for the weekly average patient expenditure on two commonly prescribed pain medications, A and B, within the United States. Their analysis encompassed five models, including Persistence, ARIMA, Multi Layered Perceptron (MLP), LSTM, and an ensemble approach that combined all the previous models' predictions. Evaluation metrics like Root Mean Squared Error (RMSE) and R-squared (R2) were amalgamated into a comprehensive objective function, with models employing walk-forward validation. Notably, the ensemble model exhibited superior performance, closely followed by MLP and LSTM, outperforming traditional methods such as ARIMA and Persistence.

Kumar et al. [14] shifted the focus to forecasting COVID-19 cases in ten highly affected countries, comparing ARIMA and Prophet models. Metrics like RMSE,

Mean Absolute Error (MAE), Root Relative Square Error (RRSE), and Mean Absolute Percentage Error (MAPE) favored ARIMA. Notably, machine learning models were absent from their evaluation, emphasizing the pragmatic applicability of these tried-and-true methods.

Lastly, Piccialli et al. [21] introduced an innovative approach to medical booking forecasting, integrating multiple data sources that span medical booking time series and weather and air-quality data. Their analysis involved causal and correlation analyses to validate interrelationships among these variables. An ensemble architecture, leveraging Convolutional Neural Networks (CNN) and LSTMs, impressively outperformed classical and machine learning methods, showcasing the potential for advanced techniques in this domain.

Mortality rate forecasting, distinct from general healthcare forecasting, being also used in actuarial work, particularly in life and pension insurance. Traditional mortality models like the Lee-Carter mortality model [15] decompose mortality into age and time components using Principal Component Analysis (PCA). Machine learning is gaining popularity in this field, with efforts to extend traditional models by Perla et al. [20] and Mathonsi et al. [18].

Perla et al. [20] aimed to forecast mortality rates using the Human Mortality Database, categorizing populations by region and gender. They used a model with three embedding layers, one of which was either a CNN or a LSTM. The CNN model outperformed the LSTM version and other state of the art methods like the DEEP model [22], demonstrating the possible benefits of applying machine learning models to mortality forecasting.

Mathonsi et al. [18] introduced the Multivariate Exponential Smoothing Long Short Term Memory (MES-LSTM) model, a hybrid approach combining multivariate exponential smoothing with LSTM. This model, based on the ES-RNN method developed by Smyl [24] that outperformed both statistical and machine learning methods in the M4 competition [17] in a multivariate context. The MES-LSTM model was tested using the Our World In Data COVID-19 dataset, incorporating 46 variables and was able to outperform the baseline models in most contexts.

2.2.2 Recent Methods for Time Series Forecasting

This section will take a look at recent sequence models that will be put to the test in the mortality forecasting task. Most of the models, namely the S4D [6], DSS [8], LS4KB and LS4PB [9] are variants of the S4 model [5], so we will take a closer look at the latter and briefly explain the variations. We will also take a look at the only convolutional model, the SGconv [16].

SSMs have two main issues when it comes to their applicability in time series forecasting, they have issues dealing with long term dependencies, and secondly, they have high computational and memory requirements due mainly to the successive matrix multiplications, as seen in Equation (8).

The first issue was solved by Gu et al. [4], with the introduction of a special initialisation for matrix A, the HiPPO matrices,

$$\text{HiPPO} = A_{nk} = \begin{cases} (2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}. \quad (9)$$

These matrices save the coefficients of the Legendre series, whose weighted sum approximates the input, as a way of allowing the state u to memorise the history of inputs x.

The second issue was solved by S4 model [5], by introducing a special parameterisation of matrix A and a special kernel computation algorithm, that, when put together, avoid the successive matrix multiplications. The parameterisation of A aims to bring the A matrix closer to the ideal case, a diagonal matrix, where matrix multiplications become trivial. However the HiPPO matrix family is not diagonalisable but it is Normal Plus Low-Rank (NPLR). Meaning,

$$A = V\Lambda V^* - PQ^* = V(\Lambda - V^*P(V^*Q)^*)V^*, \quad (10)$$

for unitary $V \in \mathbb{C}^{N\times N}$, diagonal Λ and low-rank factorization $P, Q \in \mathbb{R}^{N\times r}$ where r is the rank.

The idea behind the algorithm is not to compute the K kernel directly but rather compute its truncated generating function, $\hat{K}_L(z)$, i.e., compute it in the frequency rather than the time domain, and then recover the kernel using an inverse Fourier transform. The truncated generating function can be computed as,

$$\hat{K}_L(z) = \tilde{C}((I - \overline{A}z)^{-1}\overline{B}), \quad (11)$$

where $z \in \Omega = \{\exp(2\pi\frac{k}{L} : k \in [L])\}$, and $\tilde{C} = \overline{C}(I - \overline{A}^L z^L)$ is a constant. This now involves an inverse of matrix A, we can apply the Woodbury identity so that the inverse is reduced to terms of Λ,

$$(\Lambda + PQ^*)^{-1} = \Lambda^{-1} - \Lambda^{-1}P(1+Q^*\Lambda^{-1}P)^{-1}Q^*\Lambda^{-1}. \quad (12)$$

Note that the V matrices are not included, this is because they are not going to be trained and will remain a constant throughout. Combining Equation (12) and Equation (11) in terms of the original SSM matrices we obtain,

$$\hat{K}_{NPLR} = c(z)\times \left[k(\tilde{C}, B) - k(\tilde{C}, P)(1 + k(Q^*, P))^{-1}k(Q^*, B)\right] \quad (13)$$

where $\mathsf{c}(z) = \frac{2\Delta}{1+z}$ and $k(x, y) = \sum_i \frac{x_i y_i}{2\frac{1-z}{1+z} - \Lambda_i}$ are Cauchy Kernels, which have several fast implementations, such as the multipole method that can be done in $O(N)$ [3]. This completes the SSM implementation, all it is left is to organise the S4 layer. Each SSM defines

a 1-D sequence map, for a problem with H features we require H independent copies of the SSM, one for each feature. Then, in order to capture dependencies between features, a mixing layer composed of two fully connected feedforward networks is added to obtain the final output.

The Diagonal State Space (DSS) and the S4Diagonal (S4D) [8, 6] simplify the S4 approach, by reducing matrix A to a diagonal matrix, Λ with eigenvalues λ_i. This implies a modification on the HiPPO initialisation, both authors opted to simply drop the low-rank component, yielding the

$$\textit{Skew-Hippo}_{i,j} = \begin{cases} (2i+1)^{\frac{1}{2}}(2j+1)^{\frac{1}{2}}/2 & i < j \\ -\frac{1}{2} & i = j \\ -(2i+1)^{\frac{1}{2}}(2j+1)^{\frac{1}{2}}/2 & i > j \end{cases}. \quad (14)$$

This no longer requires the algorithm the S4 uses to be efficient and the kernels may be computed directly in the time domain. The DSS uses a zero hold assumption Equation (6) to discretise the SSM matrices and computes the kernel as,

$$K = w \cdot \Lambda^{-1} \cdot \text{row-softmax}_{\epsilon}(P), \quad (15)$$

where $P_{i,k} = \lambda_i k \Delta$ and $w \in \mathbb{C}^N$. This parameterises the state space using $\lambda, w \in \mathbb{C}^N$ and the row-sofmax ensures numerical stability, preventing exponential explosion. The S4D, on the other hand, does not require any particular descritisation and computes the kernel as

$$\begin{gathered} \overline{K} = (\overline{B}^T \odot \overline{C}) \cdot V_L(\overline{A}), \\ V_L(\overline{A})_{n,l} = \overline{A}_n^l, \end{gathered} \quad (16)$$

where $\odot$ is the Hadamard product and V_L is known as the Vandermonde matrix, which can be computed efficiently in $\tilde{O}(N+L)$ operations and $O(N+L)$ space [6].

Another variant of the S4 was introduced by Hasani et al. [9], where they introduce an input dependent state transition mechanism. This essentially means that instead of using a constant A matrix to govern state space transitions, like in Equation (5) we now have a function parameterised by A and B of the input x,

$$\begin{gathered} u_k = [\overline{A} + \overline{B}x_k]u_{k-1} + \overline{B}x_k, \\ y(t) = \overline{C}u_k. \end{gathered} \quad (17)$$

This makes it so that in the convolutional view it appears another kernel besides $\overline{K}$, the $\overline{K}_{liquid}$,

$$\begin{gathered} y_k = \overline{K} * x + \overline{K}_{liquid} * x_{correlation}, \\ \overline{K}_{liquid} \in \mathbb{R}^{\tilde{L}} := (\overline{C}\overline{A}^{(\tilde{L}-i-p)}\overline{B}^p)_{i\in[0,\tilde{L}],p\in[2,P]} \end{gathered} \quad (18)$$

where $x_{correlation}$ is all the combinations of 2^{nd} until P^{th} order correlation, a new hyper-parameter, and $\tilde{L}$ is the length of $x_{correlation}$. To compute $\overline{K}$ we can rely on the S4 parameterisation and algorithm but not for $\overline{K}_{liquid}$. The authors introduced two ways to compute this kernel, yielding two models, the LS4KB where $\overline{K}_{liquid}$ for a given order p, $\overline{K}_{liquid=p}$, can be computed as,

$$\overline{K}_{liquid=p} = [\overline{K} \odot \overline{B}^{p-1}] \cdot J_L, \quad (19)$$

where J_L is the backwards identity matrix and $\overline{K}$ is the pre-computed S4 kernel. The other computation, LS4PB, introduces a simplification where $\overline{A}$ in Equation (18) is an identity matrix, this yields,

$$\overline{K}_{liquid=p} = (\overline{C}\overline{B}^p). \quad (20)$$

The final model is the Structured Global Convolution Kernel or SGconv, developed by Li et al. [16]. This model takes some inspiration in the S4's principles of efficient parameterisation regarding sequence length and a decaying kernel structure where the weights for convolving with close neighbours should be larger than those for distant neighbours to develop a convolutional model. Each feature is assigned a kernel $K = \frac{1}{Z}[k_0, ..., k_N]$, composed of the concatenation of N sub-kernels, with $N = \log_2(\frac{L}{d}) + 1$ where L is the sequence length and d is the hyper-parameter that defines the size of each sub-kernel, we can see that the number of parameters is efficient on the sequence length due to the $\log$. In turn, each sub-kernel is defined as $k_i = \alpha^i \text{Upsample}_{d2^{\max(i-1,0)}}(w_i)$, where α determines the weight decay responsible for the second principle and $w_i \in \mathbb{R}^d$ are the parameters of the sub-kernels, the upsampling operation also ensures the efficient parameterisation, since it allows longer kernels to stem form the same number of parameters as shorter kernels, for a graphical interpretation see Figure (2).

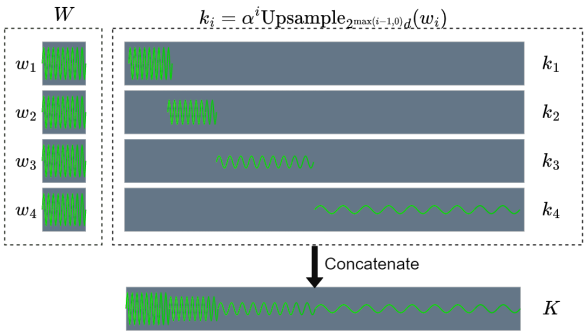

Figure 2: Graphical representation of the kernel of the structured global convolution model. Note, that despite the image not being to scale, the size of each sub-kernel is double the size of the previous sub-kernel, while having the same number of parameters. Additionally, the weights w_i are represented as being all the same, which is not necessarily the case. Illustration adapted from [16].

3. Methodology

The state space models discussed in the previous section will be tested, alongside the LSTM and Persistence baselines. These models will be trained and tested on a multivariate mortality forecasting task, which consists of forecasting the mortality rates of several CODs when given a two year context. More specifically, the task will consist of three sub-tasks, predicting the mortality rates

for the next month, the next three months and the next six months. This section is divided into three parts, Datasets and Methodological Approach, Hyper-Parameter Adjustment and Experiments.

3.1. Datasets and Methodological Approach

The data stems from the United States underlying cause of death database of the United States Center for Disease Control [2]. From this database the monthly death counts of 57 Causes of Death (COD) from 1999 to 2020 across the 5 most populous States, California, Texas, Florida, New York and Pennsylvania was collected, totalling 252×5 data points. In order to compare values across States the death counts for each year was normalised by their corresponding State population and multiplied by 100000 to obtain the deaths per 100000 inhabitants.

To get a better understanding of the data, an analysis was performed on the CODs to see which were periodical and which were non-periodical. First, they were detrended by doing a linear regression to estimate the trend and then subtract it to the data. Second, the CODs were subjected to a Welch periodogram [26] and it was determined if it had any peaks with periods below 24 months, the size of the context, with a power spectral density above 0.2 in every State. This yielded a list of 11 periodical CODs and 32 non-periodical CODs, Figure (3). Interestingly, the top 10 most common CODs are included in the 11 periodical features, which is not surprising because least common features are more susceptible to noise.

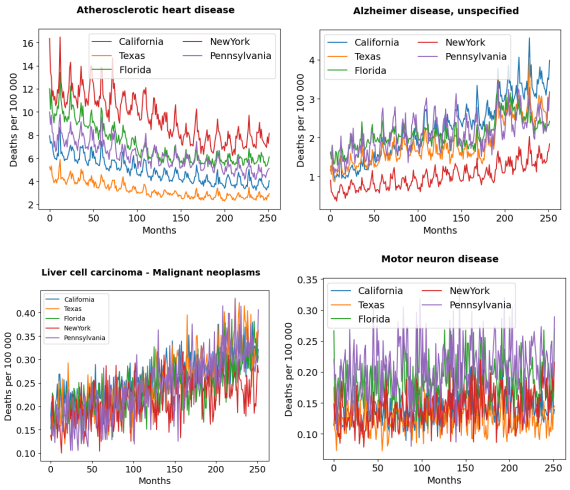

Figure 3: This figure presents four graphs representing four causes of death from 1999 to 2019 measured in deaths per 100 000 inhabitants. On the top row we have atherosclerotic heart disease and Alzheimer disease, that are included in both the top 10 most common causes of death and are also among the ones who show the most periodicity. On the bottom, there are two graphs one from liver cell carcinoma and another from motor neuron disease, which are both, in turn, in the top 10 least common causes of death and are the ones who present the least amount of periodicity.

The models will receive the 57 CODs from a 24 month window, however these features do not provide the model with information about the time of the year each data point stems from, making it difficult to get a sense of seasonality. So we introduced two sinusoidal features with a period of 6 and 12 months to provide this temporal information.

The data was divided into training and test sets, consisting of 75% and 25% of the total data or 188 and 64 months, respectively, for each State. These two sets were organised into 30 month parcels, 24 month for context and 6 month as the ground truth, to evaluate the predictions. The parcels from each state were then merged together into two big training and test sets, with a grand total of 795 training examples and 165 test examples, that were organised into batches of size 64.

The models were all implemented within a simple architecture that is common between them consisting of two linear fully connected layers, one between the input and the model and another between the output of the model layer and the actual output, called the encoder and decoder, respectively, and the working layer that consist of the models we have been discussing so far.

Now for the training itself all models made forecasts for the next 6-month, and the training loss was computed from the MSE,

$$\begin{aligned} \text{MSE}(Y) &= 1/N \sum_{i=1}^{N} ||y_i - \hat{y}_i||^2 \\ \text{MAE}(Y) &= 1/N \sum_{i=1}^{N} ||y_i - \hat{y}_i||, \end{aligned} \tag{21}$$

of all the outputs including the forecast and the models adjustment to the known values. Additionally, the sinusoidal features are removed during the computation of the loss. During testing, we collect 6 main metrics, the 1-month, 3-month and 6-month errors both MSE and MAE, (21).

3.2. Hyper-Parameter Adjustment

To find out the optimal hyper-parameters of each model, a grid search technique was employed where I select a range of values for each hyper-parameter and train each combination of hyper-parameters 10 times, to account for variability. The best combination is determined based on the average MSE across the 10 iterations of the 6-month forecast of the test set. Every model has a hyper-parameter that was the number of stacked working layers, $n_w l = [1, 2, 3]$. The state space models, namely, the S4, S4Diagonal, DSS and the Liquid-S4 both the KB and PB, also have the state space dimension, $N = [16, 32, 64, 128]$ and the number of features, $H = [30, 59, 118]$. The liquid models have an extra hyper-parameter, the maximum correlation order, $P = [2, 3, 4, 5]$. The convolutional SGconv also has the number of features, $H = [30, 59, 118]$, but instead of state space dimension we have the dimension of each sub-kernel, $d = [4, 8, 16, 30]$, note that $d = 30$ is the maximum dimension for $L = 30$ according to what we have seen in Section 2.2.2. Finally, the LSTM has the hidden size, $h = [16, 32, 64, 128]$, that corresponds to the dimensions of $\mathbf{h}$ and the likelihood of teacher forcing, $P_{tf} = [0.0, 0.25, 0.5, 0.75, 1]$, meaning the likelihood of providing the model with the ground truth to make the

following prediction versus providing its own previous prediction.

3.3. Experiments

Using the optimal hyper-parameters, we can conduct the two experiments, the first was designed to compare the models performance on the 1, 3 and 6 month forecasts. To do this we train each model 20 times, similarly to [12], and selected the one with the best 6-month MSE performance. The results were grouped by COD, and also by type of COD, periodical and non-periodical, as described earlier. This allows us to see if the models have differential performance according to the periodicity of features. The CODs have varying scales, so to facilitate comparisons the results are normalised using min max normalisation.

The second experiment, named feature variation, aims to understand the impact of additional features such as the sinusoidal features used. So we introduce the state features, consisting of a one-hot-encoding of the state of origin of a given example. We test every combination of additional features, no extra features, only the sinusoidal features, only the state features and both the state and sinusoidal features. Each combination was trained 10 times and the results were compared on the basis of the MSE of the 6-month forecast.

4. Results & discussion

This section will present and discuss the results from the protocols presented previously. Therefore, it will be divided into 3 parts, Hyper-Parameter Adjustment, Model Comparison and Feature Variation.

4.1. Hyper-Parameter Adjustment

The models behaved rather similarly, so we will briefly introduce the optimal set of hyper-parameters for all of them and then use the case of the S4, Figure (4), to better understand the training process. The results were:

- **S4**: H=118; N=32; n_{wl}=1
- **S4D**: H=118; N=16; n_{wl}=1
- **DSS**: H=118; N=32; n_{wl}=1
- **LS4PB**: H=118; N=128; n_{wl}=1, P=3
- **LS4KB**: H=118; N=16; n_{wl}=1, P=4
- **SGconv**: H=118; d=16; n_{wl}=1
- **LSTM**: h=118; P_{tf}=0.75; n_{wl}=1

We can see from Figure (4) and the listing above that the models prefers a higher number of features. Such behaviour may indicate that the features are more easily interpretable in higher dimensions. This may be due to the models decomposing the feature into several components that are easier to analyse, an example of such decomposition may be decomposing the time series into trend and seasonality and forecast them separately. Secondly, we can also observe that all the models also preferred a single working layer. This may be the case because the task at hand is sufficiently simple to be handled by a single layer. For example, a single layer of the S4 model has $5N$ trainable parameters per feature, with $N = 32$, it totals to 160 parameters per features for a task where each feature has only 24 values an increase in parameter may only lead to overfitting. Finally, when we observe the SSM size, N, in 4 we can see that it is the least significant hyper-parameter, as its impact on MSE is much less than the number of layers or the number of features.

4.2. Model Comparison

Figure (5) sums up the results from this experiment on the MSE metric. From this figure we can see that the state of the art models have similar performance, with the S4, S4D and SGvonv being slightly better off. We can also see that the baseline models are very competitive in the 1 month forecast, with the Persistence being the best performing model in this scenario. However, as the forecasting horizon increases to 3 and 6 months we see that the performance of the baselines deteriorates much more than the remaining models. Indicating that the more complex models are more suitable for longer forecasts.

Figure (6) shows a more in depth analysis where the models performance is grouped by the type of feature, periodical versus non-periodical. This figure only presents the best performing models on the periodical, the S4, and the non-periodical, the S4D, features along with the baselines. Not surprisingly the performance of the models is overall better on the periodical features.

Let us first focus on the periodical features, in the 1 month forecast the LSTM and the Persistence models perform slightly better, and the S4D is somewhat worse off. As before, increasing the forecast window has a bigger impact on the baselines, however, the LSTM is still competitive in the 3 month forecast, with the greatest decrease in performance coming on the 6 month forecast.

In the non-periodical features, the SSMs are far more competent than the baselines, especially in the 6 month forecast, with the S4D performing slightly better than the S4. This is the case for all the state of the art models, suggesting that these more complex models are better suited to deal with this kind of non-periodical features.

4.3. Feature Variation

The results from this experiment can be summed up by Figure (7) where there are 3 models presented, the S4, the S4D and the LSTM. The S4 showed no improvement when provided with the sinusoidal features, however, the addition of the State features significantly worsened its performance, this behaviour is shared with the DSS and LS4 PB models. On the other hand, the S4D model showed improvements when provided with the sinusoidal features, but its performance also worsened significantly when provided with the State features, this was also the case for the SGconv and LS4 KB models. The LSTM model was the only model that somewhat maintained its performance regardless of the features provided, the most discernible change was an increase in the average distribution when it comes to the sinusoidal features.

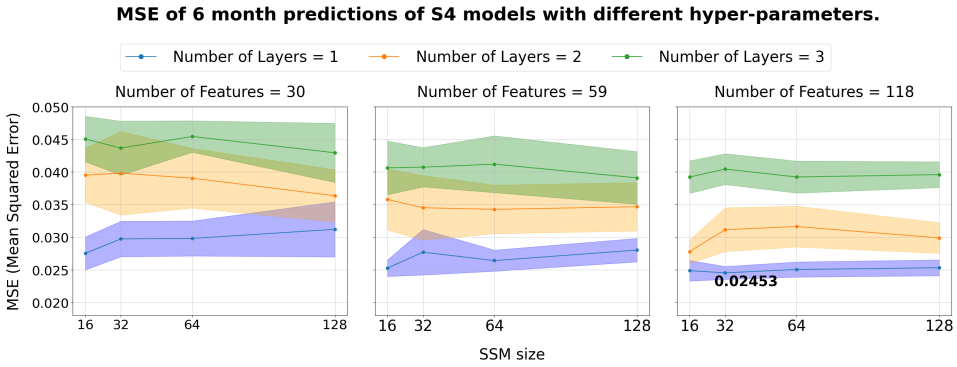

Figure 4: Hyper-parameter adjustment of the S4 model. In the x-axis we have the variation of the SSM size, N, the windows present the variations on the number of features, H, and the different colored lines encode the number of layers, n_{wl}. Each combination of these hyper-parameters was trained 10 times, the average of these iteration is displayed in solid line and dots, and the standard deviation above and below the average is shaded in. The performance of these iterations is measured in the y-axis using MSE as a metric. The MSE was computed from 6-month long forecasts.

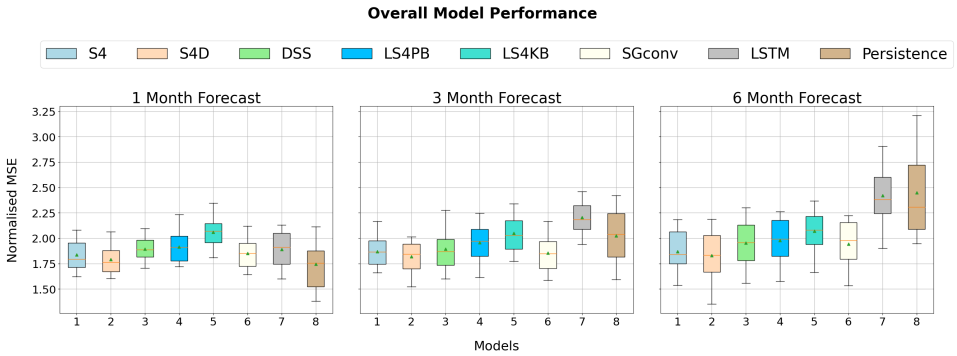

Figure 5: Boxplots of the normalised MSE of the 1, 2 and 3 month predictions for all the models. The values plotted are the average performance of each sample in the test set, meaning I averaged across States and features. Each boxplot encompasses six measurements, the bottom and top of the box are the first and third quartiles, respectively, the whiskers are the 5 and 95 percentiles, the orange line is the median and the green triangle is the average.

5. Conclusions and Future Work

This work set out to find if applying state of the art machine learning models to a mortality forecasting task would be an improvement over classical methods. We compared 6 state of the art models with the LSTM and the Persistence model as baselines. The results showed that the performance between the state of the art models was rather similar. The baseline models' performance was comparable with the state of the art in the 1 month forecast, the Persistence model even managed to outperform all the other models in this task, however, the baselines were subpar when we extended the forecasting window to 3 and 6 months, Figure (5). So, introducing models, such as the S4 and the S4D, in this context may prove to be beneficial, specially in longer forecasts.

We also observed the models performance on two sets of CODs, the periodical and the non-periodical. We have seen that some models are better suited than other for dealing with the non-periodical features, such is the case with the S4D, the same is true for the periodical features, here the S4 has the advantage. However, every model performed better in the periodical features rather than the non-periodical ones. The models were also tested with two additional sets of features, the sinusoidal features and the State features, the sinusoidal features proved to either improve or not significantly affect performance, while the State features mainly worsened the models performance. This might indicate that the models value temporally sensitive information more rather then location information.

5.1. Future Work

There are several improvements and additional work that can be performed as a continuation of this work. We have seen that the models tested have different performances according to the length of the forecast and the periodicity of the features. For example, the LSTM was very proficient in the 1 month forecast with the periodic features, while the S4 was better in the longer forecasts also with the periodic features. One way to take advantage of this difference in performance would be to use an ensemble architecture. We could use an weighted average ensemble, like kaushik et al. [12], where the final prediction is a weighted average over each individual prediction of all models, the weights of each prediction would be dependent on the models performance on that particular feature and forecast length. The problem now is to find the adequate weights, we could first train each model on a given training set, then, with the models' parameters locked in and using a different training set, we can

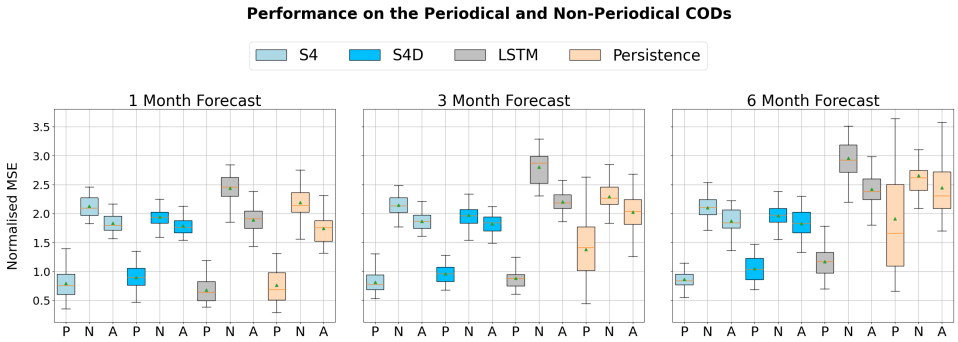

Figure 6: These graphs present boxplots of the normalised MSE of the 1, 2 and 3 month predictions for the S4, S4D, LSTM and Persistence models, the values plotted are the average of each of the samples in the test set across the States and the features. On the x axis we vary the category of features that were used, the periodical features, P, the non-periodical features, N, and all the features, A. Each boxplot encompasses six measurements, the bottom and top of the box are the first and third quartiles, respectively, the whiskers are the 5 and 95 percentiles, the orange line is the median and the green triangle is the average. Note that the best performing model changes according to both the category of the features and the length of the forecast.

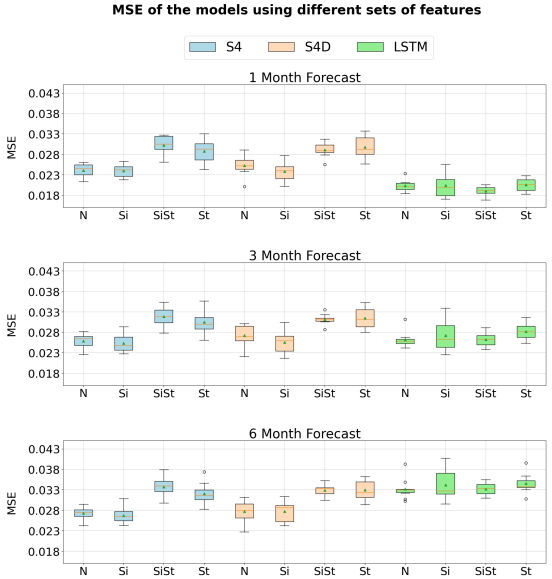

Figure 7: Results from the feature variation experiment. Each model was trained 10 times with each combination of features, displayed in the x-axis, no additional features, N, only the sinusoidal features, Si, both the sinusoidal and State features, SiSt, and only the State features, St. For each iteration I collected the average MSE of the 6 month forecast, the results across the iterations are organised into boxplots, the top and bottom of the box represent the first and third quartile, respectively, the orange line is the median and the whiskers are represent the quartiles plus or minus the inter-quartile range.

train the weights. This way, the weights would be adjusted to give the best combined forecast, emphasising a given model according to the features and forecast length where it performs the best. Another approach would be instead of leaning into the models innate preferences, try and correct them in order to obtain an overall better performing model. This could be done by using a specialised loss function, the loss function I used, the MSE, gives every single prediction the same importance, a specialised loss function would give more importance to features and forecast lengths where the model typically performs worse. For example, the S4D has a hard time with short forecasts, then the loss function would give more importance to the short forecast in an effort to improve them. Now the problem lies in how much importance we give to each forecast/feature, a way to deal with this would be through a grid search, although this may prove to be very time and resource consuming.

Another approach is to introduce additional sets of features, like Piccialli et al. [21] and Mathonsi et al. [18]. These additional features, like air quality and meteorological data, would provide the models with information that might affect the CODs. For example, if we know that low temperatures increase the cases of respiratory illnesses, then if the model sees low temperatures then it may predict higher death rates for respiratory illnesses than it would without this feature. These kinds of features present a challenge, low temperatures do not cause immediate increase in in the number of deaths by respiratory illnesses, their effect may be delayed by some time. In order to solve this issues we could perform some feature analysis, perhaps using causality and correlation analysis, like a Granger causality test [23] to determine if the additional features Granger-causes any given COD behaviour and with what lag. This way we find out which CODs are most affected by these features as well as the lag that these features have with those CODs. Having that, we provide the models with the normal CODs and the lagged versions of the additional features, according to the correlation analysis. There may be the issue where different CODs have different lag values for an additional feature, this may be solved by providing the same additional feature but with different lag values. And as we are not interested in the prediction of these features we can remove them from the training loss, like we did with the sinusoidal features.

6. Acknowledgements

I would like to thank my coordinating professor, Professor Bruno Martins, along with the INESC-ID institution that enabled the creation of this thesis in the context of my Masters degree in Biomedical engeneering.

References

[1] G. Box and G. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day series in time series analysis and digital processing. Holden-Day, 1970.

[2] Centers for Disease Control and Prevention, National Center for Health Statistics. Underlying cause of death 1999-2020, 2021.

[3] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.

[4] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Re. Hippo: Recurrent memory with optimal polynomial projections, 2020.

[5] A. Gu, K. Goel, and C. Ré. Efficiently modeling long sequences with structured state spaces, 2021.

[6] A. Gu, A. Gupta, K. Goel, and C. Ré. On the parameterization and initialization of diagonal state space models, 2022.

[7] A. Gu, I. Johnson, K. Goel, K. Saab, T. Dao, A. Rudra, and C. Ré. Combining recurrent, convolutional, and continuous-time models with linear state-space layers, 2021.

[8] A. Gupta, A. Gu, and J. Berant. Diagonal state spaces are as effective as structured state spaces, 2022.

[9] R. Hasani, M. Lechner, T.-H. Wang, M. Chahine, A. Amini, and D. Rus. Liquid structural state-space models, 2022.

[10] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[11] D. Kalita. A brief overview of recurrent neural networks (rnn). Technical report, 2023.

[12] S. Kaushik, A. Choudhury, P. K. Sheron, N. Dasgupta, S. Natarajan, L. A. Pickett, and V. Dutt. Ai in healthcare: Time-series forecasting using statistical, neural, and ensemble architectures. *Frontiers in Big Data*, 3, 2020.

[13] V. Kotu and B. Deshpande. Chapter 12 - time series forecasting. In V. Kotu and B. Deshpande, editors, *Data Science (Second Edition)*, pages 395–445. Morgan Kaufmann, second edition edition, 2019.

[14] N. Kumar and S. Susan. Covid-19 pandemic prediction using time series forecasting models. In *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–7, 2020.

[15] R. D. Lee and L. R. Carter. Modeling and forecasting u. s. mortality. *Journal of the American Statistical Association*, 87(419):659–671, 1992.

[16] Y. Li, T. Cai, Y. Zhang, D. Chen, and D. Dey. What makes convolutional models great on long sequence modeling?, 2022.

[17] S. Makridakis, E. Spiliotis, and V. Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020. M4 Competition.

[18] T. Mathonsi and T. L. van Zyl. A statistics and deep learning hybrid method for multivariate time series forecasting and mortality modeling. *CoRR*, abs/2112.08618, 2021.

[19] C. Olah. Understanding lstm networks. Technical report, 2015.

[20] F. Perla, R. Richman, S. Scognamiglio, and M. V. Wüthrich. Time-series forecasting of mortality rates using deep learning. *Scandinavian Actuarial Journal*, 2021(7):572–598, 2021.

[21] F. Piccialli, F. Giampaolo, E. Prezioso, D. Camacho, and G. Acampora. Artificial intelligence and healthcare: Forecasting of medical bookings through multi-source time-series fusion. *Information Fusion*, 74:1–16, 2021.

[22] D. Salinas, V. Flunkert, and J. Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks, 2017.

[23] A. Shojaie and E. B. Fox. Granger causality: A review and recent advances, 2021.

[24] S. Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85, 2020. M4 Competition.

[25] Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler. Long range arena: A benchmark for efficient transformers, 2020.

[26] P. Welch. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15(2):70–73, 1967.

[27] T. Zhou, Z. Ma, X. wang, Q. Wen, L. Sun, T. Yao, W. Yin, and R. Jin. Film: Frequency improved legendre memory model for long-term time series forecasting, 2022.