2017 IEEE Conference on Control Technology and Applications (CCTA)
August 27-30, 2017. Kohala Coast, Hawai'i, USA

A Deep Learning Framework for Model Reduction of Dynamical Systems

David Hartman, *Student Member, IEEE,* Lalit K. Mestha, *Fellow, IEEE*

Abstract— **A new framework is proposed for model-order reduction in which a neural network is used to learn a reduced model of a dynamical system. As technological, social, and biological models become increasingly complex, it has become necessary to find ways of reducing the number of dynamical states of a system while still maintaining model integrity. The new approach combines ideas from two existing model-order reduction methods: proper orthogonal decomposition (POD) and balanced truncation. The previously mentioned methods reduce the number of state variables by projecting the dynamics onto a linear subspace spanned by principal components of a representative data matrix. Our proposed model-reduction method has the advantage of projecting the dynamics onto a nonlinear space. The proposed method is referred to as feature decomposition," in light of the nonlinear features extracted by way of neural networks. The method is applied to both autonomous and state-space systems. In feature decomposition for state-space systems, empirical Gramians are the representative training set on which we perform the nonlinear feature learning. It is shown that under certain assumptions, feature decomposition applied to state-space systems is equivalent to balanced truncation. Finally, the method is applied to a spreading infectious disease dynamical system and the results from our reduced order model are compared to POD.**

I. Introduction

Designing controls for large scale systems is often difficult because of the large number of states in the model. Model order-reduction has historically been used to reduce the size of large-scale structural and fluid dynamics problem. Proper orthogonal decomposition (POD) or Karhunen-Loeve decomposition and balanced truncation are two such model-order reduction methods. POD is a model-order reduction technique for autonomous systems while balanced truncation is a model-order reduction technique for state-space systems.

In Section III, we review proper orthogonal decomposition, a model reduction method that relies on principal component analysis (PCA). There are two steps to POD. Step 1: Extract a lower dimensioned linear subspace of the dynamics by performing principal component analysis on a representative data matrix. Step 2: Project the dynamics onto the reduced subspace thereby attaining a reduced model.

In Section III, we extend POD by replacing step 1 with the nonlinear feature learning of a certain type of neural network called the autoencoder. In other words, instead of extracting a linear space through PCA, we extract a nonlinear embedded space through an autoencoder. An autoencoder is a neural network that can discover a reduced dimensioned representation of data and is reviewed in Section II. While it has been shown that PCA and linear autoencoders are equivalent [1], the success of deep leaning is attributed to the use of non-linear, deep-layered autoencoders. Instead of extracting a reduced, linear space as in POD, we now extract a reduced, nonlinear space referred to as the feature space. Step 2 is carried out as before, but now the dynamics are projected onto the feature space. Considering this, we call this new technique "feature decomposition."

In Section IV, we discuss feature decomposition for state-space systems. We first review the balanced truncation method for model reduction of state-space systems. Unlike in autonomous systems, in state-space systems, we must ensure that the controllable and observable spaces of the dynamics are represented in the reduced model. Balanced truncation takes this into account by projecting the dynamics onto the "modes" of a system that have the most impact on input-output behavior (Hankel modes) [2]. However, if the system is symmetric, the method of balanced truncation can be replaced with the direct truncation method and Hankel modes do not need to be considered. In direct truncation, the dynamics are projected onto the principal components of the cross Gramian, a matrix that incorporates both the controllable and observable spaces. As this is a data-driven technique, the empirical version of the cross Gramian will be used instead [3]. It is not surprising that balanced truncation and direct truncation are equivalent under the symmetric assumption as both techniques account for the controllable and observable spaces, albeit in different ways (balanced truncation through the use of Hankel modes and direct truncation through the use of cross Gramians).

Assuming a symmetric system and therefore the equivalence of balanced and direct truncation we have the proper setup to perform feature decomposition. We replace PCA of the cross Gramian in the method of direct truncation with feature learning of the empirical cross Gramian in the method of feature decomposition. In other words, the linear subspace extracted by performing PCA on the cross Gramian (direct truncation) is replaced by the nonlinear feature space extracted by training on the empirical cross Gramian (feature decomposition).

In fact, if the state-space system is linear and symmetric and a linear autoencoder is used instead of a nonlinear autoencoder, we can say even further that the extracted spaces from balanced truncation and feature decomposition (principal component subspace and feature space) are the same and linear. A proof sketch will be given for this in Section IV. This is our justification for replacing balanced truncation with our proposed feature decomposition

David Hartman, is with the Electrical Engineering Department, University of Maryland, College Park, College Park, MD, 20742 USA (corresponding author email: dhartma2@terpmail.umd.edu and phone: 917 815 9990)

Lalit K. Mestha is with the GE Global Research, Niskayuna, NY 12309 USA, (e-mail: Lalit.Mestha@ge.com and phone: 518 387 6967)

framework. Accounting for the recent advances in deep learning, we believe there are benefits to using nonlinear autoencoders in finding a nonlinear lower-dimensioned embedded space in which the states of the nonlinear dynamics lie. PCA and linear autoencoders, on the other hand, are constrained to extracting linear embedded spaces, an unfavorable restriction for nonlinear dynamics.

In Section V, we show an example of feature decomposition applied to a spreading infectious disease example. The nonlinear dynamical system of disease spread can be used to model the spread of infections, the spread of viruses in a computer network, and spread of rumors in social media.

In summary, the goal of this paper is to create a framework for learning the features of a dynamical system and then project onto this learnt space creating a reduced model. We show that our framework for both autonomous and state-space systems is nearly equivalent to traditional model-order reduction techniques: POD for autonomous systems and balanced truncation for state-space systems. The difference is that both these techniques rely on PCA while our framework relies on nonlinear autoencoders.

II. AUTOENCODERS

Autoencoders are a feedforward artificial neural network algorithm that can be used for dimensionality reduction [4]. Given a set of inputs and corresponding outputs, a neural network approximates a function between the input and output space. In an autoencoder, the inputs and outputs are the same and a function is estimated that reconstructs the original input. The reconstruction of training data occurs by way of nonlinear transformations. As a byproduct of this reconstruction, features of the data are extracted. Generally, the more training data employed and the more characteristic the data is of the underlying distribution of the problem, the more representative the features will be.

Autoencoders and neural networks have become popular in this decade as optimization algorithms used to solve them improve. Autoencoders are often an improved alternative to PCA as they are capable of learning non-linear features as opposed to linear principal components. The autoencoder algorithm as a function estimator for input reconstruction consists of two steps: the encoder and decoder step. The encoder encodes the input (x) into a hidden or feature representation (y) and then decodes that representation back into a reconstructed input ($\tilde{x}$).

A. Encoder Step

In the encoder step, a mapping (f_θ) transforms the original input (x) into the feature space through an affine transformation ($Wx + b$) followed by a non-linear transformation, s. The encoder mapping is as follows:

$$f_\theta(x) = s(Wx + b)$$

S is the sigmoid function defined as: $s(\tau) = \frac{1}{1+e^{-\tau}}$, where τ in this case equals $Wx + b$. The sigmoid function is applied element-wise to its column vector argument. W is a $r \; x \; n$ weight and b is a vector of dimension r. The value of r is specified by the user. Autoencoders can be viewed as a dimension reduction scheme if r is chosen to be less than n. Additionally, let the parameter θ equal $\{W, b\}$ and the output of $f_\theta(x)$ be the hidden representation, y.

B. Decoder Step

In the decoder step, the hidden representation y is transformed back into the original input space. The mapping for the decoder reconstructs the original input from the hidden representation. The decoder mapping is as follows:

$$g_{\theta'}(y) = s(W'y + b')$$

W' is a $n \; x \; r$ weight matrix and b' is a vector of dimension n. Let θ' equal $\{W', b'\}$ and $g_{\theta'}(y)$ be the reconstructed input, $\tilde{x}$. Note, $W' \; and \; b'$ are not the transposes of W and b (W^T and b^T). However, in some settings, one would set W' to W^T and have one less variable to optimize over. This is called tied weights.

When the size of the hidden or feature representation (r) is less than the size of the original input (n), the encoder mapping is a dimension-reduction mapping. The optimization problem associated with the autoencoder is shown below:

$$E(\Theta, \Theta') = \min \sum_{i=1}^{N} ||x_i - s(W'(s(Wx_i + b)) + b')||_2^2$$

The term inside the summation equals the error between the original input (x) and the reconstructed input ($\tilde{x}$). The optimization occurs over N training points.

The autoencoder setup above is only one type amongst many variants including the stacked denoised autoencoder, where multiple encoding and decoding layers are stacked one on top of the other. Once, the variables Θ and Θ' (W, W', b, b') are learnt by training on N data points $[x_1, \dots\dots x_N]$ one can encode a new data point into its dimensionality reduced hidden representation.

There is a deep connection between autoencoders and PCA. Baldi and Hornik [1] showed that the r principal components found from PCA equal the r learnt features of the autoencoder under the following assumptions: 1. Only a linear encoder and decoder are used without any nonlinear activation function 2. The error loss function is the squared error loss. 3. Tied weights are used. In other words, the optimization problem associated with linear autoencoders has a minimum equal to the orthogonal projection onto r principal components of PCA.

III. MODEL REDUCTION FOR AUTONOMOUS SYSTEMS

A. Proper Orthogonal Decomposition

The goal of proper orthogonal decomposition is to reduce the dimension of the states of the following autonomous dynamical system:

$$\dot{x}(t) = f(x(\text{t})) \qquad (1)$$

The method makes use of empirical data X = $[x(t_1),\dots,x(t_N)]$ to create a data matrix. The data is chosen using the method of snapshots proposed by Sirovich [5]. Proper orthogonal decomposition consists of two steps: 1. PCA is applied to the covariance data matrix (XX^T). 2. Using Galerkin projection, the dynamical equations are projected onto the subspace spanned by the principal components. We first start with a review of PCA.

B. Principal Component Analysis

Define: Empirical Covariance matrix, $\tilde{R} = XX^T$ where $\tilde{R}$ is n x N matrix, N is the number of data points, and n is the dimension of each data point. One defines this matrix as follows:

$$\tilde{R} = \sum_{j=1}^{N} x(t_j)x^T(t_j)$$

The subspace of the principal components of $\tilde{R}$ can be characterized by the projection function π_r. This can be formulated as minimizing E.

$$E(\pi_r) = \sum_{i=1}^{N} ||x_i - \pi_r x_i||_2^2$$

Where π_r is constrained to a subspace of dimension r.

Lemma 1: The minimizer of the above optimization problem, π_r^* equals $P_r{}^T P_r$ with P_r equal to a $r \; x \; n$ matrix whose r rows are equal to the r eigenvectors of the covariance matrix $\tilde{R}$ [6].

Additionally, the eigenvalues of $\tilde{R}$ show how close of an approximation the data is to the r dimensional subspace

Lemma 2: The minimum of the solution, $E(\pi_r{}^*)$ equals $\sum_{k=n-r+1}^{n} \lambda_k$, where λ_i's are eigenvalues of $\tilde{R}$ and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ [7].

C. Galerkin Projection

In POD, after the principal components are extracted from the dynamical system, the dynamics are projected onto the subspace spanned by the r largest principal components of the covariance matrix $\tilde{R}$. This projection is called the Galerkin projection. The Galerkin projection is a method of projecting a vector field onto a subspace. It has been used extensively in numerical analysis of PDEs [8]. Applying the Galerkin projection in POD to (1), the reduced model becomes:

$$\dot{x}_r(t) = P_r f(P_r^T x_r(t)) \tag{2}$$

Note, x_r is used instead of x, where x is in R^n and x_r is in R^r and $r \ll n$. Also note that while this method can be applied to both linear and non-linear equations, it suffers from the deficiency of only being able to project the vector field trajectory onto a linear subspace. P_r is described in lemma 1. The next method learns from empirical data a non-linear projection that better captures the space on which the dynamical data points reside.

D. Feature Decomposition for Autonomous Systems

The proposed substitution for proper orthogonal decomposition replaces PCA of POD with an autoencoder. Using the learnt parameters of the autoencoder, W, W', b, b', we perform a Galerkin projection on the autonomous dynamical system shown in (1) producing the following system:

$$\dot{x}_r(t) = s(W(f(s(W'x_r(t) + b'))) + b) \tag{3}$$

Equation (3) is similar to (2) with P_r replaced by $s(Wx + b)$ and P_r^T replaced by $s(W'x + b')$. Remember, s is the sigmoid function and operates element-wise on a column vector.

IV. MODEL REDUCTION FOR STATE-SPACE SYSTEMS

In this section, we extend feature decomposition of autonomous systems to state-space systems. As we turn to state-space systems, we notice that instead of one space, we are now concerned with two spaces: input to state (controllable space) and state to output (observable subspace). The state-space system is as follows:

$$\dot{x}(t) = f(x(t), u(t))$$
$$y(t) = h(x(t)) \tag{4}$$

While the method of snapshots was sufficient to capture the dynamics of an autonomous system, we need an improved data matrix that captures both the controllable and observable spaces of a state-space system.

The next method uses the empirical cross Gramian, an approximate version of the cross Gramian [3], to capture the controllable and observable spaces of a system. An autoencoder learns the parameters, W, W', b, b', by training on the empirical cross Gramian and then a Galerkin projection is performed using these parameters. In autonomous systems, the data matrix on which we train is created by the method of snapshots; in state-space systems the data matrix is created by the empirical cross Gramian, Before, we discuss feature decomposition for state space systems, we introduce balanced truncation, the traditional model-order reduction technique for state-space systems.

A. Balanced Truncation

Balanced truncation is a model reduction method that ensures reduced systems retain properties of controllability and observability. In these methods, the least controllable and observable states are truncated. To understand balanced truncation, we introduce the Hankel operator. The Hankel operator is given by map H: O·C, where OO^T equals the observability Gramian (W_o) and CC^T equals the controllability Gramian (W_c). This holds for square systems where the dimensions of inputs and outputs are equal. The controllability and observability Gramians are a measure of degrees in which a system is controllable and observable. A system is observable if the states of a system can be recovered from its outputs; a system is controllable if there exists a controller to move a state from any initial state to any final state.

In balanced truncation, the projected subspace is a set of the largest Hankel singular values, σ(H).

The reduced-order equations for balanced truncation are as follows [9]:

$$\dot{x}_r(t) = PTf(T^{-1}P^T x_r(t), u(t))$$
$$y_r(t) = h(T^{-1}P^T x_r(t)) \tag{5}$$

Where T is a change of coordinates such that TW_cT^T and $T^{-T}W_OT^{-1}$ are equal and diagonal (balanced). W_c is the controllability Gramian and W_O is the observability Gramian. Additionally, P is a $r \; x \; n$ projection matrix in the form of $[I_r \; 0]$, where I_r is an r x r identity matrix. In other words, in balanced truncation we project the dynamics onto the r largest Hankel singular values, σ(H).

B. *Direct Truncation*

As explained in the introduction, if the system is also symmetric, the balanced truncation method can be replaced with direct truncation. In direct truncation, the principal components of the cross Gramian (W_X), defined as C·O are truncated. The cross Gramian captures both controllable and observable properties of a system. For linear state space systems, [$\dot{x}(t) = Ax(t) + Bu(t);\ y(t) = Cx(t)$], the cross Gramian is defined as follows:

$$W_X = \int_0^T e^{At} BC e^{At} dt$$

Theorem 1: If (A,B,C,D) is a symmetric system, then projecting onto the set of Hankel singular values (balanced truncation) is the same as projecting onto the eigenspace of the cross Gramian (direct truncation) [10].

Proof: In a symmetric system, the Hankel operator is symmetric or, in other words, O·C = $(O \cdot C)^T$. Therefore, σ(H) = σ(O·C) = $\sqrt{\lambda(\mathrm{OC}(OC)^T)} = \sqrt{\lambda(\mathrm{OC}(OC)^T)} = \sqrt{\lambda(\mathrm{COCO})} = \sqrt{\lambda(W_X W_X)} = |\lambda(W_X)|$ [3].

C. *Definition of Empirical Cross Gramian*

Himpie defines an empirical version of the cross Gramian for square, symmetric systems where input and output spaces are dimensioned M and the state variables are dimensioned N. The empirical cross Gramian is a data matrix populated by the state and output of the system perturbed by different inputs and initial states [3]. It is an analog to Lall's [2] empirical controllability and observability Gramians. We show Himpie's full definition below.

Define: Empirical Cross Gramian (W_X)

$$W_X = \frac{1}{KLM} \sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{m=1}^{M} \frac{1}{c_k d_l} \sum_{t=0}^{T} \varphi^{klm}(t)$$

Where:

e_m = standard unit vector in $\boldsymbol{R}^M$ for $1 \leq m \leq M$

e_j = standard unit vector in $\boldsymbol{R}^N$ for $1 \leq j \leq N$

$\varphi_{ij}^{klm}(t) = x_i^{km}(t) y_m^{lj}(t)$

$x_i^{km}(t)$ is the *i*-th component of the state of the system for impulsive input $u^{km}(t) = c_k e_m \delta(t)$ and zero initial state.

$y_m^{lj}(t)$ is the *m*-th component of the output of the state with initial state $x_o^{lj} = d_l e_j$ and zero input.

The empirical cross Gramian has K·M state trajectories for perturbed impulse input with zero steady state, and L·N output trajectories for initial states with zero input. The sets $\{c_k: k = 1 \ldots K\}$ and $\{d_l: l = 1 \ldots L\}$ are chosen based on the operating region of the underlying system. M is the dimension of input and output and is indexed by m. In short, the empirical cross Gramian is an inner product between state trajectories with perturbed input and output trajectories with perturbed initial state [3].

E. *Equivalence of Balanced Truncation and Direct Truncation*

Here, we will give the result for the near equivalence of balanced truncation and direct truncation. First, we state a lemma on the equivalence of the empirical cross Gramian to the cross Gramian for linear systems..

Lemma 3: For nonempty sets of c_k and d_l, the empirical cross Gramian, is equal to the cross Gramian for linear state space systems.

See [3] for proof.

Result: Assuming a square, symmetric, linear system, balanced truncation is equivalent to projecting the dynamical system onto the eigenspace of the empirical cross Gramian.

By the symmetric property of the system, one can replace balanced truncation with a projection of the dynamics onto the eigenspace of the cross Gramian (Theorem 1). By the linearity assumption the empirical cross Gramian equals the cross Gramian (Lemma 3).

Looking ahead, we will replace the principal component analysis of the empirical cross Gramian with an autoencoder just as we replaced the principal component analysis of the covariance matrix ($\tilde{R}$) with an autoencoder in the autonomous case. In Part E, we compare the different training sets used for finding the lower-dimensioned embedded spaces in the autonomous and state-space systems.

F. *Comparison of Covariance and Cross Gramian*

Below, we compare using the empirical covariance matrix in the method of snapshots for the autonomous system and the empirical cross Gramian for the state-space system. In both, we highlight the data matrix for which PCA was applied. Subsequently, in our framework, PCA is replaced by nonlinear autoencoders, which train on these same data matrices.

Summary of Empirical Covariance Matrix ($\tilde{\mathrm{R}}$) (autonomous case**):**

- **i.** X = [$x(t_1) \ldots, x(t_j), \ldots x(t_T)$], where $x(t_j)$ is the state trajectory at time t_j of a dynamical system. **[Data Matrix]**
- **ii.** $\tilde{R} = \sum_{j=1}^{T} x(t_j)\, x(t_j)^T$ where T is the number of snapshots and $\tilde{R}$ is the empirical covariance. **[Gramian]**
- **iii.** Principal components of X = eigenvectors of $\tilde{R} = XX^T$. **[PCA]**

Summary of Empirical Cross-Gramian (W_X) (state-space case): Note: To simplify, we take $K = L = c_k = d_l = 1$ in the empirical cross Gramian definition and are left with $W_X = \frac{1}{M} \sum_{m=1}^{M} \sum_{t=0}^{T} \varphi^m(t)$.

- i. X = [$\varphi^1(t_1) \ldots \varphi^1(t_T)$, …….$\varphi^m(t_1) \ldots \varphi^m(t_T)$, ……., $\varphi^M(t_1) \ldots \varphi^M(t_T)$]. $\varphi^m(t)$ is a matrix of size $R^{N \times M}$ where M is the size of input and output system, N is the dimension of the states and T is the number of snapshots. So X is of size N x M^2T. **[Data Matrix]**
- ii. $\tilde{W}_X \tilde{W}_X^T = \sum_{m=1}^{M} \sum_{a=1}^{T} \varphi^m(t_a) \varphi^m(t_a)^{Tr}$ where T is the number of snapshots and $\tilde{W}_X \tilde{W}_X^T$ is the empirical cross Gramian. **[Gramian]**

iii. Principal components of X = eigenvectors of $\widetilde{W}_X\widetilde{W}_X^T$. **[PCA]**

G. Feature Decomposition for State-Space Systems

Using the empirical cross Gramian described above, the autoencoder algorithm now learns the controllable and observable space on which to project the state-space equations. The Galerkin projection is then performed by projecting the state space equations onto the features learnt by the autoencoder.

The reduced state-space system then becomes:

$$\dot{x}_r(t) = s(W(f(s(W'x_r(t) + b'), u(t))) + b)$$
$$y_r(t) = h(s(W'x_r(t) + b')) \tag{6}$$

Where W, W', b, and b' are learnt variables from training the autoencoder on the empirical cross Gramian and s is the sigmoid function. Compare this to (5), the balanced truncation equations. Equation (6) is similar to (3), the reduced model for autonomous systems. However, remember that the parameters W, W', b, b' in both these equations are learnt by training on different training sets, namely the method of snapshots matrix for (3) and the empirical cross Gramian for (6).

H. Summary of Replacing Balanced Truncation with Feature Decomposition

Below, we outline the algorithmic steps for replacing the model-order reduction technique of balanced truncation with feature decomposition of the state-space systems. The steps are as follows:

1: Replace balanced truncation with direct truncation of cross Gamian under symmetric assumption (Theorem 1)

2: Approximate cross Gramian by empirical cross Gramian (Lemma 3)

3: Train auotencoder on empirical cross Gramian with a hidden representation of size r.

4: Project the original dynamics onto the learnt feature space (Galerkin projection) to obtain the following reduced model:

$$\dot{x}_r(t) = s(W(f(s(W'x_r(t) + b'), u(t))) + b)$$
$$y_r(t) = h(s(W'x_r(t) + b'))$$

V. Simulation

In this section, we highlight the performance of feature decomposition of an autonomous system on an infectious disease model. The recent outbreak of Ebola and Zika virus have necessitated finding more accurate models for the spread of infectious diseases. One such model is the SIS (Susceptible/Infected/Susceptible) epidemic model, which characterizes individuals as residing in one of two compartments: "Susceptible" and "Infected." Susceptible individuals are healthy individuals who can transition from "susceptible" to "infected" with an infection rate β. Infected individuals are individuals harboring the disease who can transition from "infected" to "susceptible" with a recovery rate δ.

Furthermore, since different individuals recover and infect their neighbors at different rates, a networked version of the epidemic model was proposed and is described in [11]. In an SIS epidemic network autonomous model, we consider a directed graph of N nodes corresponding to N individuals and the associated connecting edges. The state variables of this networked dynamical system are the probabilities ($p_i(t)$) node i is infected where $1 \leq i \leq N$. Let,

$p_i(t)$ equal the probability node i is infected at time t,
$\delta_i(t)$ equal the recovery rate of node i,
$\beta_{ij}(t)$ equal the rate at which node i is infected by node j.

The SIS network model, therefore, consists of N nonlinear differential equations corresponding to N nodes (one such node is shown in fig. 1) and is described below in (7) [11].

$$\dot{p_i} = (1 - p_i)\sum_{j=1}^{N} \beta_{ij}p_j - \delta_i p_i \quad for\ i = 1\ to\ N \tag{7}$$

The matrix version of (7) is (8):

$$\dot{p} = (B - D)p - PBp \tag{8}$$

Where P is a matrix whose diagonal entries are $(p_1, \dots, p_n)$,

p is a row vector whose entries are $(p_1, \dots, p_n)$,

B is a matrix whose entries are β_{ij} and D is a matrix whose diagonal entries are $(\delta_1, \dots, \delta_n)$.

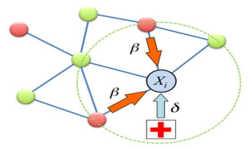

Figure 1. Example of a single node in SIS infectious disease graph with infection rate β and recovery rate δ.

Courtesy of Nowzari, et al. "Analysis and control of epidemics: A survey of spreading processes on complex networks."

In MATLAB, the SIS network model is modelled with 30 individuals. This is an autonomous system as there are no control inputs. Both the POD and the proposed feature decomposition algorithm for autonomous systems are applied to this problem to reduce the state variables p(t) in R^{30} are to $p_r(t)$ in R^3. The original dynamics in R^{30} is simulated for 100 time steps. This data is then accumulated into a data matrix of size $R^{30\,x\,100}$. Both PCA and an autoencoder are run on the data, each extracting 3 "features". The dynamical equations are projected onto the principal components as in POD as well as the nonlinear feature space learnt from the autoencoder. We now have two separate parameters for reducing the dynamics: POD reduction and feature decomposition.

Using the parameters from these two reduction techniques, the initial conditions from the original problem in R^{30} are reduced to 3 state variables and then simulated for 100 time steps using the reduced order equations. Once the reduced model results (R^3) are found for 100 time steps, they are decoded (reconstructed) back to R^{30} as in the decode step of the autoencoder (see autoencoder section).

Three plots are shown below: 1. original dynamics (fig. 2) 2. the reconstructed output of POD (fig. 3) 3. the reconstructed output of our proposed framework (fig. 4). Figure 3 is obtained by first reducing the original initial state of the model from R^{30} to R^3 using POD (2). *2*. Propagating the initial state via the reduced dynamics and retrieving a matrix in $R^{3\,x\,100}$, where the columns are the reduced state variables for each of the propagated 100 time steps. 3. Decoding or reconstructing these 100 data points in R^3 back to R^{30} to get a matrix of size, $R^{30\,x\,100}$. These data points are referred to as the reconstructed POD data points. Figure 4 is obtained in a similar fashion, but replace step 2 by reducing the dynamics via the feature decomposition framework, instead of POD. Note the improvement of figure 4 over figure 3 as an approximation of figure 2.

To obtain a metric of success, a mean square error is computed between the reconstructed data points (POD and Feature Decomposition) and the original propagated data points. Averaged over five runs, the mean squared error for the POD-based algorithm is 0.1667 and the mean squared error for the autoencoder-based algorithm is better at 0.0873. The experiments were repeated for a reduction from 100 to 3 states and a reduction from 20 to 3 states. For a reduction from 100 to 3 states, averaged over five runs, the mean squared error for the POD-based algorithm is 0.5870 and the mean squared error for the autoencoder-based algorithm is better at 0.3075. For a reduction from 20 to 3 states, the mean squared error for the POD-based algorithm is 0.0677 and the mean squared error for the autoencoder-based algorithm is better at 0.0570.

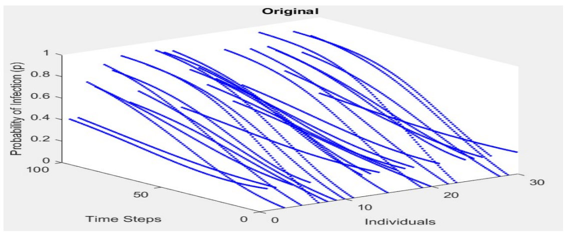

Figure 2. 100 step simulation of original dynamics of order 30

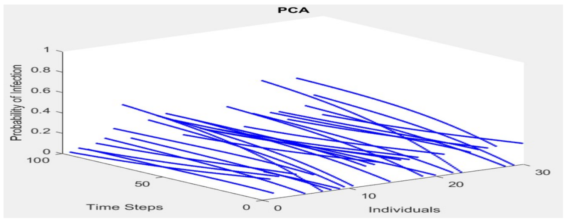

Figure 3. 100 step simulation of reconstructed (order 30) output with reduced (order 3) dynamics via POD

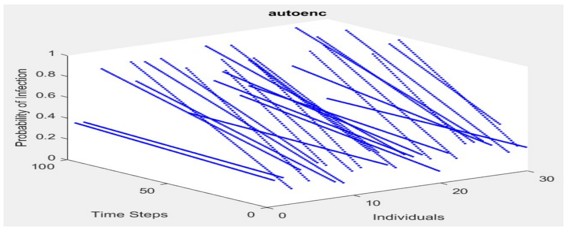

Figure 4. 100 step simulation of reconstructed (order 30) output with reduced (order 3) dynamics via Feature Decomposition

VI. Conclusion

The paper shows a framework for model-order reduction that taps into the power of deep learning. Many model-order reduction techniques at their heart rely on linear-based principal component analysis [12], a less than ideal technique for reducing nonlinear systems. The feature decomposition framework allows for extracting a nonlinear feature space on which to project the dynamics. Furthermore, for state-space model reduction, we showed that training on the empirical cross Gramian data set will give results similar to balanced truncation. Additionally, the example of the infectious disease model showed that the feature decomposition method can outperform the traditional PCA-based model-order reduction technique.

While bounds exist for proper orthogonal decomposition and balanced truncation, currently no such bounds exist for our framework due to the non-convexity of the optimization problem associated with the autoencoder. Future work is needed to find bounds for this non-convex problem. On a final note, the feature decomposition framework described above extracts features using only a single hidden layer in the autoencoder. More powerful features can be extracted by using multiple hidden layers in the stacked version of the autoencoder.

References

[1] P. Baldi, K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *in Neural networks* 2.1, 1989, pp. 53-58.

[2] S. Lall, J.E. Marsden, S. Galavski, “Empirical Model Reduction of Controlled Nonlinear Systems,” *in Proceedings of IFAC World Congress,* 1999, pp. 473-478.

[3] C. Himpie, M. Ohlberger, “Cross-Gramian-Based Combined State and Parameter Reduction for Large-Scale Control Systems,” *in Mathematical Problems in Engineering*, 2014.

[4] P. Vincent, et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *in Journal of Machine Learning Research*, 2010, pp. 3371-3408.

[5] L. Sirovich, "Turbulence and the dynamics of coherent structures. I. Coherent structures," *in Quarterly of applied mathematics* 45.3, 1987, pp. 561-571.

[6] C.W. Rowley, “Model Reductions for Fluids Using Balanced Proper Orthogonal Decomposition,” *in International Journal of Bifurcation and Chaos*, 2005, pp. 997-1013.

[7]S. Volkwein, "Model reduction using proper orthogonal decomposition," *in Lecture Notes, Institute of Mathematics and Scientific Computing, University of Graz, 201.*

[8] M. Wang, et al., "Deep Learning-Based Model Reduction for Distributed Parameter Systems," in *IEEE Transactions on Systems, Man, and Cybernetics, Systems,* 2016, pp.1664-1674.

[9] K. Willcox, J. Peraire, “Balanced Model Reduction via the Proper Orthogonal Decomposition,” *in AIAA Journal*, vol. 40.11, 2002, pp. 2323-2330.

[10] U. Baur, P. Benner, “Cross-Gramian-Based Model Reduction for Data Sparse Systems*,” in Electronic Transactions on Numerical Analysis*, 2008, pp. 256-270.

[11] C. Nowzari, Cameron, et al., "Analysis and control of epidemics: A survey of spreading processes on complex networks," *in IEEE Control Systems*, 2016, pp. 26-46.

[12] M. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," in *AIChE journal*, 1991, pp. 233-243.