# Modeling Global Dynamics from Local Snapshots with Deep Generative Neural Networks

Scott Gigante [1 †], David van Dijk [2 3 †], Kevin R. Moon[4],
Alexander Strzalkowski[5], Guy Wolf [6 ††] and Smita Krishnaswamy [2 3 †† §]
[1] Computational Biology and Bioinformatics Program, [2] Department of Genetics,
[3] Department of Computer Science, Yale University, New Haven, CT, USA
[4] Department of Mathematics and Statistics, Utah State University, Logan, UT, USA
[5] Department of Computer Science, Princeton University, Princeton, NJ, USA
[6] Department of Mathematics and Statistics, Université de Montréal, Montéal, QC, Canada

***Abstract*—Complex high dimensional stochastic dynamic systems arise in many applications in the natural sciences and especially biology. However, while these systems are difficult to describe analytically, "snapshot" measurements that sample the output of the system are often available. In order to model the dynamics of such systems given snapshot data, or local transitions, we present a deep neural network framework we call *Dynamics Modeling Network* or *DyMoN*. DyMoN is a neural network framework trained as a deep generative Markov model whose next state is a probability distribution based on the current state. DyMoN is trained using samples of current and next-state pairs, and thus does not require longitudinal measurements. We show the advantage of DyMoN over shallow models such as Kalman filters and hidden Markov models, and other deep models such as recurrent neural networks in its ability to embody the dynamics (which can be studied via perturbation of the neural network) and generate longitudinal hypothetical trajectories. We perform three case studies in which we apply DyMoN to different types of biological systems and extract features of the dynamics in each case by examining the learned model.**

## I. Introduction

It is difficult if not impossible to derive differential equations or analytical models for most natural stochastic dynamic systems. Further, in Biology it is often the case that we only have access to samples or "snapshots" from such dynamic system (e.g., pairs of consecutive points) and not to continuous observations. Here, we propose to learn a generative neural network model of dynamic systems that we call a Dynamics Modeling Network (DyMoN) that is able to learn a dynamic model of a system for which we only have sparse snapshot data, or even a single snapshot showing a multitude of instantiations of the dynamic system rather than one instantiation longitudinally. DyMoN uses a deep neural network architecture to learn fixed-memory stochastic dynamics (e.g., memoryless or $n$-th order Markov process) in an observed system. This network is trained to map a current state (or $n$ states) to a distribution of next states. DyMoN is trained by penalizing the stochastically generated output states based on maximal mean discrepancy (MMD) [1], [2] from known next-states as a probabilistic distance between the desired and generated output distributions.

DyMoN provides several advantages over existing methods, such as HMMs and recurrent neural networks. First, DyMoN provides a **representational** advantage by serving as an embodiment of the dynamics in lieu of a predetermined model, such as stochastic differential equations. This representation is deep and factored, in the sense that the "logic" of the dynamic transition is broken down into increasingly abstract steps, each of which can be visualized or examined. Second, DyMoN provides a **generative** advantage as it generates new trajectories/sequences that have not been seen previously in the system. Third, the utilization of a deep network model offers a **multitasking** advantage: while previous models, such as HMMs, can simulate trajectories, and one may use PCA (or similar methods) to visualize trajectory information, most previous methods are not designed or equipped to *simultaneously* learn multiscale features (i.e., in many levels of abstraction), visualize intrinsic underlying dynamics, and utilize them to generate new trajectories. Finally, the natural parallelizability of neural networks (e.g., with GPU-based implementations) offers **computational** advantages, as they can be used to process large volumes of noisy data. Furthermore, once trained, DyMoN can efficiently generate new trajectories faster than most existing methods.

DyMoN is well suited to model biological data, such as single-cell RNA sequencing data from cellular developmental systems. Such data is often collected at only one or a handful of time-points. Thus the dynamics are either inferred from "pseudotime" (an inferred axis of progression based on observed cells which are at different states of development), or from interpolation between discrete time points. These trajectories are not a matter of sequence completion as is often done in recurrent neural networks; the goal is to predict the near-future state of a cell given its current state. Often history information is not available in biological measurements. As such, we apply DyMoN to several biological systems for which we have access to snapshot samples of the data. These include mass cytometry of developing T cells in the mouse thymus and single-cell RNA sequencing of human embryonic stem cells developing in embryoid bodies. DyMoN is able to learn the

† These authors contributed equally. †† These authors contributed equally.
§ Corresponding author. smita.krishnaswamy@yale.edu
333 Cedar St. New Haven CT 06510 USA

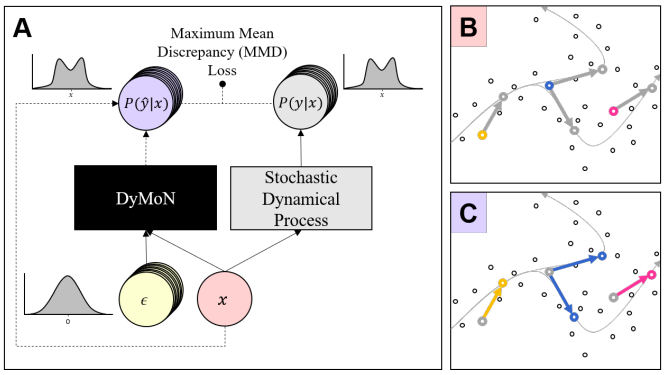


Fig. 1. (**A**) Schema of DyMoN architecture where $x$ represents the neural network input vector, $\hat{y}$ represents the predicted network output in (1), and $\epsilon$ is a random Gaussian noise vector. $P(y|x)$ represents the distribution of outputs from many samples of the stochastic dynamical process given $x$, and $P(\hat{y}|x)$ represents the distribution of outputs of DyMoN given $x$ and many different noise vectors. (**B**) Example input states. (**C**) Learned transition vectors (arrows) and output states from DyMoN.

underlying dynamics of each of these systems, reveal insights into the 'rules' driving these dynamics, as well as sample new trajectories/sequences from these dynamics.

## II. Dynamics Modeling Network (DyMoN)

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a finite dataset of $d$-dimensional states, and let $\mathcal{T} \subseteq \mathcal{X} \times \mathcal{X}$ be a (finite) collection of transitions between these states. Such transitions can either be observed by sampling a dynamical system, or constructed by geometry-revealing diffusion methods such as diffusion maps [3]. We propose to learn the dynamics in $\mathcal{T}$ and the geometry represented by them as a stochastic velocity vector field of a Markov process with a feed-forward neural network we call Dynamics Modeling Network (DyMoN).

DyMoN is formed by a cascade of linear operations and nonlinear activations. These are controlled by optimized network weights, which we collectively denote by $\theta \in \Theta$, where $\Theta$ represents the space of possible weight values determined by the fixed network architecture. To capture the dynamics in $\mathcal{T}$, DyMoN learns a *velocity vector* $\Delta_\theta(x) \in \mathbb{R}^d$ and uses it to define a transition function $T : \mathbb{R}^d \times \Theta \to \mathbb{R}^d$ that generates a Markov process

$$x_t = T(x_{t-1}, \theta) = x_{t-1} + \Delta_\theta(x_{t-1}) \qquad t = 1, 2, 3, \ldots,$$

given an initial state $x_0 \in \mathcal{X}$ and optimized weights $\theta \in \Theta$. Further, to introduce stochasticity, we treat the output $\Delta_\theta(x_{t-1})$ as a random vector whose probability distribution determines the conditional probabilities $P(x_t|x_{t-1})$ of the Markov process.

Given a fixed architecture, we now describe how DyMoN is trained using the data points in $\mathcal{X}$. Consider a source state $x \in \mathcal{X}$ with multiple transitions from it leading to target states $Y_x = \{y : (x, y) \in \mathcal{T}\}$. Let $\mathcal{P}_x(y) = P(y \mid x)$, whose support is $Y_x$, be the conditional probability of transitioning from $x$ to $y$. For ease of notation, we allow repetitions in the set notation of $Y_x$, and assume that data points in $Y_x$ are indeed distributed according to $\mathcal{P}_x(y)$ and are sufficient for estimating $\mathcal{P}_x(y)$. Further, we assume that these transitions are smooth in the sense that if $x$ is similar to $x'$ then $P(y \mid x)$ is similar to $P(y \mid x')$. Therefore in practice we may replace $Y_x$ in the following training procedure with $\cup_{x' \approx x} Y_{x'}$, which includes target points from neighbors of $x$ to increase the robustness of DyMoN.

The stochastic output of DyMoN is enabled by a random input vector $\varepsilon \in \mathbb{R}^n$ sampled from a probability distribution $\mathcal{F}$ with zero mean and unit variance. This input can be explicitly written into DyMoN function as

$$T_\varepsilon(x, \theta) = x + f_\theta(x, \varepsilon), \tag{1}$$

where $f_\theta : \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R}^d$ represents suitable feed forward layers for combining the training input $x$ with the random input $\varepsilon$ to provide an instantiation of the transition velocity vector $\Delta_\theta(x)$.

Given $x \in \mathcal{X}$ and $\theta \in \Theta$, one can consider the distribution $\widehat{\mathcal{P}}_x^{(\theta)}$ of the random variable $T_\varepsilon(x, \theta)$, $\varepsilon \sim \mathcal{F}$ and estimate it by $m$ i.i.d. instantiations $\boldsymbol{\epsilon} = \{\varepsilon_j \sim \mathcal{F}\}_{j=1}^m$ passed through the network to form $\hat{Y}_{(x,\theta)}^{(\boldsymbol{\epsilon})} = \{T_{\varepsilon_1}(x, \theta), \ldots, T_{\varepsilon_m}(x, \theta)\}$. To conform with the training data, DyMoN is optimized so that this distribution approximates the distribution $\mathcal{P}_x$, as captured from the training data by $Y_x$. This optimization is given by $\arg\min_\theta \mathbb{E}\left[\{H_x(\theta) : x \in \mathcal{X}\}\right]$ with $H_x(\theta) = \mathrm{MMD}(\widehat{\mathcal{P}}_x^{(\theta)}, \mathcal{P}_x)$. The MMD is computed using a Gaussian kernel over $\hat{Y}_{(x,\theta)}^{(\boldsymbol{\epsilon})}$ and $Y_x$. Given trained weights $\theta \in \Theta$ and an initial state $x_0 \in \mathcal{X}$, a random walk is generated by $x_t = T_{\varepsilon_t}(x_{t-1}, \theta)$, where $t = 1, 2, \ldots$, and $\varepsilon_1, \varepsilon_2, \ldots \overset{\text{i.i.d.}}{\sim} \mathcal{F}$.

The training and application of DyMoN to learn deterministic systems and higher-order Markov processes extend naturally from the stochastic memoryless (i.e., first-order) DyMoN, a) by ignoring the random input; and b) providing additional previous states as input respectively. Training details, mathematical background and further evaluation of DyMoN can be found at [4].

## III. Empirical Validation

In this section we demonstrate the performance and accuracy of DyMoN and compare it to a range of other methods. We show that we can reliably use DyMoN to 1) learn the transition probabilities at each state; 2) generate stochastic trajectories within a system; and 3) visualize the data.

**Trajectory generation:** We demonstrate the ability of DyMoN to generate paths on a single and double pendulum, providing one and two previous states as input for the single and double pendulums respectively.

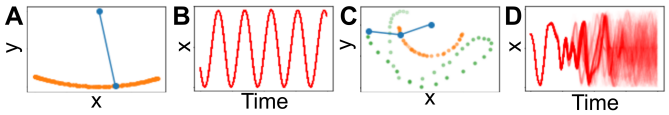


Fig. 2. Paths generated by the second-order DyMoN trained on a single and double pendulum. A single path is shown for each system in (**A**) and (**C**). (**B**) and (**D**) show the x coordinate of 500 paths starting from an epsilon-difference for the single and the lower of the double pendulums, respectively.

Figure 2 shows the Euclidean coordinates of DyMoN-generated paths of both pendulums over time, with only the

| | EMD | TRAIN | INFERENCE |
|---|---|---|---|
| DYMON | 0.150 | 9 MIN | 19.7 S |
| RNN | 1.61 | 269 MIN | 3209 S |
| KF | 0.357 | 650 S | 15.1 S |
| HMM | 0.159 | 92 S | 10.5 S |

TABLE I
PERFORMANCE OF LEARNING METHODS FOR DYNAMICAL SYSTEMS ON GENERATING SAMPLES FROM A GAUSSIAN MIXTURE MODEL.

lower pendulum shown in the case of the double pendulum. Both predicted pendulums show smooth trajectories; the single pendulum shows periodic behavior and the double pendulum shows chaotic behavior.

We also trained DyMoN on the Frey faces dataset [5] to demonstrate DyMoN's capacity to learn an empirical model of a stochastic system for which no generating distribution exists. Figure 3 shows 1000 samples generated by DyMoN on the Frey faces dataset visualized using PCA and ten samples from this trajectory with uniform spacing in time. DyMoN samples a large range of states and generates a realistic new trajectory.

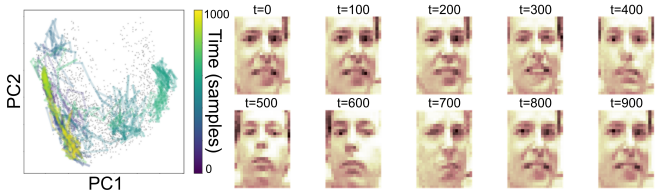


Fig. 3. Chain generated by the DyMoN visualized on a PCA embedding of the Frey faces dataset (left). The chain is shown in color (indicating time) superimposed over the training data in gray. 10 equally spaced faces generated by DyMoN are also shown (right).

**Stationary distribution comparison to other methods:** We train DyMoN to sample a Gaussian mixture model (GMM) to show it can generate samples that match the expected data distribution, avoiding the "mode collapse" problem which affects Generative Adversarial Networks, in which the model generates samples from only a single mode of a multimodal distribution. We generate a 1D Gaussian mixture model using Metropolis-Hastings sampling and compare the performance of Recurrent Neural Networks (RNNs), Hidden Markov Models (HMMs) and Kalman Filters (KFs) (Table I). DyMoN performs similarly to the HMM, both in accuracy of the sampled distribution measured by Earth Mover's Distance (EMD) and in inference time. RNNs fall into an infinite loop reproducing the same data point. Kalman filters are inherently difficult to train on a system without predefined states and transitions, and as such under-perform in an unsupervised setting.

**Visualization:** A DyMoN with a low-dimensional latent layer can also be used to produce a visualizable embedding of the data. When trained on a video of a rotating teapot, [6] DyMoN's embedding layer is homeomorphic to a circle (Fig. 4), while PCA on the same dataset produces spurious intersections, and an RNN does not produce an interpretable embedding, as it does not simply model the transition but has memory that may confound the embedding.

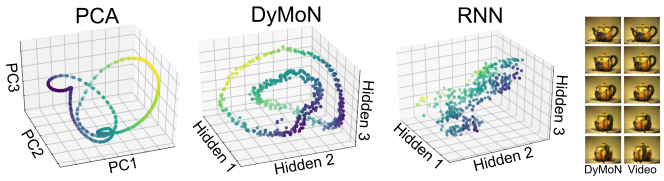


Fig. 4. Embedding of the teapot video dataset colored by the x axis using PCA (left), a three-node hidden layer of DyMoN (center), and a three-node hidden layer of an RNN (right). We also show sequential images from the original video and a DyMoN-generated chain (far right).

## IV. BIOLOGICAL CASE STUDIES

### A. T Cell Development in the Thymus

Next, we use DyMoN to learn transitions in single-cell data. Since single-cell measurements are destructive, we cannot follow a single cell through time with current technologies; instead, the entire population of cells is used to derive potential temporal cell trajectories [7].

Here, we use a mass cytometry dataset measuring developing T cells (adaptive immune cells) from a mouse thymus [7], using pseudotime from Wishbone [7] to sample local transitions. After training, we sample trajectories by initializing DyMoN with undifferentiated cells. We obtain two types of trajectories (Fig. 5.) We examine marker expression as a function of trajectory progression (Fig. 5Cii and Ciii). The two trajectories follow the known progression of developing T cells, beginning at CD4-/CD8- immature T cells, developing into CD4+/CD8+ poised T cell progenitors, and finally diverging into T helper cells (CD4+/CD8-) and cytotoxic T cells (CD4-/CD8+) respectively.

To investigate the DyMoN learned internal representation, we compute the Jacobian of the transition vector with respect to the inputs at two points on the trajectory: at the branch point (Fig. 5Di) and at the CD8+/CD4- branch (Fig. 5Dii). DyMoN learns different associations between genes at each point in the ambient space. We find a negative association between CD4 and CD8 in the Jacobian obtained at the branch point, reflecting the known decision between downregulating either CD4 or CD8. In addition, we find that in the Jacobian of the CD8+ branch Gata3, a developmental marker, goes down with CD4 and CD8, confirming the developmental characteristic of the trajectory. Finally, in the same CD8+ branch Jacobian, Foxp3 and CD25 are positively associated, consistent with their role as regulatory T cell markers. Analysis of DyMon concurs with established literature on T cell development, showing that DyMoN can both learn meaningful trajectories and provide insights into the network's dynamical model of the system.

### B. Human Embryonic Stem Cell Differentiation

We now apply the generative capabilities of DyMoN on a biological system of newly measured single-cell RNA sequencing data of human embryonic stem cells (hESCs) grown as embryoid bodies (EB). Cells were distributed over a 27-day

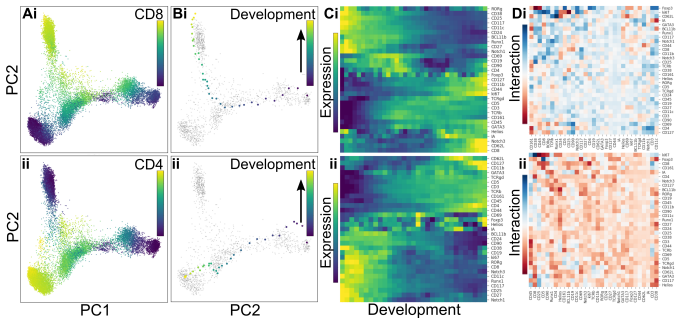


Fig. 5. DyMoN on high-dimensional protein abundance data of T cell development. (**A**) PCA plots of all 17,000 cells colored by CD8 (**Ai**) and CD4 (**Aii**) expression. (**B**) PCA plots with all cells in grey and DyMoN trajectories in color. (**Bi**) DyMoN trajectory of CD4+/CD8- T helper cells. (**Bii**) DyMoN trajectory of CD4-/CD8+ cytotoxic T cells. (**C**) Shows row z-scored heatmaps of marker expression as a function of the trajectory for each of the two DyMoN trajectories with hierarchically clustered genes on the rows and cells on the columns. (**D**) Heatmaps of the Jacobians obtained at the branch point (**Di**) and at the end of the CD8+/CD4- branch (**Dii**).[2]

time course and measured using the 10X Chromium platform. We train DyMoN on Markov transitions from the diffusion geometry of the dataset. We sample neighbors $\mathbf{y}$ of $x$ over weighted affinities defined by a Gaussian kernel, retaining neighbors for which $y$ is defined to be "later" than $x$ (defined from a smoothed estimate of the discrete time variable.)

Fig. 6, shows two generated trajectories: neural progenitor development (A) and bone progenitor development (B). Fig. 6ii shows that each of these paths initially expresses known stem cell markers NANOG and POU5F1 and follow a common transition into the ectoderm (LHX2) before diverging, with path A taking the neural progenitor branch distinguished by SOX1, and path B taking the bone progenitor branch distinguished by ALPL. We observe a novel set of transcription factors distinguishing each stage of differentiation (Fig. 6iii), and propose them as a potentially novel reprogramming protocol to obtain each respective mature cell type. Thus, DyMoN provides a novel form of hypothesis generation enabled by deep abstract models of dynamical systems.

## V. Conclusion

Here we presented DyMoN, a neural network framework for modeling stochastic dynamics from observed samples of a system. DyMoN is well-suited for modeling $n$-th order Markovian stochastic dynamics from the types of data that occur in biological settings, especially in systems for which the generative process cannot be described by differential equations. The flexibility of this framework is enabled by several aspects of DyMoN. First, since the networks encode Markovian dynamics, they do not require continuous longitudinal data. Second, the MMD penalty used to train the next-state generation can be enforced via samples or a known probability distribution, making the networks trainable on top of shallow models such as diffusion operators and pseudotime inferences. DyMoN enables generation of new trajectories, extraction of feature

[2]High-resolution figures available in [4].

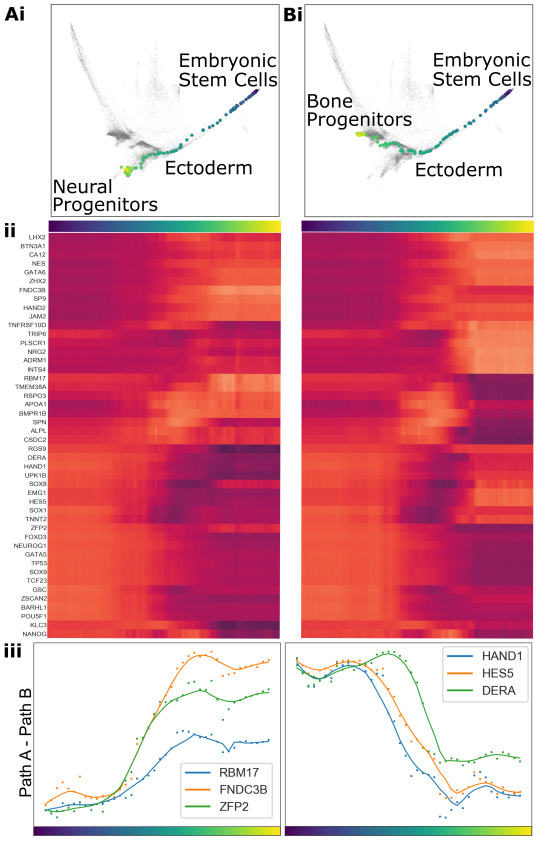


Fig. 6. Paths generated by DyMoN sampling the neural progenitor (A) and bone progenitor (B) cell states. Paths are shown both on MDS (i) and as heatmaps of selected transcription factors (ii). From these paths, we propose a novel cellular programming protocol shown in (iii).[2]

dependencies, and visualization of the dynamic process, and will enable inference of driving forces (transcription factors, mutations, etc.) of progression dynamics in biomedical data.


## Acknowledgments

This research was partially funded by: the Gruber Foundation [*S.G.*]; IVADO (l'institut de valorisation des données) [*G.W.*]; and the Chan-Zuckerberg Initiative (grant ID: 182702) [*S.K.*].


## References


[1] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *JMLR*, vol. 13, no. Mar, pp. 723–773, 2012.

[2] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani, "Training generative neural networks via Maximum Mean Discrepancy optimization," *ArXiv e-prints*, May 2015, arXiv:1505.03906.

[3] R. R. Coifman and S. Lafon, "Diffusion maps," *ACHA*, vol. 21, no. 1, pp. 5–30, 2006.

[4] S. Gigante, D. van Dijk, K. Moon, A. Strzalkowski, G. Wolf, and S. Krishnaswamy, "Modeling dynamics of biological systems with deep generative neural networks," *ArXiv e-prints*, 2018, arXiv:1802.03497.

[5] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[6] K. Q. Weinberger, F. Sha, and L. K. Saul, "Learning a kernel matrix for nonlinear dimensionality reduction," in *PMLR*. ACM, 2004.

[7] M. Setty, M. D. Tadmor, S. Reich-Zeliger, O. Angel, T. M. Salame, P. Kathail, K. Choi, S. Bendall, N. Friedman, and D. Pe'er, "Wishbone identifies bifurcating developmental trajectories from single-cell data," *Nature Biotechnology*, vol. 34, no. 6, pp. 637–645, 2016.