AI-assisted Biology: Predict the Conditional Probability Distributions from Noisy Measurements

Shangying Wang[1,2] and Simone Bianco[1,2]

[1]IBM Almaden Research Center, 650 Harry Rd, San Jose, CA 95120, USA
[2]Center for Cellular Construction, San Francisco, CA, 94158, USA

October 7, 2021

Abstract

The relationship between the genotype, defined as the set of genetic information encoded in the DNA, and the phenotype, defined as the macroscopic realization of that information, is still unclear. The emergence of a specific phenotype may be linked not only to gene expression, but also to environmental perturbations and experimental conditions. Moreover, even genetically identical cells in identical environments may display a variety of phenotypes. This imposes a big challenge in building traditional supervised machine learning models that can only predict determined phenotypic parameters or categories per specific genetic and/or environmental conditions as inputs. Furthermore, biological noise has been proven to play a crucial role in gene regulation mechanisms. The prediction of the average value of a given phenotype is not always sufficient to fully characterize a given biological system. In this study, we develop a deep learning algorithm that can predict the conditional probability distribution of a phenotype of interest with a small number of observations per input condition. The key innovation of this study is that the deep neural network automatically generates the probability distributions based on only few (10 or less) noisy measurements for each input condition, with no prior knowledge or assumption of the probability distributions. This is extremely useful for exploring unknown biological systems with limited measurements for each input condition, which is linked not only to a better quantitative understanding of biological systems, but also to the design of new ones, as it is in the case of synthetic biology and cellular engineering.

1 Introduction

Phenotypic variation is ubiquitous in biology and is often traceable to underlying genetic and environmental variation. However, identical genotype and environmental conditions are not sufficient to guarantee a unique phenotype. This is mainly due to the inherent stochastic nature of biological processes. Random fluctuations may alter the levels of the biochemical components and drive the system to a specific phenotype. For example, in the case of protein synthesis, the bursty nature of messenger RNA (mRNA) expression causes the expression level of proteins to always fluctuate. Proteins typically have longer lifetimes than these random "bursts", which results in a buffering effect, thus ensuring biochemical robustness of cellular homeostasis. Moreover, noise in the expression of a given gene can propagate to generate additional noise in the expression of downstream genes. Furthermore, many important cellular processes rely on actual physical interaction and hence spatial proximity between reactants (for example, RNA polymerase and DNA) and other physical criteria which, given the diffusive dynamic nature of the cell, occur stochastically. Noise is a key component of a well functioning biological system, and plays an essential role in key cellular activities (1; 2; 3; 4; 5). However, the study of the nature and effect of noise on biological systems is still poorly studied and hence only partially understood. This implies that the use of machine learning tools in biology is somehow limited by biological fluctuations apparent in the data.

Particularly, building machine learning tools to elucidate the relationship between underlying genetic and environmental conditions to phenotypic observations is a challenge. One critical deficiency of this type of machine learning predictors is the general inability to call a given observation as an outlier relative to the data that the model has been trained and tested on (6). More specifically, there is a distribution of outputs (all possible phenotypic values) corresponding to each unique set of inputs (genetic and environmental condition).

A naive deep learning predictor will make a prediction based on the mean of all available observations for each unique set of inputs. This is problematic in the case of stochastic processes in the presence of limited observations, where the mean value cannot represent the entire dynamics of the system. Based on the central limit theorem in statistics, the sampling distribution of the mean for a variable will approximate a normal distribution given a sufficiently large sample size. The sampling distributions of the mean cluster more tightly around the population mean as the sample sizes increase. Conversely, the sampling distributions of the mean for smaller sample sizes are much broader. For small sample sizes, it's not unusual for sample means to be further away from the actual population mean. Furthermore, even with a very large sample size and a reliable machine learning predictor are trained, the sample mean cannot provide insightful information for specific inputs since not only the average, but the whole probability distribution is important in understanding the whole characteristic of the population.

Figure 1 reports a simple examples of the difficulties in correctly mapping the inputs to the outputs if limited measurements are obtained. Let us assume that

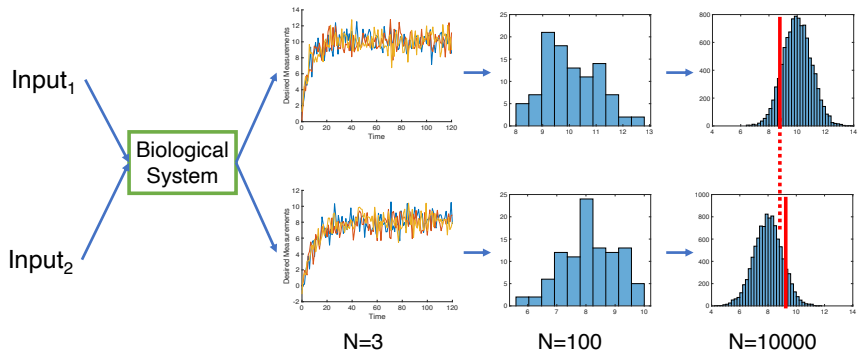

Figure 1: Mapping between inputs and outputs for a biological system. Here, we assume the probability distribution of the output is normal. The figure shows 3 time courses for two different inputs (Input_1 and Input_2) of the biological system respectively, as well as the histogram of the steady state value with 100 and 10000 measurements.

there are two different inputs (Input_1 and Input_2) to the biological system. For each input, experiments are performed and some measurements are taken. In Figure 1 we show the noisy time course of a desired phenotype with 3 repeated measurements for each input. We also show the histogram of the steady-state value with 100 measurements and 10000 measurements. The red line represents the sample mean. One can see that the higher the sample size, the smoother the histogram. Based on the histogram, it is possible to conclude that the output of Input_2 is different from the output of Input_1, with The histogram in the former case shifting to the left. As well known, the analysis of small datasets may not allow to correctly capture the different phenotypes relative to Input_1 and Input_2. Moreover, in this synthetic example, we have assumed the probability distribution of the output to be Normal. However, for real biological systems, the type of probability distribution can be quite different with different inputs, which can further complicate the inference.
The aim of this work is to introduce a machine learning method capable of inferring the conditional probability distribution $CPD(y|x)$ of the output variable y, conditional on inputs x without knowing the prior knowledge or assumption of the distribution itself.

2 Related Work

2.1 Predictive machine learning algorithms to infer the sample mean with reduced noise

There has been little work devoted to reducing the noise of observations and increase the prediction accuracy of the mean value of the population. Among these,

we cite ensemble techniques, distance-based algorithm and single-learning-based techniques to identify noisy instances (7). However, as mentioned in the introduction, the essence of noise in biological systems may prevent from calling an observation an outlier. The whole probability distribution has essential information and it is needed to help understand the characteristic of the biological phenomenon.

2.2 Inferring probability distribution vs noise reduction with increased sample size

Probability distributions for each input condition, i.e., conditional probability distributions, are generally estimated with sufficient observations, either from experiments or stochastic simulations (8; 9). Such methods are often expensive, time consuming or both.

If one can estimate the conditional probability distribution of a target variable for a small subset of input combinations, a Long Short-Term Memory (LSTM) algorithm can be used to to map the inputs to the probability distribution of some variables training on a relative small input parameter combinations. The trained algorithm can then be used to explore a much larger input parametric space, as in the example of the myc-e2f pathway in cell-cycle progression (10). The model describing this specific biological system consists of 10 stochastic differential equations and 24 trainable parameters. For each combination of parameters, 10^4 repeated stochastic simulations were sampled to get the conditional probability density function of the variables. In total, 10^5 parameter combinations were collected. The data collected is actually very sparse compared to the high input dimensions. Then an ensemble of LSTM neural networks were employed to map the parameter set to the corresponding probability density functions and demonstrated pretty high prediction accuracy.

Absent a sufficiently large sample size, the estimation of the conditional probability may not be possible. With this work, we aim at solving this problem. Figure 2 is a schematic plot describing the inference of conditional probability distributions from limited observations. With varying input conditions, such as temperature, growth rate, lysis rate, etc., for each set of inputs, limited observations can be obtained. We aim at using deep learning to estimate the conditional probability distribution from few observations per input condition.

2.3 Inferring distribution hyperparameters for a pre-assumed probability distribution type

It is often common to assume a probability distribution for the variable of interest (11) when it is unknown, through various ad hoc assumptions on the nature of the data. For example, for continuous and effectively unbounded outputs (within the region of probable values), a Gaussian distribution is typically used; For continuous and positive outputs, a Weibull distribution can be used. For discrete outputs (only integer values are allowed) and have a lower bound of zero, a negative binomial distribution is typically used.

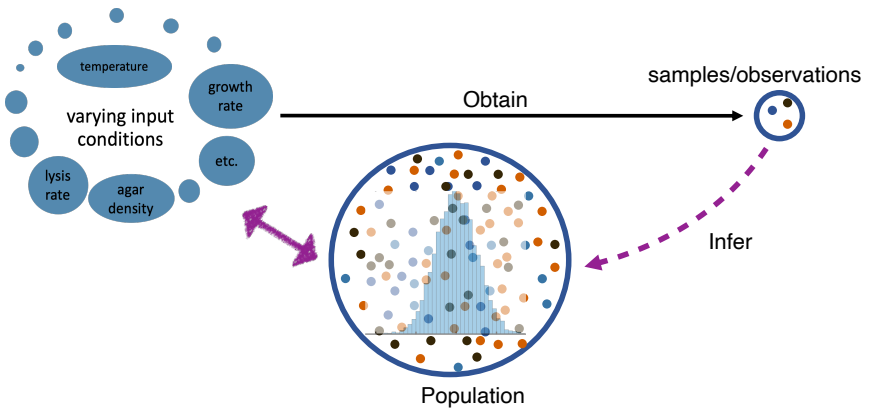

Figure 2: Research goal: infer the conditional probability distribution from few observations.

Indicate the preassumed probability distribution with the function $P(y|\theta)$, where y represents all possible values of the target variable and θ represents the hyperparameter(s) which set the exact shape of the distribution. For example, the hyper-parameters of the Gaussian distributions are the mean μ and the variance δ. With the chosen distribution, a simple machine learning model can be built to predict the value(s) of θ, then the probability distribution function $P(y|\theta)$ can be calculated.
In order to use this approach, a fixed distribution type needs to be chosen. However, for biological processes, with different input parameter combinations, the probability distribution can be quite different, and may not be known apriori. In the next section, we introduce a machine learning framework capable of inferring the probability distribution of the output of a biologic system using a limited set of observations per given biological input.

3 System and Methodology

3.1 A simple gene regulatory network

Given the observations of burstiness of mRNA transcription, a conceptual model of the most elementary multi-state promoter, namely the two state promoter, as described in Figure 3, can be constructed. The gene itself randomly transitions between transcriptionally inactive "off" state and active "on" state. There are four parameters: k_{on}, the rate at which the gene transitions from the inactive to the active state, which determine the rate of transcription bursts; k_{off}, the rate at which the gene transitions from the active to inactive state, which determines the duration of transcription bursts; v, the rate of transcription when the gene is in the active state, which determine the burst size, i.e., how many mRNAs are produced during each burst; and δ, the rate of mRNA degradation. Such a

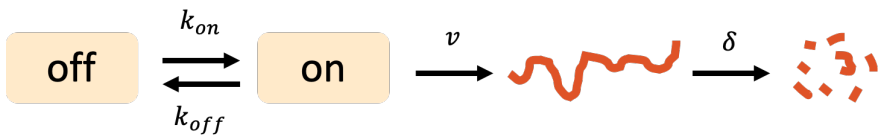

Figure 3: Scheme of a simple gene regulatory network. The gene randomly transition from an *on* to an *off* state and viceversa, with rates k_{on} and k_{off}, respectively. The gene can transcribed in the *on* state at a rate v, and degrades at rate δ.

model was first analyzed by Peccoud and Ycart (12). We choose to apply our framework on this system because it is simple and has an analytic solution of the probability density function of mRNA (Eqn. 1), and therefore allows for direct validation of our method (13).

$$P(m|k_{on}, k_{off}, v, \delta) = \frac{1}{m!}\frac{\Gamma(m+a)\Gamma(b)}{\Gamma(m+b)\Gamma(a)}\mathcal{F}, \tag{1}$$

where $a = \frac{k_{on}}{\delta}$, $b = \frac{k_{on}+k_{off}}{\delta}$ and $\mathcal{F} = {}_1F_1(m+a, m+b; -\frac{v}{\delta})$. ${}_1F_1(a, b; y) = \sum_{n=0}^{\infty} \frac{(a)_n}{(b)_n n!} y^n$ is the confluent hypergeometric function of the first kind, where $(a)_0 \equiv 1$ and $(a)_m \equiv a(a+1)(a+2)...(a+m-1)$.
Based on the analytical results (Eqn. 1), we can see that there are only three dimensionless parameters will affect the outcome: $\frac{k_{on}}{\delta}$, $\frac{k_{off}}{\delta}$ and $\frac{v}{\delta}$. Thus we fixed $\delta = 1$ and only vary input parameters k_{on}, k_{off} and v for simplicity. Experimental observations have indicated that the half-life of the mRNA is on the order of $5 - 10$ minutes in bacteria (14), around 10 hours in human (15). Any scaling will result in the same stationary distribution, as such distributions are, by definition, independent of time.
For this model, there is not only intrinsic noise, but also measurement noise. The measurement noise is due to the restriction that all the measured mRNA numbers shall be integers.
Figure 4 illustrate that even for this simple gene regulatory system, with different parameter values, the probability distribution can take various forms. It can sometimes be monotonically or exponentially decreasing, bimodal, a step function or a Gaussian. Thus we cannot use a pre-assumed distribution type approach for this system.

3.2 Data Preparation

For the system described above, we randomly generate 13000 parameter combinations by tuning k_{on}, k_{off} and v. ($0.01 < k_{on}, k_{off} < 1000$ and $0.01 < v < 200$) (3). (10000 is used for the training and 3000 is used for the test).We calculate the probability distribution for mRNA number from 0 to 300 for each parameter combinations, which is not used when training the neural network, but to measure the neural network performance after training.

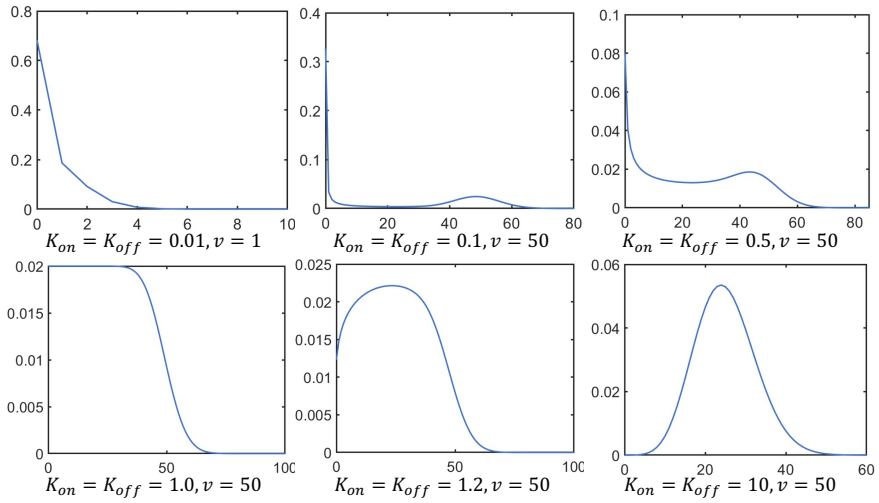

Figure 4: Examples of theoretical probability distributions for the biological system of Figure 3.

We numerically computed the cumulative distribution function (cdf) for the system based on Eqn. 1. For any random variable, the cdf is always a non-decreasing function with $0 \leq cdf \leq 1$. Thus by drawing a random number r ($0 \leq r \leq 1$) and then by finding the corresponding inverse value on the abscissa, a sample value of random variable can be generated. This method is called inverse transform sampling (16).
By using the inverse transform sampling method, we randomly draw n samples ($n = 1$, 3, 10 and 100) for each parameter combinations to represent limited observations, which are used in the training process to minimize the loss function.

3.3 Machine learning framework

Classic regression models would take a set of inputs X, and train machine learning models to predict the value of the target variable Y. The value is usually a single observation value, or a mean value based on a few observations. The loss function is chosen to minimize the neural network predictions with the mean value of the observations. In this study, we present a Recurrent Neural Network (RNN) based model for the data-driven inference of the conditional probability function of noisy biological systems. While most of the previous deep learning models assume the Gaussian or other types of distributions, the proposed RNN model aims at directly predicting the probability distribution function without any prior knowledge or assumption, except for continuity.
Our framework uses an LSTM, which is a special kind of RNN, to predict the distributions (Figure 5). LSTM is proposed to overcome the difficulties in learning a long-term dependency structure (17; 18). LSTM uses multiple gating

functions to conserve the information stored in its internal state for a longer period of time. Our model also employs a negative log-likelihood function as the loss function (Eqn. 2). The loss function is trained with limited output observations (e.g., the number of mRNA produced). The framework can be applied to any biological system (e.g., a cell) whose input parameter space is large and virtually impossible to explore experimentally (e.g., genetic modifications, physical modifications, environmental modifications), and whose output is a desired phenotype (e.g., cellular growth, cellular adhesion, cellular sensing).
The use of a RNN with a negative likelihood loss function has never been attempted to estimate an unknown probability distribution of a biological phenotype. The combination of the minimization of the loss function using observations (which are not a ground truth), and the training of the RNN using input parameters is also non-obvious.
Suppose there is a data set $D = \{(y_i, x_i); i = 1, ...M\}$. The parameters of the artificial neural network Θ are estimated by minimizing a negative log likelihood function $l(\Theta)$.

$$l(\Theta) = -\sum_{i=1}^{M}\sum_{j=1}^{n} log(P(o_{ij}|X_i))) \tag{2}$$

where M is the data size, n is the number of observations, X_i represents the ith input parameter combination, o_{ij} represents the jth output observation corresponding to the ith input parameter combination, and P is the probability function of the observations to the parameters, which is first initialized to be uniform during the neural network initialization process. The loss function is trained using a limited set of collected observations in order to remove the uncertainties that are inherent in a RNN. Its purpose is to maximize the probability that the observed data is within the predicted probability distribution of the RNN. It does this by calculating the sum of all of the probabilities of the observed data within each parameter. Because the loss function is a negative log-likelihood, its integration into the RNN minimizes the loss of observations that would otherwise fall outside of a typical mean analysis. The first step in the training of the RNN described herein is the establishment of the loss function. During training of the RNN, the loss function compares the prediction outcomes to the desired output resulting in output values throughout the time series and propagation of the loss function back through the RNN to update the input weights; thus, every node that has participated in the calculation of the output associated with the loss function has its weight updated to minimize the error throughout the RNN.

3.4 Pipeline

3.4.1 Data Collection/Generation/Cleaning

We assume that data is collected either through experiments or *in silico* for various input conditions if a stochastic formulation the biological system is avail-

able. Input values can be chosen randomly. Our methods accepts normalized to a standard scale input values. If the observations are discrete numbers (mRNA counts, protein counts, etc.), they need to be integer values, and are included as one of the predicting points for the neural network (NN) outputs. If the observations are continuous values (concentrations, optical density measurements, etc.), they formally need to be bound from above to ensure that observations are within the prediction range of the neural network.

3.4.2 Neural Network Construction

The input layer of the neural network inputs all the input conditions. The output layer of the neural network outputs the probability distribution of the neural network. If the observations are discrete numbers, the neural network predict the probability value for all possible discrete numbers and a Softmax layer is implemented to ensure that the sum for the predicted probability for any input condition equals 1. If the observations are continuous values, it is necessary to discretize the possible observation range into bins. The neural network is only predicting the probability value for the center of the bins. The probabilities for other values are interpolated. The last layer is normalized with a normalization factor to make sure the cumulative trapezoidal numerical integration of the probability distribution equals 1.

3.4.3 Neural Network Training

Neural network nodes are randomly initialized. The output probability distribution are initialized to a uniform distribution and modified each training epoch to minimize the loss function. The gradient was clipped to prevent exploration. The neural network is trained using a training data set consisting of the input conditions and one output observation. To test the performance, measurements of a small batch of data is repeated multiple times to estimate a sample probability distribution.

3.4.4 Neural Network Predictions

After training, the neural network is ready to be used for predicting the probability distribution for any input. Trained neural network can be used to facilitate the quantitative understanding of biological systems as well as the design of synthetic gene circuits.

4 Results

4.1 Conditional probability distributions can be inferred with limited observations

The input of our algorithm is parameter combinations, specifically, k_{on}, k_{off}, and v for the proposed system. The output is the probability distribution of the

	n=100		n=10		n=3		n=1	
	Train	Test	Train	Test	Train	Test	Train	Test
Loss	1.40908	1.41247	1.40535	1.41159	1.40645	1.41636	1.40508	1.42919
R^2	0.9768	0.9759	0.9762	0.9749	0.9770	0.9760	0.9735	0.9725
RMSE	0.00630	0.00641	0.00631	0.00643	0.00615	0.00628	0.00651	0.00662

Table 1: Model Performance Compare

intrinsically noisy observations. Figure 6 shows the neural network performance for the test dataset for sample size $n = 100$, 10, 3 and 1 respectively. The examples are randomly chosen from the training and test datasets. Table 1 listed the values of the loss function, R^2 and the root mean squared error (RMSE) for the training and test dataset for$n = 100$, 10, 3 and 1 respectively. The accuracy is calculated by comparing the predicted probability distribution vs the real probability distribution. This is done after the neural network training is finished. Since the probability distribution is discretized into 301 points, the RMSE is calculated based on comparing $(301 * datasize)$ values between the prediction and the ground truth. All the values are the average value from 3 neural networks.

Our model works very well in predicting the probability distribution. It can be used to infer probability distribution of unknown types. Even with only 1 observation per input condition, our method can still predict with high reliability. We demonstrate that repeated measurements under a fixed condition are not necessary to obtain a reliable estimation of the sample probability distribution. Moreover, since noise plays an important roles in biological functions, being able to generate the conditional probability function with limited data can aid the understanding of the system dynamics in various parts of the parameter space. By using our method, it is possible to easily detect different dynamical regimes, as well as give a reliable estimate of the variability of the outputs.

A consequence of the accuracy of the predictions of the distribution using deep learning is a higher accuracy in the estimation of the sample parameters and distribution momenta. As an example, in Figure 7 we report the sample mean and the mean predicted using the NN vs the real mean, calculated based on Eqn. 1). Larger sample size can largely reduce the noise and make the sample mean closer to the real population mean (top panel). By using our framework, the predicted mean calculated based on the predicted conditional probability distribution from the neural network can be estimated for significantly smaller sample sizes.

5 Conclusion and future work

Biological processes are intrinsically stochastic, which often causes difficulties in our ability to investigate and understand biological or clinical data. Specifically, it hinders our understanding of the relationship between the underlying genetic, environmental conditions and phenotypic observations, which has important

consequences for fields like synthetic biology and cellular engineering, where high reliability of the output prediction may be critical.
In this study, we develop a deep learning algorithm which is able to predict the probability distribution of the phenotype observations with as few as one observation per input combination. The key innovation of this study is that deep neural network automatically generates the probability distributions based on few observation for each input condition, without prior knowledge or assumption of the probability distributions.
Our approach Overcomes the bottleneck of the need to collect many observations per input condition to eliminate the effect of the noise of phenotype observations or reconstruct the conditional probability distribution of a stochastic process, which is often expensive and time consuming. It also reduces the barrier in implementing predictive machine learning model for "noisy" biological data.

Code Availability

Source code is available at: https://github.com/ShangyingWang/RNN_NLL_2021.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. DBI-1548297.

Competing Interests

The authors declare that they have no competing financial interests.

Correspondence

Correspondence and requests for materials should be addressed to Simone Bianco (email: sbianco@ibm.com).

References

[1] Raser, J. M. & O'shea, E. K. Noise in gene expression: origins, consequences, and control. *Science* **309**, 2010–2013 (2005).

[2] Eldar, A. & Elowitz, M. B. Functional roles for noise in genetic circuits. *Nature* **467**, 167–173 (2010).

[3] Munsky, B., Neuert, G. & Van Oudenaarden, A. Using gene expression noise to understand gene regulation. *Science* **336**, 183–187 (2012).

[4] Tsimring, L. S. Noise in biology. *Reports on Progress in Physics* **77**, 026601 (2014).

[5] Liu, Y., Beyer, A. & Aebersold, R. On the dependency of cellular protein levels on mrna abundance. *Cell* **165**, 535–550 (2016).

[6] Ching, T. *et al.* Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface* **15**, 20170387 (2018).

[7] Gupta, S. & Gupta, A. Dealing with noise problem in machine learning data-sets: A systematic review. *Procedia Computer Science* **161**, 466–474 (2019).

[8] Swain, P. S., Elowitz, M. B. & Siggia, E. D. Intrinsic and extrinsic contributions to stochasticity in gene expression. *Proceedings of the National Academy of Sciences* **99**, 12795–12800 (2002).

[9] Mettetal, J. T., Muzzey, D., Pedraza, J. M., Ozbudak, E. M. & Van Oudenaarden, A. Predicting stochastic gene expression dynamics in single cells. *Proceedings of the National Academy of Sciences* **103**, 7304–7309 (2006).

[10] Wang, S. *et al.* Massive computational acceleration by using neural networks to emulate mechanism-based biological models. *Nature communications* **10**, 1–9 (2019).

[11] Blake, S. A guide to generating probability distributions with neural networks. https://medium.com/hal24k-techblog/a-guide-to-generating-probability-distributions-with-neural-networks-ffc4efacd6a4 (2019). Accessed: 2019-12-16.

[12] Peccoud, J. & Ycart, B. Markovian modeling of gene-product synthesis. *Theoretical population biology* **48**, 222–234 (1995).

[13] Wang, S. *Quantifying gene regulatory networks.* Ph.D. thesis, Duke University (2014).

[14] Taniguchi, Y. *et al.* Quantifying e. coli proteome and transcriptome with single-molecule sensitivity in single cells. *science* **329**, 533–538 (2010).

[15] Yang, E. *et al.* Decay rates of human mrnas: correlation with functional characteristics and sequence attributes. *Genome research* **13**, 1863–1872 (2003).

[16] Devroye, L. Sample-based non-uniform random variate generation. In *Proceedings of the 18th conference on Winter simulation*, 260–265 (1986).

[17] Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9**, 1735–1780 (1997).

[18] Gers, F. A., Schmidhuber, J. & Cummins, F. Learning to forget: Continual prediction with lstm. *Neural computation* **12**, 2451–2471 (2000).

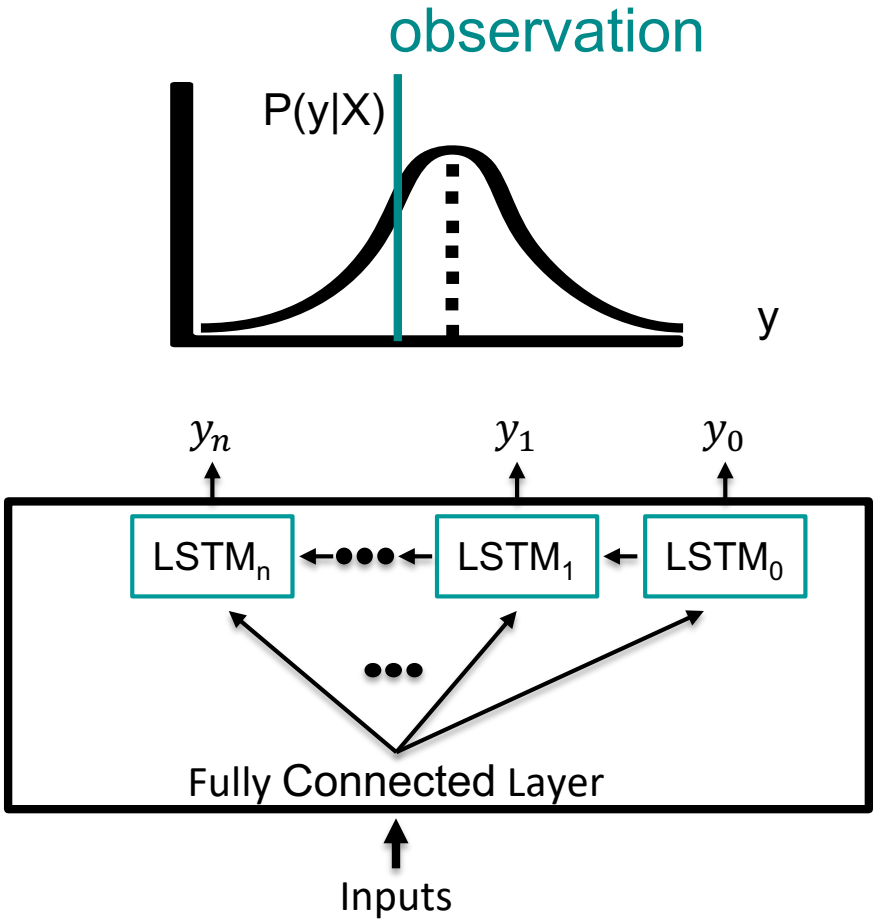

Figure 5: Machine learning framework.

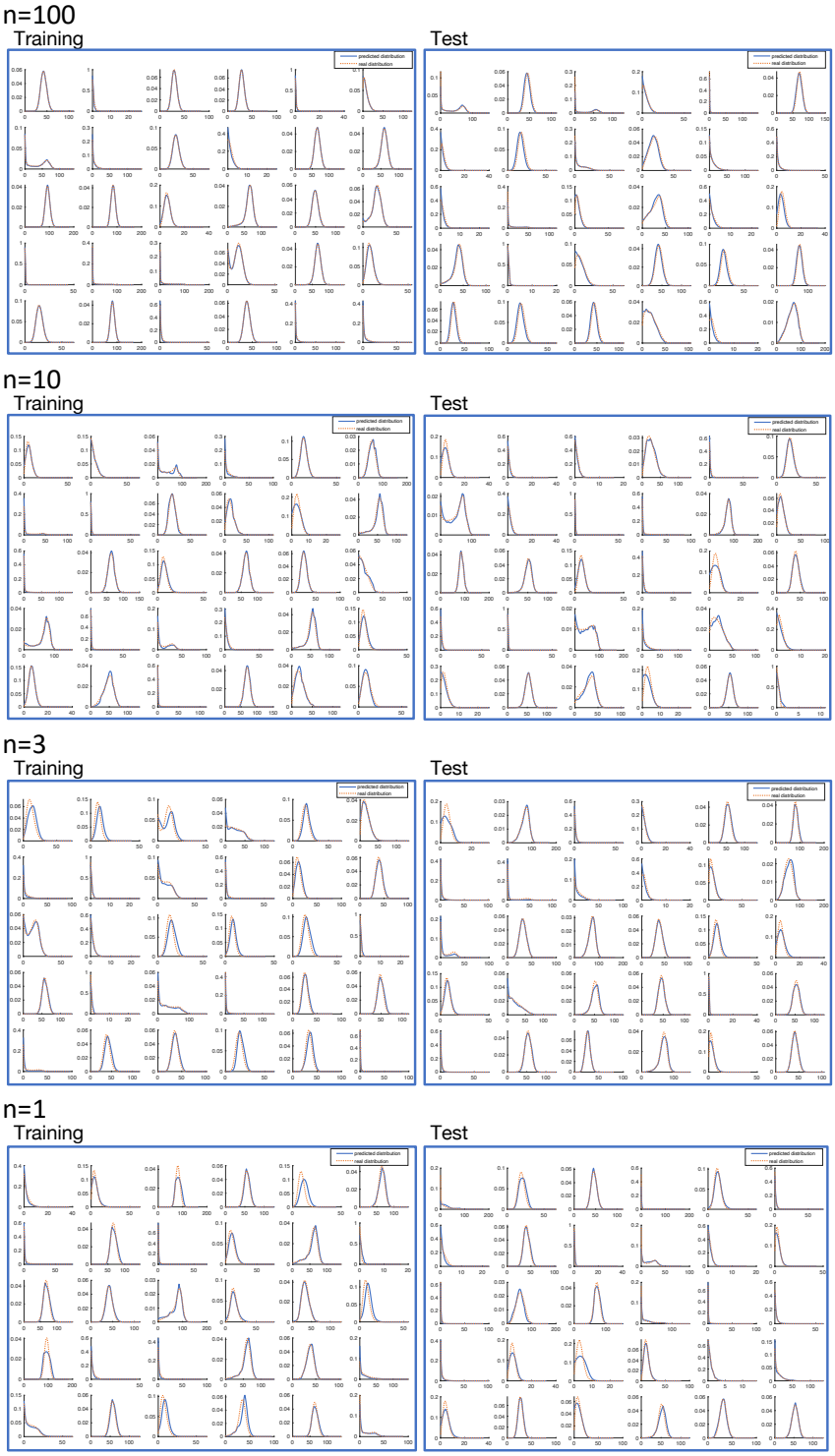

Figure 6: Neural Network performance of the test data.

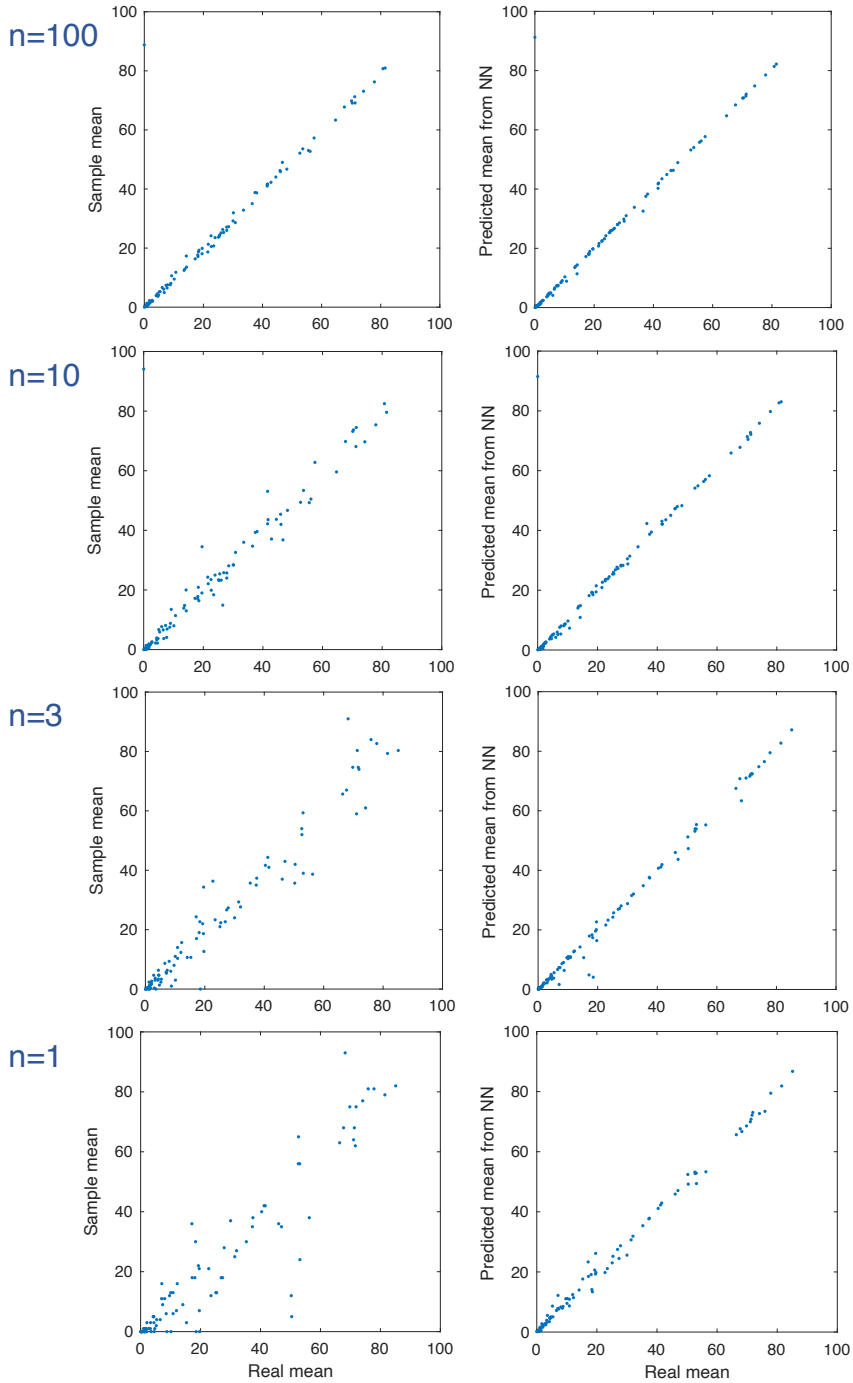

Figure 7: Sample mean, predicted mean vs real mean.