Using Deep Kernels in Time Varying Networks for Reverse-Engineering of Gene Interactions

Yiwen Yuan* [1] **Xueqian Li*** [1] **Dong Wang*** [1]

Abstract

Time varying network has different active edges when time points changed, which presents various interactions between network nodes. For a dynamic network changing through time elapsing, accurate and proper models are needed to engineer such developmental process. In this paper, We explore the performance of deep kernels on reverse-engineering of gene interactions, which incorporates both the structural properties of deep learning architectures with the non-parametric flexibility of kernel methods. Inspired by KELLER (kernel-reweighted logistic regression method) (Song et al., 2009), we utilize pairwise MRF (Markov random field) to model interactions between genes. Deep kernels, constructed from logistic regression and LSTM (long short-term memory) are used for learning the parameters of the pairwise MRF. Furthermore, we show generalizability by extending our method in learning wind power load prediction.

1. Introduction

KELLER (Song et al., 2009) is an algorithm for recovering time-varying networks on a fixed set of genes from time series of expression values, specifically for the task of reverse engineering the gene interaction across different stages of its development. The key assumptions is that the temporal sequence underlying biological processes vary smoothly across time. Under this assumption, to estimate the network at a particular time step, observations from all time steps can be used. Moreover, in KELLER, the authors assume that for learning each time step, closer time steps have more importance than further time steps and use a RBF kernel as the re-weighting parameter for observations from different time steps.

RBF (radius basis function) kernels have been a popular choice in many applications for time-evolving network learning. However, because of its non-parametric nature, it cannot discover the meaningful representations in high-dimensional data. In the setting of reverse-engineering gene interaction, it makes the assumption that for every pair of genes, the interaction of them is inversely related to their distance. This can be a strong assumption because genes have different stages in their life cycle and certain stages plays a more important role. Therefore, the use of deep kernels can be a better candidate for capturing time dependencies.

Our method is a continuation of Song *et al.* (Song et al., 2009)'s work on estimating time varying interactions between genes. In this paper, we adopt the deep kernel design from deep kernel learning (Wilson et al., 2016), starting from a base kernel $k(\mathbf{x}_i, \mathbf{x}_j\theta)$ with hyperparameters θ, we transform the inputs X as $k(\mathbf{x}_i, \mathbf{x}_j\theta)k(g(\mathbf{x}_i, \mathbf{w}),\ g(\mathbf{x}_j, \mathbf{w}), \mathbf{w})$ where $g(\mathbf{x}, \mathbf{w})$ is a non-linear mapping given by a deep architecture, in our case a neural network and an LSTM. For the base kernel $k(\mathbf{x}_i, \mathbf{x}_j\theta)$, we use the RBF kernel.

Our contributions are as follows,

- we replace the non-parametric kernel with a deep neural network kernel and a LSTM module, which learns the pairwise factors in time varying networks step by step using a parametric kernel;
- we compare the three different kernels' performance, and make some biological explanation, which exhibit the advantages of using LSTM kernel;
- apart from the reverse-engineering of gene interactions, we also re-evaluate our method for the wind power load prediction task, where the LSTM kernel has fast convergence.

2. Background & Related Work

In order to better model a gene regulatory process, Song *et al.* (Song et al., 2009) proposed a kernel-reweighted logistic regression to recover the evolving network. They assumed

*Equal contribution [1]Carnegie Mellon University, Pittsburgh, 15213. Correspondence to: Yiwen Yuan <yiweny@andrew.cmu.edu>, Xueqian Li <xueqianl@andrew.cmu.edu>, Dong Wang <dongw1@andrew.cmu.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

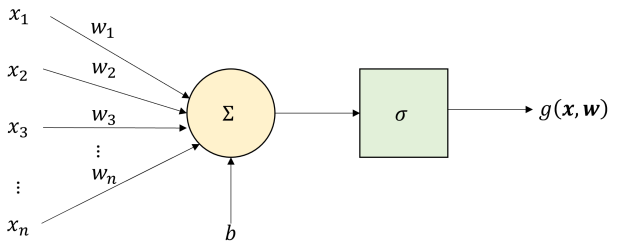


Figure 1. Logistic Regression is used as the non-linear mapping g

that the time-varying evolution process is relatively negligible, thus neighboring temporal networks have more common edges. This assumption allowed them to treat the dynamic network as the aggregation of static networks, and applied the l_1-regularized logistic regression. They used the MRF to model the distribution of genes. Then they considered weighting scheme by applying Gaussian RBF kernel with l_1-regularization, since the observations which are nearby current time matter more. Still, utilizing the Euclidean projected gradient to make the network more efficiently. [But the pairwise factors in the RBF are found to be predefined in the open-sourced code, they may get these factors from their domain knowledge]

Recurrent models recently become popular in solving problems with sequential structure. LSTM (Hochreiter & Schmidhuber, 1997) is a more powerful variant of vanilla recurrent neural network architecture. Its special memory cell and gating mechanism stabilize the flow of the back-propagated errors and improve the learning process of the model. LSTM not only achieves state-of-the-art(SOA) results on solving speech and language problems (Alan Graves & Hinton, 2013), but also quantifies uncertainty (Gal & Ghahramani, 2016).

Maruan *et al.* (Al-Shedivat, 2017) proposed closed-form kernel functions for Gaussian function processes, to encapsulate the structural properties of LSTM. In addition, a new provably convergent semi-stochastic gradient descent algorithm was proposed to jointly learn kernel functions and optimize the recurrent model. And by decomposing the covariance matrix into Kronecker products, the whole runtime was significantly improved, thus enabling scale-able training and prediction on sequential data.

3. Methods

3.1. Estimate Time-varying Network

One way to estimate the time varying network $\mathcal{G}^{(t)}$ is to investigate the pairwise relationship of two vertices at time step i. In our study, we set foot specifically on the task of learning and estimating the time-varying network of gene interaction. Specifically, we want to measure the development of pairwise gene interactions through recovering at each time step the set of genes that each gene u is interacting with, i.e., $\mathcal{N}^{(t)}(u) := \{v \in \mathcal{V} \mid (u, v) \in \mathcal{E}^{(t)}\}$. The intuition is that if we can estimate the neighborhood of each gene at each time step, then we can recover the network $\mathcal{G}^{t_i}$, by joining these neighborhoods. The distribution of the expression values of the genes at any given time point t is normalized to $\{0, 1\}$, where 0 represents the pair is independent and 1 otherwise. They are modeled as the pair-wise Markov Random Field(MRF):

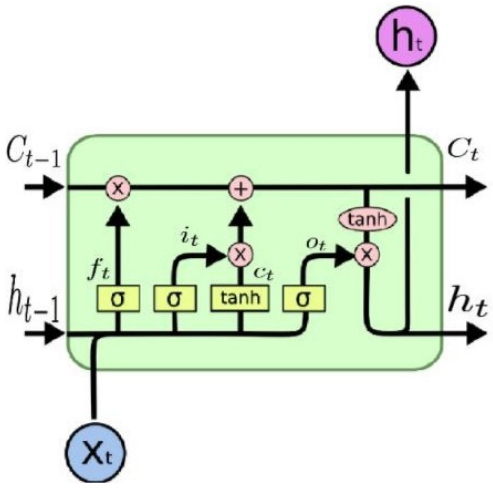


Figure 2. LSTM structure

$$\mathbb{P}_{\theta^{(t)}}(X^{(t)}) := \frac{1}{Z(\theta^{(t)})} \exp \sum_{(u,v)\in\mathcal{E}^{(t)}} \theta_{uv}^{(t)} X_u^{(t)} X_v^{(t)} \quad (1)$$

where $\theta_{uv}^{(t)} = \theta_{vu}^{(t)} \in \mathbb{R}$. The θ is the indicator random variable for the dependence of a pair of gene (u, v).
We can decompose the joint distribution in Equation 1 into a product of distributions of a gene u conditioned on all other genes, noted as $\backslash u := V - \{u\}$. In particular, each conditional distribution takes the form of a logistic regression.

$$\mathbb{P}_{\theta_{\backslash u}^{(t)}}(X_u^{(t)} \mid X_{\backslash u}^{(t)}) = \frac{\exp(2X_u^{(t)}\langle\theta_{\backslash u}^{(t)}, X_{\backslash u}^{(t)}\rangle)}{\exp(2X_u^{(t)}\langle\theta_{\backslash u}^{(t)}, X_{\backslash u}^{(t)}\rangle) + 1} \quad (2)$$

where $\langle a, b\rangle = a^T b$ denotes inner product and $\theta_{\backslash^{(t)}} := \{\theta_{uv}^{(t)} \mid v \in \backslash u\}$ is a vector of parameters associated with gene u, representing if a different gene is independent regarding to u.
With equation 2, we can decompose the task of calculating the network at time step t into learning $\theta_{\backslash u}^{(t)}$ for each gene u. Our goal would be to minimize the log-likelihood of an observation x under equation 2 as $\gamma(\theta_{u^{(t)}}; x) = \log \mathbb{P}_{\theta_{\backslash u}^{(t)}(x_u|x_{\backslash u})}$.
By the assumption of the time varying network, it varies

smoothly across time. Therefore, this assumption allows us to treat each observations as i.i.d.. This allows us to borrow information across time by re-weighting the observations. In KELLER (Song et al., 2009), a Gaussian RBF kernel is adopted as the re-weighing parameter w under the assumption that time steps that are more closely related would have more importance. However, it might not be adaptive to some cases where certain stages of development have more importance over the others. This will be discussed further in later sections

Additionally, we assume that our network is sparse. This sparsity assumption holds in most cases, e.g. research (Davidson, 2001) has shown that a transcription factor only controls a small fraction of target gene under some specific condition. Then, given a time series of gene expressions $D = \{x^{(t_1)}, x^{(t_2)}, ..., x^{(t_n)}\}$, we can estimate $\theta^{(t)}_{\setminus u}$ using an $l1$ penalized log-likelihood maximization, equivalently

$$\hat{\theta}^{(t)}_{\setminus u} = \text{argmin}_{\theta^{(t)}_{\setminus u} \in R^{p-1}}(-\sum_{i=1}^{n} w^{(t)}(t_i)\gamma(\theta^{(t)}_u; x^{(t_i)}) + \lambda\|\theta^{(t)}_{\setminus u}\|_1) \quad (3)$$

where p is the total number of genes

3.2. Deep Kernel Learning

Deep Kernel Learning (Andrew Gordon Wilson & Xing, 2016) introduces the idea of incorporating deep architectures in kernels in Gaussian Process framework. In our work, we use deep kernels for learning the weighting parameter k for each gene u across all the time steps as they can capture more information of latent states and possibly improve the convergence in later stages of training.

Starting from a base kernel $k(x_i, x_j \mid \theta)$, with hyper parameter θ, we can transform the input space into

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j \mid \boldsymbol{\theta}) \rightarrow k(g(\boldsymbol{x}_i, \boldsymbol{w}), g(\boldsymbol{x}_j, \boldsymbol{w}) \mid \boldsymbol{\theta}, \boldsymbol{w})$$

where g is a deep architecture, such as neural network or LSTM, parametrized by weights $\boldsymbol{w}$.

KELLER (Song et al., 2009) adopts the symmetric non-parametric RBF kernel, which will serve as the base kernel for our study.

$$k_{RBF}(\boldsymbol{x}, \boldsymbol{x'}) = \exp(-\frac{1}{2}||\boldsymbol{x} - \boldsymbol{x'}||/l^2) \quad (4)$$

Specifically, the bandwidth l is chosen as the median of time distances and x are the time steps t. As for the choice of g, we use a logistic regression and LSTM (Hochreiter & Schmidhuber, 1997).

In the model, we learn the parameters of the deep kernel $\boldsymbol{w}$ jointly with the parameters $\boldsymbol{\theta}$, by maximizing the log-likelihood of the pairwise MRF model. Another assumption we make is that the re-weighting parameter is universal across all genes for each time step. As we are training the $\theta_{\setminus u}$ for each gene u, we keep the parameters of the deep kernel from the last iteration, the high level process is presented in algorithm 1.

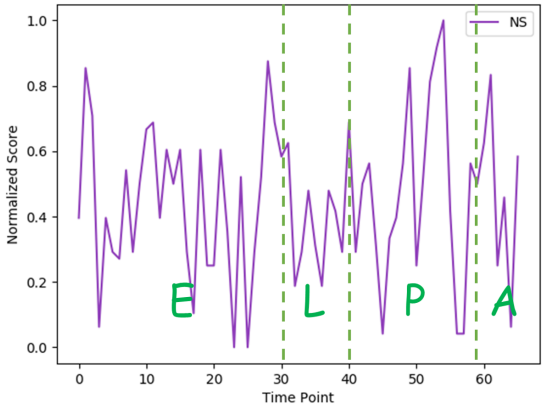
Algorithm 1 Joint Training of Pair-Wise MRF and Deep Kernel

for *time step* t **do**
  Initialize the re-weighting parameter
  $$w_t = k_{RBF}(g(x, w), g(x, w) \mid l)$$
  where g is a deep structure with parameter w and l is the median time distance.
  for *gene* u **do**
    Learn $\hat{\theta}^{(t)}_u$ based on equation 3.
    Learn w with respect to $\gamma(\theta_{u^{(t)}}; x)$
  end
  for *each pair of genes* (i, j), *where* $i \neq j$ **do**
    if $\hat{\theta}_{ij} = 1$ *or* $\hat{\theta}_{ji} = 1$ **then**
      $\theta^{(t)}_{ij} = 1$
    else
    end
  end
end

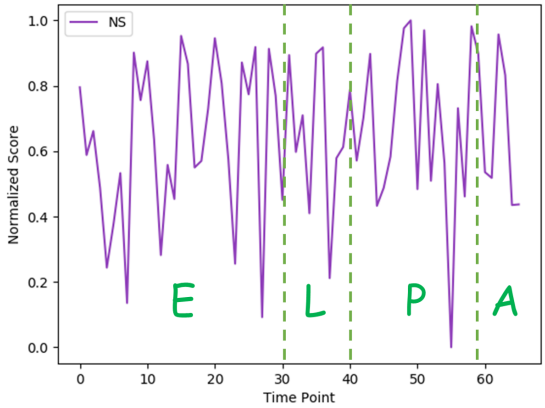
4. Experiments

There are two different dataset for our experiments. For the first dataset, we use the data collected from *Drosophila* to show that our method with LSTM kernel and logistic regression kernel can estimate a biologically time-evloving network, which shows more interesting properties of gene over its course of development. In the reverse-engineering of the gene interactions, we run three different experiments for non-parametric RBF kernel, deep neural network kernel and the LSTM kernel. For each experiments, we compute the network parameters which is defined as θ in Sec. 3. Afterwards, we compare several gene interaction activities during the 4 developmental stage of *Drosophila* (Arbeitman et al., 2002). We re-evaluate the network on the wind power load dataset (Hong et al., 2014) to further improve our network and show the advantages of deep, recurrent network when learning the pairwise factors. During this experiment, we have comparison of some essential parameters in our network, loss, log likelihood, which reveal the learning ability, fast convergence of the deep kernel.

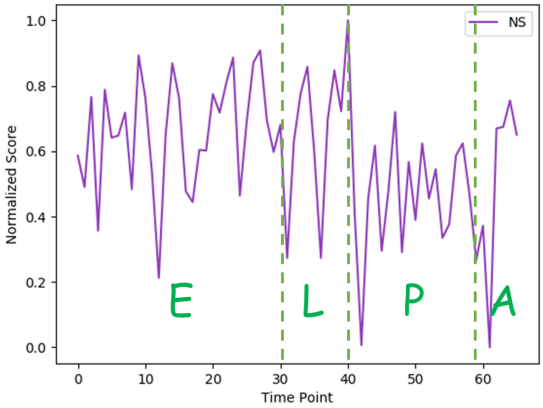
For all the *Drosophila* training experiments, we use a single NVIDIA GeForce GTX 1080-Ti GPU and a Intel(R) Core(TM) i7-6850K CPU at 3.60GHz. Testing experiments are performed on Intel(R) Core(TM) i7-8850H CPU at 2.60GHz. All the wind power load experiments ran on



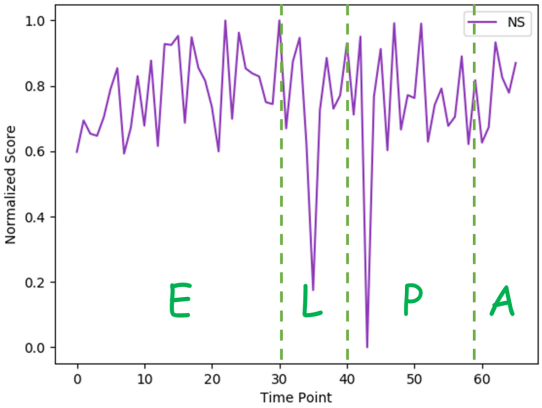
(a) Network size measurement using RBF kernel



(b) Network size measurement using 100000 iteration RBF kernel



(c) Network size measurement using deep neural network kernel



(d) Network size measurement using LSTM kernel

Figure 3. Network size measurement using 3 different kernels for Sec. 4.1. Using deep kernels show clearer pattern of network size changing over the developmental stage. For deep neural network kernel, we observe a slight drop during the pupal and adult stage, while the LSTM kernel learns larger network size during embryonic and adult stages, network size dropping during larval and pupal stages. E for embryonic stage, L for larval stage, P for pupal stage, and A for adult stage.

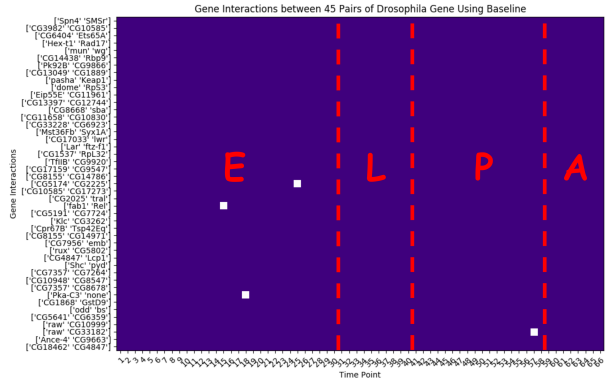
Intel(R) Core(TM) i7-3615QM CPU at 2.30GHz.

4.1. Recovering *D.melanogaster* genes interactions using time-varying network

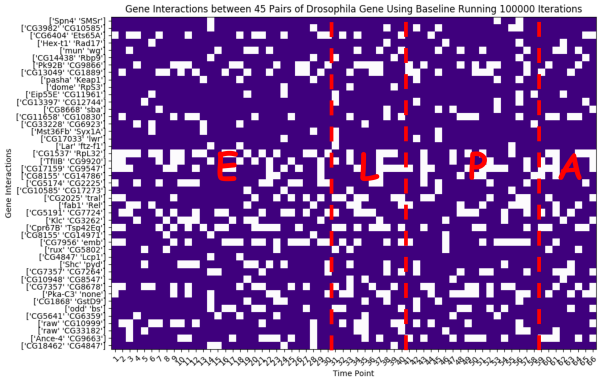
In *D.melanogaster*, the functionalities of genes and their interactions between each other are determined as dynamical and stochastical over its development course. A time-varying rather than time-invariant network can better describe its context-dependence and exhibit continuing systematic rewiring (Song et al., 2009). In this section, using the gene expression measurements from *Drosophila*, we implement three different methods corresponded to different kernels in estimating a biological time-varying network.

During the full development cycle of *D.melanogaster*, 4 different stages will be spanned by 66 time points (1-30 time point: embryonic; 31-40 time point: larval; 41-58 time point: pupal; 59-66 time point: adult) (Song et al., 2009). In our experiment, due to large number of gene dataset, we only focus on 588 genes that are related to the developmental process. The regularization parameter and bandwidth parameter of the network need to be tuned during the implementation.

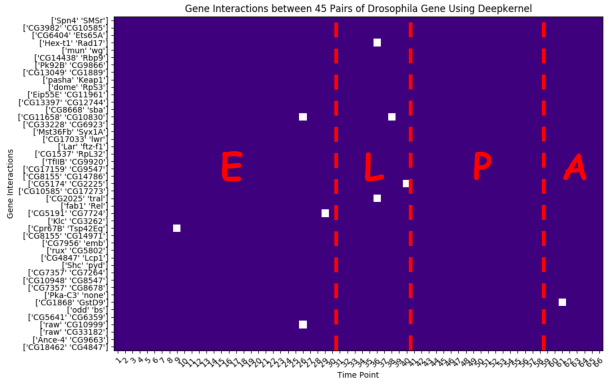
In particular, in order to measure the interactions between genes, we evaluate the statistics of the gene-regulatory networks during 4 developmental stages: the network size, which is defined as the number of edges, measuring the overall connectedness of the networks and the average local clustering coefficient measures the average connectedness of the neighbourhood of each gene. We normalize both statistics to $[0, 1]$ for comparison (Song et al., 2009). For



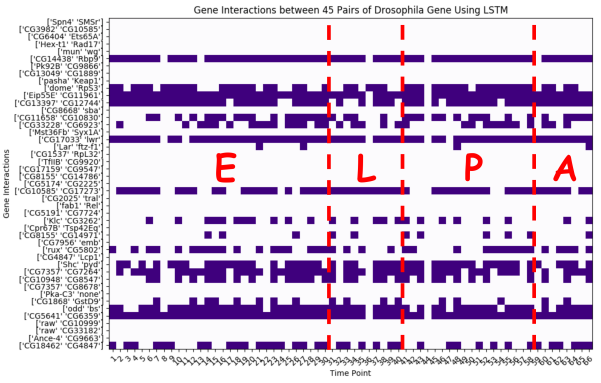
(a) Gene interactions using RBF kernel



(b) Gene interactions using RBF kernel with 100000 iterations



(c) Gene interactions using deep neural network kernel



(d) Gene interactions using LSTM kernel

Figure 4. Interactions of gene pairs at different time point during embryonic stage, larval stage, pupal stage and adult stage using 3 different kernels for Sec. 4.1. The purple block indicates the interactions, where the white block denotes absent interactions. Using baseline trained for 100000 iterations helps find certain pattern compared to 2048 iterations. LSTM kernel converges fast for 2048 iterations compared with deep neural network kernel and the RBF kernel. E for embryonic stage, L for larval stage, P for pupal stage, and A for adult stage.

all the experiments which are not specified, we use 2048 iterations.

4.1.1. Using RBF non-parametric kernel

We train the time varying network with a non-parametric RBF kernel. At the first stage, we trained the network with 2048 iterations, then we increase iterations to 100000. The network size remains large during the whole developmental stage of the cell, and fluctuates (see Fig. 3a 3b), which indicates no clear pattern learned using RBF kernel, as compared to the real biological genes.

Meanwhile, we provide a figure (Fig. 4a 4b) which lists 45 pairs of gene interactions we choose randomly within the 588 developmental genes and the exact time when they occur during 4 stages. The figure can not only be compared to the given prior knowledge of gene interactions in order to check whether the results predicted by the networks are biologically plausible, but also provide some other gene interactions which have not been experimentally verified. The results showing here denotes irregular interactions between those genes when using RBF trained with 2048 iterations, which genes are connected in a disordered manner. However, increasing iteration to 100000 does show certain while we do statistics on all the 588 genes. The percentage of active interactions between genes for RBF kernel learning is 0.99, 1.0, 0.99, 0.99 (2048 iterations) and 0.80, 0.77, 0.81, 0.79 (100000 iterations) for each developmental stage. There is less gene interactions during the larval stage observed with training 100000 iterations, which indicates less development occurred.

To understand the local interactions between neighboring genes, we use chord plot to show interactions between 7 selected genes learnt from RBF kernel training with 100000 iterations in Fig. 5. Judging from various number of gene interactions as time varying, we can observe different gene effects during 4 developmental stages which is consistent with the biological statistics that different genes play various roles in the development. Take CG14438 for example, the length of each chord corresponds to the active gene inter-

actions at specific time step. We can see there is a clear interaction drop during the larval and the adult stage, which is coincide with the result we shown in the statistic data.

4.1.2. Replace the kernel with deep neural network

Utilizing the deep neural network as specified in Sec. 3.2, we are able to recover the time-evolving network with an embedding of the differentiable module, which can be faster for learning.

We replace the RBF kernel with a deep neural network structured with a MLP (multilayer perceptrons) followed with a tanh function and a ReLU (rectified linear unit) activation function. Results are shown in Fig. 3c and Fig. 4c. The network size shows less gene interactions during pupal stage and the adult stage, while the gene interaction image shows no clear structure. The statistic percentage of active genes during development is $0.99,\ 0.99,\ 1.0,\ 0.99$ for each developmental stage.

The reason behind the bad performance of the deep neural network lies in 2 folds. First, we have not trained for longer iterations due to exhaustive training time expense (nearly 7 hours for a time step trained with 100000 iterations), where our network might gain better strength with fully-trained model as we observed in the result. Second, we only use 1 MLP, which lacks capacity to learn the whole developmental structure. The deep neural network can learn limited faster than the RBF kernel, however, the network capacity is still smaller than the network need.

4.1.3. Learn the pairwise factors with LSTM

As mentioned in (Song et al., 2009), because gene samples used for testing in the experiments normally come from different regions and stay at different level of developmental stage, microarray measurements are hard to be considered as exact values of the expression level. It is more reasonable to consider their qualitative level. The method in that paper uses fixed pairwise factor sequences along the timeline. However, based on our data, we find that using deep parametric kernel to evaluate microarray measurements can make predicting the time dependencies of pairwise factor sequences. By using LSTM neural network combined with MRF kernels, the model can be more robust and flexible to produce more plausible microarray measurements.

In out network, the LSTM module functions as a recurrent part that can synthesize the sequential data at different time points and better classify the gene interaction patterns. As we mentioned above, the LSTM helps the network to learn dynamic components within the gene interaction in various stages. We use a LSTM which weights through all the time steps in the developmental stage. The network structure is set to be similar to what we mentioned in Sec. 4.1.2.

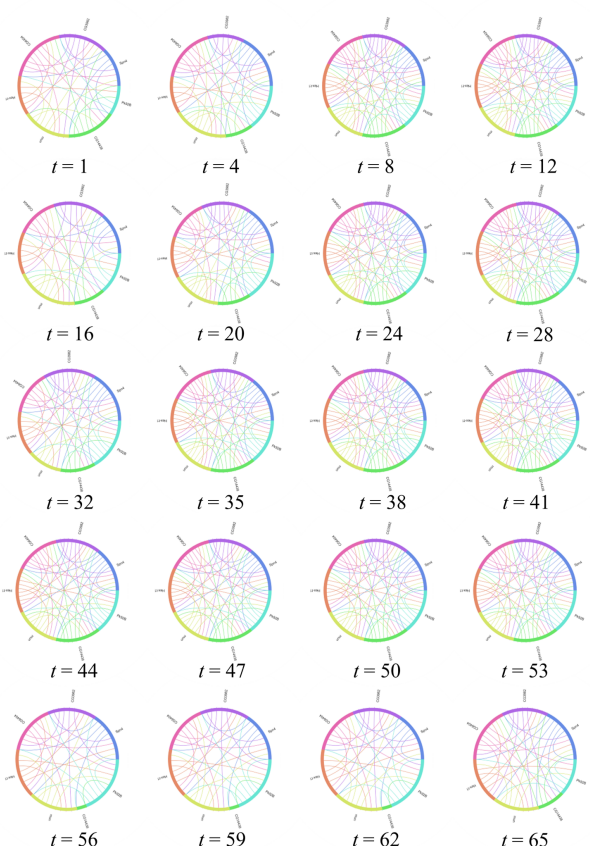


Figure 5. Chord plot for gene interactions for 7 selected developmental-related genes using RBF kernel trained with 100000 iterations. Though there is no clear interactions due to high numbers of edges, we can still notice the different length of local gene interactions. Purple: CG3982, Blue: Spn4, Cyan: Pk82B, Green: CG14438, Yellow: mun, Orange: Hex-t1, Pink: CG6404

The following experiments also follow the previous settings in Sec. 4.1.1, instead, we use a LSTM to replace the former RBF module. Using the deep generative model, we get pairwise factors for evaluation. We evaluate the gene interactions and the network size in various developmental stage for our network. Shown in Fig. 3d, we observe an obvious gene interaction drops during the larval and pupal stages. while the embryonic stage and adult stage, number of active genes is high. When we visit the gene interaction graph as shown in Fig. 4d, there is a clearer pattern during the 4 developmental stages. Our network can learn a better graph which coincide with the biological data. The statistic percentage at each developmental stage lies in $0.27,\ 0.26,\ 0.26,\ 0.27$. Using LSTM helps us to model the time-varying network more accurately with faster convergence.

The average steps for the 20 genes 'Su(z)12', 'emp',

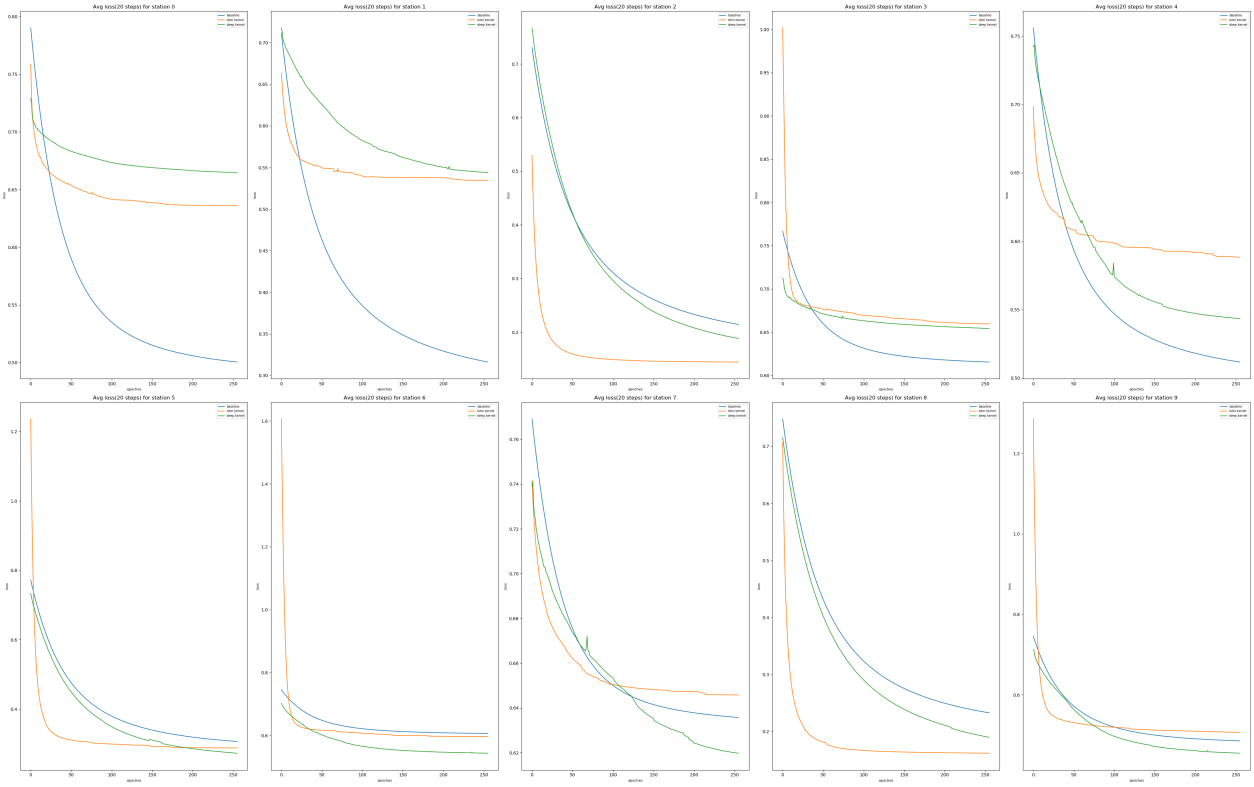


Figure 6. train loss for the first 10 stations temperature observation averaged by 20 time steps

	Suz	emp	CG2	crc	pros	ne	R5	R7	Mp	W	bw	fd5	e	ben	rdo	lin	tra
baseline	20	*	20	15	15	15	*	*	20	12	*	*	*	12	*	13	14
deep-kernel	**15**	*	15	10	12	9	*	*	10	9	*	*	*	10	*	11	9
LSTM-kernel	**15**	**1**	**1**	*	**2**	**2**	**2**	*	**2**	**1**	**1**	**2**	**1**	**1**	*	**1**	**1**

Table 1. Average steps for the probability of genes to converge, rows stands for different methods and columns stands for different genes **bold number** is the least steps for that the probability of that gene to converge where * represents non-convergence.

'CG2678', 'crc', 'pros', 'neur', 'Rab5', 'Rab7', 'Mmp2', 'W', 'bowl', 'fd59A', 'e', 'ben', 'rdo', 'lin', 'tra', 'tra2', 'cathD', 'd' to converge is list as in Table. 1 , by "convergence", we mean the log of the probability reaches a probability which is greater than 0.5 and will not reduce in later epochs. We find that for each gene we tested here, LSTM kernel converges the fastest of all the other methods. Deep neural network kernel is faster than the RBF kernel. The results we found is also consistent with the training loss we observed during the training.

For each time step, as we proceed to genes, using weights generated from deep kernels enables faster convergence than the non-parametric RBF kernels. In particular, LSTM does a better job in learning sequential weights than one-layer neural network. There remains some issues for future improvements: we only use one-layer MLP that can further replace with multiple layers; the training time can extend longer; LSTM can be designed using our own specificity instead of the canonical setting of PyTorch.

4.2. Reevaluate network using wind power load dataset

In order to test the robustness of our network, we use our model to test on wind power load dataset from a kaggle competition [1]. The dataset consists of power load observations from 20 locations and temperature observations from 11 locations. We use only the temperature data for it has lower data dimensions and thus less computing resource demanding. The temperature history data ranges across from year 2004 to year 2008. In order to adapt this dataset to our algorithm framework, we first choose a time range with 100 observations, then find the 11 stations' temperature observa-

[1] https://www.kaggle.com/c/global-energy-forecasting-competition-2012-load-forecasting/data

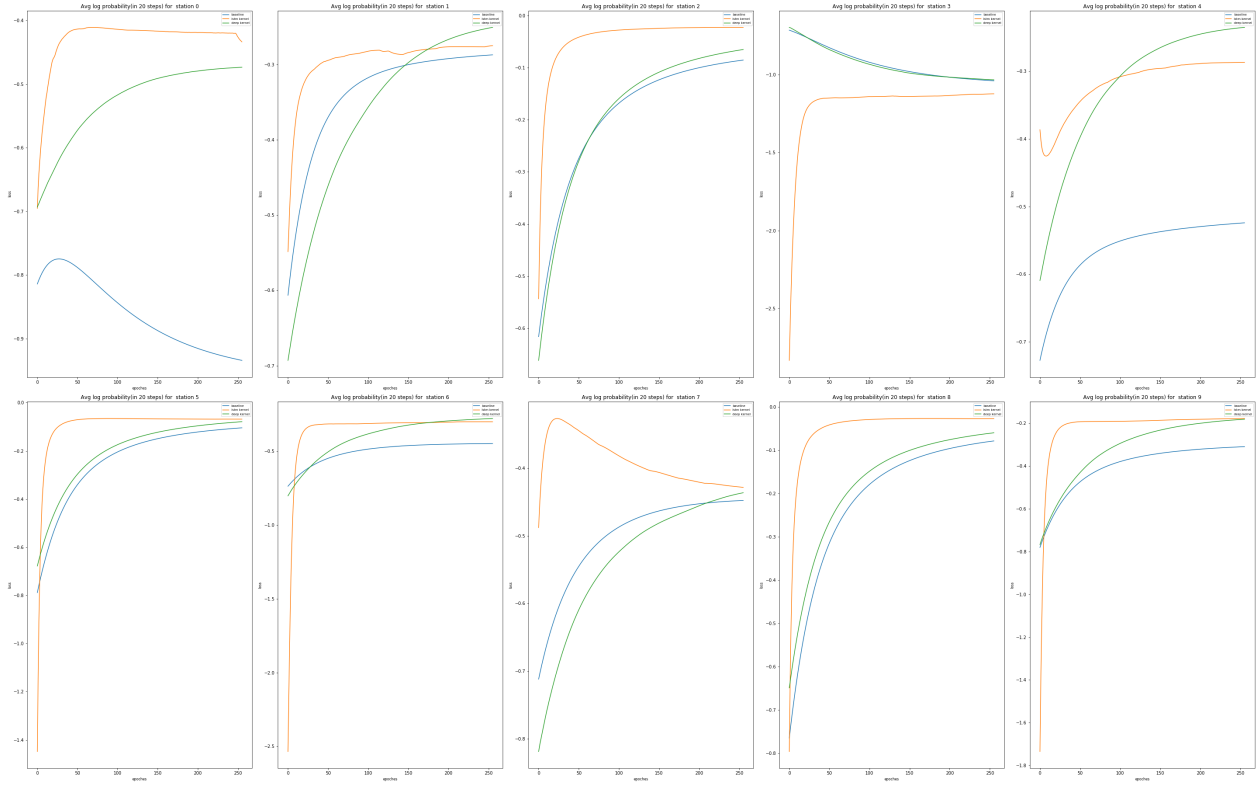


Figure 7. log probability for the first 10 stations temperature observation averaged by 20 time steps

tion data in this time range. Further we select out the data of the first 10 stations in 20 time steps for simplicity. Also we discrete the data to be in $\{0, 1\}$ just like the previous gene dataset. Specifically, we find a threshold related to the median of the temperature observation, and binaries the temperature by setting those observation to 0 which are less than the threshold and setting those those to 1 which are greater than the threshold. For all the experiments which are not specified, we use 2048 iterations. we can process the data as this way when we assume the temperature observations also has some Markov pairwise relations, i.e, the temperature distribution of one station can be expressed by the conditional probability of all other observation stations. This is reasonable since the temperature is a result of global sea circulation in a broad sense and the sea circulation is a composition of interaction activities between different geographic locations.

We plot the training loss with the three methods in previous section and it is in figure 6. There are some obvious conclusions we can draw from the train loss plot.

During training the first 2 stations, the loss for baseline method performs decreases the most rapidly, we can infer human's domain knowledge takes effect since the pairwise kernel weights comes out of our knowledge, then the LSTM kernel and the deep kernel begin to learn the pairwise kernel weights and gradually surpass the baseline method in performance of loss reduction. Noticeably, LSTM method becomes the best method since the station 6, LSTM might infer out the hidden relation structures between these stations with the data of only the first 6 stations. Also we can find that deep kernel method performs better than the baseline method as for convergence speed. In conclusion, this experiment shows that neural network based methods(Deep Kernel, LSTM) is good at infer the hidden structure or relations, even better than human domain knowledge in this case.

To evaluate how the likelihood of a station appears at a timestep, we plot the log probability of each station's observed temperatures during training, it is shown at figure 7.

As in the gene experiment, we also evaluated the average steps for the log probability of station temperature observation, which is expressed as the log conditional probability on all other stations , to converge and show the result as in the table 2. This table is inferred from the figure 7.

5. Conclusion

We propose the use of deep kernels for re-weighting observations from different time steps in learning the network for

	Station0	Station1	Station2	Station3	Station4	Station5	Station6	Station7	Station8	Station9
baseline	*	200	250	220	250	230	200	250	230	250
deep-kernel	200	150	200	230	200	220	200	250	250	250
LSTM-kernel	**50**	**50**	**40**	**50**	**50**	**40**	**50**	**40**	**40**	**50**

Table 2. Average steps for the probability of genes to converge, rows stands for different methods and columns stands for different stations **bold number** is the least steps for the probability of that station to converge

one particular time step in time-varying networks, modeled by pair-wise Markov Random Field. The deep kernel is trained jointly with the parameters from the pair-wise MRF model and shared among all genes for one time step. Logistic Regression Kernel and LSTM Kernel with RBF base function are implemented and tested against using only the RBF model on *D.melanogaster* (Arbeitman MN, 2002) and wind power load data.

In the *D.melanogaster* data set, LSTM kernel has shown a faster convergence and lower training loss compared to logistic regression (one-layer MLP) kernel and RBF kernel, where logistic regression kernel still has faster convergence than the RBF kernel. Moreover, gene interaction pattern which learnt by LSTM kernel considers to be coincides with biological statistics. By comparison, the RBF kernel trained with 100000 iterations show certain improvement for learning time-varying network.

In wind power load data, LSTM kernel has demonstrated faster and more stable convergence in later stages of the training. Deep kernel method also shows its advantage over human designed pair-wise weights respect to training speed and convergence rate. Generally speaking, neural network based methods are good at finding hidden structures and uncertainty properties between different but with similar property identities.

Our experiments has shown that using LSTM kernel for time-varying networks modeled by pair-wise MRF facilitates and expedites the convergence of training compared to non-parametric RBF kernel. This result implies that it can be used in learning involving large data sets. In the future, our goal is to test out the deep kernels on different graphical structures and possibly draw a more extensive conclusion on the use of deep kernels in graphical models.

6. Supplementary

We have some supplementary materials regarding training loss for 3 methods we mentioned before in gene interaction engineering. (Fig. S1)

References

Al-Shedivat, M., W. A. S. Y. H. Z. X. E. Learning scalable deep kernels with recurrent structure. *Journal of Machine Learning Research*, 18(82):1–37, 2017. 2

Alan Graves, A.-r. M. and Hinton, G. Speech recognition with deep recurrent neural networks. *In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 6645–6649, 2013. 2

Andrew Gordon Wilson, Zhiting Hu, R. S. and Xing, E. Deep kernel learning. *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, 51:370–378,, 2016. 3

Arbeitman, M. N., Furlong, E. E., Imam, F., Johnson, E., Null, B. H., Baker, B. S., Krasnow, M. A., Scott, M. P., Davis, R. W., and White, K. P. Gene expression during the life cycle of drosophila melanogaster. *Science*, 297(5590):2270–2275, 2002. 3

Arbeitman MN, Furlong EE, I. F. J. E. N. B. B. B. K. M. S. M. D. R. W. K. Gene expression during the life cycle of drosophila melanogaster. *Science*, 2002. 9

Davidson, E. Genomic regulatory systems. *Academic Press*, 2001. 3

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *roceedings of the 33rd International Conference on Machine Learning(ICML 2016)*, pp. 1050–1059, 2016. 2

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2, 3

Hong, T., Pinson, P., and Fan, S. Global energy forecasting competition 2012, 2014. 3

Song, L., Kolar, M., and Xing, E. P. Keller: estimating time-varying interactions between genes. *Bioinformatics*, 25(12): i128–i136, 2009. 1, 3, 4, 6

Wilson, A. G., Hu, Z., Salakhutdinov, R. R., and Xing, E. P. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pp. 2586–2594, 2016. 1

Figure S1. Training loss over epoch for 20 different genes.