Finding signatures of active loop extrusion – a machine learning approach

Simon Grosse-Holz

The formation of chromatin loops is by now widely accepted as a fundamental principle in genome organization throughout the kingdoms of life. A candidate for the underlying process is active loop extrusion, i.e. the presence of a protein complex that actively moves along chromatin and extrudes loops. While this has been directly observed in vitro for yeast condensin, it remains unclear exactly how loops form in other species and alternative mechanisms have been proposed. A specific issue currently under discussion is whether loop formation is an active or passive process (i.e. driven by ATP hydrolysis or diffusion), which should manifest in the dynamics of individual loci. In this work, we trained a neural network to distinguish between active and passive trajectories of such loci, achieving a precision of $(94.3 \pm 1.1)\%$.

INTRODUCTION

The spatial organization of the genome in the cell's nucleus is a crucial factor in determining gene expression and thus biological functionality. It is also a mathematically and physically interesting problem: the human genome has a linear length of about 2m while the diameter of a DNA strand is about 2nm [1]. It can therefore be thought of as a long, fine thread that is coiled up inside a small volume. But since the genome should also be able to fulfill its biological functions, this coil has to be well-organized, which is a non-trivial and multiscale problem.

It has now become widely accepted that an important process in organizing the genome on intermediate scales is the dynamic formation of chromatin loops [2–5]. This loop formation is usually assumed to be mediated by some loop extrusion factor (LEF), which is able to bind to two nearby chromatin sites and pull them together, c.f. fig. 1. Then moving along the fiber, it is able to pull in more and more chromatin, thereby extruding a loop. The specifics of this process, however, remain unclear and are likely to be different in eukaryotic and prokaryotic cells: while the original model assumed the LEF to actively pull on both strands symmetrically [3], it has now been shown by direct in vitro imaging that the process is asymmetrical for yeast condensin [6, 7], which is a well-known SMC complex (Structural Maintenance of Chromosomes). It is shown to anchor to a fixed site on chromatin with one side, while "reeling in" the fiber on the other side, which is proven to be an ATP dependent process.

In mammals, loop extrusion is thought to be mediated by the related complex cohesin [4, 5]. A crucial question concerns the activity of cohesin: is it a molecular motor, like its relative condensin has been shown to be, or is it passively diffusing along the chromatin fiber? Using the latter model, the authors of [4] were able to reproduce signatures usually linked to active loop extrusion. It thus remains unclear whether cohesin actually is a molecular motor.

An experimentally viable setup to investigate the activity of cohesin is tracking of genomic loci via flourescence microscopy [8, 9]. The basic idea is that almost all of the chromatin fiber is free to diffuse in the cytoplasm, while at each time two specific loci are held together by the cohesin. Since cohesin moves along the fiber (either as an active motor or diffusively), these two sites will be pulled together upon the cohesin approaching, reach a minimal separation, and diffuse away from each other after the cohesin stepped over them as shown schematically in fig. 1. Tracking several loci with time resolved flourescence microscopy, it should be possible to find such events and from their dynamics infer whether the process involved is actively driven or not. For example one would expect an active process to be asymmetric in time. However, due to large thermal fluctuations and other sources of noise, this asymmetry—while being a characteristic meanfield feature—is hard to detect in individual events [9] (see also below).

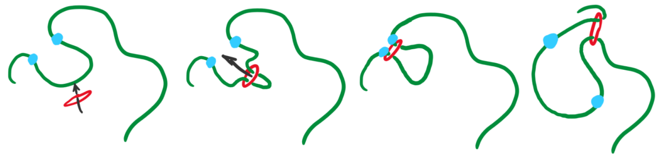


FIG. 1. Mechanism of loop extrusion, including labelled loci. Green: chromatin fiber, red: extrusion factor, blue: labelled loci. From left to right: the extruder associates to chromatin, starts extruding a loop, pulls together the two labelled loci and finally steps over them.

In this work we aim at discerning active and passive encounters of genomic loci. From the discussion above, we classify three possible types of such events:

- two sites being pulled together by an active extrusion factor
- a passive extrusion factor diffusing along the fiber and bringing subsequent pairs together
- encounters which do not involve any extrusion factor, i.e. are purely due to diffusion of the fiber.

Clearly for a long fiber which is not too densely crowded with extrusion factors, most events will be of the purely passive type. The challenge is then to filter out the ones mediated by an extruder. For the time being, we shall focus on the (presumably) easier problem of discerning actively extruded from purely passive events and leave the intricate case of a passive extruder (where the events are expected to be much more similar to the purely passive ones) to future investigations. We develop our analysis on a simulated dataset, which will be discussed below.

In the simulated dataset it is of course easy to tell active and passive events apart, since we can simply track the active extruder. This makes the problem susceptible to machine learning, because it is basically a binary classification problem: can we find a pattern that can tell us for individual events whether they are due to an active extruder or simply diffusive? Indeed we were able to train a neural net and classify events with an error rate of $\sim 5\%$. The question then shifts to understanding what the neural net does and according to which criteria it classifies.

PRELIMINARY WORK

Simulations

The simulations on which this work is based are molecular dynamics simulations of a polymer composed of 2000 monomers. The extruder is modelled as a harmonic bond between two of the monomers, moving forward to the next monomer pair every `stp` simulation timesteps. The extruder's velocity is therefore inversely proportional to the simulation parameter `stp`. We ran simulations for `stp = 50, 100, 200, 400, 1600`.

Inbetween the steps of the extruder, the polymer is free to diffuse in the implicit solvent. Implicit solvent in this context means that the solvent is not actually included in the simulation (which would incur high computational costs), but is modelled as giving the polymer random force kicks every now and then. Note that for computability we had to reduce the frequency of these force kicks to an unphysically low value, which results in ballistic behavior of the polymer on short timescales. On longer timescales, this approximation does give the correct behavior [9]. The output of the simulation is a time series of polymer configurations, each separated by `stp` simulation timesteps from each other, such that there is one configuration for each position of the extruder. These polymer configurations are the starting point of the present work; the simulations were done by J. Nuebler [9].

Close encounter events

Nomenclature: An *event* is the close encounter of two monomers that have a given minimum separation along the polymer. Tracking these two monomers, we obtain a *trajectory*, which plots the relative distance of the monomers over a *window* of time around the event. Therefore, all trajectories are normalized, in the sense that they all have the same length and there is always an event at the same time.

We used a minimum separation of 40 monomers, as well as a minimum separation of 20 monomers from the end of the polymer. The window extends 100 extruder steps before and after the event.

As discussed in the introduction, we are interested in two types of events: the ones due to the extruder actively pulling sites together, which in the following we call "active", and the ones happening due to diffusion of the polymer, which we will call "passive". Finding active events in the simulation data is straight-forward, since the position of the extruder is known for each configuration. For any given time we can therefore look up the monomers currently bound by the extruder, track them for the given time window before and after the event and thus compose a trajectory. For two pairs of monomers, these trajectories are expected to be highly correlated, if the events are close in time[1]. Especially in regard to training a neural network on these data later on, we need the trajectories to be as independent as possible. We therefore do not consider every possible event, but randomly sample a given number of time points, ensure that they are suitably far apart (we take this minimum separation to be the width of the window) and compose trajectories.

In principle, obtaining passive trajectories is not as straightforward, since we first have to find the corresponding events. However, these are sufficiently frequent that we can follow a similar algorithm: sample times at random, find the two monomers that are closest together (and have some minimum distance from the current extruder position and from each other) and compose their trajectory.

With the algorithms described above, we created a trajectory dataset from the simulation raw data. This dataset contains 1000 active and 1000 passive trajectories per value of the parameter `stp`, totaling to 10000 trajectories. To be able to assess the performance of the neural net on data that it has not seen during training, we created a second dataset of the same size. Due to the random sampling involved in the assembly of these datasets, we can assume them to be independent enough

[1] This does imply that the pairs are close along the monomer, in the sense that the left monomer of pair 1 is close to the left monomer of pair 2, and analogous for the right side. However, the converse is not true: two pairs that are close along the polymer may be bound at different times by different extruders. These events would of course be completely independent; it is therefore really only the temporal separation that matters here.

to provide a good measure for the performance of the network. We will refer to these two datasets as the *training set* and the *validation set* respectively (see below).

Manual event analysis

Before blindly handing the dataset we described in the previous section to a neural network, let us take a look at it. In fig. 2 we show the averaged trajectories, for each value of `stp` and grouped into active/passive. The first observation to make is that the active trajectories are obviously not symmetric in time, while the passive ones seem to be. While one might expect this clear difference to be a detectable signature of activity in the trajectories, fig. 3 shows that this is in fact difficult: although showing the expected difference in their behavior on average, the thermal fluctuations in individual trajectories are too big to admit reliable classification according to such a scheme.

The double logarithmic plots on the right side of fig. 2 show the scaling behavior of the trajectories. The passive trajectories should not depend on the speed of the extruder (i.e. the value of `stp`)—or in fact even its presence—since their events are well separated from the current position of the extruder. The reason why the curves do not collapse in the linear plot is that the horizontal axis shows time measured in steps of the extruder. Correcting this for each curve, they collapse to one single curve as expected and shown in the logarithmic plot. As we can also see from the plot, the scaling behavior of the trajectories is different for long and short timescales: for short timescales they roughly follow a square root law, while for long times the exponent decreases and is closer to $\frac{1}{4}$. The latter is actually the behavior one would expect for a site on a freely diffusing polymer chain[2][9]. As was noted above, the short time behavior of the simulation data is expected to be unphysical, since we are working at unrealistically low solvent densities. For this work, which is intended to be a first exploration of the method, we will put this problem aside.

Finally we come to the plot in the lower right of fig. 2, which shows that for the active trajectories the behavior after the event does depend on the speed of the extruder. Note that here the horizontal axis again shows steps of the extruder, as opposed to actual simulation time. Thus from this plot we see that the behavior of the monomers after an active event is mainly determined by how far the extruder has gone since, instead of the time that has passed. This behavior should be expected, if the motion of the extruder is slower than the thermalization of the polymer chain connecting the two sites. Indeed, the trajectories from the slowest extruder (`stp = 1600`) do agree with a square root scaling. For the faster extruders, again large parts of the trajectories belong to the short time regime, where the dynamics become unphysical (see previous paragraph).

[2] This follows from the Rouse model: while free monomers would do a random walk (and thus their distance would scale as a square root), bound monomers are slowed down by the rest of the chain, that they have to drag along. The effective size of this chain grows diffusively (i.e. as a square root) and enters the mean square displacement of the sites linearly, leading to a decrease of the root mean square displacement from square root to quartic root scaling.

CLASSIFYING TRAJECTORIES WITH A NEURAL NETWORK

In the last section we saw that although on average, active and passive trajectories do have very different characteristics, telling individual trajectories apart is nontrivial (fig. 3) and so far, there are no statistical measures that can reliably predict whether a given trajectory is active or passive. Since this on the other hand is a simple classification problem, it should be susceptible to a machine learning approach. We thus implemented a neural network and trained it to predict whether a given trajectory is active or passive. In the following we will assume the reader to be familiar with the basic ideas of neural networks [10, 11].

Structure of the network

The input layer of the network has one neuron per datapoint of a trajectory, such that we can immediately use the datasets mentioned earlier as input data. In the output layer we have two neurons in a one-hot encoding (i.e. one of them is active if the trajectory is active, while the other one reacts to passive trajectories). Inbetween these two, we insert a number of densely connected layers. The precise structure of these hidden layers seems to be largely irrelevant, as long as there is a sufficient number of neurons. For definiteness in the following we always work with 7 hidden layers, each consisting of 200 neurons (same size as the input layer). Anticipating the fact that we later want to analyze what the network does and whether it reacts to specific features in a trajectory, we insert another hidden layer of ten neurons before the output. Ideally, these ten neurons would train to code for features in the trajectory, whose combination would then determine whether it is active or passive. This could give us a clue to the criteria the network uses.

Training on the full dataset

After training the network on the full training set described above, we evaluated it on the validation set. Since

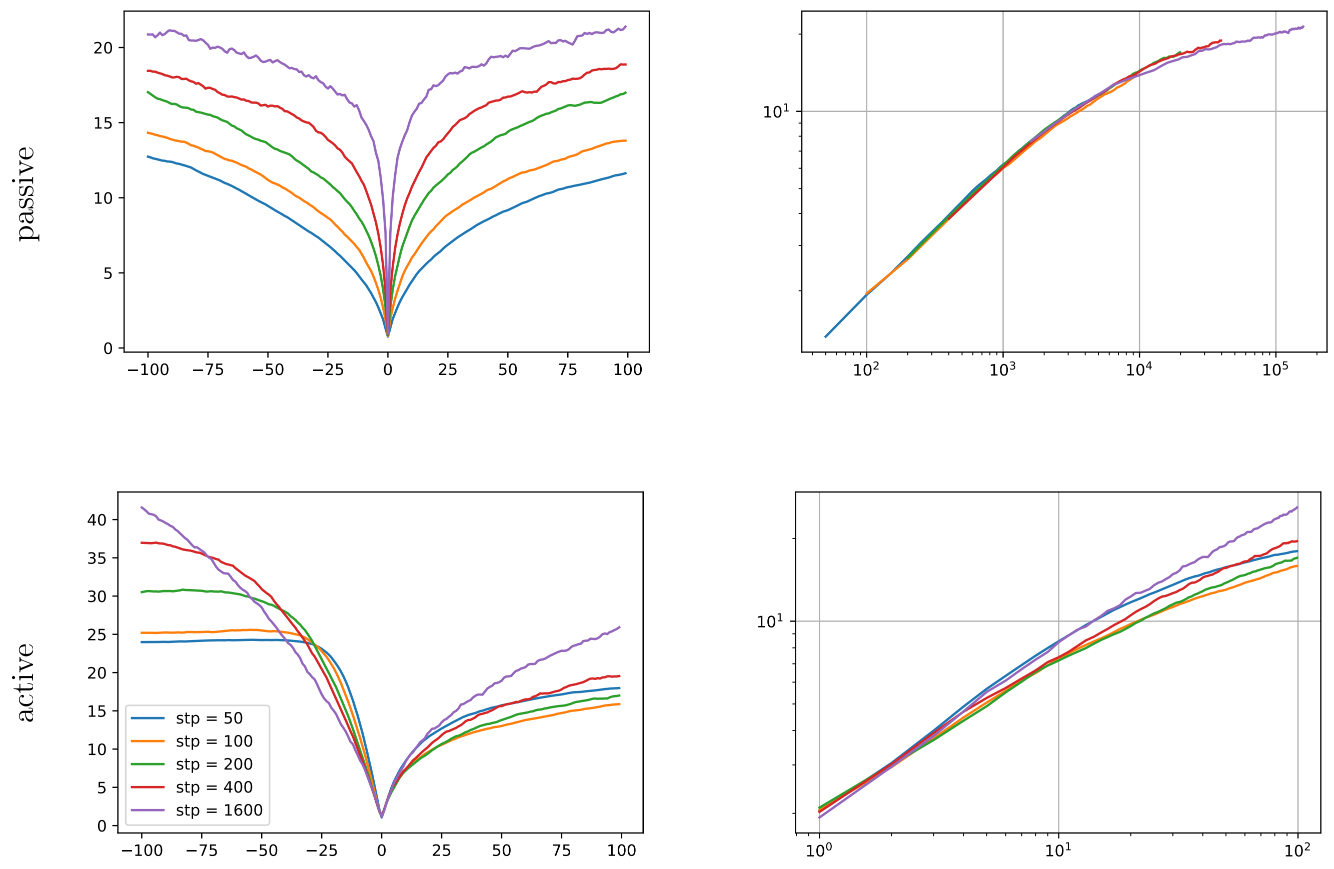


FIG. 2. Averaged trajectories. The top two plots show averages over the passive trajectories, the lower plots apply to active trajectories. Similarly, the left two plots show the full trajectories in linear scale, while the right two plots show a double logarithmic plot of the part after the event, where the loci are free to diffuse. Each curve is the average over the 1000 trajectories pertaining to a given value of the parameter `stp`. The legend in the lower left applies to all plots. The vertical axis shows the relative distance of the two sites, given in multiples of the monomer diameter, while the horizontal axis is in steps of the extruder for all plots except the top right one. There the horizontal axis is time, i.e. it is scaled with `stp` for each curve. See main text for discussion.

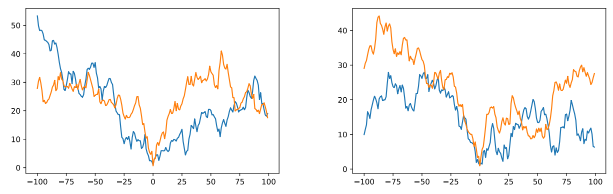
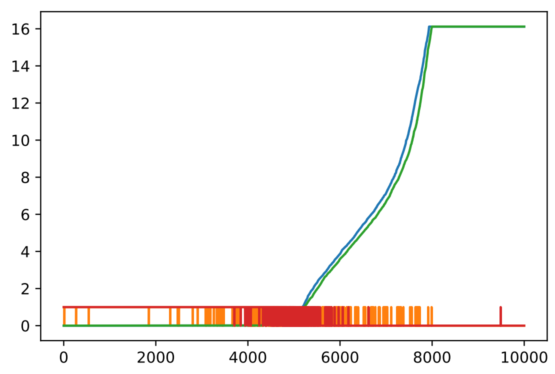


FIG. 3. Examples of individual trajectories, here for `stp = 400`. Blue: active, orange: passive. While the average trajectories nicely show the expected symmetric/non-symmetric behavior (fig. 2), the individual trajectories might not. In fact, in these examples one would rather guess that orange should be the active trajectory. The neural net we will describe below correctly predicts all four trajectories as active/passive respectively. Note: these trajectories are taken from the validation set, i.e. the net has not seen them in training.

training the net is a random process (the weights of the net are initialized at random and the training set is shuffled), we repeated training and evaluation several times to get an estimate on the stability of the performance of the net under reinitialization. It proved to be fairly stable, such that over 11 repeats we found an error rate of $(5.7 \pm 1.1)\%$. Of the wrongly classified trajectories, $(48 \pm 22)\%$ were actives classified as passive. Thus on average the classification works equally well for active and passive trajectories.

A first try at analyzing the behavior of the network might now be to take a look at the trajectories it classifies incorrectly and see whether we can find some pattern there. This is a recursion of the original problem of finding patterns in trajectories and consequently suffers the same problem: individual trajectories are to a large extent dominated by thermal motion, which makes it hard



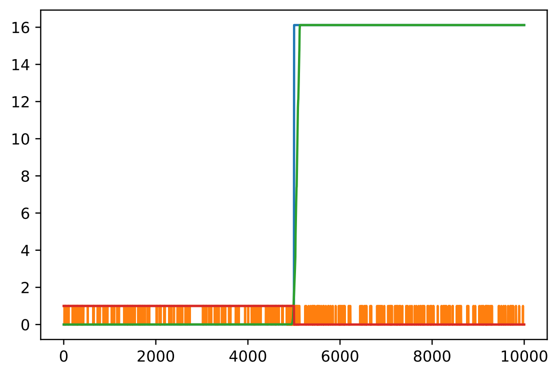


FIG. 4. Cost functions of the decisive neurons in the analysis of the ten neuron layer. Blue: cost function on the training set, green: on the validation set. For each curve, the trajectories on the horizontal axis are sorted according to the cost function, such that the plots are monotonous. The bar on the bottom of each plot then shows the nature of the trajectories: 1 for active, 0 for passive, where red encodes the training set and orange the validation set. Top: trained for 100 epochs, bottom: trained for 1000 epochs, reached perfection around epoch 500 (note the step function in red).

to see patterns. Averaging the error trajectories, one can pursue a meanfield comparison to the averages shown in fig. 2. This shows that—as one might expect—the passive error trajectories tend to have some asymmetries, thus resembling active ones, and vice versa. Apart from this, we did not find any distinctive features in the trajectories pertaining to erroneous classifications.

To further analyze what the net does and how it classifies trajectories, we cut off the output layer (after training) and evaluate the net again, storing the values $-\log n_i$ for each neuron n_i of the ten neuron layer (which is now the topmost layer). Here $n_i \in [0,1]$ is the value of the neuron in question. The given expression is the so-called `'categorical_entropy'` cost function. It is (up to a constant) the Kullback-Leibler divergence from the ten component output vector of the network

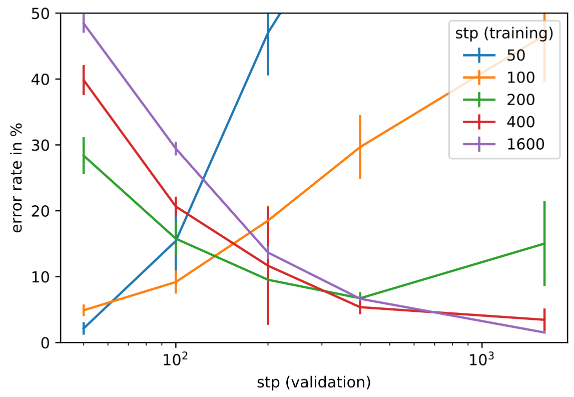


FIG. 5. Testing the transfer capabilities of the network. Colors code the value of `stp` that was used in training, while the values used for validation are given on the horizontal axis. Error rates are averaged over 20 repeats, errorbars indicate standard deviation.

to the Cartesian unit vector corresponding to n_i. For each neuron, we can then ask, which trajectories have the lowest cost function, i.e. strongest excite this specific neuron. In fact the individual trajectories minimizing the cost function turn out to be not very special, again because thermal fluctuations dominate any pattern one might hope to see. However, it turns out to be instructive to plot the cost function over the whole data set, as done in fig. 4. Producing such plots for all ten neurons, one can usually identify one decisive neuron, such as the ones shown in fig. 4, while the remaining 9 neurons are often rather unspecific, in the sense that their cost function does not correlate significantly with active or passive trajectories. Another effect that can be seen in fig. 4 is overfitting when training for too long on a restricted dataset: the decisive neuron adapts more and more to the specific training trajectories, until it reaches perfection, i.e. has learned to classify all trajectories in the set correctly. However, this does not increase the performance on the validation dataset anymore, since the net has simply memorized which trajectory of the training set is active, instead of learning a pattern.

Transfer tests

Ultimately we want to use the neural network—trained on simulated trajectories—to classify real experimental data, which will presumably differ significantly from the simulations. It is thus a crucial question, how well the net can classify data that differ in structure from what it has already seen. Fortunately, we already have such diverse data at hand: the datasets we work on contain trajectories for different speeds of the extruder. We can therefore train the net on only the subset coming from one speed and then evaluate on a subset belonging to another setting. This gives the plots in fig. 5, where we see that the net performs reasonably well for values of the parameter `stp` that are not too far from the training data, while for values at opposite ends of the parameter space the classification does not work anymore.

DISCUSSION

In this work we applied a machine learning approach to the problem of finding signatures of active processes, in this case active loop extrusion. We trained a neural net to discriminate between active and passive trajectories with an error rate of $\sim 5\%$. While there are certainly open problems and room for further optimization (as indicated at suitable positions in the text), this demonstrates that in principle it should be possible to use a neural net to classify these trajectories. More importantly, it shows that there exists a discriminatory criterion for this question, which one might doubt when considering the trajectories in fig. 3. The key question left unanswered in this short exploration is of course how the network manages to classify most of the trajectories. Indeed, this is a well known problem of neural nets and topic of ongoing research.

[1] A. Bates and A. Maxwell, *DNA topology*, 2nd ed. (Oxford University Press, New York, 2005).

[2] J. R. Dixon, S. Selvaraj, F. Yue, A. Kim, Y. Li, Y. Shen, M. Hu, J. S. Liu, and B. Ren, Nature **485**, 376 (2012).

[3] G. Fudenberg, M. Imakaev, C. Lu, A. Goloborodko, N. Abdennur, and L. Mirny, Cell Reports **15**, 2038 (2016).

[4] C. Brackley, J. Johnson, D. Michieletto, A. Morozov, M. Nicodemi, P. Cook, and D. Marenduzzo, Physical Review Letters **119** (2017), 10.1103/PhysRevLett.119.138101.

[5] G. Fudenberg, N. Abdennur, M. Imakaev, A. Goloborodko, and L. Mirny, Cold Spring Harbor Symposia on Quantitative Biology **82** (2018), 10.1101/sqb.2017.82.034710.

[6] T. Terakawa, S. Bisht, J. M. Eeftens, C. Dekker, C. H. Haering, and E. C. Greene, Science **358**, 672 (2017).

[7] M. Ganji, I. A. Shaltiel, S. Bisht, E. Kim, A. Kalichava, C. H. Haering, and C. Dekker, Science **360**, 102 (2018).

[8] S. Shao, W. Zhang, H. Hu, B. Xue, J. Qin, C. Sun, Y. Sun, W. Wei, and Y. Sun, Nucleic Acids Research **44**, e86 (2016).

[9] J. Nuebler, Private communication.

[10] F. Marquardt, "Machine Learning for Physicists," (2017), lecture series.

[11] P. Mehta, M. Bukov, C.-H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, arXiv:1803.08823 [cond-mat, physics:physics, stat] (2018), arXiv: 1803.08823.