Learned Contour Deformations in Lattice Field Theory

Ali Shakir

Supervisor: Dr. Gurtej Kanwar

School of Physics and Astronomy

The University of Edinburgh

August 19, 2025

Declaration

I declare that this work was composed entire by myself, Ali Mehlam Shakir.

Chapter 2 contains a review of the necessary theory around the XY-Model and Markov Chain Monte Carlo (MCMC) simulation. The relevant derivations are not new work, however implementation of the model simulation using existing approaches was carried out entirely by myself.

Chapter 3 contains a review of work previously carried out by my supervisor Dr. Gurtej Kanwar, and his colleagues, along with the details of how we adapt those developments to the specific problem tackled in this work.

The model design and implementation in Chapter 4 is entirely new work carried out by myself with guidance from Dr. Kanwar, and similarly the results reported in Chapter 5 have not been previously explored in the literature to our knowledge. The machine learning approaches that we employed are ones that have been circulating in the literature for some time, however novel modifications were made to adapt them to this particular use case.

Personal Statement

I spent the early days of the project familiarizing myself with key concepts in lattice field theory, and brushing off my skills coding in C so that I could first implement a simulation of the XY-Model first using a simple Metropolis-Hastings update and then using the more sophisticated Wolff cluster update.

Once Dr. Kanwar and I were confident the simulation results were sensible, and that we were able to evaluate key observables of the XY-model accurately, I began reading about the theory behind contour deformations for reducing noise in the two-point correlator. I also spent a large portion of time focused on familiarizing myself with PyTorch so that we could begin studying the use of neural networks to produce optimal contour deformations for the XY-model.

We first began by exploring the use of a one-layer convolutional network to better understand if improving noise on the two-point correlator via contour deformation would be possible. After confirming a substantial improvement, we moved on to more sophisticated methods.

I then spent the majority of the summer refining the design of model we ultimately sought to use, the U-Net, and making modifications to the architecture to incorporate a temperature-dependence.

Throughout August I primarily focused on refining figures and writing up this report full-time.

Acknowledgements

I would like to thank my supervisor Dr. Kanwar for making this project possible and for his invaluable feedback and guidance throughout the research and writing phases of the project. I would also like to thank the Higgs Centre for the financial support offered to me throughout my studies via the Higgs Scholarship.

Abstract

In this project we explore the use of vertical shift contour deformations to improve noise on the two-point correlation function of the two-dimensional XY-model.

The XY-model suffers from noisy estimates of the two-point correlator due to sign problems which we find can be alleviated signficantly through the use of countour deformations whereby the path integral of the system is carried out over a manifold in complexified space. Vertical shift deformations make up a small subset of possible contour deformations in which only an imaginary offset is added to the single real parameter at each lattice site.

We first explore the feasibility of such a procedure using a 1-layer convolutional neural network to produce a the vertical shifts for each site at fixed temperature, then move on to the use of a much larger U-Net based architecture to learn generally how to produce an optimal shift field given an arbitrary temperature.

We find that substantial improvement can be attained through the use of the U-Net architecture, and study the shape of the shift field as temperature and distance are varied, as well as produce new plots of the correlation function to illustrate the improvement in estimation of the correlation length.

Contents

List of Figures

List of Tables

1 Introduction

In this work we are concerned with evaluating two-point correlators of the XY Model in $D = 2$. The XY model in two dimensions assigns a two-component unit vector or "spin" (specified completely by a real angle) to each point on a square lattice with a nearest neighbor interaction proportional to the alignment between spins, and hence is an example of a nonlinear $O(2)$ model. The two-point correlator then measures the degree (on average) to which spins at two separate points are aligned. This model defines the universality class for thin film superfluids (in analogy with superconductivity, superfluidity describes a fluid that flows with zero viscosity), and for this reason is of great practical and theoretical significance.

As the lattice size grows large, explicitly computing the path integral for the theory, which would integrating over all possible directions of all spins on the lattice, quickly becomes intractable, so we turn to Markov Chain Monte Carlo to do importance sampling from the distribution governing field configurations. Given a set of samples, we can compute the alignment of two particular spins in each sample and average over all samples to arrive at the correlator.

Kosterlitz and Thouless [9] showed in 1973 that, despite the Mermin-Wagner theorem forbidding a phase transition due to spontaneous symmetry breaking in two dimensions, the XY model exhibits a phase transition of infinite order at finite temperature. The phase transition is characterized by a diverging exponential correlation length, and so the two-point correlation function of the model is of particular interest in simulation. Precise estimates of the KT transition temperature have previously been made using other observables [7], however a precise understanding of the correlator is limited by the tendency of correlation estimates at large separation to suffer from a severe sign problem. Sign problems arise with highly oscillatory observables, for which estimation becomes noisy when sampling is not also balanced around the center of oscillation.

Past work by Detmold et. al. [2] has demonstrated the feasibility of path integral contour deformations, that is, evaluation of the path integral over a manifold in complexified space, to alleviate the issue of decaying signal-to-noise in estimates of Wilson loops for lattice $SU(N)$ gauge theory. Numerical optimization was used to search for optimal manifolds within a subset of possible deformations dubbed vertical shift deformations, where an imaginary offset is added to the real-valued parameters at each site. Further work [3] has also demonstrated the ability of sufficiently expressive convolutional neural networks to determine optimal vertical shift deformations when trained to minimize variance in a particular observable over a set of Monte Carlo samples.

In this work we seek to apply these developments to evaluation of the two-point correlator in the XY model. We first investigate the use of a simple one-layer convolutional network to evaluate the correlator at fixed temperature and separation, and then move on to a far more expressive U-Net [13] based architecture which additionally implements feature wise linear modulation [12] to incorporate separation and temperature dependence. In both cases, a fully-convolutional network (FCN) is applied to an input image, independent of the field variables themselves, inspired *a-priori* by the geometry of the correlator to generate a field of vertical shifts which are applied to the real-valued angle at each lattice site.

2 Background

In this section we provide an overview of the theory around the XY model and the Kosterlitz-Thouless transition before turning to the key concepts in computational lattice field theory required for simulating the theory and extracting estimates of observables.

I The XY-Model in 2D

a The XY Hamiltonian

The Hamiltonian for the XY model on a square lattice Λ with spacing a is as follows:

$$H[\theta] = -\sum_{\mathbf{r}r\in\Lambda}\sum_{\mu=1,2}\cos(\theta(\mathbf{r}+a\hat{\boldsymbol{\mu}})-\theta(\mathbf{r}))$$

Where we write the two-component vector describing the position of an angle in the lattice as $\mathbf{r}$ and $\hat{\mu}$ is the unit vector pointing in the μ-th direction. The cosine simply describes a dot product between two-component unit vectors, which are fully described by one real-valued angle. It can be seen immediately that the interaction is highly localized — each spin interacts only with its nearest neighbors. This fact places the XY model in the regime described by the Mermin-Wagner theorem, which states that continuous symmetries (here $O(2)$ symmetry) cannot be broken spontaneously at finite temperature in less than three dimensions.

The partition function for the system is then readily written as

$$\mathcal{Z} = \prod_{\mathbf{r}\in\Lambda}\int_{-\pi}^{\pi} \mathrm{d}\theta(\mathbf{r})\ \exp\left(-\beta H[\theta]\right) \tag{1}$$

integrating the Boltzmann factor over all degrees of freedom, i.e. all $L\times L$ sites on a lattice of length L. In principle we are concerned with the infinite L regime however for the purposes of simulation we simply choose L to be large enough to study the long-range correlations of the system. In this work we will use units in which $k_B = 1$ so that β may be interchanged simply with $1/T$

In order to make progress with studying the model analytically, it is necessary to go to the continuum limit. Here we follow the derivation shown in [4]. We start by considering the regime where the field θ is slowly varying, so we may expand the cosine to second order:

$$\cos(\theta(\mathbf{r}+a\hat{\boldsymbol{\mu}})-\theta(\mathbf{r})) \approx 1-\frac{1}{2}\left(\theta(\mathbf{r}+a\hat{\boldsymbol{\mu}})-\theta(\mathbf{r})\right)^2$$

as well as expand the field around $\mathbf{r}$

$$\theta(\mathbf{r}+a\hat{\boldsymbol{\mu}}) \approx \theta(\mathbf{r}) + a\partial_\mu\theta(\mathbf{r})$$

Then, bringing out a factor of a^2 and carrying out the sum over μ allows us to take $a\to 0$ and change the sum over sites for a continuous integral:

$$\int \mathrm{d}^2 r \, \frac{1}{2} \left(\nabla\theta(\mathbf{r})\right)^2 \tag{2}$$

Where we have dropped the infinite constant associated with the zero-th order term in the expansion of cosine.

This form of the Hamiltonian allows for explicit calculation of some observables, particularly the pointwise correlation in the slowly varying (i.e. low temperature) regime. The continuum form of the Hamiltonian will again be relevant later as we discuss the analytic approach to deforming the path integral.

b Correlation Functions

The form of the Hamiltonian shown in eq. 2 can be easily fourier transformed, and the lattice spacing reintroduced:

$$\int \frac{\mathrm{d}^2 k}{(2\pi)^2} \frac{1}{2} k^2 \left|\theta(\mathbf{k})\right|^2 \to \frac{a^2}{L^2} \sum_{\mathbf{k}} k^2 \left|\theta(\mathbf{k})\right|$$

where, for a real-valued field we uses $\theta^*(\mathbf{k}) = \theta(-\mathbf{k})$ and the sum runs over the first Brillouin zone.

Shown in detail in [4], the field can be decomposed into its real and imaginary parts, so that the spin-spin correlation

$$C(\mathbf{r}) = \langle e^{i\theta(\mathbf{r}) - i\theta(\mathbf{0})} \rangle = \frac{1}{\mathcal{Z}} \int \mathcal{D}\theta \, e^{i\theta(\mathbf{r}) - i\theta(\mathbf{0})} e^{-\beta H[\theta]}$$

may be evaluated using the standard techniques of gaussian integration.

The correlation function depends only on the magnitude of the separation and takes a power-law form proportional to

$$g(r) \propto \left(\frac{\pi r}{a}\right)^{-\eta(\beta)}$$

illustrating that, in the low temperature regime the system exhibits an infinite correlation length ξ, as ξ is defined through (in two dimensions)

$$g(r) \propto \frac{e^{-r/\xi(\beta)}}{r^{\eta}}$$

in the language of statistical field theory, the XY model has a quasi-long-range order at low temperature, and therefore must undergo a phase transition at finite temperature before reaching the disordered phase. This is the Kosterlitz-Thouless Transition.

c Observables

The particular observable which is thought to jump discontinuously at the KT transition is the helicity modulues Υ which measures the response of the lattice (with periodic boundary conditions) to a twisting of the spins along the boundary by an angle α. On the lattice, the so-called Nelson-Kosterlitz jump is smoothed out by finite-size effects, however the drop becomes increasingly sharp as L is increased.

Υ is formally defined through

$$\Upsilon = -\frac{\partial^2 \mathcal{Z}(\alpha)}{\partial \alpha^2}\Big|_{\alpha=0} = \\ \frac{1}{L^2}\left\langle \sum_{\mathbf{r}\in\Lambda} \cos(\theta(\mathbf{r}) - \theta(\mathbf{r} - a\hat{\mathbf{x}}))\right\rangle - \frac{\beta}{L^2}\left\langle \left(\sum_{\mathbf{r}\in\Lambda} \sin(\theta(\mathbf{r}) - \theta(\mathbf{r} - a\hat{\mathbf{x}}))\right)^2\right\rangle \quad (3)$$

The choice of the x direction here is arbitrary, and if so desired, Υ can be recomputed exchanging x for y to average the two results and drive noise down further.

Other quantities of diagnostic interest are the magnetization per site, obtained from averaging together all spin vectors on the lattice:

$$\left\langle \frac{\mathbf{M}}{L^2} \right\rangle = \left\langle \frac{1}{L^2}\left(\sum_{\mathbf{r}\in\Lambda} \cos(\theta(\mathbf{r})),\ \sum_{\mathbf{r}\in\Lambda} \sin(\theta(\mathbf{r}))\right)\right\rangle = \left\langle \frac{1}{L^2}\left(\mathrm{Re}\left[\sum_{\mathbf{r}\in\Lambda} e^{i\theta(\mathbf{r})}\right],\ \mathrm{Im}\left[\sum_{\mathbf{r}\in\Lambda} e^{i\theta(\mathbf{r})}\right]\right)\right\rangle$$

and the related magnetic suscptibility (per spin) obtained by analyzing the response of the magnetization to the addition of an external field $\mathbf{h} = (h,\ 0)$ which contributes a term $-h\sum_{\mathbf{r}\in\Lambda}\cos(\theta(\mathbf{r}))$ to the Hamiltonian:

$$\chi = \frac{\partial \langle |\mathbf{M}| \rangle}{\partial h}\Big|_{h=0} = \frac{1}{T}\left(\langle |\mathbf{M}|^2\rangle - \langle |\mathbf{M}|\rangle^2\right) \quad (4)$$

And the specific heat capacity:

$$c_V = \frac{\langle H^2\rangle - \langle H\rangle^2}{L^2 T^2} \quad (5)$$

As the hamiltonian H for the XY model is also the total energy associated with a configuration.

Previous work by Hasenbusch et. al. [7] has simulated the model up to very large lattice sizes, utilizing several months of compute time to study finite-size effects and narrow in estimates of the KT temperature.

II Lattice Field Theory

a Markov Chain Monte Carlo

The partition function for the XY model on an $L \times L$ square lattice 1 necessarily involves an integral over all $L \times L$ real variables. Brute-force evaluation of this integral quickly becomes infeasible as L grows large, therefore we employ random sampling from the distribution to approximate the integral. Markov Chain Monte Carlo randomly walk around the domain of integration, preferentially moving to areas with large contributions to the integral.

The basic idea is to draw field configurations sequentially $\theta^{(0)} \to \theta^{(1)} \to ... \to \theta^{(t)}$ that will be distributed according to the Boltzmann measure as $t \to \infty$. The update that takes $\theta^{(i)} \to \theta^{(i+1)}$ must satisfy four conditions [6]:

- The probability of sampling a configuration $\theta^{(t)}$ depends only on the previous configuration $\theta^{(t-1)}$
- Any field configuration must be reachable in a finite number of updates
- The chain of configurations sampled must be aperiodic
- The detailed balance condition: $e^{-\beta H(\theta^{(t)})}\mathrm{Pr}(\theta^{(t)} \to \theta^{(t')}) = e^{-\beta H(\theta^{(t')})}\mathrm{Pr}(\theta^{(t')} \to \theta^{(t)})$

As pointed out in [6], the last of these conditions is perhaps the most nebulous. In brief, if the chain converges to some distribution asymptotically, it must then be invariant under translation in time, or *stationary*. Enforcing this property gives rise to detailed balance. Detailed balance is a sufficient but not necessary condition for an MCMC algorith. Some popular update strategies, like the Swendsen-Wang algorithm, do not satisfy detailed balance.

In practice we must also account for *thermalization time* t_0, that is, if the initial field configuration is very far from the energetic minimum, some number of steps will be required before the field is consistently fluctuating about equilibrium.

Once a set of field configurations has been gathered, estimating an observable $\mathcal{O}$ that is a function of the field is as simple as computing

$$\langle \mathcal{O} \rangle \approx \frac{1}{t - t_0} \sum_{i=t_0}^{t} \mathcal{O}[\theta^{(i)}]$$

This estimate will become exact as t runs to infinity.

We may additionally be interested in estimating the noise on our observable, in which case the more sophisticated methods of bootstrap or jackknife resampling are required. There is also the issue of *autocorrelation time*. We first define the the *autocorrelation*, which measures how correlated the samples in a sequence are with one another, as:

$$\rho(t) = \frac{\left\langle \mathcal{O}[\theta^{(i)}] \cdot \mathcal{O}[\theta^{(i+t)}] \right\rangle}{\sqrt{\mathrm{Var}[\mathcal{O}[\theta^{(i)}]] \cdot \mathrm{Var}[\mathcal{O}[\theta^{(i+t)}]]}} \in [0,\, 1]$$

For a *stationary* series, which sequences of MCMC samples are, this should depend only on t, not on i. Without delving too deeply into the rich field of time series analysis which involves the study of sequences of random variables, we will simply state that the autocorrelation is expected to decay exponentially with some characteristic time.

The Newey-West estimator for the *integrated autocorrelation time*, which estimates the number of correlated samples that are equivalent to one independent sample, is

$$\hat{\tau} = 1 + 2 \sum_{k=1}^{N} \hat{\rho}(k) \left(1 - \frac{k}{N+1}\right) \tag{6}$$

With hats denoting estimators of these quantities using the available data. This is basically a "total" correlation, which estimates the area under the autocorrelation curve. We can use this integrated autocorrelation time to determine the size of the blocks required to block bootstrapping, a method where we break the sequence into overlapping blocks and resample them with replacement to estimate how much the mean or variance of the sequence would itself vary if we were to run the MCMC over and over again.

Autocorrelations mean in practice that we only keep every n-th sample with n on the order of τ to optimize the tradeoff between gaining new information and the computational cost of saving out field configurations and/or observable values. The presence of correlations between successive data points means that estimates of the mean of those points will converge to the true value far more slowly.

A common choice of MCMC update strategy is the Metropolis-Hastings algorithm, however Metropolis-Hastings is only designed to update one spin on the lattice at a time, therefore it suffers from critical slowing down near the phase transition. Unlike for quantum field theories, there is a solution to this problem for spin systems in the form of the Wolff cluster update [14].

b The Wolff Cluster Update

The Wolff update works by, on each iteration, selecting a random spin and forming a cluster of spins which are all flipped at once. The algorithm works as follows, denoting the spin at site $\mathbf{r}$ by $\mathbf{s}(\mathbf{r})$:

- First a random site $\mathbf{r}_0$ is selected along with a randomly oriented plane (a line in the $D = 2$ case) with normal vector $\hat{\mathbf{n}}$
- The spin is reflected over this plane
- Each nearest-neighbour to the spin with location $\mathbf{r}_1$ is added to the cluster with with probability
$$1 - \exp\{\min[0, 2\beta(\hat{\mathbf{n}} \cdot \mathbf{s}(\mathbf{r}_0))(\hat{\mathbf{n}} \cdot \mathbf{s}(\mathbf{r}_1))]\}$$

When no neighbors are added to the cluster, cluster formation stops and al. spins are flipped over the chosen plane.

This update can be shown to satisfy the conditions for a Markov Chain. In addition to circumventing the problem of critical slowing down, the Wolff update, particularly at low temperature, drastically reduces autocorrelation between successive samples. When only a single spin is flipped, successive samples are extremely similar — on the other hand, when large portions of the lattice are updated, successive samples become decorrelated, and the system thermalizes faster. Clusters tend to consume much larger portions of the lattice at low temperatures, as is clear from the β dependence of probability for adding neighbors.

The Wolff update implicitly assumes a nearest-neighbor dot-product interaction, and therefore additionally avoids recalculation of the Hamiltonian at each update step — an expensive operation that requires looping over every site in the lattice. As an aside, an update strategy of this kind is applicable to all spin system with spin vectors lying in any number of dimensions, allowing its use in simulation of the related Heisenberg and Ising models as well.

c Sign Problems

The two-point correlator is an example of a highly oscillatory observable when the signal becomes weak. In order for the average value of the correlator to approach zero, as it should at large separation, individual samples must necessarily be almost evenly distributed around the unit circle in the complex plane (recall that the correlator is an exponential of a pure imaginary phase factor). 1 provides an elementary illustration of the problem for a hypothetical observable that oscillates in one dimension.

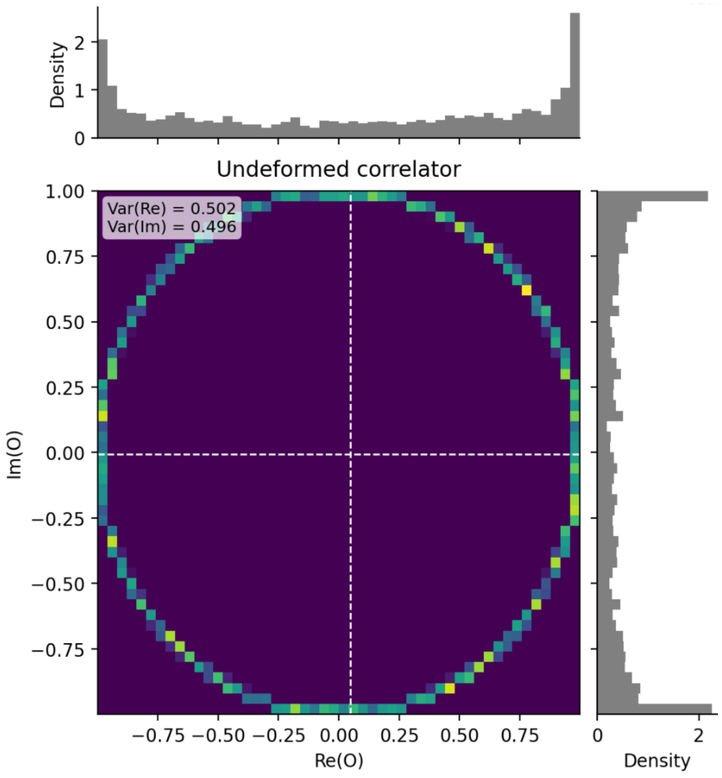

Figure 1: An illustration of a numerical sign problem in one of our simulations of the two-point correlator at temperature $T = 1$ and maximum separation on a 128×128 lattice with periodic boundary conditions, $\mathbf{r} = (64, 64)$. Notice that the points are distributed evenly around the unit circle in the complex plane, meaning that the average will be a result of many cancelling signs. This leads to very noisy estimates of the average that are highly sensitive to small changes in the sample distribution.

d Observable Results

We also present the temperature dependence of the observables 4, 3, and 5 using $18,000$ thermalized samples at each temperature plotted. We assess the autocorrelation time of each observable and apply block bootstrapping [10] to compute 95% confidence intervals on each point. The implementation of our MCMC sampler and data post-processing is discussed at length in section 4.

3 Methodology

I Contour Deformations

a General Theory

The essential idea behind deforming the path integral is to evaluate observables along a manifold in complexified space.

The derivation here is adapted from [2]. Without loss of generality for some arbitrary real-valued field(s) ϕ and an observable $\mathcal{O}$ that is a functional of the field(s), we start with:

$$\langle \mathcal{O} \rangle = \frac{1}{\mathcal{Z}} \int \mathcal{D}\phi \, \mathcal{O}[\phi] \, e^{-\beta H[\phi]} \tag{7}$$

And introduce a change of variables

$$\phi \to \tilde{\phi}(\phi)$$

The observable can then be equivalently written as an integral over the manifold in complex space $\mathcal{M}$ over which the deformed field $\tilde{\phi}$ varies:

$$\frac{1}{\mathcal{Z}} \int \mathcal{D}\tilde{\phi} \, \mathcal{O}(\tilde{\phi}) \, e^{-\beta H[\tilde{\phi}]} \tag{8}$$

The integral 8 can be performed with respect to the undeformed field by introducing a factor of the complex Jacobian $J[\phi] = \det\left(\delta\tilde{\phi}/\delta\phi\right)$

$$\int \mathcal{D}\phi \, J[\phi] \, \mathcal{O}[\tilde{\phi}(\phi)] \, e^{-\beta H[\tilde{\phi}(\phi)]}$$

We then reintroduce the undeformed Hamiltonian and push the Jacobian into the exponential to write

$$\int \mathcal{D}\phi \, \mathcal{O}[\tilde{\phi}(\phi)] \, e^{-\beta\left(H[\tilde{\phi}(\phi)] - H[\phi]\right) + \log J[\phi]} \, e^{-\beta H[\phi]} = \int \mathcal{D}\phi \, \mathcal{Q}(\tilde{\phi}(\phi)) \, e^{-\beta H[\phi]} \tag{9}$$

Where we have defined the deformed observable $\mathcal{Q}$ through

$$\mathcal{Q}(\tilde{\phi}(\phi)) = \mathcal{O}[\tilde{\phi}(\phi)] \, e^{-\beta\left(H[\tilde{\phi}(\phi)] - H[\phi]\right) + \log J[\phi]}$$

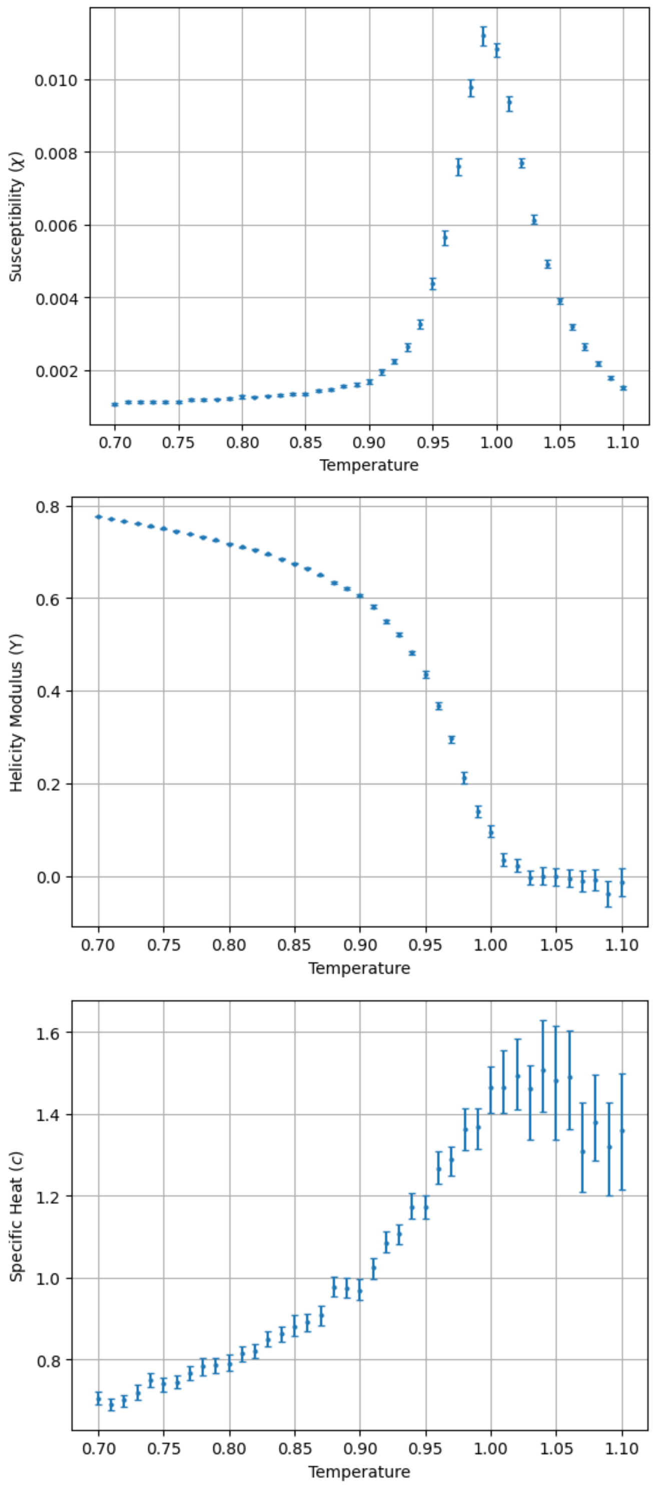

Figure 2: Susceptibility, helicity modulus, and specific heat as a function of temperature with block bootstrapped 95% confidence intervals. We see the characteristic drop in helicity modulus to zero, as well as the peaking of the susceptibility characteristic of the phase transition. The peak in the specific heat is not due to the phase transition but rather another known feature of the model that confirms our sampling is working correctly.

So long as the observable $\mathcal{O}$ has an analytic continuation to the manifold $\mathcal{M}$, Cauchy's integral theorem guarantees (as long as the integrand is holomorphic over the entire domain complexified space) $\langle \mathcal{Q} \rangle = \langle \mathcal{O} \rangle$. The key, of course, is that the variance of the deformed observable need not be the same. In particular, we can see by writing

$$\begin{aligned} \mathrm{Var}[\mathrm{Re}\,\mathcal{O}] &= \frac{1}{2}\left\langle \left|\mathcal{O}^2\right| \right\rangle + \frac{1}{2}\left\langle \mathcal{O}^2 \right\rangle - \mathrm{Re}\left[\langle \mathcal{O} \rangle^2\right] \\ \mathrm{Var}[\mathrm{Re}\,\mathcal{Q}] &= \frac{1}{2}\left\langle \left|\mathcal{Q}^2\right| \right\rangle + \frac{1}{2}\left\langle \mathcal{Q}^2 \right\rangle - \mathrm{Re}\left[\langle \mathcal{Q} \rangle^2\right] \end{aligned} \tag{10}$$

that the first two terms of each equation are not the same, while the final is guaranteed to be the same.

Crucially, as seen in the form of 9, $\mathcal{Q}$ is being sampled with respect to the original Boltzmann measure $\mathcal{D}\phi\, e^{-\beta H[\phi]}$ rather than attempting sampling with respect to the measure of 8. This means we need not make sense of the complex-valued measure (which no longer satisfies the conditions to be a probability distribution). In addition we may still employ the same MCMC algorithm to evaluate the deformed observable.

This is of particular importance for spin systems like the XY mode, which, as aforementioned, make use of the much more sophisticated Wolff cluster update strategy to drastically reduce thermalization time and autocorrelations.

b Vertical Shifts

Evidently exploring the full space of potential deformations is infeasibly, and likely unnecessary to achieve significant reduction in observable variance. Past work by Detmold. et. al [2] has explored restricted spaces of possible deformations, dubbed *vertical shifts* where each lattice site acquires an imaginary offset. In the following equations we return to the discretized case, where the field is represented by $L \times L$ variables $\{\phi_i | i \in 1, 2, ..., L \times L\}$.

This family of deformations is only sensible for variables periodic on some interval $[\phi_0,\, \phi_1]$ where the contour may begin and end at any point on the lines $\mathrm{Re}(\tilde{\phi}) = \phi_0,\, \phi_1$. A visual depiction is given in 3.

$$\tilde{\phi}(\phi) = \phi + i f(\phi) \rightarrow J[\phi] = \det\left(\delta_{ij} + i\frac{\partial f_i}{\partial \phi_j}\right) \tag{11}$$

In [2], the cases of a purely upper triangular Jacobian are explored, as well as the use of numerical optimization for the determination of fourier coefficients for the shift function. In this work we specialize even further to the case where f is simply a position-dependent constant rather than a function of the angles at each site, so the second term in the determinant of 11 drops out and only the identity is left.

For the remainder of our discussion in this section we trade ϕ back for θ, and represent the position-dependent vertical shifts as ψ. The deformed Hamiltonian now reads (using $\langle ij \rangle$ to denote angles i and j are nearest neighbors):

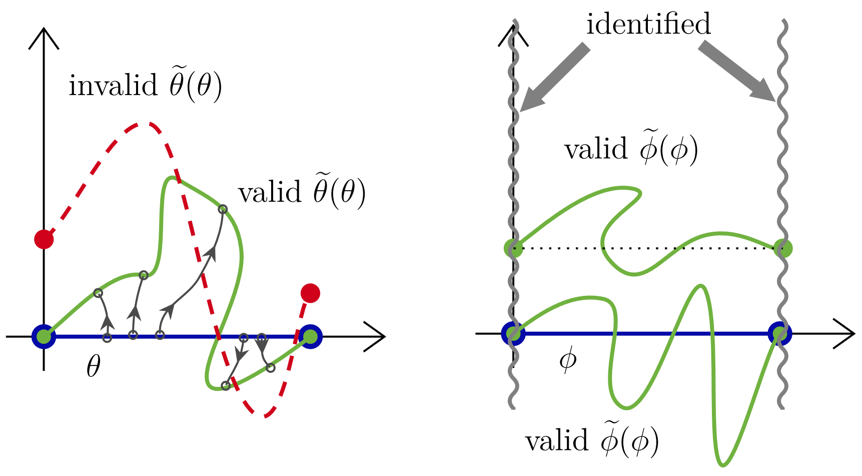
invalid $\widetilde{\theta}(\theta)$
valid $\widetilde{\theta}(\theta)$
θ
identified
valid $\widetilde{\phi}(\phi)$
ϕ
valid $\widetilde{\phi}(\phi)$

Figure 3: Allowed and disallowed contours for a general variable (courtesy of [2]) θ and for a periodic variable ϕ. In the case of a periodic variable the slice along the imaginary axis at each end of the interval is identified, so vertical shift deformations are allowed.

$$H[\phi + i\psi] = -\sum_{\langle ij\rangle} \left(\cos(\theta_i - \theta_j)\cosh(\psi_i - \psi_j) - i\sin(\theta_i - \theta_j)\sinh(\psi_i - \psi_j)\right) \tag{12}$$

The deformed Hamiltonian picks up a phase factor that can be tuned to interfere destructively with the phase of the observable of interest. Additionally, we see that terms like $\cosh(\psi_i - \psi_j)$ which, when entering the exponential, will contribute by reweighting samples. Evidently the shift field cannot vary too quickly, as cosh blows up for large gradients in the shifts. Large gradients will then result in samples being reweighted by the exponent of a very large negative value, in which case information is lost and the variance will suffer.

We also remind the reader that the derivation in this section in no way relies on the change of coordinates chosen. No matter what manifold we choose to evaluate the integrand over, so long as the integrand is holomorphic over that manifold, we are guaranteed the same average value of the observable.

c Deforming the Correlator

As was done to reach 2, we can make analytical progress with 12 by going to the continuum limit and expanding the (hyperbolic) trigonometric functions for the case of small variations in the field and the vertical shifts. sin and sinh do not have a second-order term, while cos and cosh do, making the analytics particularly straightforward. Keeping only terms quadratic in the shifts and in the field:

$$cos(\theta_i - \theta_j)\cosh(\psi_i - \psi_j) \rightarrow 1 - \frac{1}{2}(\partial_\mu\theta(\mathbf{r}))^2 + \frac{1}{2}(\partial_\mu\psi(\mathbf{r}))^2$$
$$i\sin(\theta_i - \theta_j)\sinh(\psi_i - \psi_j) \rightarrow i(\partial_\mu\theta(\mathbf{r}))\cdot(\partial_\mu\psi(\mathbf{r}))$$

Again carrying out the sum over μ and changing the sum for an integral we obtain the continuum deformed Hamiltonian:

$$H[\phi + i\psi] = \int \mathrm{d}^2 r \left(\frac{1}{2}(\nabla\theta(\mathbf{r}))^2 - \frac{1}{2}(\nabla\psi(\mathbf{r}))^2 + i\nabla\theta(\mathbf{r}) \cdot \nabla\psi(\mathbf{r}) \right)$$

It is clear that the phase contributed by the deformed Hamiltonian is

$$-\beta \int \mathrm{d}^2 r \, \nabla\theta(\mathbf{r}) \cdot \nabla\psi(\mathbf{r}) \tag{13}$$

We now require 13 to interfere with the phase of the two-point correlator of the deformed field, which is the exponential of the term

$$i(\theta(\mathbf{r}) - \theta(\mathbf{0})) - (\psi(\mathbf{r}) - \psi(\mathbf{0}))$$

So the overall phase of the deformed correlator can be rewritten as (using $\mathbf{x}$ as the integration variable to avoid ambiguity)

$$\int \mathrm{d}^2 x \, ((\delta(\mathbf{x} - \mathbf{r}) - \delta(\mathbf{x}))\theta(\mathbf{x}) - \beta \, \nabla\theta(\mathbf{x}) \cdot \nabla\psi(\mathbf{x}))$$

In the thermodynamic limit where the boundary lies at infinity, we can integrate the second term by parts at the cost of a sign. Then, setting the phase to zero produces a Poisson equation with a positive source at $\mathbf{0}$ and a negative source at $\mathbf{r}$. The "charge" of each source is scaled by $T = \frac{1}{\beta}$:

$$\nabla^2\psi(\mathbf{x}) = -T \, (\delta(\mathbf{x} - \mathbf{r}) - \delta(\mathbf{x})) \tag{14}$$

This is a strikingly simple result, however we have only considered the behavior of the deformed Hamiltonian to first order, and when gradients of the field become large (i.e. high temperature) the approximations that led to this form no longer hold. With this in mind, the main focus of this work comes into view — finding a numerical approach to determining the optimal shift field. In particular, we will investigate the use of convolutional neural networks to learn the optimal form of ψ.

There is past work [5] on numerical optimization of vertical shift deformations for reducing the sign problem in the $1 + 1$ dimensional XY model with nonzero chemical potential, which we refer the reader to as an example of past work with extended versions of the model we use in this project. To our knowledge however, no past work has applied neural networks for learning the form of the optimal vertical shift contour.

II Machine Learning

Before delving into our particular use case, we will take a brief detour through the theory of convolutional neural networks for readers without a strong background in machine learning. A strong understanding of how convolutions and pooling operations are applied to images or stacks of images is crucial for building the models we use throughout this project.

To set the stage, a neural network is not much more than a series of linear operations applied to an input, with nonlinear *activation* functions stitching together successive outputs before applying the next linear operation. Each of those operations is essentially a tensor of learnable parameters, or *weights* that need to be tuned, or *trained* to optimize the output.

Training the network involves searching the space of possible *weights* in the model to locate a set of parameters that achieves minimization of a loss function. Optimization of networks is a constantly evolving field and the methods we employ in this work have a rich development history. We do not focus heavily on this aspect here, but will include relevant sources as the discussion proceeds. In all cases, the gradient of the loss with respect to each parameter is required, these gradients being computed through *backpropagation* of gradients from the final layer of the network to the earliest layers.

In the case of contour deformations, the loss is formulated in a way that quantifies the total variance of the deformed correlator, possibly across a range of separations and temperatures. As the network weights are updated according to a form of gradient descent, it will learn to find contours that drive this total variance down as far as possible.

a Convolutional Networks

The vertical shift field itself being an $L \times L$ lattice is particularly conducive to the machinery of fully-convolutional (neural) networks (FCNs). A fully convolutional network works by accepting an image and performing a series of convolution and pooling operations to produce another image.

More simply, a convolutional network can be thought of as essentially applying a series of filters to an input image to produce an output image. As opposed to the filters we typically apply to photographs to blur or compress them, which simply involves taking combinations of the values of nearby pixels — these filters are optimized to output an image that minimizes an objective function.

Discrete convolutions in 2 dimensions utilize a $K \times K$ grid of *learnable* weights, or kernel, which is swept across an input grid, using the weights to take linear combinations of values in the input grid as it passes over them and form a new image composed of those combinations. In addition to the $K \times K$ weights, an additional additive factor known as a *bias* is also included. In practice, the convolution may also be applied with nonzero *stride*, in which case the kernel is shifted by more than one site per iteration. This results in a smaller output image. A simple convolution is illustrated in 4

Images may also be padded with extra values around the edges before convolution if the image size is not divisible by the kernel size. Padding is an important step in our use case to ensure periodicity is preserved by wrapping values from one end over to the other in each dimension.

Convolution can be used to increase the size of an input as well, through an operation called *transpose convolution* wherein the kernel of weights moves across one site at a time, magnifying it to a $K \times K$ grid. All the resultant grids are then lined up according to the position they multiplied in the input layer, and overlapping values are summed. A simple transpose convolution is illustrated in 5

We may also desire the input layer at an intermediate convolution to have more than one

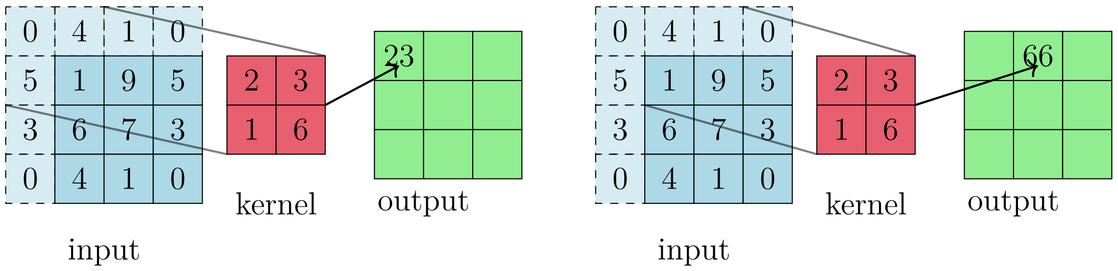

Figure 4: How a 2x2 convolution kernel sweeps over a 3x3 input grid with periodic padding to produce a new 3x3 grid. The first two iterations are shown. Notice the output grid is 3x3, since the kernel can be translated to the right (and down) 3 times before reaching the edge of the input. A stride of 2 for example would produce a 2x2 output

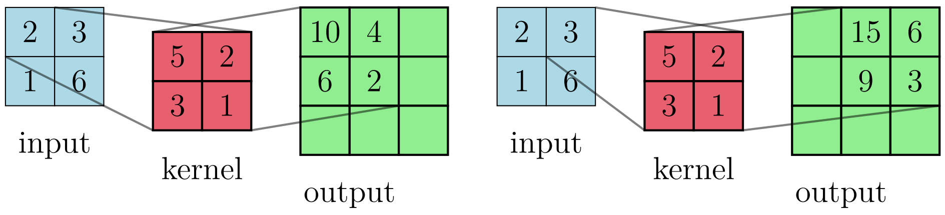

Figure 5: How the transpose operation works by magnifying each site of the input image. Notice that the next iteration shown on the right will overlap the values 4 and 2 from the first iteration. After all iterations are performed, the resultant grids are summed together (with the blank spaces being zero)

layer, or *channel*, as well as to produce an output image also with multiple channels. As an example, consider a typical RGB image, which necessarily has 3 channels describing the amount of each color present in each pixel. We may think of the stack of three images as a 3-dimensional object, which is now swept over by a kernel of size $3 \times K \times K$. Alternatively, this can be thought of as sweeping a kernel over each channel and adding together the results from each channel at each iteration to "collapse" the channels to a single channel. If we then want the output image to have N channels, we'll require N independent $3 \times K \times K$ kernels, each producing one of the N output channels.

Finally, pooling operations also sweep a kernel across the input with some stride and padding, however rather than multiplying values in the input they simply perform an operation such as averaging the values overlapped by the kernel, taking the maximum or minimum, or some other simple arithmetic operation. No learnable parameters are involved which must be optimized. Pooling operations are typically implemented with a stride that is equal to the size of the kernel so that the size of the input image is decreased significantly, as the goal is the compress the input information into a smaller amount of meaningful data.

We can form highly expressive networks by chaining the aforementioned operations together with nonlinear operations, or*activation functions* in between applied to each pixel/site in the layer outputted by the previous convolution.

This is exactly what we will do in creating the U-Net architecture described later in this section. In the past, the primary use of FCNs has been for the task of *image segmentation*, which involves detecting the outlines of objects in an image. The input in that case is the image itself, and the output a usually black-and-white image depicting the outlines of various objects. In our case the input image is a single channel mask inspired *a priori* by the geometry of the correlator.

b Multilayer Perceptrons

We will also require the use of *Multilayer Perceptrons* (MLPs). These objects are designed to operate on scalar or vector inputs rather than images, and are often used in conjunction with convolutional networks to produce scalar outputs for regression or classification tasks.

The MLP is most simply thought of as a form of nonlinear regression. An input vector $\mathbf{x}$ with n components passes through a linear transformation $\mathbf{A} : \mathbb{R}^n \to \mathbb{R}^m$ to produce a vector with m components, with a single learnable *bias* also added to all components. Each component passes through a nonlinear function f before the processes repeats again some number of times. Each linear transformation is a matrix of learnable weights.

We can represent these networks with flowchart-like diagrams, where intermediate layes are represented by a set of nodes for each component and lines between nodes indicate a learnable weight. An example of such a diagram appears in 6.

c The Input Mask

As we saw in 14, the optimal vertical shift deformation for the two-point correlator takes the form of a source at the origin and a sink at the point of interest. In the continuum limit and with a small gradients approximation, we found the shift field that cancels the phase of the correlator to take the same form as the electric potential due to two point charges in two dimensions.

While we expect that the more sophisticated neural network architecture should be able to account for higher-order effects that our approximation drops, the general process for producing the shift field should be by "smearing out" opposite-sign point sources at the origin and second point. 7 demonstrates this idea with a shift mask that behaves like an electric potential in two dimensions.

In our implementation, discussed in the next section, we will handle generation of the mask internally, so that the network is only required to take a coordinate (x, y) on the lattice as the input.

To summarize, the network will seek to learn a map from this input mask, informed by the structure of the correlator, to a field of vertical shifts that will be used to deform the correlator. We construct an objective function that quantifies the variance in the deformed correlator obtained using the network's output, and numerically optimize the network's weights by minimizing this loss. It is also worth reiterating as well that no matter what contour is learned, we have a theoretical guarantee that the average will

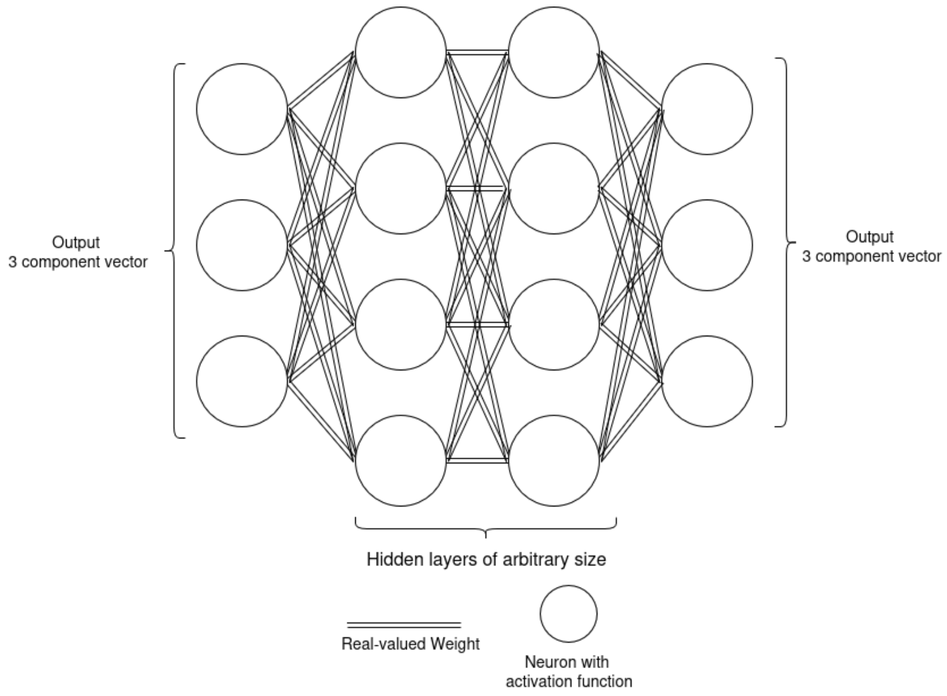

Figure 6: A diagram demonstrating the structure of a typical MLP. Here we show a 3-dimensional input going to a 3-dimensional output with hidden layers in between. Each node of every layer receives a linear combination of all the nodes from the previous layer, and then applies an activation function before passing it on to the next layer. The input and output layers can be tuned to the specific task, and sometimes a transformation is applied to the output depending on the regression task.

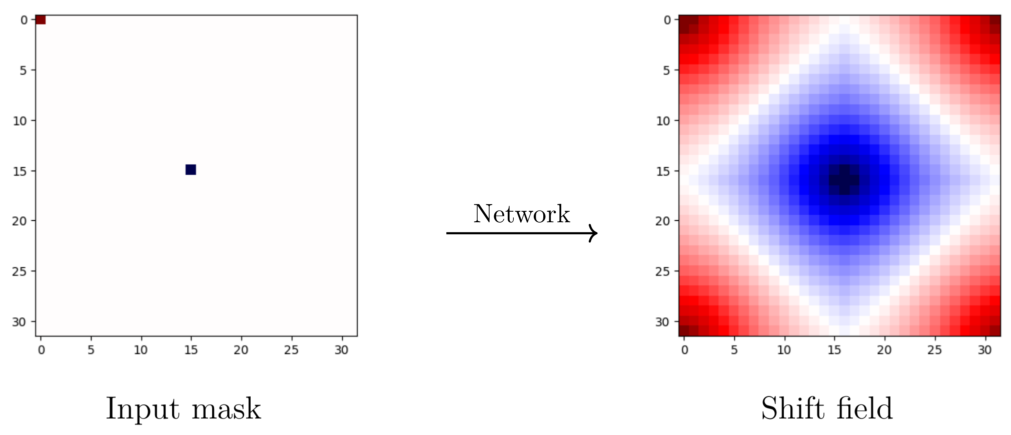

Figure 7: The input mask (left) with point sources at $(0,0)$ and $(16,16)$ on a 32×32 lattice passes through a convolutional network. to produce a shift field which is "smeared out" across the entire lattice.

converge to the same value as in the undeformed case. Noisiness or bias (which we always want to avoid regardless) in the training process has no bearing on that result.

d 1-Layer Network and Loss Function

As a springboard for further investigation, we began by using a single convolution applied to the input mask. With only one convolution, the network only has $K \times K$ parameters, meaning the training routine to optimize these parameters is quick, as is generating the shift masks. The limited expressivity of such a network means that we cannot expect it to generalize well to arbitrary separation and temperature.

With one convolutional layer there is also the fact that the point sources on the input mask can only be smeared out as far as $K/2$ from their position. For this reason we might expect the 1-layer network to work best when the "true" shift field is highly localized around the source and sink. Another way of looking at this is using the analogy to electrostatics — the electric field due to two point sources that are sufficiently far apart will mimic the field due to only one of the charges when in the vicinity of that charge.

Now is also a good time to discuss the basic single separation, single temperature loss function that will later become a component of the generalized loss used for training a more sophisticated model.

Inspired by the form of the variance for the deformed observable 10, we optimize for the variance of the real part of the deformed correlator, writing the loss function for the network as

$$\mathcal{L} = \left\langle (\mathrm{Re}\,\mathcal{Q})^2 \right\rangle \tag{15}$$

We could also use the imaginary part, but the real part is inspired by the undeformed observable:

$$e^{i\theta(\mathbf{r})-i\theta(\mathbf{0})} = \cos\left(\theta(\mathbf{r}) - \theta(\mathbf{0})\right) + i\sin\left(\theta(\mathbf{r}) - \theta(\mathbf{0})\right)$$

The $O(2)$ symmetry of the model mandates that the second term average to zero, in fact it will be symmetrically distributed around zero. As we will see later when discussing results, minimizing the square of the real part of the deformed observable will have the effect of "squeezing" the distribution of the real part as much as possible via vertical shift deformation, while of course the mean is preserved by Cauchy's theorem.

e The U-Net, FiLM, and Generalized Loss

In order to build a network that is sufficiently expressive for generalizing to arbitrary temperature and separation, we turn to the U-Net architecture, originally proposed in [13], with some adjustments.

The U-Net was originally developed for biomedical image segmentation, to identify the structures formed by neurons in 2D scans from electron microscopes. The original U-Net consists of three basic steps:

- The *encoder*: Each encoder level consists of a convolution to double the number of channels followed by a SiLU activation 8

  Feature-wise linear modulation (FiLM), described further below, is then applied across the channels before another convolution preserving the number of channels with a SiLU is performed.

  Max pooling is then applied to reduce each dimension of the input by a factor of two before the process is repeated.

- The *decoder* step: Each decoder level first performs a transpose convolution to double each dimension and halve the number of channels. Then, a convolution halves the number of channels again (to offset doubling by the skip connection) before another convolution. The process then repeats

- *Skip connections* The skip connections grab the final stage channels of each encoder level and carry them over to be appended to the corresponding decoder level, doubling the number of channels in the first stage of that level.

  The idea behind the skip connection is that it allows the network to preserve long-range/large-scale features that are identified in the larger images during encoding and carry that information over to decoding.

The general shape and description of the operations involved in the U-Net is shown in 9. The only unfamiliar operation here is FiLM, which we will take a brief detour to describe next.

The original implementation of the U-Net required only these three steps. Because we have a scalar input (temperature) in addition to the mask, we need a way to feed that information into the encoder steps effectively. The work of [12] provided inspiration for how to do this effectively. The implementation of Perez et. al. is significantly more involved than what we require, but the basic idea is easily adaptable to any scenario where we might seek to condition a convolutional network on information that is not conducive to an image-like representation.

If at some stage of the network, in our case right after the first convolution of each encoder step, we have an image with C channels, we

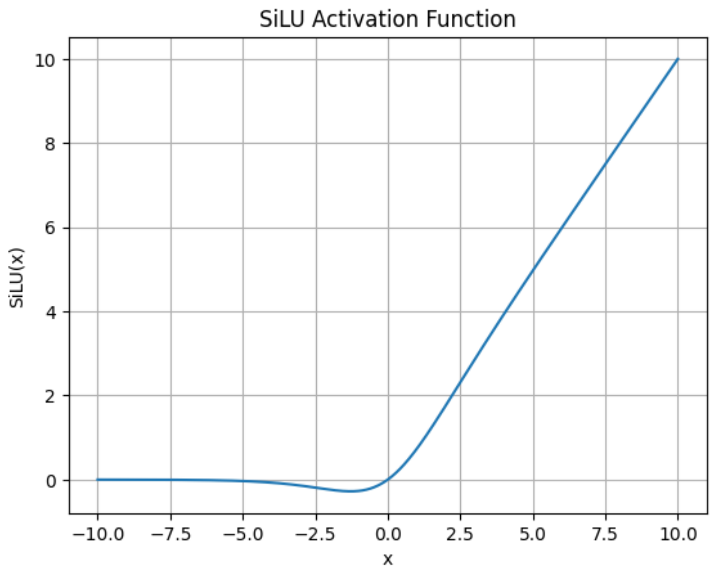

Figure 8: SiLU activation function

- treat each channel as an element of a C component vector γ_i with $i \in [1, 2, ..., C]$
- We then use an auxilliary network, in our case the MLP shown in the bottom part of 9 to generate C component vectors α_i and β_i which are *scaling* and *bias* factors, respectively
- Pass the channels $\alpha_i \cdot \gamma_i + \beta_i$ into the next convolution

We will ultimately be working with a 128×128 lattice, and we choose to let the image size at the bottom of the U-Net be 4×4, meaning there will be six FiLM MLPs. In total, there are $120,993$ learnable parameters. The full structure of the model is as described in 1.

In order to train the model to generalize to arbitrary temperature and separation, we must improve on the loss function 15, in which $\mathcal{Q}$ is explicitly a function of $\mathbf{r}$ and implicitly depends on temperature T. We write the generalized loss as

$$\mathcal{L} = \sum_{\mathbf{r}\in\Lambda} g(\mathbf{r}) \int_{T_{\text{min}}}^{T_{\text{max}}} \mathrm{d}T\, f(T) \left\langle (\mathrm{Re}\, \mathcal{Q}(\mathbf{r}))^2 \right\rangle \tag{16}$$

where f and g have *support* on the sets $\{\mathbf{r} \,|\, \mathbf{r} \in \Lambda\}$ and $[T_{\text{min}}, T_{\text{max}}]$ will do. As discussed in [1], optimizing loss functions of this type will lead the network to individually minimize each contribution to the sum/integral, which is exactly what we hope to achieve — optimal performance over all separations and temperatures. We opt for a particular simple choice of the functions f and g, setting them to unity and treating all separations and temperatures on equal footing. If we instead wanting sampling to be uniform in $\beta = 1/T$, we would choose $\rho(T) = |d\beta/dT| = 1/T^2$.

f Training Procedure

In practice, we use 18000 thermalized configurations for temperatures between 0.7 and 1.1 in increments of 0.01, leading to nearly $200,000$ total samples for training of the U-

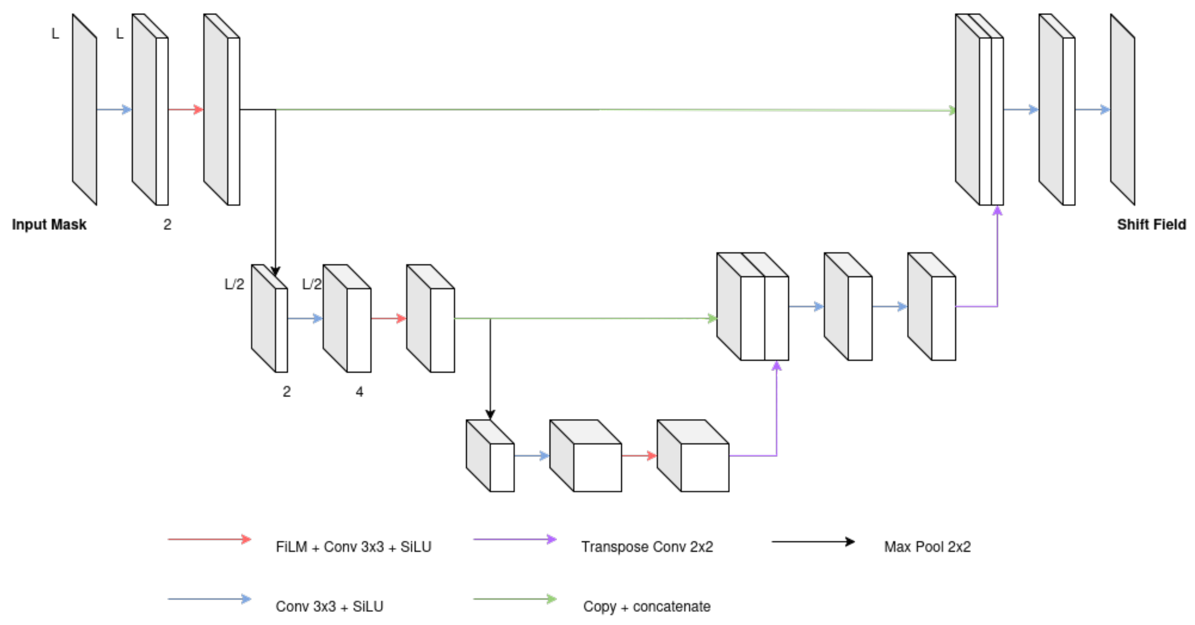

Figure 9: The U-Net architecture we use in this project. Here we only depict the network with 3 levels to the "U", but it is easy to extend to an arbitrary number of levels, as long as the length and width of the layer remains integer, of course. At each level during the encoder step, a FiLM layer is applied to introduce a dependence on temperature. The final layer at each decoder step is saved and appended to the corresponding layer at each decoding step to preserve important long-range structure information. The 3rd dimension of each block in the diagram represents the number of channels, and is labeled in the first two levels. We double the number of channels at each encoder level, and halve them at each decoder level.

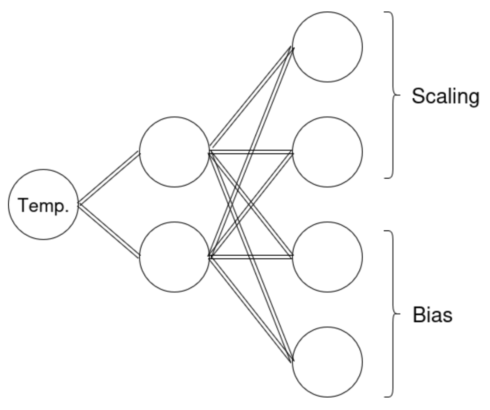

Figure 10: The MLP implicit in each FiLM layer. The scalar input is the temperature and the output is two times the number of channels at each decoder level. Half of the outputs are used for scaling, the other half for bias.

Stage	Operations	Kernel/Stride	Channels	Out Size
Enc 1	Conv (Expand) $\to$ SiLU $\to$ FiLM $\to$ Conv (Refine) $\to$ SiLU $\to$ Pool	$3 \times 3/1$ (Conv) $2 \times 2/2$ (Pool)	$1 \to 2$	64×64
Enc 2	Conv (Expand) $\to$ SiLU $\to$ FiLM $\to$ Conv (Refine) $\to$ SiLU $\to$ Pool	$3 \times 3/1$ (Conv) $2 \times 2/2$ (Pool)	$2 \to 4$	32×32
Enc 3	Conv (Expand) $\to$ SiLU $\to$ FiLM $\to$ Conv (Refine) $\to$ SiLU $\to$ Pool	$3 \times 3/1$ (Conv) $2 \times 2/2$ (Pool)	$4 \to 8$	16×16
Enc 4	Conv (Expand) $\to$ SiLU $\to$ FiLM $\to$ Conv (Refine) $\to$ SiLU $\to$ Pool	$3 \times 3/1$ (Conv) $2 \times 2/2$ (Pool)	$8 \to 16$	8×8
Bottom	Conv (Expand) $\to$ SiLU $\to$ FiLM $\to$ Conv (Refine) $\to$ SiLU	$3 \times 3/1$ (Conv)	$16 \to 32$	4×4
Dec 1	ConvT $\to$ Concat $\to$ Conv (Reduce) $\to$ SiLU $\to$ Conv (Refine) $\to$ SiLU	$2 \times 2/2$ (ConvT) $3 \times 3/1$ (Conv)	$32 \to 16$	8×8
Dec 2	ConvT $\to$ Concat $\to$ Conv (Reduce) $\to$ SiLU $\to$ Conv (Refine) $\to$ SiLU	$2 \times 2/2$ (ConvT) $3 \times 3/1$ (Conv)	$16 \to 8$	16×16
Dec 3	ConvT $\to$ Concat $\to$ Conv (Reduce) $\to$ SiLU $\to$ Conv (Refine) $\to$ SiLU	$2 \times 2/2$ (ConvT) $3 \times 3/1$ (Conv)	$8 \to 4$	32×32
Dec 4	ConvT $\to$ Concat $\to$ Conv (Reduce) $\to$ SiLU $\to$ Conv (Refine) $\to$ SiLU	$2 \times 2/2$ (ConvT) $3 \times 3/1$ (Conv)	$4 \to 2$	64×64
Out	Conv	$3 \times 3/1$ (Conv)	$2 \to 1$	128×128

Table 1: The U-Net structure in tabular format. Expand/Reduce/Refine refer to doubling/halving/preserving the number of channels, respectively. Note that we are implementing pooling and transpose convolution with size 2 to halve/double the image size.

Net. On top of this, there are 16, 384 possible choices of $\mathbf{r}$ on any one of those lattices. Instead of computing $\mathcal{L}$ using the full dataset, we approximate it using small batches. On average, this means the optimization procedure will become noisier, but this stochasticity can be a boon when trying to escape shallow local minima of the loss function.

Each batch will therefore contain B configurations sampled from the total 18000×41 configurations across all temperatures, along with the associated temperatures and a set of randomized separations. Averaging $(\mathrm{Re}\,\mathcal{Q})^2$ across the batch will then provide a noisy estimate of the full loss. Once the full dataset has been iterated through in batches, one *epoch* is completed.

We utilize the ADAM optimizer as proposed originally in [8], which is a variant of stochastic gradient descent that incorporates a momentum an memory of past gradients to boost performance. We refer the reader to the original paper for the details, as this is not a main focus of the project.

Both the 1-layer and U-Net loss functions were optimized in the same way.

4 Implementation

The entire codebase for this project is available here on GitHub. In this section we review the software used in the MCMC sampling, machine learning models, and analysis of the results.

I Data Generation and Storage

The MCMC sampling is done in C. The algorithm first "thermalizes" by running the first 10% of iterations without writing out any data, then writing out every tenth data point over the remaining number of iterations. For example to generate the training data as described in the previous section, 2×10^5 iterations were done to result in $0.9 \times 2 \times 10^4$ samples. The lattice is stored in a one-dimension array of length 128×128 and row-major indexing is used to reference individual angles for computing the Wolff probability.

At each step, the cluster is formed by storing the memory location of each angle added to the cluster in a first-in-first-out stack, and the indices of the angles in the parent array are stored in separate stack so that when cluster formation terminates the stack can be emptied in linear time, updating the angles to reflect the directions of the spins once flipped.

Every tenth iteration, all bits in the lattice array are written out in sequence to a binary file. We then loop over temperature and generate a new file for each temperature. Doubles are used so each angle is allotted 64 bits of precision at the sampling level.

II Observables and Machine Learning Routines

Using the NumPy package, the binary files written out by the MCMC sampler can be read into a Python program in the format of an $N_{\text{samples}} \times 128 \times 128$ NumPy array. We then convert the configurations into PyTorch [11] tensors with angles truncated to 32 bits. Tensors in PyTorch, like NumPy arrays, are computationally must faster to work with as operations on them are heavily parallelized in either C++ or CUDA/HIP depending on the availability of a suitable GPU. PyTorch tensors are also equipped to store gradients with respect to their elements, so as a series of operations are carried out on them, PyTorch builds a compute graph so the gradients can be backpropagated for optimization.

PyTorch also provides a rich neural network class, which can be subclassed to access various modules for convolution and pooling, activation functions, and linear layers for MLPs. Neural networks in PyTorch are also intended to operate directly on batches, so during optimization tensors of shape $B \times 128 \times 128$ along with 1D tensors of length B for associated temperature and ramdom x and y separations are fed into the network. The network output is then used to compute the deformed correlator so the scalar batch loss (actually a 1×1 tensor) can be computed. PyTorch then updates the weights using the ADAM procedure.

Computations were done locally on a laptop equipped with an AMD Radeon RX 7700S, so HIP could be leveraged.

5 Results

In this section we compute the correlator on a *validation* dataset, which is a smaller dataset containing 4500 configurations at temperatures between 0.8 and 1.0 in increments of 1.01. We use a smaller dataset mostly for time, as evaluating the shift field and deformed Hamiltonian are expensive operations. We also require a separate dataset than the data the networks have been trained on to demonstrate that improvements are not simply due to overfitting.

I 1-Layer Network Preliminary Results

The 1 layer network is investigated primarily to understand if a learned vertical shift field can provide meaningful reduction in variance as the theory predicts. As aforementioned, the 1 layer network is only trained on one separation and temperature at a time, which limits its use in computing the correlation function over the entire lattice. We investigate the variance reduction the 1 layer network is able to achieve at maximum separation in both directions, over a range of temperatures, three of which are displayed in 11. We use kernel of odd-number width, 21×21 so that the source and sink will both be centered within the local shifts that are learned in their vicinity.

We also investigate the shape of the shift field learned at temperature 1.0, which is shown in figure 12. We see that the source and sink are learned to be the absolute minima and maxima of the shift field, as we expect from theory, and the magnitude of the shifts falls off smoothly as distance from the source/sink increases. Note that in these plots the y-axis is mirrored, so the origin is the top left.

The deformation has the effect of "squeezing" the distribution of points in both the imaginary and real direction, so that points no longer lie on the unit circle in the complex plane. As seen in the marginal distributions of 11, the points are more tightly distributed around the mean in both directions, demonstrating visually the significant reduction in variance.

Seeing this improvement even with a relatively small kernel provides strong grounds for moving a more expressive network architecture, so that we can train once on the full dataset and produce shift fields for any separation and temperature to compute the deformed correlator across the full lattice.

II U-Net Results

We begin our analysis of the U-Net performance by looking at the same histograms as for the 1-layer network, of course now deforming using the shift field produced by the trained U-Net. These histograms are shown in 13. We immediately see with the same data that there is significantly more improvement across the board, especially at lower temperatures where the data collapses to a tight unimodal distribution around the mean value.

We also investigate the autocorrelation between the undeformed and deformed samples at these temperatures. We might expect that because the deformed observable is now nonlocal and involves a sum over the whole lattice that autocorrelations will be stronger. Recall that the samples drawn are also not at successive timesteps, but rather at every 10th timestep — meaning that for the deformed samples to be sufficiently decorrelated, the lattice must change fairly significantly every ten steps. We plot the autocorrelation as a function of how many 10-step intervals of ten the samples are separated by in 14. It appears that across the temperatures included in the data there is no meaningful degradation introduced by deformation — the autocorrelation curves before and after deforming decay over a similar timescale.

Next we investigate the geometry of shift field over a range of temperatures still at max separation in 15. Before discussing the behavior we note there is an ambiguity in the definition of the vertical shifts which has not been of particular significance until now.

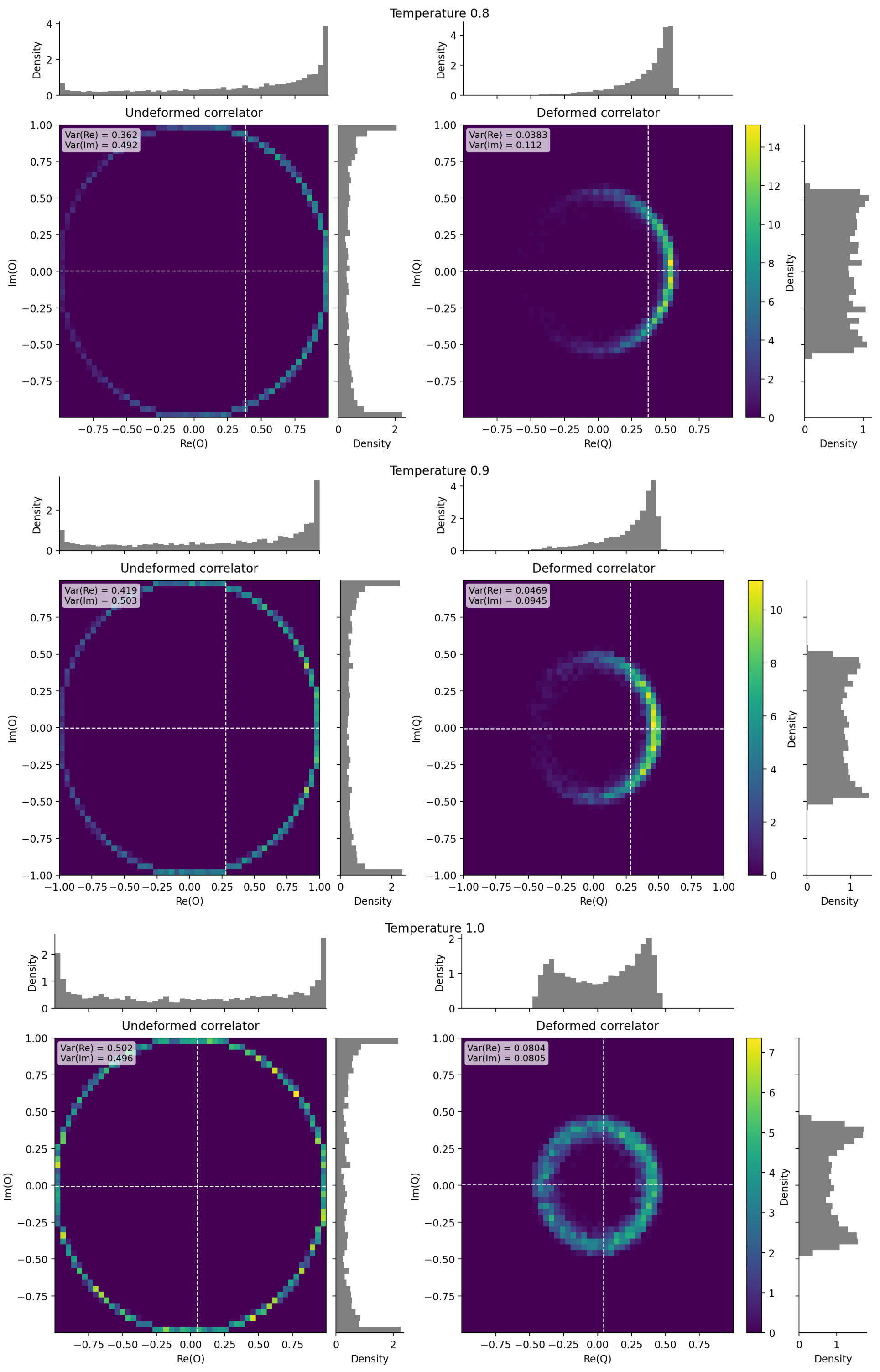

Figure 11: Histograms of the undeformed and deformed correlator using the shift field generated by the 1 layer network at various temperatures and at maximum separation in both directions. It is clear that the 1 layer network struggles more with lower temperatures. Notice the 'squeezing' of the histogram when deformation is applied, bringing points off of the unit circle in the complex plane. We also see that as temperature increases, the marginal distributions slowly flow from unimodal to bimodal.

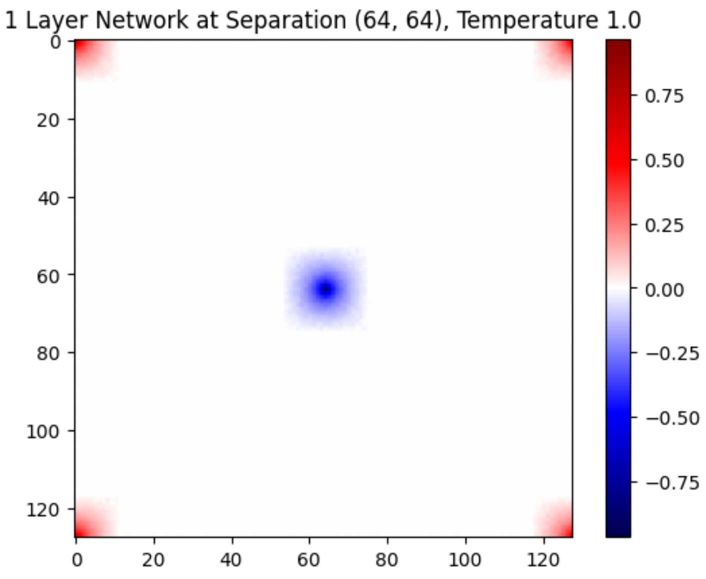

Figure 12: Shift field learned by the 1 layer network at maximum separation for temperature $T = 1.0$. The shifts are confined to within 10 sites of the source and sink. We can clearly see that the source and sink are the absolute minima and maxima of the shift field, as we expect from the theory.

The deformed observable will be agnostic to an overall constant added to the shift field, as the deformed Hamiltonian is only sensitive to differences between neighboring shifts. The U-Net therefore may therefore assign a different overall constant to different separations and temperatures. To address this, and to compare various temperatures and separations on the same color scale, we compute the mean across the full shift field and subtract it off. In theory we expect the mean to be halfway between the source and the sink values, however due to noise this may not always be exactly the case.

We see visually that the form of the shift field is highly reminiscent of an electric potential, although the exact shape is expected to differ due to noise and higher order corrections to the shift field. Even in the regime where higher order effects are suppressed, the U-Net demonstrates an ability to correctly regularize a solution to Poisson's equation, and handle periodic boundary conditions correctly.

We also provide the shift field at an arbitrarily chosen separation in 16 to demonstrate how the field changes when the source and sink are moved relative to each other. Again the behavior is very reminiscent of the solution to a Poisson equation.

a Projected Correlations

In lattice field theory we are typically interested in *momentum projected* correlation functions, where in the case of $1 + 1$ dimensional theories we would integrate out the spatial dependence to retain only the correlations between different points in time. In 2 spatial dimensions (mappable onto the $1 + 1$ case by Wick rotation), we integrate out either the x or y coordinate. This is referred to as momentum projection as integrating over one dimension is the same as setting the corresponding coordinate in momentum space to zero. Projecting correlations onto one coordinate also makes it much easier for us to compare the undeformed and deformed case.

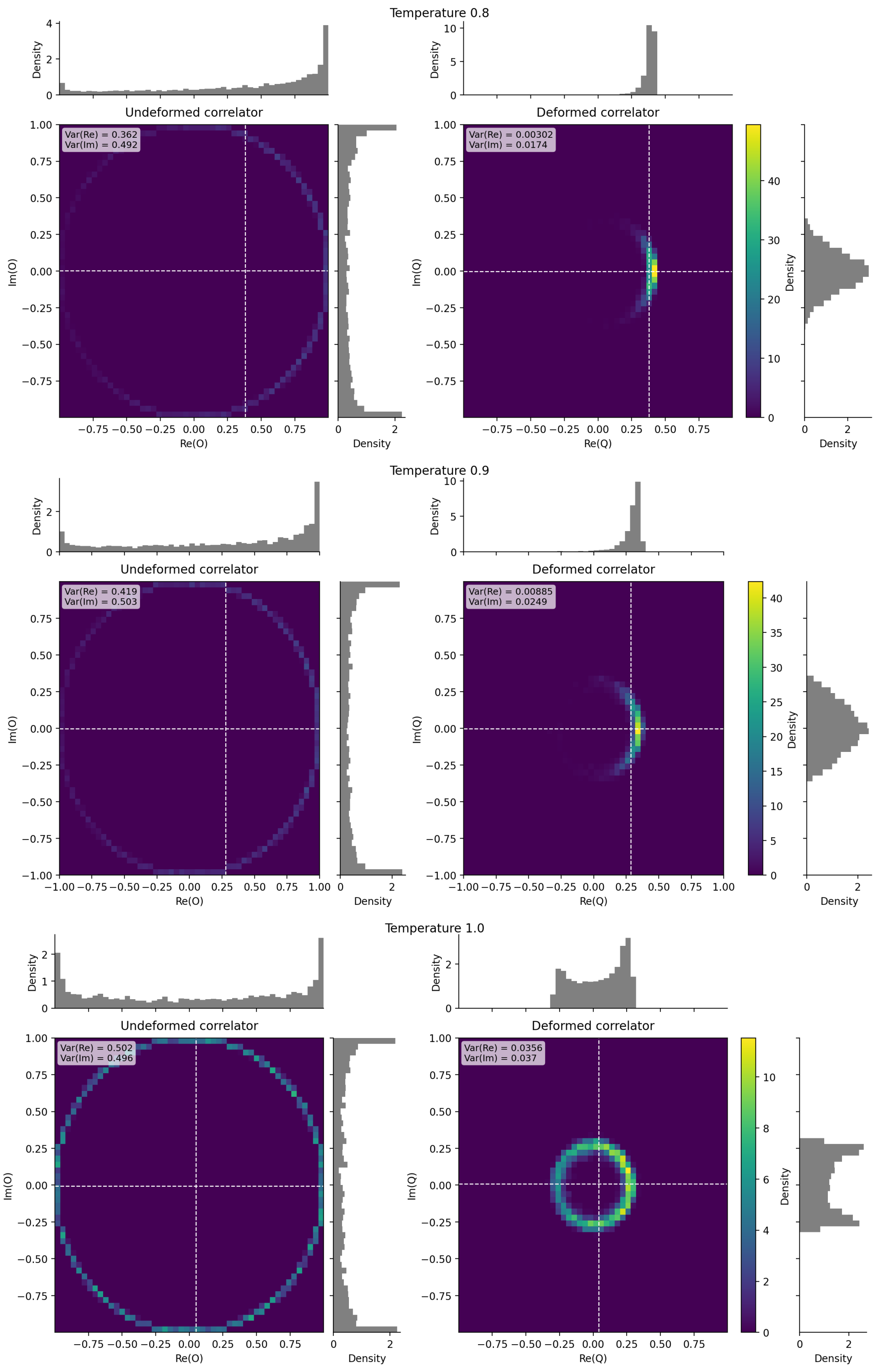

Figure 13: Histograms of the undeformed and deformed correlator using the shift field generated by the U-Net at various temperatures and at maximum separation in both directions. The U-Net clearly outperforms the 1-layer network at all temperatures — the histograms are "squeezed" far more heavily, and, in fact, points flow to the right around the circle to a larger degree as well.

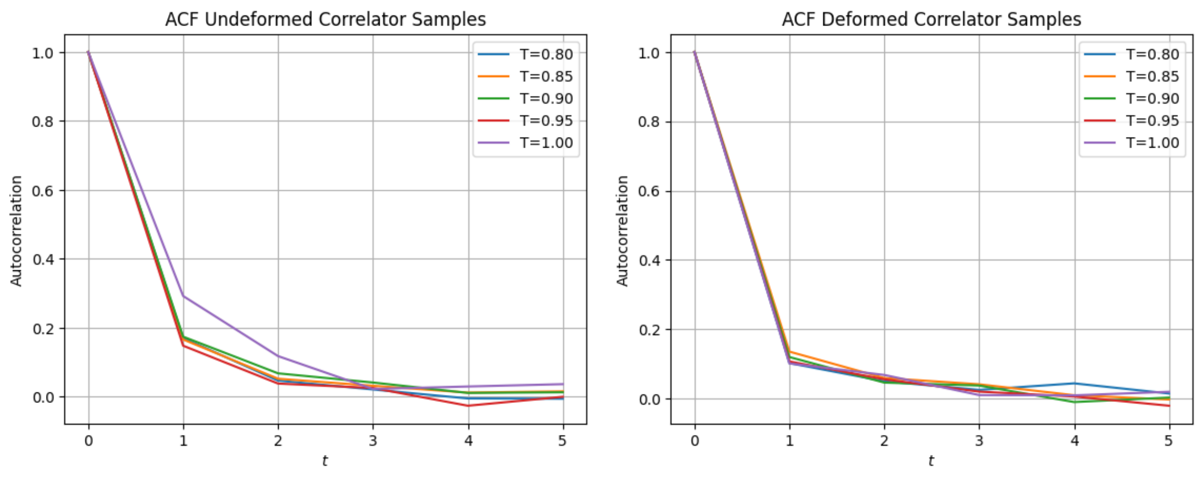

Figure 14: Autocorrelation functions for the undeformed and U-Net deformed observable at maximum separation and various temperatures across the validation data.

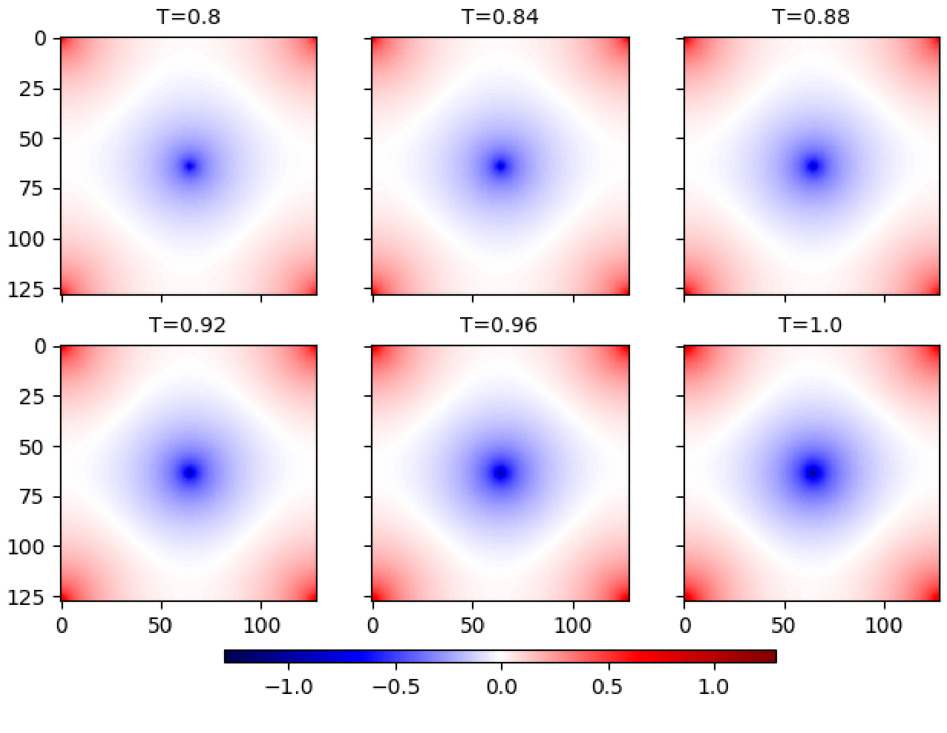

Figure 15: Shift fields produced by the U-Net over a range of temperatures for the maximum possible separation. We see again that the U-Net preserves the source and sink as local minima/maxima, this time learning a shift field that spans the full lattice and is visually similar to an electrostatic potential.

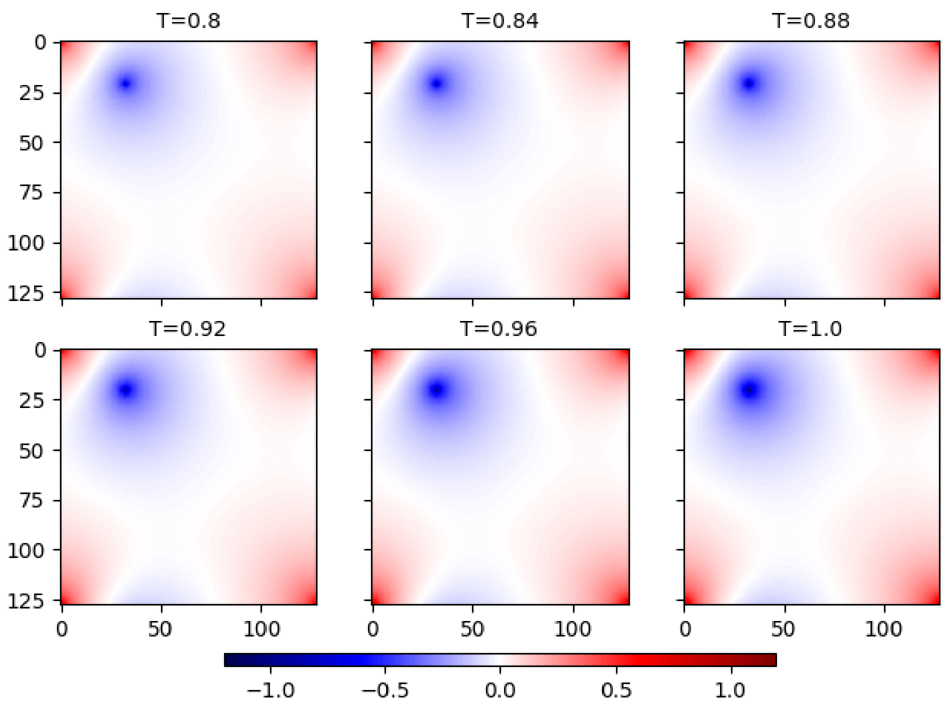

Figure 16: Shift fields produced by the U-Net over a range of temperatures for the separation $(21, 32)$, to illustrate how the field behaves as the relative positions of the source and sink are varied.

When computing these projected correlations, we can also leverage the translational invariance of the correlator — i.e. for some particular separation (x, y) we can compute

$$e^{i\theta(\mathbf{r}+\mathbf{r}_0)-i\theta(\mathbf{r}_0)}$$

or its deformed counterpart, and average over all or some choices of $\mathbf{r}_0$. This is also equivalently to taking all of the lattices in our configuration and translating them by $-\mathbf{r}_0$ then calculating the correlator with respect to the origin as usual. We adopt the latter approach with the deformed correlator as it turns out to be simpler to implement given the existing structure of the code.

There is an even quicker and cleaner method to evaluate the translationally averaged projected correlator in the undeformed case. Breaking the notation $\mathbf{r}$ into (x, y), we first write (considering only the real part of the correlator for reasons discussed previously):

$$C(x, y, x_0, y_0) = \text{Re}\left\langle e^{i\theta(x,y)-i\theta(x_0,y_0)}\right\rangle$$

We then want to sum over x_0, y_0 to perform translational averaging, as well as over y to perform momentum projection:

$$C(x) = \mathrm{Re}\left(\frac{1}{L^2}\sum_{x_0}\sum_{y_0}\sum_{y} C(x_0+x, y, x_0, y_0)\right) = \frac{1}{L^2}\mathrm{Re}\left\langle \sum_{x_0}\sum_{y_0}\sum_{y} e^{i\theta(x_0+x,y)-i\theta(x_0,y_0)}\right\rangle \tag{17}$$

The sum is easily rewritten as a product of two sums:

$$= \mathrm{Re}\left\langle \sum_{x_0}\left(\frac{1}{L}\sum_{y} e^{i\theta(x_0+x,y)}\right)\left(\frac{1}{L}\sum_{y_0} e^{-i\theta(x_0,y_0)}\right)\right\rangle = \mathrm{Re}\left\langle \sum_{x_0}\Phi(x_0+x)\Phi^*(x_0)\right\rangle \tag{18}$$

Where it is understood that the index will wrap around when it exceeds the lattice width. These variables $\Phi(x) = \frac{1}{L}\sum_y \exp(i\theta(x,y))$ are computationally very cheap to evaluate over an ensemble of lattices as opposed to evaluating the triple-nested sum of 17 . We can theoretically achieve a polynomial reduction in the variance by doing this averaging, which cannot be done so simply in the case of a nonlocal deformed observable.

With the deformed observable, we do not do full translational averaging, as the computational cost on the available hardware is too great given the scope of this project. Instead we randomly sample a few (specifically 4 in our analysis) translations for each separation being evaluated. We then opt to, as aforementioned, translate the lattice backward by that amount to allow us to use the exact same shift field and deformation procedure as when the reference point is the origin.

The hope is that the variance reduction from deformation improves exponentially as separation increases, and at some point it "wins" for improving precision. We investigate this next.

6 Conclusion

Appendix A KT-Theory

Appendix B Statistics

References

[1] Michael S. Albergo and Eric Vanden-Eijnden. *NETS: A Non-Equilibrium Transport Sampler*. 2025. arXiv: 2410.02711 [cs.LG]. URL: https://arxiv.org/abs/2410.02711.

[2] William Detmold et al. "Path integral contour deformations for observables in $SU(N)$ gauge theory". In: *Physical Review D* 103.9 (May 2021). ISSN: 2470-0029. DOI: 10.1103/physrevd.103.094517. URL: http://dx.doi.org/10.1103/PhysRevD.103.094517.

[3] William Detmold et al. *Signal-to-noise improvement through neural network contour deformations for 3D $SU(2)$ lattice gauge theory*. 2023. arXiv: 2309.00600 [hep-lat]. URL: https://arxiv.org/abs/2309.00600.

[4] Victor Drouin-Touchette. *The Kosterlitz-Thouless phase transition: an introduction for the intrepid student*. 2022. arXiv: 2207.13748 [cond-mat.stat-mech]. URL: https://arxiv.org/abs/2207.13748.

[5] Matteo Giordano et al. "Exponential reduction of the sign problem at finite density in the 2+1D XY model via contour deformations". In: *Physical Review D* 106.5 (Sept. 2022). ISSN: 2470-0029. DOI: 10.1103/physrevd.106.054512. URL: http://dx.doi.org/10.1103/PhysRevD.106.054512.

[6] Masanori Hanada. *Markov Chain Monte Carlo for Dummies*. 2018. arXiv: 1808.08490 [hep-th]. URL: https://arxiv.org/abs/1808.08490.

[7] Martin Hasenbusch. "The two-dimensional XYmodel at the transition temperature: a high-precision Monte Carlo study". In: *Journal of Physics A: Mathematical and General* 38.26 (June 2005), 5869–5883. ISSN: 1361-6447. DOI: 10.1088/0305-4470/38/26/003. URL: http://dx.doi.org/10.1088/0305-4470/38/26/003.

[8] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[9] J. M. Kosterlitz and D. J. Thouless. "Ordering, metastability and phase transitions in two-dimensional systems". In: *Journal of Physics C Solid State Physics* 6.7 (Apr. 1973), pp. 1181–1203. DOI: 10.1088/0022-3719/6/7/010.

[10] Hans R. Kunsch. "The Jackknife and the Bootstrap for General Stationary Observations". In: *The Annals of Statistics* 17.3 (1989), pp. 1217 –1241. DOI: 10.1214/aos/1176347265. URL: https://doi.org/10.1214/aos/1176347265.

[11] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[12] Ethan Perez et al. *FiLM: Visual Reasoning with a General Conditioning Layer*. 2017. arXiv: 1709.07871 [cs.CV]. URL: https://arxiv.org/abs/1709.07871.

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV]. URL: https://arxiv.org/abs/1505.04597.

[14] Ulli Wolff. "Collective Monte Carlo Updating for Spin Systems". In: *Phys. Rev. Lett.* 62 (4 1989), pp. 361–364. DOI: 10.1103/PhysRevLett.62.361. URL: https://link.aps.org/doi/10.1103/PhysRevLett.62.361.