Applications of Transformer Neural Networks in Correlated Matter

Thesis submitted to
Indian Institute of Science Education and Research Pune
in partial fulfilment of the requirements for the BS-MS Dual Degree Programme

by
Abhinav Suresh

Supervisor: Prof Annabelle Bohrdt
Department of Theoretical Physics, University of Regensburg

May 14, 2024

Certificate

This is to certify that this dissertation entitled "Applications of Transformer Neural Networks in Correlated Matter" towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune, represents the study/work carried out by Abhinav Suresh at the University of Regensburg under the supervision of Annabelle Bohrdt, Professor, Department of Theoretical Physics during the academic year 2023-2024.

Prof Annabelle Bohrdt

Committee:

Prof Annabelle Bohrdt

Dr Sreejith G J

Declaration

I hereby declare that the matter embodied in the report entitled " Applications of Transformer Neural Networks in Correlated Matter" are the results of the work carried out by me at the Department of Physics , Indian Institute of Science Education and Research (IISER) Pune, under the supervision of Prof Annabelle Bohrdt, and the same has not been submitted elsewhere for any other degree. Wherever others contribute, every effort is made to indicate this clearly, with due reference to the literature and acknowledgement of collaborative research and discussions.

Abhinav Suresh

20191024

Acknowledgement

I want to extend my sincere appreciation to my supervisor, Prof Annabelle Borhdt, for her guidance and help in completing this thesis. Thanks for being patient with me and supportive the whole time. I extend my gratitude to Henning Schlömer for giving valuable suggestions and helping me throughout this project. Thanks a lot for your mentorship and the weekly discussions. I would like to thank Hannah Lange for her advice in doing neural quantum states, without which I would not have been able to do it in this limited time. I want to thank Hosein Hashemi for useful tips on the transformer. Without the support and facilities of the Grifoni Chair, University of Regensburg, this work would not be the same, and I am grateful for this. I am also extremely grateful to my parents and all my teachers who have shaped my life so far.

Contents

List of Figures

Abstract

The thesis explores in great detail how the advancements in machine learning could be leveraged to further fundamental research in physics. We use the transformer architecture to classify different phases of matter with interpretable models. We propose a modified architecture, the correlator transformer, that provides full interpretability in terms of correlation functions. We also show that transformers could be a good variational ansatz generator for quantum many-body systems and find good approximations for ground state energies for different Heisenberg systems.

A significant portion of the thesis is dedicated to applying vision transformers for phase classification in different systems. Considerable work has been done in regard to phase classification using neural networks, and with great success. But, despite their effectiveness, they tend to operate as "black boxes". This work seeks to shed some light on these black boxes in terms of physically relevant observables. By training vision transformers on a range of systems, from lattice models like the Ising and Z_2 gauge models to continuous systems such as the Fermi gas, the thesis provides new insights into how a neural network "perceives" these phases of matter. The goal is not only to classify these phases accurately but also to understand the underlying physical correlations the models use to make their decisions, thus addressing the critical challenge of interpretability in deep neural networks.

Deep neural networks have the potential to serve as universal function approximators, and this capability has been used to successfully solve quantum many-body systems variationally. Thus, the second part of the thesis focuses on this aspect of the transformer. We show that a vision transformer can parameterise the quantum many-body wave functions and approximate their ground state and other observables accurately.

Chapter 1

Introduction

Our search for gaining new knowledge has always been limited by the tools we have. Understanding the complexities of quantum many-body systems is among the most exciting and challenging areas of physics. Most models used to study many-body systems are not exactly solvable, and we are forced to rely on numerical tools. Thanks to mathematical simplifications and clever algorithms [1, 2, 3, 4, 5, 6, 7] we have been able to explore many interesting many-body phenomena. Thus, developing new and improving existing tools to further our numerical capabilities is crucial to fundamental research.

The advent of technological innovations in machine learning (ML) (particularly in deep learning) has catapulted many fields to unprecedented levels of efficiency, accuracy, and discovery. These advancements have not only reshaped existing industries but have also paved the way for entirely new areas of application. From healthcare [8, 9, 10] and finance [11, 12, 13] to autonomous vehicles and environmental conservation, ML is revolutionizing how we analyze data, make decisions, and understand the world around us. Deep learning utilizes computational models with many hidden layers to learn data representations at varying levels of abstraction [14]. This strategy has pioneered advanced performance in areas like speech recognition, visual object recognition, and object detection, as well as fields including drug discovery [15, 16, 17] and genomics [18]. Deep learning uncovers complex patterns within large datasets through the backpropagation of gradients.

As ML techniques have become more prevalent in industrial settings, there has been a growing interest among physicists to explore its capabilities, especially in the study of quantum many-body systems [19, 20, 21]. Models in physics are built on physical laws and constraints, and we can interpret the outcomes of the model based on these laws. While in deep learning, models are agnostic about the underlying rules that govern the data. Neural networks are trained to uncover these hidden laws or features the data possesses. This makes understanding how deep neural networks arrive at their decisions a significant challenge. To successfully model a physical system using a neural network, it is crucial that we know what the model is "doing". Deep learning models are infamously hard to interpret due to their inherent layered structure.

Among the plethora of deep learning models, the Transformer neural network, intended

to improve natural language processing [22], has single-handedly allowed us to develop more efficient large language models, for e.g., Chat-GPT, Gemini ...etc. The transformer marks its divergence from existing architecture in its ability to capture long-range dependencies and its inherent parallelizability. This has paved the way for groundbreaking advancements in sequence modelling tasks. With this great power comes great opaqueness in interpretability. Although it gives state-of-the-art benchmarks, from a physicist's point of view, the interpretability of such models remains a problem that still needs to be tackled. The first part of this thesis endeavours to shed some light on the black box and explain why a model can accurately classify two phases in terms of physical observables.

In this seminal paper [20], the authors show that a class of neural networks called Restricted Boltzmann Machines (RBM) can be used to generate variational ansatz and thereby solve quantum many-body Hamiltonians using variational Monte Carlo. The second part of the thesis focuses on using the transformer architecture to find good approximations to ground state energies for a few correlated systems.

This thesis is structured as follows:

- Chapter 2: Reviews the Transformer architecture, focusing on its origins and evolution.
- Chapter 3: Discusses phase classification with an emphasis on interpretability. Results for each physical system are presented in different sections.
- Chapter 4: Although related to interpretable phase classification, in this separate chapter we discuss our modification to the standard transformer architecture to make phase classification interpretable with regards to correlation functions.
- Chapter 5: Discusses Neural Quantum States (NQS) and ground state search using transformers
- Chapter 6: Concluding remarks on this work and discusses possible future directions.

Therefore, this thesis is dedicated to exploring the applications of transformer neural networks in correlated systems.

Chapter 2

The Transformer

The transformer, initially developed to enhance sequence-to-sequence problems such as natural language processing (NLP) was introduced by[22]. It has now become the state of the art in not only NLP but also image classification.

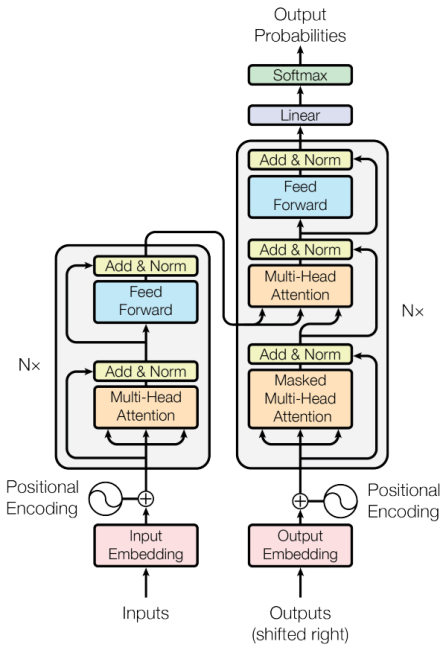

Figure 2.1: Schematic of a transformer decoder and encoder modules. Fig from [22]

The transformer architecture (Fig: 2.1) is composed of an encoder and a decoder, each constructed from a series of identical layers. The core of both the encoder and decoder layers is the self-attention mechanism. Within the context of NLP, this makes the model capable of attending to words at a different location in the input sentence when encoding or decoding a particular word. This mechanism gives the transformer the ability to understand the context of a word within a sentence, which is critical for accurate translation and text generation.

- Encoder: The encoder processes the input data and creates a set of representations that

the decoder can use. Every layer in the encoder consists of two sub-components: the first is a multi-head self-attention mechanism, and the second is a densely connected feed-forward network. The transformer's self-attention mechanism employs scaled dot-product attention, which computes the relevance of all words in the input sequence w.r.t each other words (tokens).

- Decoder: The decoder handles the generation of output sequences. Similar to the encoder the decoder has two sub layers but it also has an additional sub layer that computes attention over the encoder output. This aids the decoder in concentrating on important segments of the input sequence, improving its ability to generate contextually appropriate outputs.

- Self-Attention: Self-attention is a function that evaluates and links various locations within a single sequence to derive a representation of the same sequence. That's why the name "self". What this allows is to enable the model to generate a contextual understanding of the tokens in a sequence. This is crucial in all NLP tasks. The self-attention is used in three different ways: in the encoder for handling the input, in the decoder for handling the output, and between the encoder and decoder (encoder-decoder attention) to help the decoder attend to relevant parts of the input.

2.1 Attention Mechanism

The attention mechanism is among the most important concepts in contemporary machine learning, especially in the field of NLP. It provides a way for models to concentrate on the most pertinent segments of the input for decision-making purposes, similar to the way humans pay attention to certain aspects of our environment over others.

Attention was first introduced as a sequence-to-sequence (seq2seq) model component to improve machine translation. In early seq2seq models the encoder compressed the whole input sequence (source language text) into a fixed-length vector, from which the decoder produced the output sequence (text in the target language). However, this architecture struggled with long sequences because the fixed-length vector became an information bottleneck. To address this,[23] introduced an attention mechanism that allowed the decoder to look back at the input sequence and dynamically focus on different parts of it during the translation process. This not only improved the translation quality but also made the internal decision-making process of the model more interpretable.

One way to think about the attention mechanism is transforming a query along with a collection of key-value pairs into an output, where the output is values weighted by the similarity score between the query and key. This similarity score can be computed in different ways, but the most common is scaled dot product attention.

The Scaled Dot-Product Attention is described by,

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V. \tag{2.1}$$

Where Q is the query matrix, K is the key matrix V is the value matrix, d is the dimensionality of the keys (query, value), and the scaling factor is used for stability for gradient propagation.

The softmax function is applied across the keys for each query, resulting in weights that sum to 1 to give the attention probabilities. The output is values weighted by the degree of similarity between the query and its associated key.

The Transformer model extends this idea with "Multi-Head Attention," which allows the model to collectively attend to short-range and long-range information from different representational spaces. This is achieved by performing the attention function in parallel for each head and then concatenating the results:

$$\begin{aligned} MultiHead(Q,K,V) &= Concat(head_0, head_1, ..., head_n)W^0 \\ &\text{Where each head is computed as:} \\ head_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.2}$$

All W are projection matrices that are learned during training.

The advent of attention mechanisms has been pivotal in the field of large language models as it enabled the development of more sophisticated models that can handle long sequences and complex relationships within the data. It has also improved the interpretability of models, as the attention weights give information about which part of the data is more relevant to the task at hand.

2.2 Positional Embedding

Positional embeddings are a crucial component of the Transformer architecture, which enables the model to take into account the order of the sequence data it processes. Since Transformers lack the inherent sequential processing of models like RNNs, positional embeddings are necessary to provide context that the relative position of tokens in a sequence can convey. We will briefly go through the three main types of positional embeddings: sine-cosine, learnable and relative positional embeddings.

Sine-Cosine Positional Embedding

The original Transformer model uses fixed, non-learnable positional embeddings (absolute) based on sine and cosine functions of different frequencies. The intuition is that these functions

can provide a unique signal for each time step, and their periodic nature allows the model to generalize to sequence lengths unseen during training. The positional information is usually encoded after a linear projection of the patched (to N patches) snapshot to a hidden dimension subspace (d). This is done to make the model more generalizable. The embedding can be written as,

$$PE_{(i,2j)} = \sin\left(\frac{i}{10000^{2j/d}}\right)$$

$$PE_{(i,2j+1)} = \cos\left(\frac{i}{10000^{2j/d}}\right).$$

Here, $i \in \{0,1,...N-1\}$ is the position of N patches and $j \in \{0,1,....,(d-1)/2\}$ where d is the embedding dimension. Each positional encoding dimension is represented by a sinusoidal function, where the wavelengths progress geometrically from 2π to $10000 \cdot 2\pi$. This type of encoding allows each token to have a distinct positional encoding.

Learnable Positional Embedding

Alternatively, positional embeddings can be parameters that the model learns during training. This approach treats positions like words in NLP, assigning each position a vector just like word embedding, and these vectors are learned as part of the model's training process. The advantage here is that the model can learn complex positional relationships that are specific to the task at hand. The embeddings are initialized randomly and then adjusted through backpropagation. Since they are trainable parameters, this increases the complexity of the model compared to absolute positional embedding.

Relative Positional Embedding

Relative positional embeddings were introduced to improve upon the absolute positional embeddings used in the original Transformer. The idea is to embed the relative positions of tokens with respect to one another rather than their absolute position in the sequence. This is especially helpful in data where an element's relative location is more significant than its absolute position within the sequence. The Transformer can learn to use these relative positions effectively. The relative positions can be either fixed or learnable.

Relation-aware positional embedding that allows for translational invariance was introduced by [24] in which they add a term to the attention computation that represents the relative position. This was modified by [25] to have both parallizabilty and interpretabilty.

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + F_p\right)V \tag{2.3}$$

Here, F_p is the Toeplitz matrix parametrized by scoring function $f_\theta.(F_p)_{ij} = f_\theta(j-i)$,

where $f_\theta(j-i)$ models the how the i^{th} position relate to the j^{th} and can be written as [25],

$$f_\theta(k) = \sum_{s=1}^{S} \alpha_s \exp(-|\beta_s|(k-\gamma_s)^2) \tag{2.4}$$

where $k = (i-j)$ and $\alpha_s, \beta_s, \gamma_s$ are trainable parameters for the kernel s. The kernels can be thought of as different realizations of these set of weights (usually we use 1 to 3).

Each of these approaches to positional embedding has its advantages and may be more suitable for different tasks or datasets. Absolute embeddings offer generalization capabilities and are computationally simple, while learnable embeddings provide flexibility and can potentially capture more complex positional information. Relative embeddings focus on the context of token pairs and can provide a more dynamic understanding without large computational overhead.

2.3 Vision Transformer

The vision transformer introduced by [26] modified the transformer to classify image-like data. Traditional convolutional neural networks or CNNs have been the go-to models for image-related tasks, utilizing layers with convolutions that process local kernels and build up an understanding of spatial features of the input data. ViT challenges this norm by leveraging the Transformer's self-attention mechanism to process images and, thereby, allowing the model to have long-range dependencies. Images contain a vast number of pixels and, hence, features

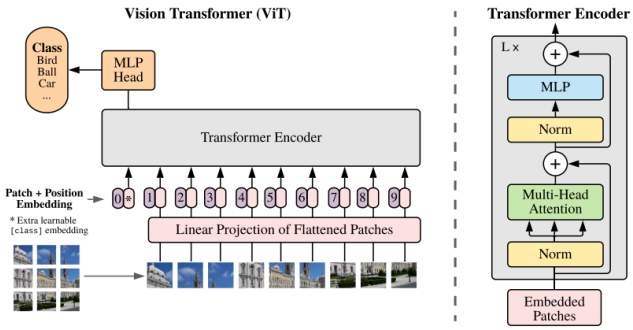

Figure 2.2: Schematic of Vision Transformer (ViT). Fig from [26]

compared to words in a sentence. The Vision Transformer addresses this issue by viewing images as a series of flattened 2D patches, similar to words in a sentence, rather than as a grid of pixels. The basic concept behind ViT is to divide a picture into fixed-size patches, use a linear embedding to reduce the dimension, add position embeddings, and then run the sequence of these embeddings through a conventional Transformer encoder.

- Patch Embedding: ViT begins by dividing an image into patches. These patches are flattened and changed into a series of vectors with reduced dimensions. through a trainable linear projection. This process is similar to tokenization in NLP, with image patches being the visual tokens.

- Position Embeddings: Since the order of the input is not automatically taken into account by the Transformer, ViT encodes the patch embeddings with the positional embedding to retain information about the location of each patch within the image.

- Transformer Encoder: The resulting sequence of patch embeddings, with positional information included, is then fed to a Transformer encoder. Multi-headed self-attention and MLP blocks (multi-layer perceptrons) are arranged in alternating layers to form the encoder, just like the original Transformer used in NLP. Layer normalization and residual connections are also included, enhancing training stability and performance.

- Multi-Head Attention: The self-attention mechanism lets the model focus on the most important segments of the image as it processes each patch. This allows the model to recognize and incorporate dependencies over long distances and understand global contexts within the image.

We tune the standard ViT architecture according to the demands of the data. A watered-down simpler model is considered for simpler systems like classical ising, and a more complex architecture is considered for Cooper pair classification. The specifics of the model and data are given in the next section.

Chapter 3

Interpretable Phase Classification

3.1 Overview

Machine learning, with its ability to discern patterns and relationships in large datasets, offers powerful tools for phase classification. Neural networks, when trained with appropriate non-linear function and depth, can be applied to snapshots to identify various order parameters and extremely non-trivial states such as topological phases. [19, 27] demonstrated that we can recognize phases and phase transitions in a variety of physical systems using fully connected and convolutional neural networks. [28] demonstrated the scheme of confusion learning to find critical points. Other works using unsupervised learning approaches [29, 30, 31, 32, 33], addressing the sign problem[34, 35] and classification of topological phases [36, 37, 38, 39, 40] has been demonstrated successfully with the use of neural networks.

Most of the time, even though the neural network successfully classifies non-trivial phases of matter, the interpretability or "why" or "what" the model learns remains a black box. Many works have been done towards more interpretable, physics-friendly neural networks. [41, 42] shows support vector machines (SVM) learning the mathematical form of physical observables that can discriminate the two phases, [43] explores the learned weights of neural networks for interpretability, [44] uses principle component analysis (PCA) to understand phase transitions and fermionic critical points. Other works in this direction include but are not limited to obtaining analytical forms of nematic order parameters[45] and looking at the learned correlation functions [46] with CCNN. This work intends to extend the ideas of CCNN [46] with Transformers that can handle continuous quantum systems and long-range correlations.

In this chapter, we explore interpretability by employing different methods, including the attention maps from the transformer, PCA analysis and looking at the learned weights to find correlations.

3.2 Classical Ising 2D

The transverse field Ising Hamiltonian is given by

$$H = -J \sum_{<i,j>} \sigma_i^z \sigma_j^z - h \sum_i \sigma_i^x. \tag{3.1}$$

Where J is the coupling between the nearest neighbours and h is the external magnetic field. We train the model with snapshots ranging from $T = 0.25$ to $T = 4$. The dataset is for $h = 0$ with ferromagnetic interaction $J > 0$. At low temperatures, we have a magnetic order; all spins are either up or down. In contrast, at higher temperatures, the state has a random magnetic order.

3.2.1 Results

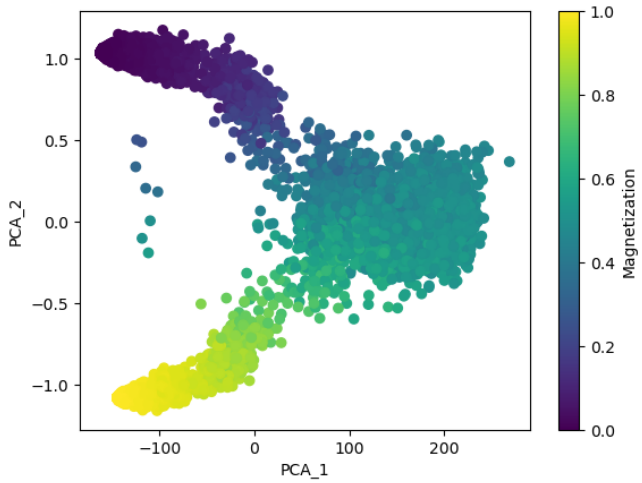

Figure 3.1: Principal component analysis of classification tokens 10000 snapshots.

The results are for the given hyperparameters below, and we achieve 100% accuracy.

Hidden dim $d = 4$

Patch size $= 2$

Number of layers $= 1$

Number of attention heads $= 1$

Hidden drop out $= 0.0$

Attention drop out $= 0.0$

Learning rate $= 0.005$

Epochs $= 5$

Batch size $= 64$

We use the standard ViT with classification token but without any positional embedding. The ground state data is the ferromagnetic state, and the excited state has random magnetization. The model achieved 99.98% accuracy in the classification. From the principal component analysis (PCA) of the model output Fig:3.1, we can see that the model learns the magnetization of the system to classify between the excited state and the ground state. We find three distinct domains corresponding to $+1, 0(0.5)$ and $-1(0)$ magnetization. We can conclude that the model learns the magnetization of the system to classify.

3.3 Z_2 LGT in $(1+1)$ D

3.3.1 ViT

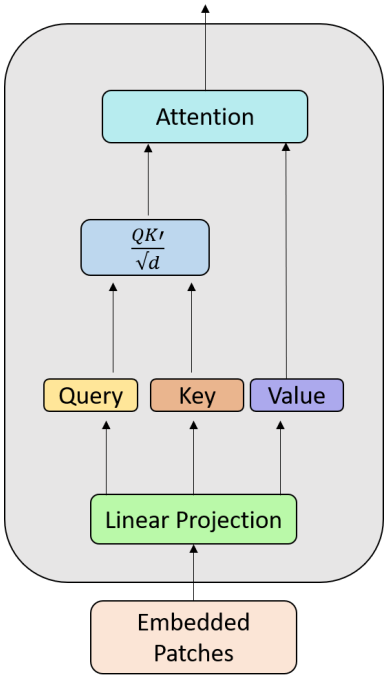

Figure 3.2: Schematic of vision Transformer Encoder

The Attention in Fig 3.2 is given by the expression,

$$\text{Attention}(Q,K,V) = \left(\frac{QK^T}{\sqrt{d}}\right)V. \tag{3.2}$$

Where d is the hidden dimension we chose for the model, here, we do not implement a softmax activation function to control the order of correlations the model accesses. Next, we look into how the transformer works on the level of pixels. Consider an image of dimension (h,c), c is the number of channels. First, we patch the image into grids of size $(p * c)$, p designate the desired patch size.

The patched input image X has shape $(h/p, cp)$. This patched image is then linearly projected by a projection matrix W^p (Eqn 3.3) to the hidden dimension space d. We do not have any bias terms for the linear projections.

$$\begin{aligned} X^p &= XW^p, \text{ where } W^p \text{ has shape } (cp, d) \\ X^p_{ij} &= \sum_{a=1}^{cp} X_{ia} W^p_{aj}, \text{ where } X^p \text{ has shape } (h/p, d) \end{aligned} \tag{3.3}$$

We include a learnable positional embedding (P) by element-wise multiplication for all the projected patches.

The Query (Q), Key (K), and Value (V) are obtained by further linear projections (by the projectors W^q, W^k, W^v) of these projected patches, which are described below (Eqn 3.4).

$$\begin{aligned} Q &= X^p W^q \\ Q_{ij} &= \sum_{b=1}^{d} X^p_{ib} W^q_{bj} \\ &\text{Similarly,} \\ K^T_{ij} &= \sum_{k}^{d} W^{k^T}_{ik} X^{p^T}_{kj} \\ V_{ij} &= \sum_{m=1}^{d} X^p_{im} W^q_{mj} \end{aligned} \tag{3.4}$$

The Q, K and V serve as the representations of the image in the hidden dimension subspace. The attention map is then computed in the following manner,

$$\begin{aligned} a_{ij} &= \sum_r Q_{ir} K^T_{rj} = \sum_{r,b,m=1}^{d} X^p_{ib} W^q_{br} W^{k^T}_{rk} X^{p^T}_{kj} \\ a_{ij} &= \frac{1}{\sqrt{d}} \sum_{r,b,m=1}^{d} X^p_{ib} X^{p^T}_{kj} W^q_{br} W^{k^T}_{rk} \end{aligned} \tag{3.5}$$

The final attention is computed by weighing these attention scores (a) with values (V).

Attention $(A) = aV$

$$\begin{aligned} A_{ij} &= \frac{1}{\sqrt{d}} \sum_{r=1}^{h/p} \sum_{p,b,k,m=1}^{d} \sum_{a,l,n=1}^{cp} X_{ia} X^T_{lr} X_{rn} W^p_{ab} W^p_{nm} W^{p^T}_{kl} W^q_{bp} W^{k^T}_{pk} W^v_{mj} \\ &= \frac{1}{\sqrt{d}} \sum_{r=1}^{h/p} \sum_{p,b,k,m=1}^{d} \sum_{a,l,n=1}^{cp} X_{ia} X^T_{lr} X_{rn} P_{ib} P^T_{kr} P_{rm} W^p_{ab} W^{p^T}_{nm} W^p_{kl} W^q_{bp} W^{k^T}_{pk} W^v_{mj}. \end{aligned} \tag{3.6}$$

For the final classification, we take the mean over all the tokens before further projecting it

into Logits.

$$Logits\ (\hat{y}) = A^{mean}W + B \tag{3.7}$$

where W is a $dX2$ weight matrix and B are the biases. We use the Cross-Entropy loss function, which is given by,

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{3.8}$$

where N is the number of classes, y is the true distribution and $\hat{y}$ is the predicted distribution.

3.3.2 Theory and data

The Hamiltonian of the Z_2 Lattice Gauge Theory in $(1+1)$-D is described as follows [47]

$$\hat{H}_0 = J\sum_{j=1}^{L-1}\left(\hat{a}_j^{\dagger}\hat{\tau}_{j,j+1}^{z}\hat{a}_{j+1} + \text{H.c.}\right) - h\sum_{j=1}^{L}\hat{\tau}_{j,j+1}^{x}, \tag{3.9}$$

Where $\hat{a}_j^{\dagger}, \hat{a}_j$ are the bosonic ladder operators acting on matter site j, $\tau_{j,j+1}^{z}$ denotes the gauge field between the matter site j and $j+1$ and $\tau_{j,j+1}^{x}$ denotes the electric field between the matter site j and $j+1$. The local generator $\hat{G}$ of the gauge invariance is defined as

$$\hat{G}_j = (-1)^{\hat{n}_j}\hat{\tau}_{j-1,j}^{x}\hat{\tau}_{j,j+1}^{x}, \tag{3.10}$$

$$\hat{G}_j|\psi\rangle = g_j|\psi\rangle \tag{3.11}$$

with eigenvalues $g_j = \pm 1$.

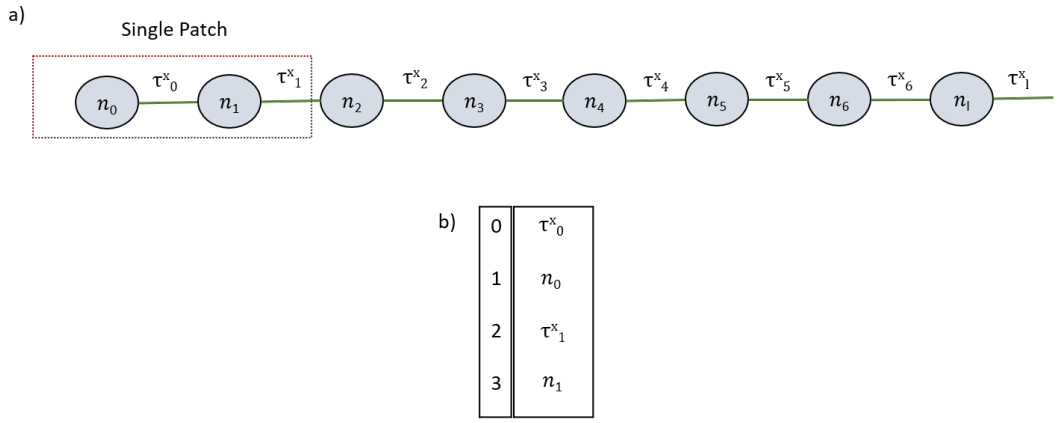

Figure 3.3: a)1d chain indicating the matter (n_j) and links (τ_j^x) and the patching for patch size 2, b) the indices of elements inside a patch (P).

We generate a $1D$ chain of length 16, with matter values measured in the Fock space basis

$(0,1)$ and link values measured in the τ_x basis $(-1,1)$. For the convenience of interpretability we change the matter values to $(1,-1)$ prior to feeding the data into the transformer.

3.3.3 Results

The results are for the given hyperparameters below, and we achieve 99.89% accuracy.

Hidden dim $d = 16$

Patch size $= 2$

Number of layers $= 1$

Number of attention heads $= 1$

Hidden drop out $= 0.0$

Attention drop out $= 0.0$

Learning rate $= 0.01$

Epochs $= 10$

Batch size $= 64$

From Eqn: 3.6 we can construct the weights of the 3 point correlator in the following way

$$C^{ir}_{aln} = \sum_{p,b,k,m=1}^{d} P_{ib} P^T_{kr} P_{rm} W^p_{ab} W^{p^T}_{nm} W^p_{kl} W^q_{bp} W^{k^T}_{pk} W^v_{mj}. \tag{3.12}$$

Where $a,l,n \in \{0,1,2,3\}$, denotes the 4 elements inside a patch (Fig: 3.3 b)). We fix $i = r = 0$. That is, we look for the correlations inside the first patch specifically (and ignore the superscript for convenience). Of the physically relevant correlations, the ones we are interested in of the index C_{023}, which would contain the relevant terms from Gauss's Law (Eqn: 4.14).

Since we changed the matter values from $0,1$ to $1,-1$, the $(-1)^{n_j}$ would just become nj, allowing us to directly get the Gauss's Law from the 3^{rd} order correlation. From Fig: 3.4 we can see that the weights of the term 023, which denotes the terms of the kind $\hat{n}_j \hat{\tau}^x_{j-1,j} \hat{\tau}^x_{j,j+1}$ have greater weightage. We sum over all permutations of a given index. It is important to note that the weights of the correlators of the type $\{C_{aaa}, C_{ann}, C_{aan}...\}$ etc are non-zero (at most the same order). But of the unique (of the type C_{aln}) 3^{rd} order correlators, we find that the transformer is focusing on the most physically relevant one, i.e. those which constitute the Gauss's law.

Fig: 3.5 depicts the weights of all possible 3^{rd} order correlations. All third order terms that has index 1 (C_{1an}) has significantly reduced weights compared to the other. The large weights of correlators such as $C_{220}, C_{200}, C_{000}$ ... indicates towards the transformer giving similar weights to the individual elements $0,2,3$ and ignoring 1.

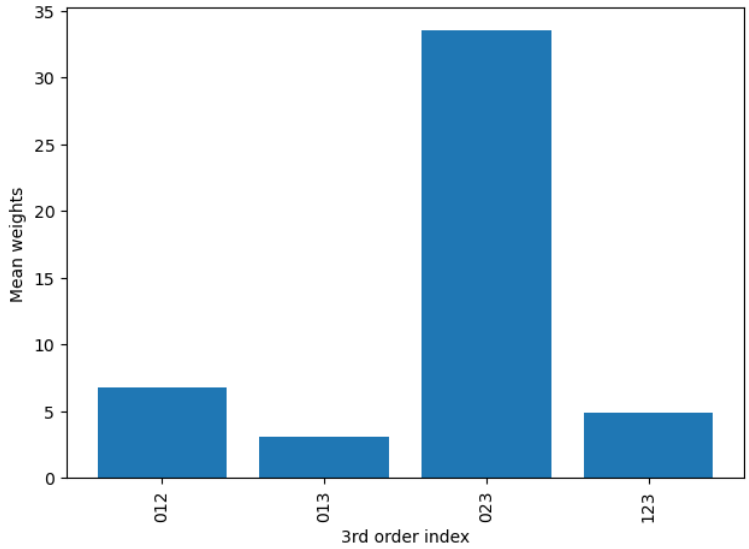

Figure 3.4: Absolute weights of unique 3^{rd} order terms in the Attention averaged over 50 different realizations. The indices denote the position of the relevant elements as described in Fig:3.3

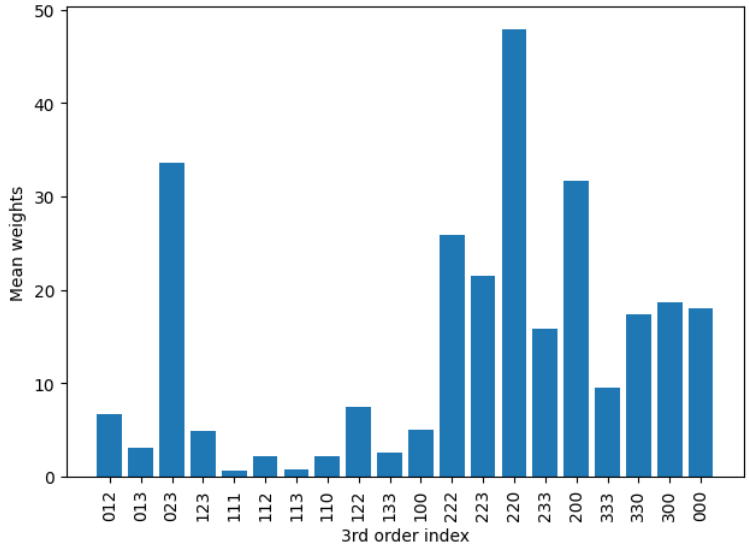

Figure 3.5: Absolute weights of all possible 3^{rd} order terms in the Attention averaged over 50 different realizations

3.4 Ising Gauge Theory 2D

3.4.1 ViT

The Attention in Fig 3.6 is given by the expression,

$$\text{Attention}(Q,K,V) = \left(\frac{QK^T}{\sqrt{d}}\right)V. \tag{3.13}$$

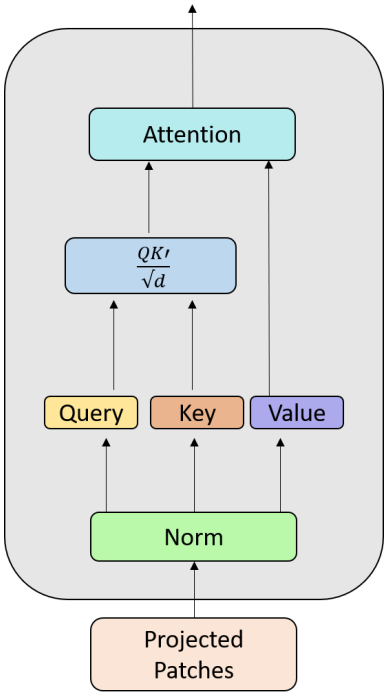

Figure 3.6: Schematic of vision Transformer Encoder

Where d is the hidden dimension we chose for the model. Here, we do not implement a softmax activation function to control the order of correlations the model accesses. Next, we look into how the transformer works on the level of pixels. Consider an image of dimension (h,h,c), c being the number of channels. First, we patch the image into grids of size $(p * p)$, p designate the desired patch size.

The patched input image X has shape $((h/p)^2, cp^2)$. This patched image is then linearly projected by a projection matrix W^p (Eqn 3.14) to the hidden dimension space d and appended by the classification token. The classification token is a learnable parameter of shape $(1,d)$. We do not have any bias terms for the linear projections except for the final classification projection.

$$
\begin{aligned}
&X^p = XW^p, \text{ where } W^p \text{ has shape } (cp^2, d) \\
&X^p_{ij} = \sum_{a=1}^{cp^2} X_{ia} W^p_{aj}, \text{ where } X^p \text{ has shape } ((h/p)^2, d) \\
&\text{Now the classification token } (1,d) \text{ is added as the first element of } X^p \\
&\text{where } X^p \text{ now has shape } ((h/p)^2 + 1, d)
\end{aligned} \tag{3.14}
$$

Importantly, we do not implement positional embedding. Though positional embedding helps to capture inter-patch correlations, the discriminating feature for the Ising gauge data lies within the patches. Therefore, positional embedding could be omitted to make the model simpler and more accurate.

The Query (Q), Key (K), and Value (V) are obtained by further linear projections (by the projectors W^q, W^k, W^v) of these projected patches. The attention map is calculated as follows,

$$a_{ij} = \sum_r Q_{ir} K^T_{rj} = \sum_{r,b,m=1}^{d} X^p_{ib} W^q_{br} W^{k^T}_{rk} X^{p^T}_{kj}$$
$$a_{ij} = \frac{1}{\sqrt{d}} \sum_{r,b,m=1}^{d} X^p_{ib} X^{p^T}_{kj} W^q_{br} W^{k^T}_{rk} \tag{3.15}$$

And as discussed before, the attention is given by,

$$\text{Attention } (A) = aV \tag{3.16}$$
$$A_{ij} = \sum_{r=1}^{(h/p)^2+1} \sum_{p,b,k,m=1}^{d} \sum_{a,l,n=1}^{cp^2} X_{ia} X^T_{lr} X_{rn} W^p_{ab} W^p_{nm} W^{p^T}_{kl} W^q_{bp} W^{k^T}_{pk} W^v_{mj}$$

For the final classification, the transformer receives only the classification token part of the attention output $(A^{cls}_{1j} = A_{1j})$.

$$A^{cls}_{1j} = \frac{1}{\sqrt{d}} \sum_{p=1}^{(h/p)^2+1} \sum_{k,r,b,m=1}^{d} X^p_{1b} X^{p^T}_{kp} X^p_{pm} W^q_{br} W^{k^T}_{rk} W^q_{mj} \tag{3.17}$$

X^p_{1b} represents the classification token. This is then linearly projected to get the logits for the loss function.

$$Logits\ (\hat{y}) = A^{cls} W + B$$
$$\hat{y}_i = \frac{1}{\sqrt{d}} \sum_{p=1}^{(h/p)^2+1} \sum_{s,k,r,b,m=1}^{d} X^p_{1b} X^{p^T}_{kp} X^p_{pm} W^q_{br} W^{k^T}_{rk} W^q_{ms} W_{si} + \sum_{s=1}^{d} B_s \tag{3.18}$$

where W is a $dX2$ weight matrix and B are the biases. We use the Cross-Entropy loss function to optimize the weights and biases.

3.4.2 Theory

The Z_2 Ising gauge theory in 2D is a model of tremendous interest in quantum computation and condensed matter physics, known for its applications in understanding topological order and quantum error correction. The theory is described in terms of spins on the links of a two-dimensional lattice.

The Z_2 Ising Lattice Gauge Theory Hamiltonian in two dimensions is defined as

$$H = -J \sum_p \prod_{l \in p} \sigma^z_l \tag{3.19}$$

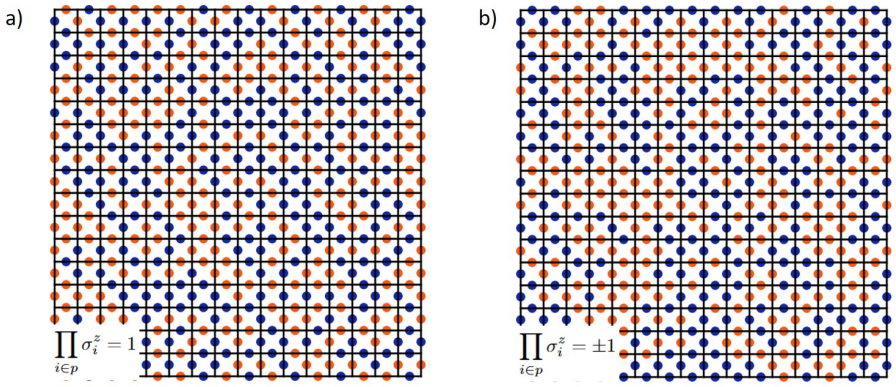

Figure 3.7: a) Ground state where Gauss's law is satisfied, b) Excited state where Gauss's law is violated.

Here, the coupling constant is denoted by J, and the first sum covers all plaquettes p in the lattice.

Gauss's Law Constraint:

For the gauge invariance, each plaquette p of the lattice must satisfy the Gauss's law:

$$G_p = \prod_{l\in p} \sigma_l^z = 1 \tag{3.20}$$

The system has a highly degenerate ground state, comprising all possible configurations where $G_p = 1$ is obeyed. At finite temperatures, the gauge constraint is violated locally. This excited state is characterised by

$$G_p = \prod_{l\in p} \sigma_l^z = \pm 1 \tag{3.21}$$

3.4.3 Results

We consider a system of size $16X16$, with 2 channels for horizontal and vertical links. The results are for the given hyperparameters below, and we achieve 100% accuracy.

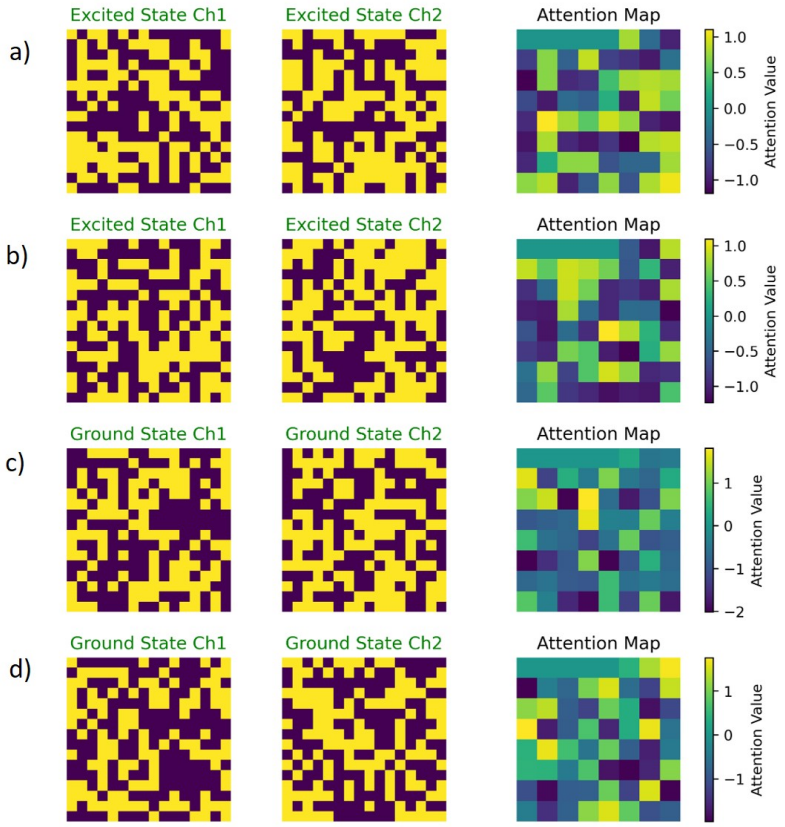

Figure 3.8: a),b) Excited state snapshots and their corresponding attention maps, c),d) ground state snapshots and their corresponding attention maps

Hidden dim $d = 8$

Patch size $= 2$

Number of layers $= 1$

Number of attention heads $= 1$

Hidden drop out $= 0.0$

Attention drop out $= 0.0$

Learning rate $= 0.01$

Epochs $= 20$

Batch size $= 64$

Surprisingly, the transformer is able to classify the phases with weighted 2-point correlators. To understand this, we describe the possible sum of 2^{nd}-order correlations, represented diagrammatically in Fig 3.9. And find it possible to arrive at a constraint similar to Gauss's law.

Fig 3.8 shows the attention map for the two phases, which unfortunately does not give much insight into how the transformer is able to classify. So, we look into the weights inside a $(2X2X2)$ patch (Fig 3.10). We can construct the weights of these 2-point weight correlations

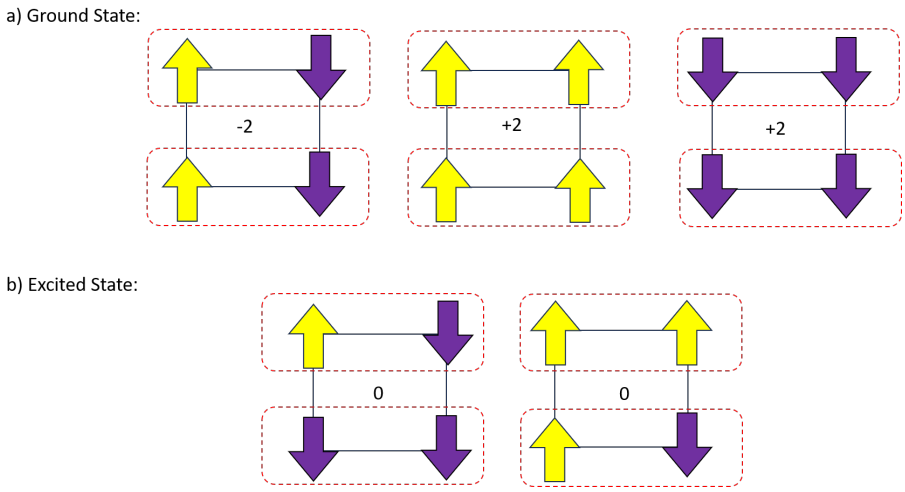

Figure 3.9: We look for a possible sum of 2-point correlators in a plaquette that could differentiate the phases. The 2-point correlation is calculated w.r.t the spins inside the red dots and summed for each possible plaquette configuration. a) The ground state can take the values ± 2, b) while the excited state can take 0 and ± 2

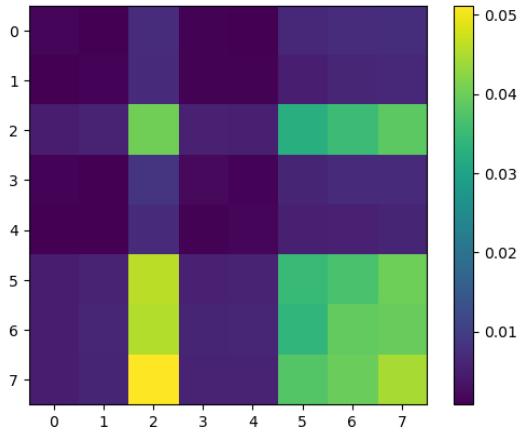

Figure 3.10: The weights (C) of the attention map in the first patch. The weights are computed for 100 realizations and summed over by taking the absolute value.

from the weights of the attention maps Eqn: 3.15

$$C_{bk}^{ij} = \sum_{r,b,m=1}^{d} W_{bn}^{p} W_{pj}^{p^T} W_{br}^{q} W_{rk}^{k^T} \tag{3.22}$$

We fix $i = j = 0$ to analyse the weights inside a single patch. Fig 3.10 shows the weights of the attention map (C_{bk}) for the first patch averaged over 100 realizations. It is important to note that a single patch consists of 8 link variables (spins), and only 4 of them constitute a plaquette. We take the absolute value of each realization to minimize any loss of information. The weights suggest links $2,5,6,7$ contribute to the classification. This pattern remains the same for any given patch. We calculate Gauss's Law using these 4 spins for the 2 phases (Fig: 3.11).

By design, the transformer output (of the classification token) computes all possible 2-point

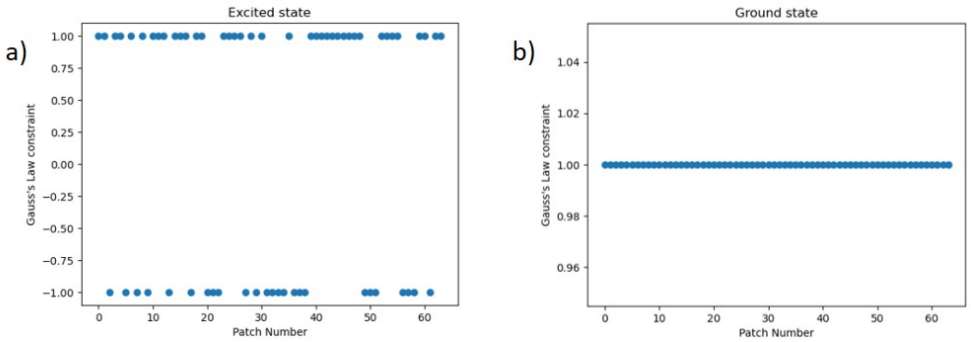

Figure 3.11: a) Gauss' law for the high-temperature phase, b) Gauss's law for zero temperature state

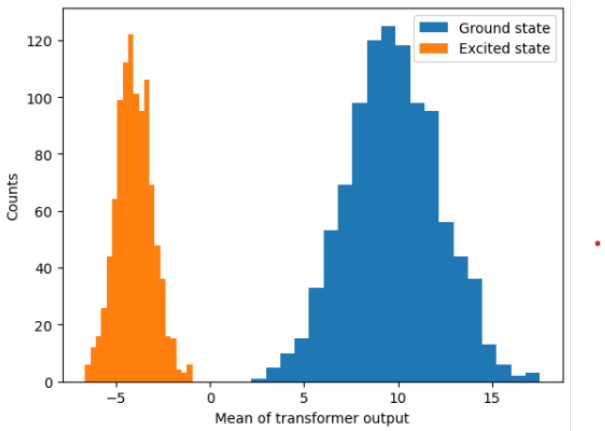

Figure 3.12: Mean of the transformer output(of the classification token) for each phase.

correlations. Fig: 3.12 shows a histogram of the transformer out for the 2 phases. We can conclude that the model doesn't need to look into 4th order correlations as the theory suggests to differentiate the phases accurately.

3.5 Cooper Pairs

3.5.1 ViT

The Attention in Fig 3.13 is given by the expression,

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V. \tag{3.23}$$

Where the softmax is applied on both the dimensions with the tokens, and d is the hidden dimension we chose for the model. Next, we delve a bit deeper into how the transformer works on the level of pixels. Assume an image of dimension (h,h,c), c being the number of channels. First, we patch the image into grids of size $(p * p)$, p designate the desired patch size.

The patched input image X has shape $((h/p)^2, cp^2)$. This patched image is then linearly

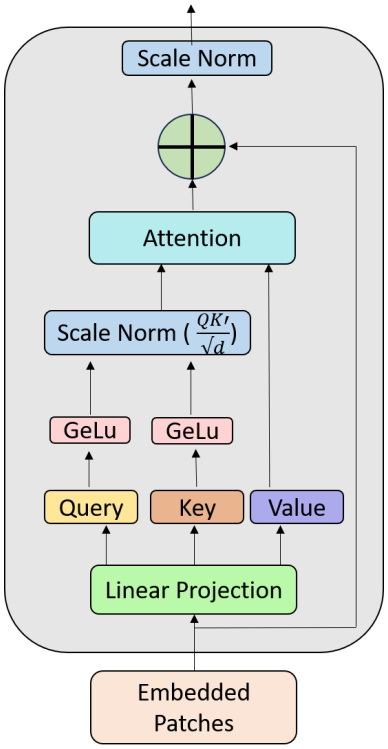

Figure 3.13: Schematic for Vision Transformer Encoder module

projected by a projection matrix W^p (Eqn 3.24) to the hidden dimension space d and appended by the classification token. The classification token is a learnable parameter of shape $(1,d)$. (The bias terms are present but are not explicitly written for the sake of simplification)

$$
\begin{aligned}
&X^p = XW^p, \text{ where } W^p \text{ has shape } (cp^2, d) \\
&X^p_{ij} = \sum_{a=1}^{cp^2} X_{ia} W^p_{aj}, \text{ where } X^p \text{ has shape } ((h/p)^2, d) \\
&\text{Now the classification token } (1,d) \text{ is added as the first element of } X^p \\
&\text{where } X^p \text{ now has shape } ((h/p)^2 + 1, d)
\end{aligned} \tag{3.24}
$$

Next, we add positional embedding(P_{ij}) to X^p to obtain the Embedded Patches in Fig 3.13. We choose a learnable positional embedding. Interestingly, we find that the element-wise multiplication of positional embedding provides better results than the traditional approach of adding them. The Query (Q), Key (K), and Value (V) are obtained by further linear projections (by the projectors W^q, W^k, W^v) of these embedded patches. The attention map is computed as

$$
\begin{aligned}
a_{ij} &= \frac{1}{\sqrt{d}} QK^T = \frac{1}{\sqrt{d}} \sum_{p=1}^{d} \sum_{b=1}^{d} X^p_{ib} W^q_{bp} P_{ib} \sum_{k=1}^{d} W^k_{pk} X^{p^T}_{kj} P^T_{kj} \\
a_{ij} &= \frac{1}{\sqrt{d}} \sum_{p,b,k=1}^{d} X^p_{ib} X^{p^T}_{kj} P_{ib} P^T_{kj} W^q_{bp} W^k_{pk}
\end{aligned} \tag{3.25}
$$

We apply Scale Norm (SN), which is

$$SN(x) = \frac{x}{||x||a} \tag{3.26}$$

where $||.||$ is the L_2 norm and a is a learnable parameter. Now, we take softmax along the last two dimensions, which denotes the tokens, to attenuate irrelevant portions and focus more on the relevant parts of the snapshots that contribute to classification. This gives us the attention probabilities (Eqn 3.27).

$$\text{softmax}\left(a_{ij}\right) = \frac{exp\left(\frac{1}{\sqrt{d}}\sum_{p,b,k=1}^{d} X_{ib}^{p} X_{kj}^{p^T} P_{ib} P_{kj}^{T} W_{bp}^{q} W_{pk}^{k}\right)}{\sum_{m=1}^{(h/p)^2+1}\sum_{n=1}^{(h/p)^2+1} exp\left(\frac{1}{\sqrt{d}}\sum_{p,b,k=1}^{d} X_{mb}^{p} X_{kn}^{p^T} P_{mb} P_{kn}^{T} W_{bp}^{q} W_{pk}^{k}\right)} \tag{3.27}$$

The final attention output is obtained by weighing these attention probabilities by their respective Value (V).

$$\begin{aligned}
\text{Attention (A)} &= \text{softmax}\,(a)\,V \\
A_{ij} &= \sum_{l=1}^{d} \text{softmax}\,(a_{il})\,V_{lj} \\
&= \sum_{l=1}^{(h/p)^2+1} \left(\frac{exp\left(\frac{1}{\sqrt{d}}\sum_{p,b,k=1}^{d} X_{ib}^{p} X_{kl}^{p^T} P_{ib} P_{kl}^{T} W_{bp}^{q} W_{pk}^{k}\right)\sum_{r=1}^{d} X_{lr}^{p} P_{lr} W_{rj}^{v}}{\sum_{m=1}^{(h/p)^2+1}\sum_{n=1}^{(h/p)^2+1} exp\left(\frac{1}{\sqrt{d}}\sum_{p,b,k=1}^{d} X_{mb}^{p} X_{kn}^{p^T} P_{mb} P_{kn}^{T} W_{bp}^{q} W_{pk}^{k}\right)} \right) \\
X^{p} &= X^{p} + A
\end{aligned} \tag{3.28}$$

For the final classification, the transformer receives only the classification token part($X_{cls}^{p} = X_{1j}^{p}$) Eqn (3.29) of the transformer output (X^p). $\sum X_{1j}^{p}$ represents the learnable classification token.

$$X_{1j}^{p} = X_{1j}^{p} + \sum_{l=1}^{(h/p)^2+1} \left(\frac{exp\left(\frac{1}{\sqrt{d}}\sum_{p,b,k=1}^{d} X_{1b}^{p} X_{kl}^{p^T} P_{1b} P_{kl}^{T} W_{bp}^{q} W_{pk}^{k}\right)\sum_{r=1}^{d} X_{lr}^{p} P_{lr} W_{rj}^{v}}{\sum_{m=1}^{(h/p)^2+1}\sum_{n=1}^{(h/p)^2+1} exp\left(\frac{1}{\sqrt{d}}\sum_{p,b,k=1}^{d} X_{mb}^{p} X_{kn}^{p^T} P_{mb} P_{kn}^{T} W_{bp}^{q} W_{pk}^{k}\right)} \right) \tag{3.29}$$

Which is then further linearly projected to get the logits for the loss function. We use the Cross-Entropy loss function, which is given by,

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{3.30}$$

where N is the number of classes, y is the true distribution and $\hat{y}$ is the predicted distribution.The

explicit expression for the predicted distribution can be written as

$$\hat{y} = X^p_{cls} W + B$$
$$\hat{y_j} = \sum_{s=1}^{d} X^p_{1s} W_{sj} + B_s$$
$$= \sum_{s=1}^{d} \left(X^p_{1s} + \sum_{l=1}^{(h/p)^2+1} \frac{exp\left(\frac{1}{\sqrt{d}} \sum_{p,b,k=1}^{d} X^p_{1b} X^{p^T}_{kl} P_{1b} P^T_{kl} W^q_{bp} W^k_{pk}\right) \sum_{r=1}^{d} X^p_{lr} P_{lr} W^v_{rs}}{\sum_{m=1}^{(h/p)^2+1} \sum_{n=1}^{(h/p)^2+1} exp\left(\frac{1}{\sqrt{d}} \sum_{p,b,k=1}^{d} X^p_{mb} X^{p^T}_{kn} P_{mb} P^T_{kn} W^q_{bp} W^k_{pk}\right)} \right) W_{sj}$$
$$+ \sum_{s=1}^{d} B_s \tag{3.31}$$

Where W and B are the weights and biases of the final linear layer.

3.5.2 Cooper Pair Data

Cooper pairs are pairs of opposite spin momentum fermions bounded together at low temperatures. We generate synthetic data of momentum space snapshots of $2D$ Fermi gas in a harmonic oscillator potential with tunable interaction strength. At low or zero interaction strength, there is no pairing, while at sufficiently large interaction strength, the opposite spin fermions tend to pair across the Fermi surface. This pairing is important for superfluidity and superconductivity [48]. We generate configuration on a $50X50$ (denoting p_xXp_y) grid with cooper pairs and without while maintaining the total number of particles. The red and blue particles denote opposite spin fermions. The excited state (with high interaction strength) has an opposite spin pair (red and blue denote spin up and spin down, respectively) distributed over the Fermi surface (denoted by the dotted circle). Meanwhile, the ground state has a random distribution. Fig 3.14 a)

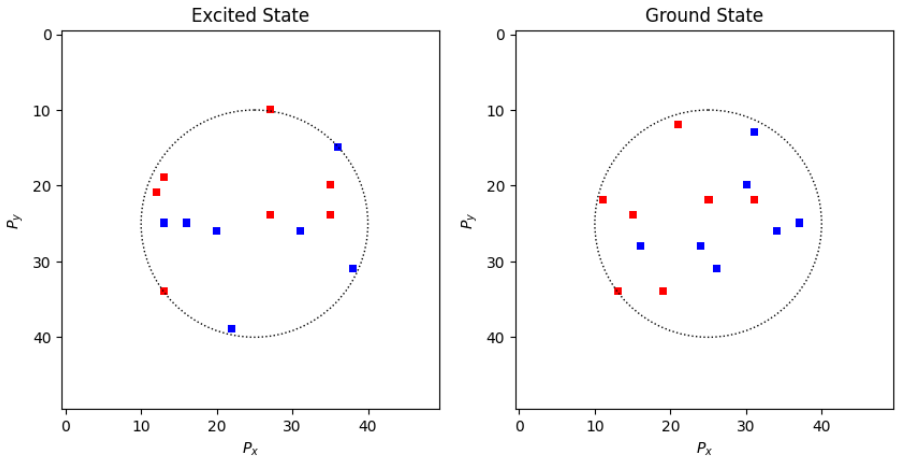

Figure 3.14: a) The excited state has two pairs of particles on the Fermi momenta b) the ground state corresponds to randomly distributed particles. The dotted lines represent the Fermi momenta.

represent one of the classes (Excited State) where there are two pairs of particles on the Fermi

surface, and Fig 3.14 b) are snapshots where the particles are randomly distributed. The model is trained on a set of these 2 classes of snapshots.

3.5.3 Results

The model is trained using the following hyperparameters.

$$\begin{aligned}
&\text{Hidden dim } d = 32\\
&\text{Patch size} = 5\\
&\text{Number of layers} = 1\\
&\text{Number of attention heads} = 4\\
&\text{Hidden drop out} = 0.2\\
&\text{Attention drop out} = 0.2\\
&\text{Learning rate} = 0.005\\
&\text{Epochs} = 20\\
&\text{Batch size} = 64
\end{aligned}$$

We are able to achieve an accuracy of 96.10%.

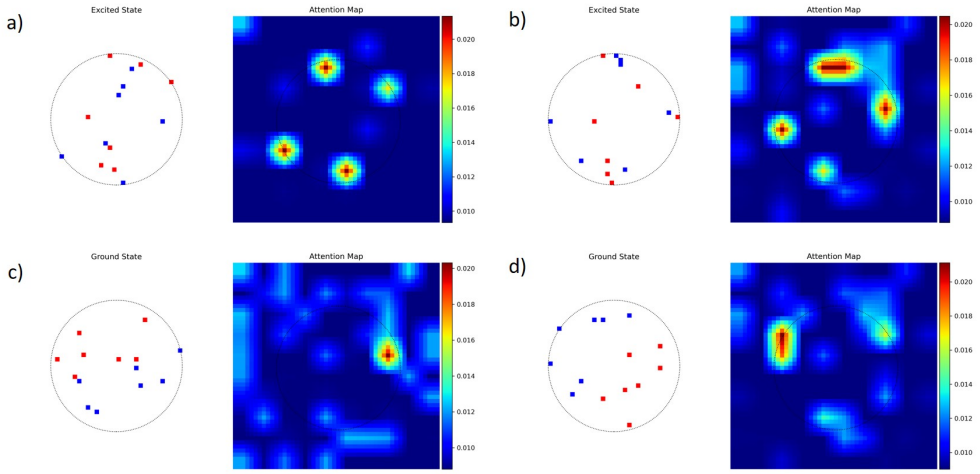

Figure 3.15: a),b) the excited state snapshots and their respective attention maps, c),d)the ground state snapshots and their respective attention maps

Fig 3.15 a) and b) show how the attention map focuses on the cooper pairs of the excited state. While Fig 3.15 c) and d) shows the attention maps for the ground state, and we can see the Transformer trying to locate the particles on the Fermi momenta but cannot find more pairs. All attention maps are computed w.r.t the classification token. From these attention maps, we can conclude that the transformer can identify the long-range correlations in our snapshot by locating the cooper pairs.

3.5.4 Analysis of Attention Maps

The attention map is computed by averaging over the attention maps of all the heads. Looking into each attention map in the individual heads gives insights into what each head is focusing on.

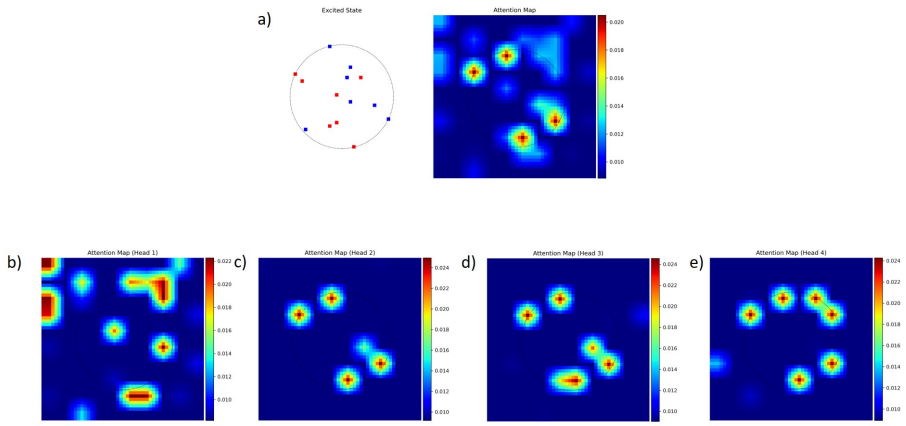

Figure 3.16: a) Excited state snapshot and the corresponding attention map, b),c),d),e) are the attention map for head1, head2, head3 and head4, respectively

Fig 3.16 shows how different attention heads attend to different parts of the excited state snapshot.

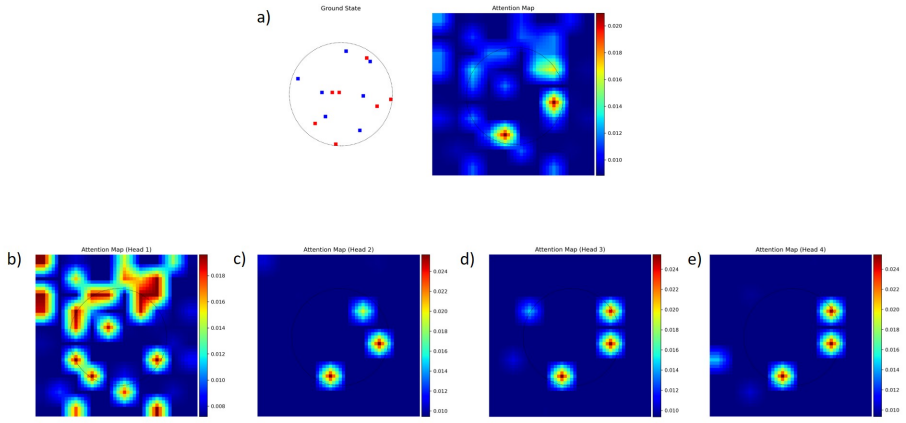

Figure 3.17: a) Ground state snapshot and the corresponding attention map, b),c),d),e) are the attention map for head1, head2, head3 and head4, respectively

Fig 3.17 shows how different attention heads attend to different parts of the ground state snapshot. These attention maps show that the transformer is looking for the Cooper pairs at the surface. We can also infer that head 2 contributes more to the underlying physics for fixed hyperparameters and seeds. Analysing these attention maps one can arrive at the conclusion that the transformer is focusing on the particles on the surface (Cooper pairs) to make a distinction between the two classes of snapshots.

3.6 Percolation

3.6.1 ViT

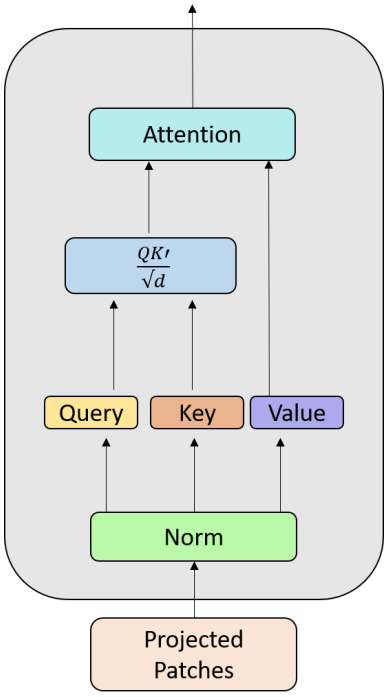

Figure 3.18: Schematic of vision Transformer Encoder

We use the same Transformer as we used for the $2D$ Ising Gauge Theory (Fig: 3.18) but with positional embedding. And here, this is crucial for the model's high performance because it has to learn long-range correlations.

As discussed earlier, the patched input image X has shape $((h/p)^2, cp^2)$. This patched image is then linearly projected by a projection matrix W^p (Eqn 3.32) to the hidden dimension space d and appended by the classification token. The classification token is a learnable parameter of shape $(1,d)$. We also use completely learnable positional embedding.

$$X^p = XW^p, \text{ where } W^p \text{ has shape } (cp^2, d)$$

$$X^p_{ij} = \sum_{a=1}^{cp^2} X_{ia} W^p_{aj}, \quad \text{where } X^p \text{ has shape } ((h/p)^2, d)$$

Now the classification token $(1,d)$ is added as the first element of X^p

where X^p now has shape $((h/p)^2 + 1, d)$

Then we element wise multiply a learnable position embedding P.

$$X^p_{ij} = \sum_{a=1}^{cp^2} X_{ia} W^p_{aj} P_{ij} \tag{3.32}$$

The Query (Q), Key (K), and Value (V) are obtained by further linear projections (by the

projectors W^q, W^k, W^v) of these projected patches. The attention map is given by,

$$a_{ij} = \sum_r Q_{ir} K^T_{rj} = \sum_{r,b,k=1}^{d} X^p_{ib} W^q_{br} W^{k^T}_{rk} X^{p^T}_{kj}$$
$$a_{ij} = \frac{1}{\sqrt{d}} \sum_{r,b,k=1}^{d} X^p_{ib} X^{p^T}_{kj} W^q_{br} W^{k^T}_{rk} \tag{3.33}$$

And as discussed before, the attention is given by,

$$\text{Attention } (A) = aV \tag{3.34}$$
$$A_{ij} = \sum_{p=1}^{(h/p)^2+1} \sum_{r,b,k,m=1}^{d} \sum_{a,l,n=1}^{cp^2} X^p_{ib} X^{p\ T}_{kp} X^p_{pm} W^q_{br} W^{k^T}_{rk} W^v_{mj}$$

For the final classification, the transformer receives only the classification token part of the attention output ($A^{cls}_{1j} = A_{1j}$).

$$A^{cls}_{1j} = \frac{1}{\sqrt{d}} \sum_{p=1}^{(h/p)^2+1} \sum_{k,r,b,m=1}^{d} X^p_{1b} X^{p^T}_{kp} X^p_{pm} W^q_{br} W^{k^T}_{rk} W^q_{mj} \tag{3.35}$$

X^p_{1b} represents the classification token. This is then linearly projected to get the logits for the loss function.

$$Logits\ (\hat{y}) = A^{cls} W + B$$
$$\hat{y}_i = \frac{1}{\sqrt{d}} \sum_{p=1}^{(h/p)^2+1} \sum_{s,k,r,b,m=1}^{d} X^p_{1b} X^{p^T}_{kp} X^p_{pm} W^q_{br} W^{k^T}_{rk} W^q_{ms} W_{si} + \sum_{s=1}^{d} B_s \tag{3.36}$$

where W is a $dX2$ weight matrix and B are the biases. We use the Cross-Entropy loss function to optimize the weights and biases.

3.6.2 Results

Data 1

We generate a $16X16$ lattice of random spins directed from one edge to the opposite edge. We classify the 2 classes, those that reach the opposite edge as percolated and those that don't as a non-percolated state. The idea here is to test if the transformer can learn long-range correla-

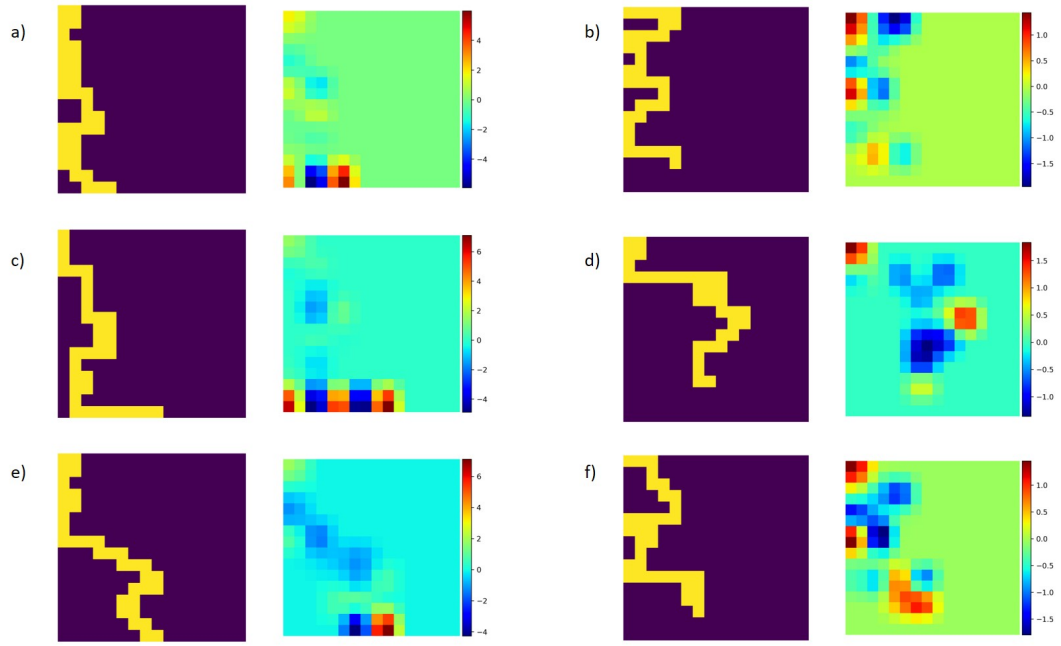

Figure 3.19: a),c),e) Attention maps for a few examples of percolated snapshots, b),d),f) attention maps for a few examples of non-percolated snapshots

tions. We get an accuracy of 99.99% for this classification.

Hidden dim $d = 8$

Patch size $= 2$

Number of layers $= 1$

Number of attention heads $= 1$

Learning rate $= 0.01$

Epochs $= 20$

Batch size $= 64$

Fig: 3.19 shows the attention map w.r.t the classification token (these attention maps are interpolated to the image size). There is a concentration of attention score at the lower edge where the percolated spins end. It seems the transformer is looking at whether a snapshot reaches the bottom edge to decide whether it is percolated.

Data 2

Previously, we considered a simplified model for percolation, i.e. the percolated lattice random spins directed from one edge to the opposite edge. Here, we add more complexity by randomly rearranging the spins in the percolated images to make the non-percolated snapshots. This is done so that the transformer can't just look at the lower edge to make the classification. The

data still consists of a $16X16$ lattice with constant magnetization. The results are for the given hyperparameters, and we achieve 98.69% accuracy.

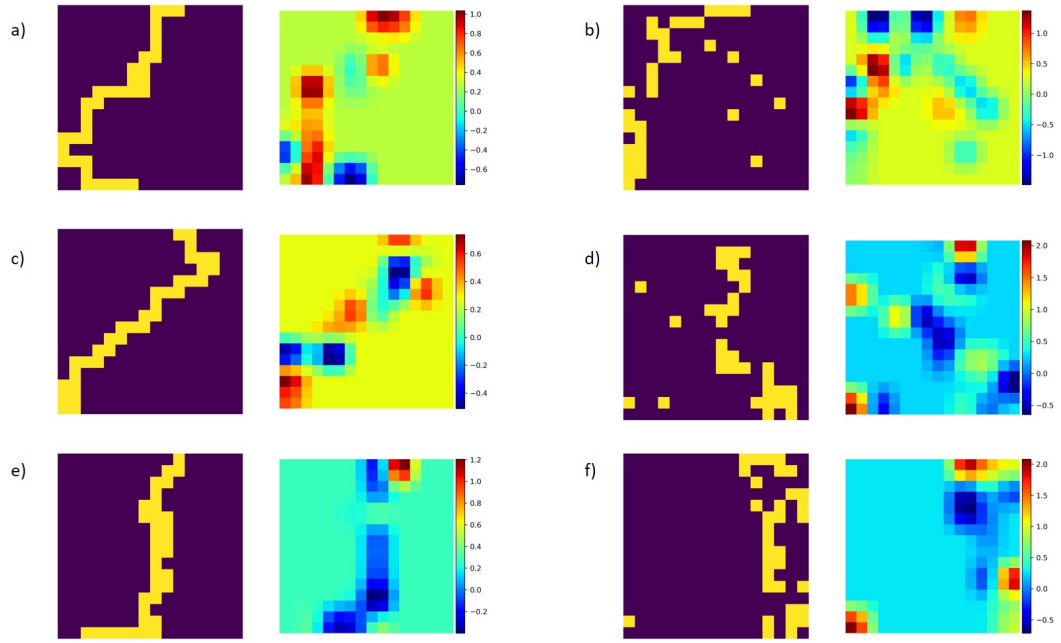

Figure 3.20: a),c),e) Percolated snapshots and corresponding attention map, b),d),f) non-percolated snapshots and corresponding attention map

Hidden dim $d = 8$

Patch size $= 2$

Number of layers $= 1$

Number of attention heads $= 1$

Learning rate $= 0.01$

Epochs $= 10$

Batch size $= 64$

Although the total magnetization is constant across both phases, the high accuracy suggests that the model is aware of the spatial distribution of the spins. This is confirmed by the attention maps in Fig 3.20. The attention map seems to roughly follow the spins' distribution in both phases.

Fig: 3.21 shows the principle component analysis of the classification token.

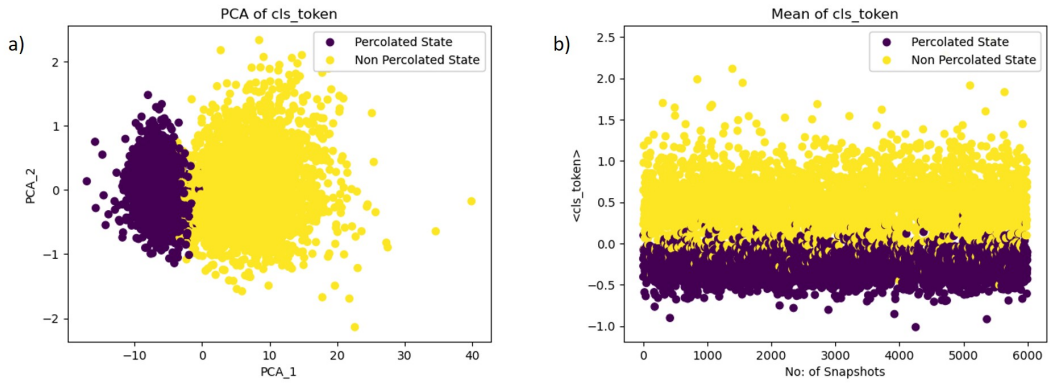

Figure 3.21: a) PCA analysis of the classification token, b) The mean of classification token of the two phases after training

3.7 Other Synthetic Data

3.7.1 Data

To simulate a bilayer material (similar to graphene bilayers) with antiferromagnetic coupling between the layers, we generate snapshots with antiferromagnetic order about the vertical axis (Fig:3.22). The snapshots are $16X16$ with each side representing 1 layer; therefore, the snapshots are unfolded bilayer. We train the transformer to see whether it can learn to classify this ordered state against a disordered one (random magnetization).

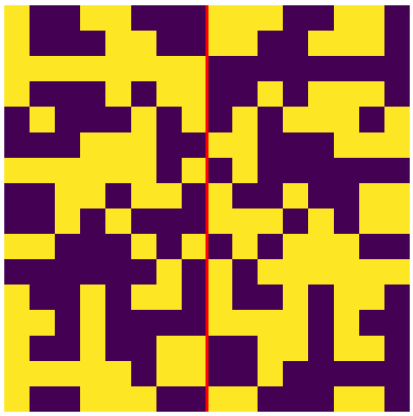

Figure 3.22: Snapshot with antiferromagnetic coupling about the red dotted line, which is the folding axis of the bilayer. The left-hand side would be 1 layer and the right-hand side the other.

3.7.2 Results

The results are for the given hyperparameters, and we achieve 99.99% accuracy.

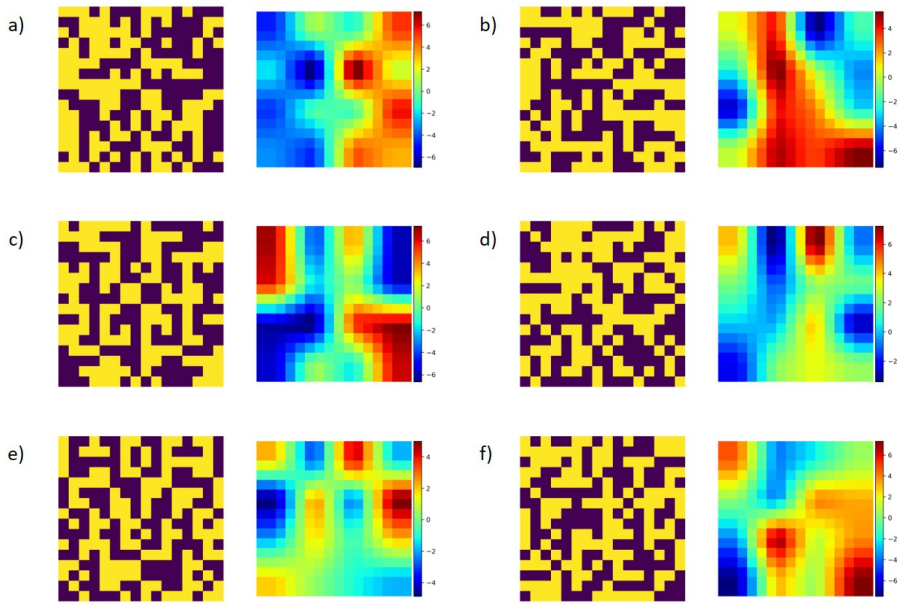

Figure 3.23: a),c),e) Snapshots with antiferromagnetic bilayer coupling and their attention maps, a),c),e) Phases with random magnetization and their attention maps

Hidden dim $d = 16$

Patch size $= 4$

Number of layers $= 1$

Number of attention heads $= 1$

Learning rate $= 0.01$

Epochs $= 10$

Batch size $= 64$

Fig: 3.23 shows the attention map obtained from the transformer. We can clearly see that the attention map for the antiferromagnetic phase reflects the underlying pattern. All the attention maps are computed w.r.t the classification token.

3.7.3 Data

We generate spins in $16X16$ lattices with mirror symmetry about the vertical axis and train it against a random configuration of spins. The aim of these generated data is to see whether the transformer learns this symmetry. This could then be extended for systems with much more complicated symmetries.

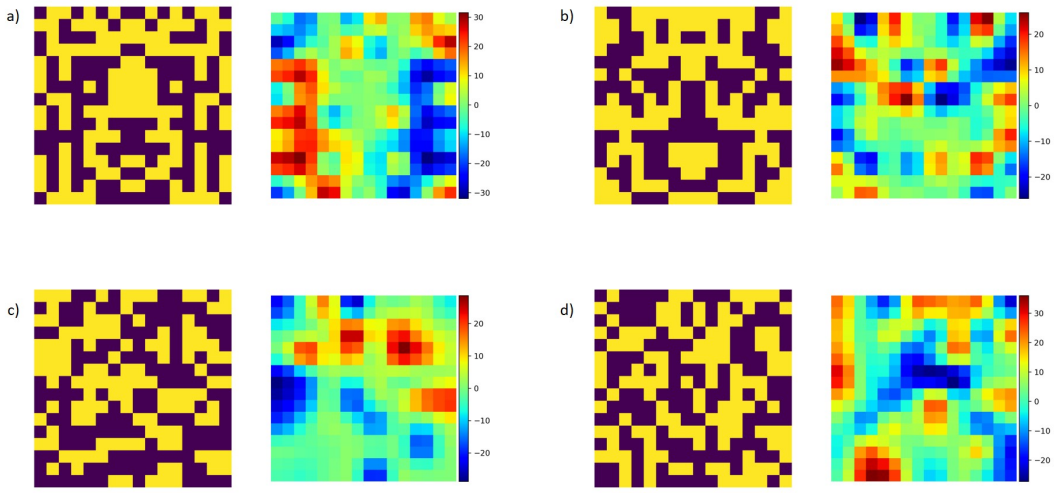

Figure 3.24: a), b) Mirror symmetric snapshots and their respective attention maps, c), d) random snapshots and their respective attention maps

3.7.4 Results

Hidden dim $d = 16$

Patch size $= 2$

Number of layers $= 1$

Number of attention heads $= 1$

Learning rate $= 0.01$

Epochs $= 10$

Batch size $= 64$

The model achieves an accuracy of 99.99%. We see that the attention maps in the figure Fig: 3.24 a) and b) do not have mirror symmetry but are similar to the case of bilayer Fig: 3.23. We conclude that the transformer attention maps do not always reflect the exact underlying symmetry.

Chapter 4

Correlator Transformer

4.1 Architecture

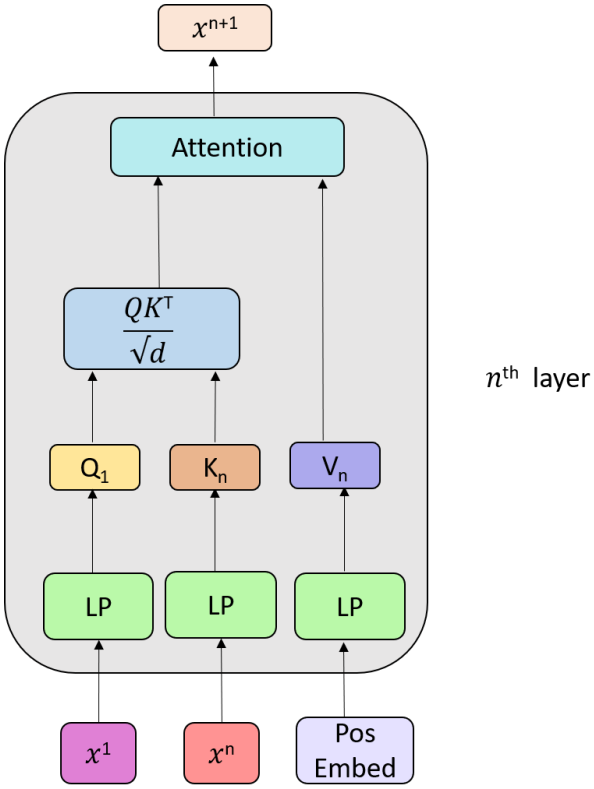

Figure 4.1: Schematic of the correlator transformer encoder

Here, the Attention in Fig: 4.1 can be written as

$$\text{Attention} = \frac{1}{\sqrt{d}} QK^T V \tag{4.1}$$

Consider a snapshot of dimension (h,h,c) where h,c are the height and number of channels. We first divide this snapshot into equal patches of dimension pXp, where p is the patch size.

The snapshot now will have the dimension $((h/p)^2, cp^2)$. After this patching, we have a linear projection of the hidden dimension subspace d. This is done to make the model more generalizable. This projected patch (x^1) can be used as it is for the input of the correlator attention or can be provided with additional learnable positional embedding depending on the input data. We call this the first order term x^1 in Fig: 4.1, which remains the first input of all subsequent correlator transformer layers. By default, we don't use any layer norm (denoted by Norm in Fig: 4.1), and it is used only if it is crucial for the model's performance. This is chosen to keep the model as simple as possible for better interpretability. The projected snapshot X^1 can be written as,

$$\begin{aligned} X^1 &= XW^p, \text{ where } W^p \text{ has shape } (cp^2, d) \\ X^1_{ij} &= \sum_{a=1}^{cp^2} X_{ia} W^p_{aj}, \text{ where } X^p \text{ has shape } ((h/p)^2, d) \end{aligned} \tag{4.2}$$

In the first layer, both the query and key projections are computed from X^1, which returns the second-order terms X^2. The query projection in the second layer remains the same, while the key projection is now computed from the output of the first layer X^2. This now computes the third-order terms X^3. Therefore, using n transformer layers would enable us to compute up to $(n+1)^{th}$ order terms. We use a dropout layer to guide the network in obtaining the most critical features. The query and key for the first transformer layer can be written as

$$\begin{aligned} Q_1 &= X^1 W^Q_1 \\ K_1 &= X^1 W^K_1 \end{aligned} \tag{4.3}$$

Meanwhile, the value is a linear projection of the sine cosine embedding. The sine cosine positional embedding is,

$$\begin{aligned} PE_{(i,2j)} &= \sin\left(\frac{i}{10000^{2j/d}}\right) \\ PE_{(i,2j+1)} &= \cos\left(\frac{i}{10000^{2j/d}}\right). \end{aligned}$$

Here, $i \in \{0, 1, ...N-1\}$ is the position of N patches and $j \in \{0, 1, ..., d-1\}$ where d is the embedding dimension. Since these are not learnable parameters, they do not increase the complexity of the model. The value, then, is

$$V_1 = (PE)W^v_1 \tag{4.4}$$

The attention score of the first layer (X^2) can now be written as,

$$\begin{aligned} X^2 &= Q_1 K_1^T V_1 \\ &= X^1 W_1^Q W_1^{K^T} X^{1^T} (PE) W_1^v \end{aligned} \tag{4.5}$$

The n^{th} layer attention can be defined as,

$$A_{n+1} = Q_1 K_n^T V_n = X^{n+1} \tag{4.6}$$

Where Q_1 denotes the query projection from X^1, K_n denotes the key projection from X^n and V_n is the value projection from the positional embedding at each layer n. This attention can be thought of as weighing the different correlations in QK^T with their positional information.

We take a mean over each order correlator and have a linear projection to form the logits for the Cross-Entropy loss function coupled with an $L1$ regularization loss with strength λ.

After the training, we collect these learned correlators and do a regularization path analysis to understand which order terms influence the classification the most.

This modified architecture shares the same number of parameters as that in a traditional vision transformer.

4.1.1 Regularization Path Analysis

Regularization path analysis is a method for feature selection. We use the L_1 regularization or the lasso regression. The idea is to collect all the learned correlators x^n and do a lasso regression to see which of these $n-$ point correlators is the most important for phase classification.

$$y = \beta_0 + \beta_1 x^1 + \beta_2 x^2 + ... + \beta_n x^n \tag{4.7}$$

Now, we need to find the appropriate β that would give us the best fit (Linear regression), and by looking at which β_n is the more prominent, we can say x^n is the important order of correlation in the system. To make sure that we isolate the crucial β_n we add a regularization term L

$$L = \lambda(|\beta_1| + |\beta_2| + ... + |\beta_n|) \tag{4.8}$$

Now we do a linear regression and find the β_n by minimizing

$$\left(y - \beta_0 - \sum_{j=1}^{n} \beta_j x^j \right)^2 + \lambda \sum_{j=1}^{n} |\beta_j| \tag{4.9}$$

We solve Eqn:4.9 for varying strengths λ starting from very high where the regression fails and slowly reduces it. And by looking at the coefficients β_n for all λ, we can deduce which

particular order of correlation is the most crucial in a quantum phase.

4.2 Heisenberg 2D

The Hamiltonian is given by

$$\hat{H} = -J \sum_{\langle i,j \rangle} \left(\hat{S}_i^x \hat{S}_j^x + \hat{S}_i^y \hat{S}_j^y + \hat{S}_i^z \hat{S}_j^z \right) \tag{4.10}$$

Where J is the nearest neighbour coupling constant and $S^{x,y,z}$ are the spin operators. We train the transformer on a class of antiferromagnetic state (AFM) ($J > 0$) against completely random magnetization. The AFM ground states are projective measurements of S^z on each site. Here,

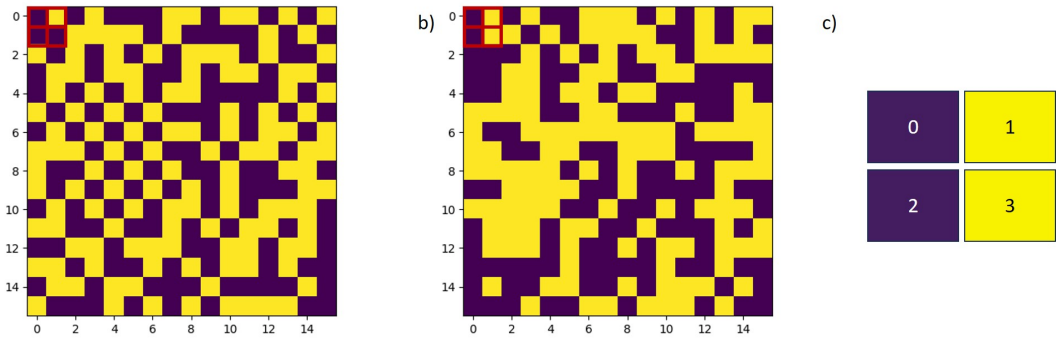

Figure 4.2: a) A ground state configuration, b) a random configuration, c) the position of the spins inside a $2X2$ patch denoted by the red square

we multiply an additional learnable positional embedding to x^1 before it enters the attention mechanism. The results are for the given hyperparameters:

Hidden dim $d = 16$

Patch size $= 2$

Number of layers $= 3$

Learning rate $= 0.02$

Epochs $= 50$

$\lambda = 0.001$

Fig: 4.3 shows the accuracy and loss for the classification. Since the transformer is able to classify the ground state and excited state of the Heisenberg AFM, we ask which are the important correlations that the model learns to do the classification with such high accuracy. For this, we do a regularization path analysis of all order terms that the transformer computes. Fig: 4.4 depicts the regularization path for each 4 order correlators the transformer learns, and it says that the 2^{nd} order correlations are the most crucial for the classification between the

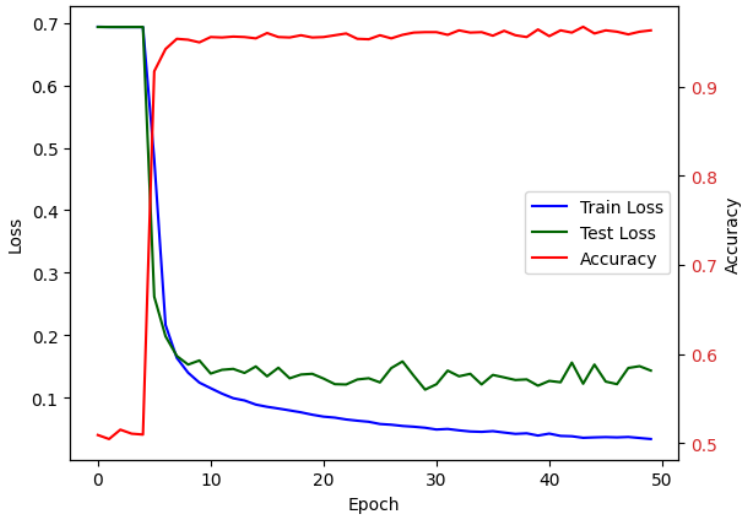

Figure 4.3: Accuracy and loss metrics for Heisenberg AFM classification

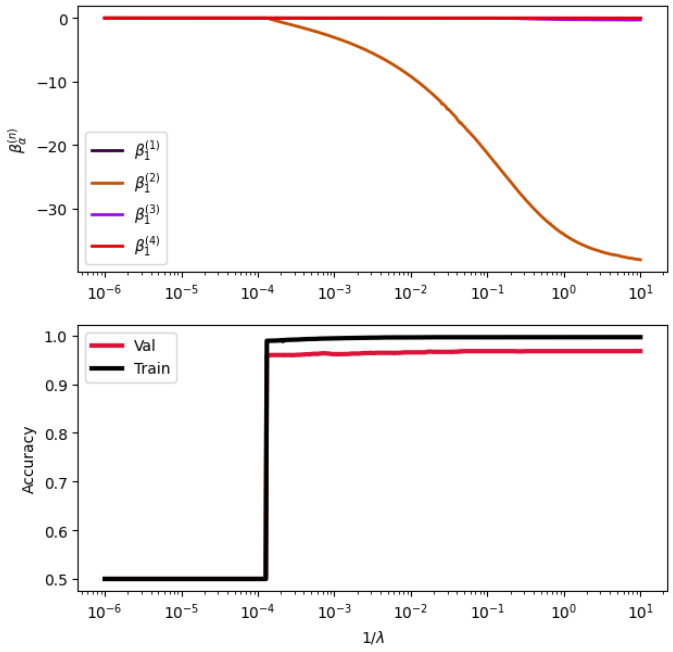

Figure 4.4: Regularization path analysis for the given hyperparameters

2 phases. Now we ask which 2^{nd} order correlator is important, so we compute the 2^{nd} order weight matrix inside a patch Eqn: 4.11.

$$C^2_{rs} = \sum P_{ik} P^T_{ln} W^p_{rk} W^{p^T}_{ls} W^{1Q}_{km} W^{1K^T}_{ml} \tag{4.11}$$

Here, the sum r,s goes through the 4 spins inside a patch (first patch), and P denotes the learnable positional embedding, W^p denotes the linear projection matrix for the hidden dimension projection whereas W^{1Q}, W^{1K} are query and key projections of the first layer. Fig: 4.5 represents the weights of 2^{nd} order correlations between the 4 spins of a patch. This indicates that the transformer is focusing more on the nearest neighbour correlations. That is, the correlations

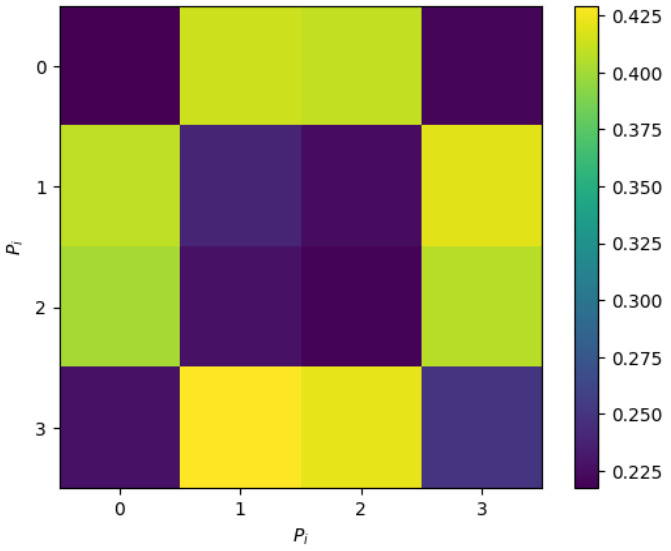

Figure 4.5: The mean absolute weights inside a patch over different realizations. P_i goes over the spins inside the patch.

between the nearest neighbour spins $01, 02$ have more weight than the next nearest neighbour spin correlator 03.

4.3 Z_2 LGT in $(1+1)$ D

We use the same data as we discussed in Chapter 3, the Hamiltonian being

$$\hat{H}_0 = J \sum_{j=1}^{L-1} \left(\hat{a}_j^\dagger \hat{\tau}_{j,j+1}^z \hat{a}_{j+1} + \text{H.c.} \right) - h \sum_{j=1}^{L} \hat{\tau}_{j,j+1}^x, \tag{4.12}$$

Where $\hat{a}_j^\dagger, \hat{a}_j$ are the bosonic ladder operators acting on matter site j, $\tau_{j,j+1}^z$ denotes the gauge field between the matter site j and $j+1$ and $\tau_{j,j+1}^x$ denotes the electric field between the matter site j and $j+1$.

The local generator $\hat{G}$ of the gauge invariance is defined as

$$\hat{G}_j = (-1)^{\hat{n}_j} \hat{\tau}_{j-1,j}^x \hat{\tau}_{j,j+1}^x, \tag{4.13}$$

$$\hat{G}_j |\psi\rangle = g_j |\psi\rangle \tag{4.14}$$

with eigenvalues $g_j = \pm 1$. Our ground state is where $g_j = 1 \forall\ j$, and the excited state is where $g_j = \pm 1$.

We use a layer norm layer before the linear projection of x^1 and x^n. The results are for the

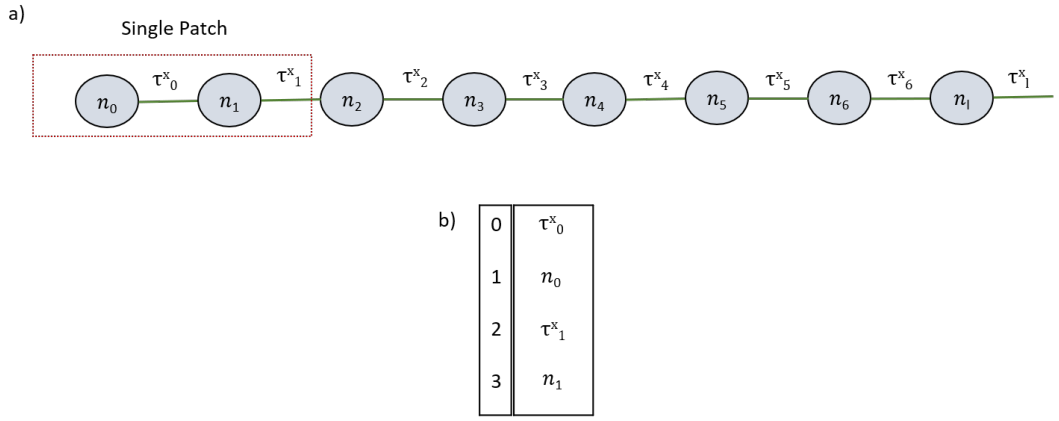

Figure 4.6: a) Schematic of the $1D$ Z_2 LGT, b) the positions of the elements inside a patch (only elements $0, 2, 3$ contribute to the gauge constraint

given hyperparameters:

$$\text{Hidden dim } d = 8$$
$$\text{Patch size} = 2$$
$$\text{Number of layers} = 3$$
$$\text{Learning rate} = 0.001$$
$$\text{Epochs} = 100$$
$$\lambda = 0.002$$

Fig: 4.7 shows the classification results, and we achieve near-perfect accuracy. Now, we

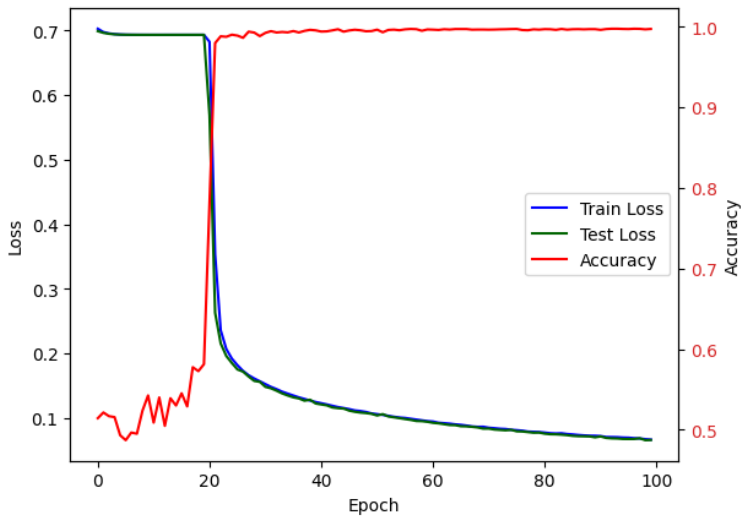

Figure 4.7: Accuracy and loss metrics for the 1D LGT classifications

look into the regularization path (Fig: 4.8) to see which order correlations the transformer focuses on. The transformer focuses on the 3^{rd}-order correlations needed to calculate the gauge constraint. Now, we plot the weights of correlations inside a single patch Eqn: 4.15(which

constitutes 4 elements, as detailed in Chapter 3).

$$C^{3}_{drs} = \sum P_{ia} P^{T}_{kj} P_{nl} W^{p}_{da} W^{p^{T}}_{kr} W^{p}_{sl} W^{1Q^{T}}_{mk} W^{2Q}_{ac} W^{1K}_{lm} W^{2K^{T}}_{cb} W^{1V^{T}}_{bn} \tag{4.15}$$

Where W^{iM} denotes the projection of i^{th} layer and $M \in Q, K, V$ denotes the query, key and value projection. From Fig: 4.9, we see that it gives more weightage to the correlations between the terms that constitute the 3^{rd}-order gauge constraint. We can conclude that the transformer is learning 3^{rd} order gauge constraint to classify the $1D$ Z_2 LGT phases.

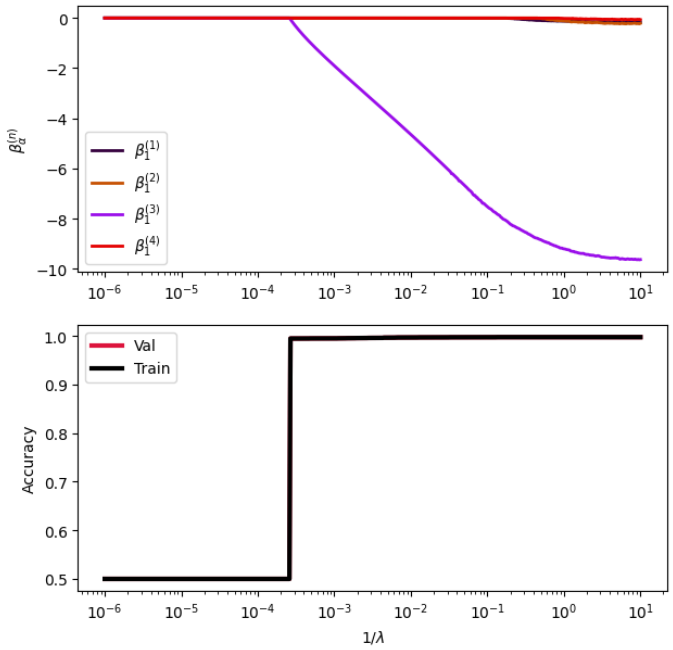

Figure 4.8: Regularization path analysis for 1d LGT

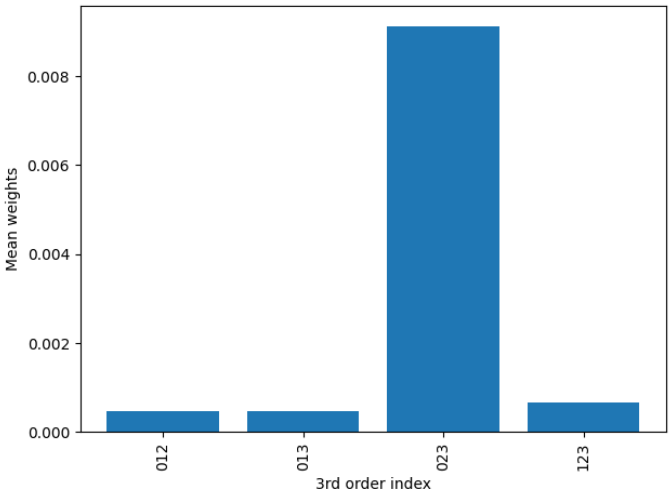

Figure 4.9: 3^{rd} order weights corresponding to all the combinations of the elements inside a patch over 10 different realizations

4.4 Ising LGT 2D

We use the same data as we discussed in Chapter 3, the Hamiltonian is given by

$$H = -J \sum_{p} \prod_{l \in p} \sigma_l^z \tag{4.16}$$

Here, the coupling constant is denoted by J, and the first sum covers all plaquettes p in the lattice. Fig: 4.10 depicts both channels of a sample ground state snapshot and the positions of the spins inside a $2X2X2$ patch.

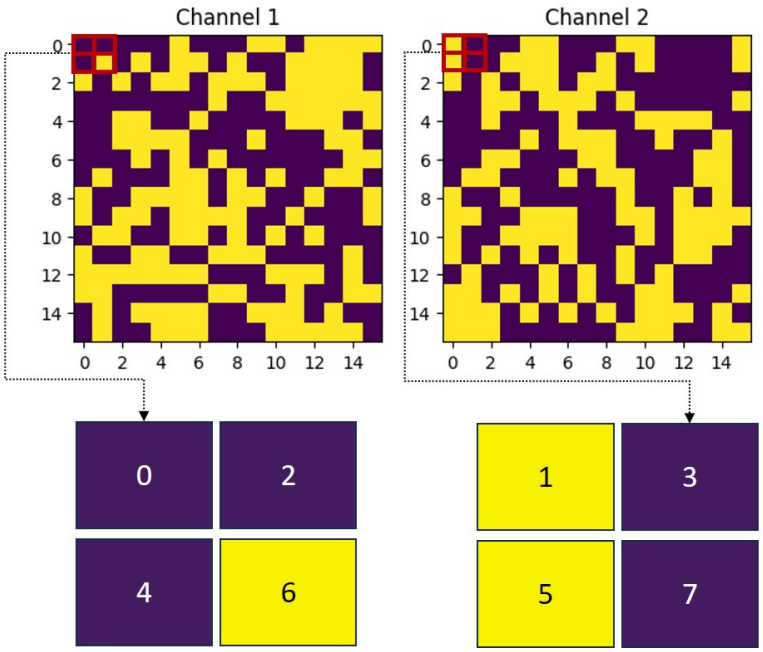

Figure 4.10: Position of the 8 spins (links) inside a patch. Only the spins $2,5,6,7$ forms the plaquette

We use a layer norm layer before the linear projection of x^1 and x^n. The results are for the given hyperparameters:

$$\begin{aligned}
&\text{Hidden dim } d = 6 \\
&\text{Patch size} = 2 \\
&\text{Number of layers} = 3 \\
&\text{Learning rate} = 0.01 \\
&\text{Epochs} = 100 \\
&\lambda = 0.0001
\end{aligned}$$

Fig:4.11 shows the classification results of the transformer on the 2 phases. We now look into the regularization path analysis Fig:4.12, which indicates that the transformer is learning 4^{th}-order correlations to differentiate between the 2 phases. This is promising because the gauge

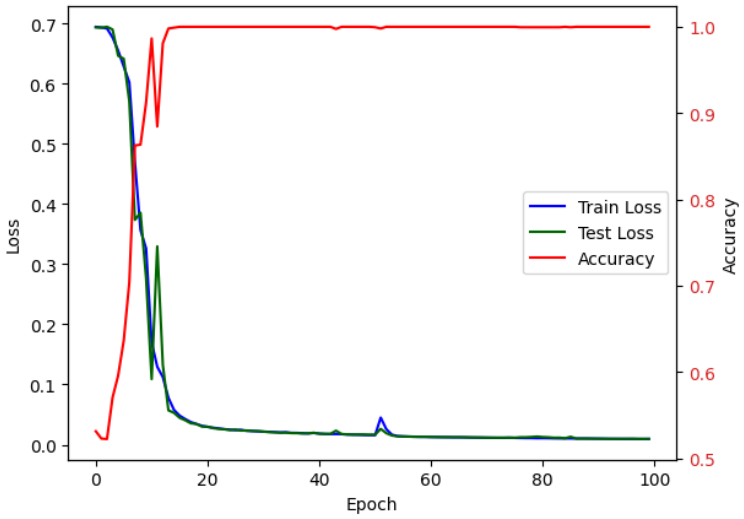

Figure 4.11: Accuracy and loss metrics for the Ising LGT classifications

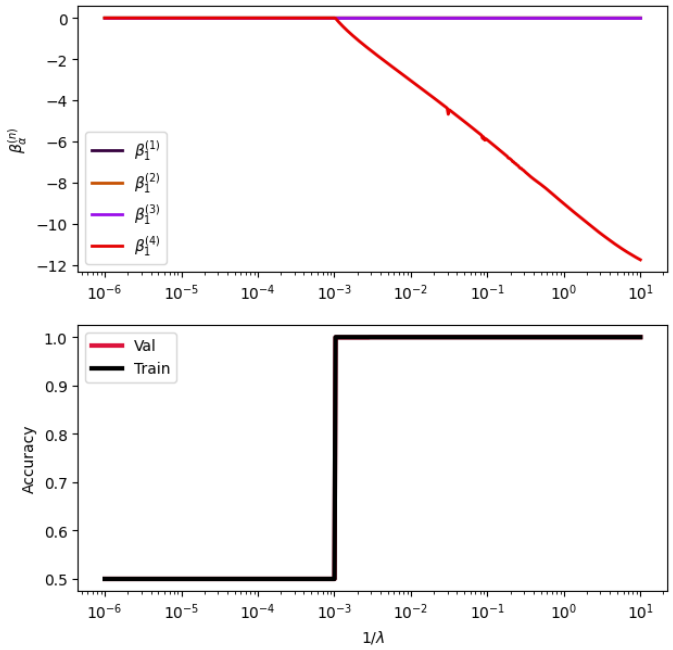

Figure 4.12: Regularization path analysis for the Ising LGT snapshots

constraint is a 4-point correlator. To check whether our model is learning the correct correlator, we construct the 4-point correlation matrix (Eqn: 4.17), where the indices go through all 8 spins inside a single (2X2X2) patch.

$$C^4_{hdrs} = \sum W^p_{hg} W^{p^T}_{ad} W^p_{rk} W^{p^T}_{ls} W^{1Q}_{km} W^{2Q^T}_{ca} W^{3Q}_{gp} W^{1K^T}_{ml} W^{2K}_{bc} W^{3K^T}_{pq} W^{1V}_{nb} W^{2V^T}_{qf} \tag{4.17}$$

Fig:4.13 depicts the weights of the 4-point correlators that the transformer learns, and we can see that the index 2567 has more weightage. As discussed in the classification in Chapter 3, we know that the spins at position $2,5,6,7$ include the 4 links that form the plaquette. We can clearly say that the transformer is learning the 4-point gauge constraints (Gauss's law) to

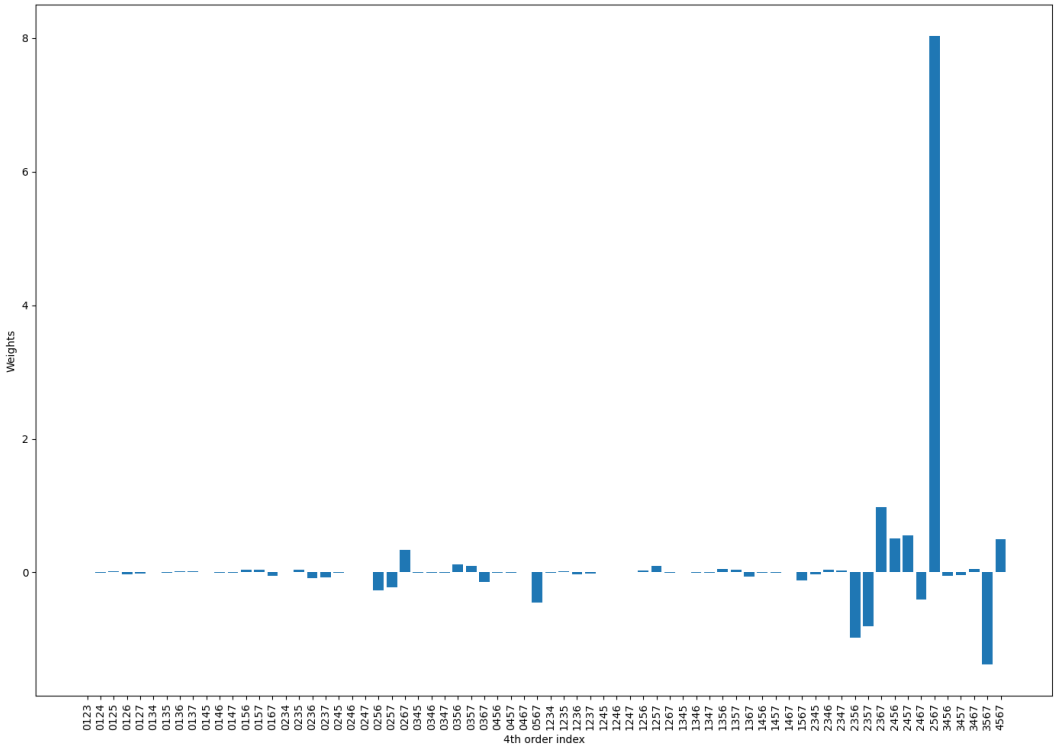

Figure 4.13: The 4^{th} order correlator weights learned by the transformer

distinguish between the zero-temperature and high-temperature phases.

4.5 Cooper pairs

Here, we demonstrate the ability of the transformer to capture long-range correlation, which most existing architectures find challenging. The synthetic data depicting momentum space snapshots are generated by distributing $+1$ for the spin up and -1 for the spin down on a $30X30$ grid with a Fermi radius of 10. The excited state has 2 cooper pairs and 2 pairs randomly distributed inside the Fermi surface. The ground state has 2 pairs on the Fermi surface, but none forms a pair. This is done so that the transformer won't just count the particles on the surface to make the classification. Fig: 4.14 shows one example from both the classes.

Here, we multiply an additional learnable positional embedding to x^1 before it enters the attention mechanism. The results are for the given hyperparameters:

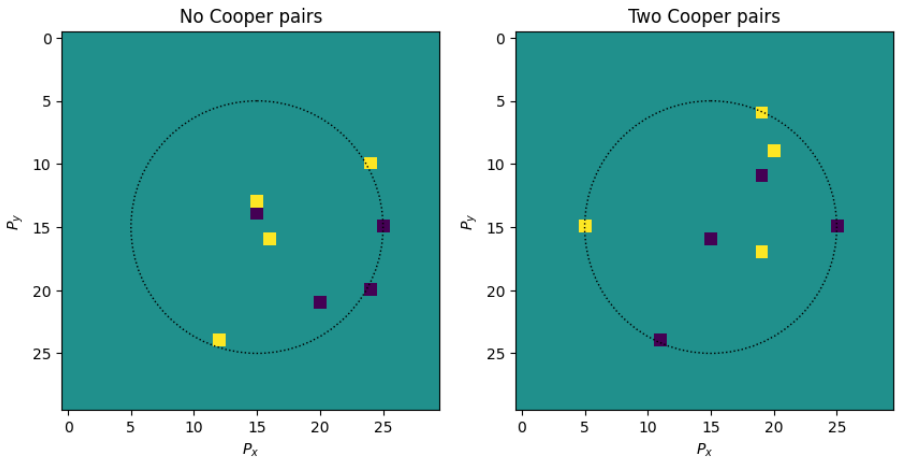

Figure 4.14: a) A ground state snapshot with zero cooper pairs, b) an excited state snapshot with 2 cooper pairs

Hidden dim $d = 16$

Patch size $= 2$

Number of layers $= 3$

Learning rate $= 0.01$

Epochs $= 50$

$\lambda = 0.0001$

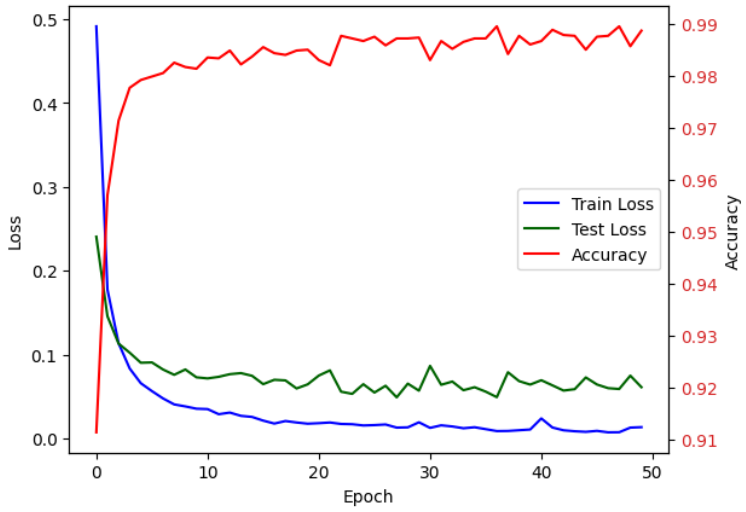

Figure 4.15: Accuracy and loss metric for Cooper pair classification

Fig: 4.16 indicates that the 2^{nd} order correlations are the most important for discriminating the two snapshot classes. We plot the 2^{nd}-order attention map to see which 2^{nd}-order correlator is the most important. Fig: 4.17 shows the attention map w.r.t the highlighted patch in red

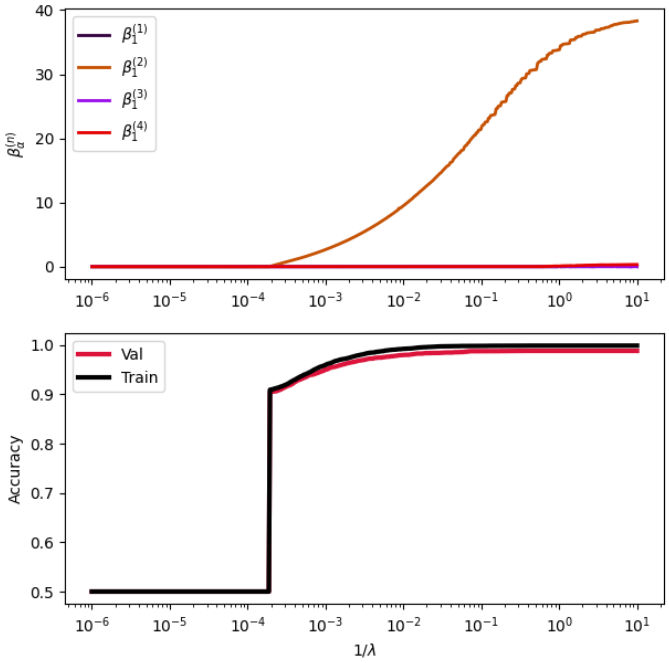

Figure 4.16: Regularization path analysis for the Cooper pair data

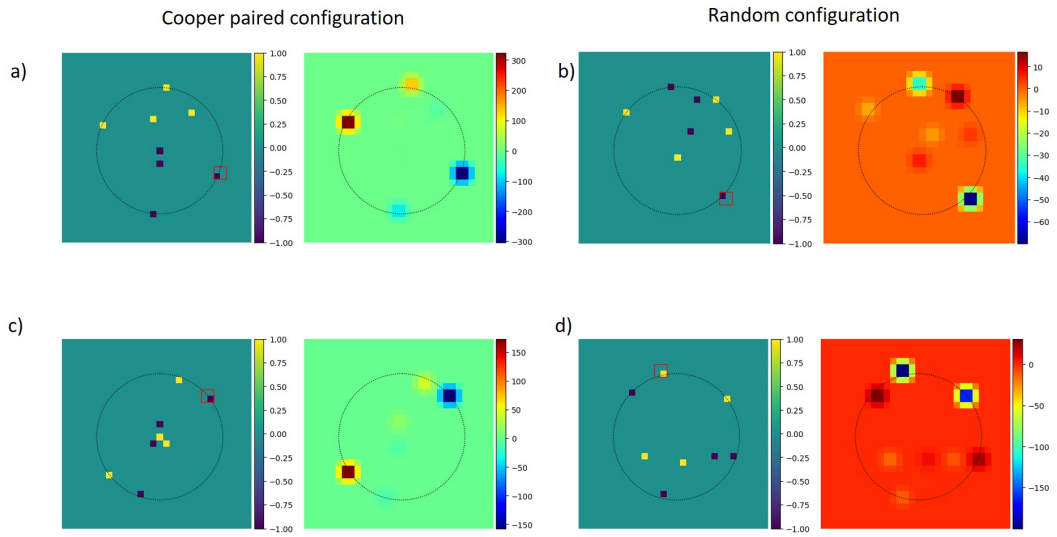

Figure 4.17: Attention maps w.r.t the highlighted particles on the Fermi surface.

square. We see that the transformer focuses on the opposite momentum pair in a cooper-paired state, while no such structure exists in the randomly distributed phase. We can conclude from the regularization path analysis and the attention maps that the transformer is learning to identify the long-range cooper pairs to distinguish between the two classes of snapshots.

4.6 Percolation

We use the same data from Chapter 3: percolated snapshots are connected spins from one edge to the other, and the non-percolated snapshots are where this connection is broken. Here,

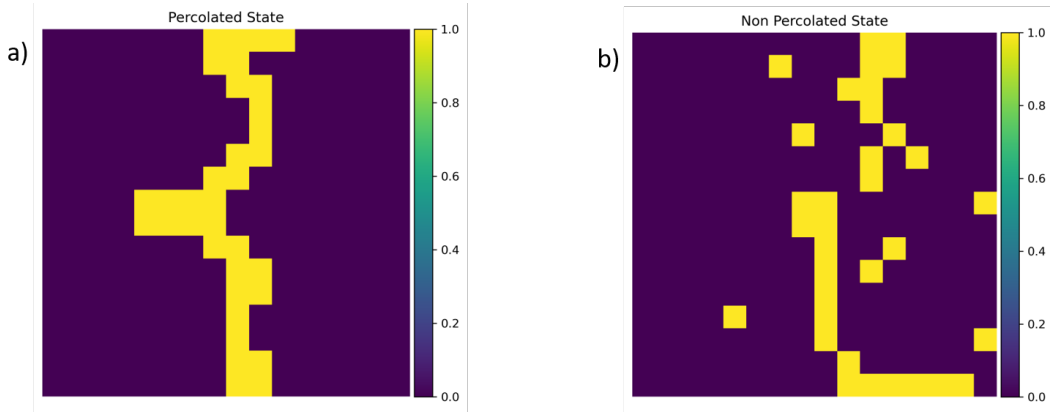

Figure 4.18: a) A percolating snapshot, b) a non-percolating snapshot

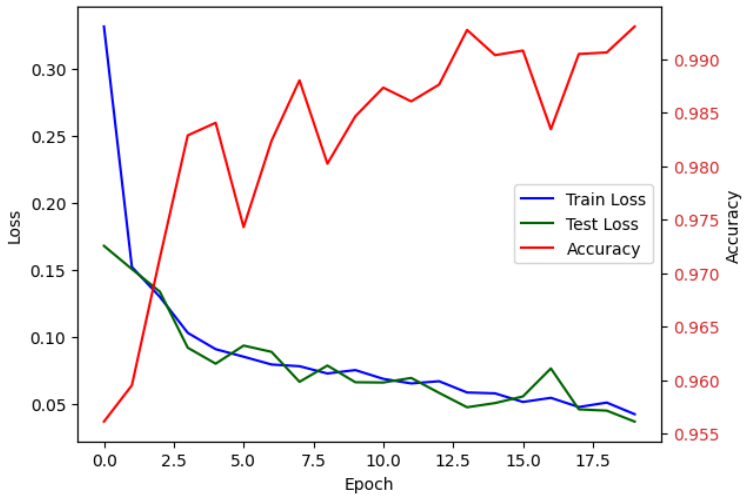

Figure 4.19: Accuracy and loss metric for the percolation classification

we multiply an additional learnable positional embedding to x^1 before it enters the attention mechanism. The results are for the following hyperparameters:

$$\begin{aligned}
&\text{Hidden dim } d = 16\\
&\text{Patch size} = 2\\
&\text{Number of layers} = 3\\
&\text{Learning rate} = 0.01\\
&\text{Epochs} = 20\\
&\lambda = 0.001
\end{aligned}$$

Fig: 4.19 depicts the classification metric, and we reach a very good accuracy from the beginning. The regularization path analysis Fig:4.20 tells us that the transformer uses 2^{nd}

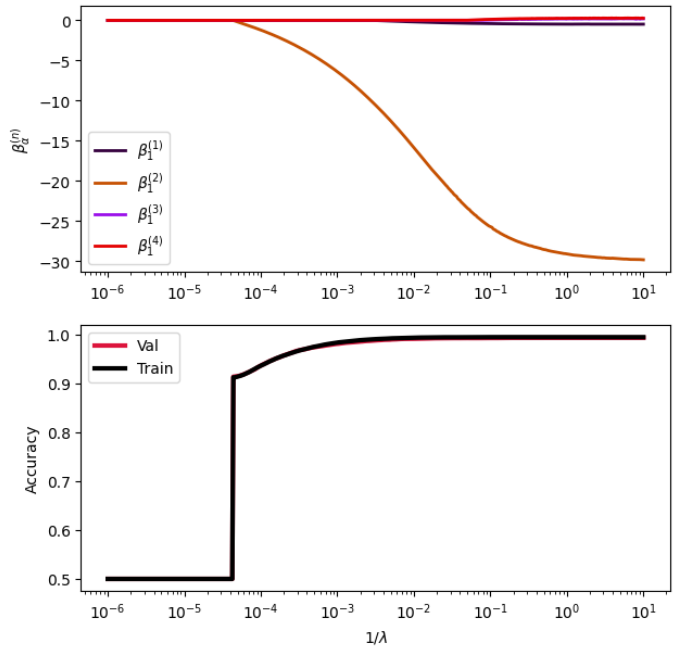

Figure 4.20: Regularization path analysis for the percolation problem

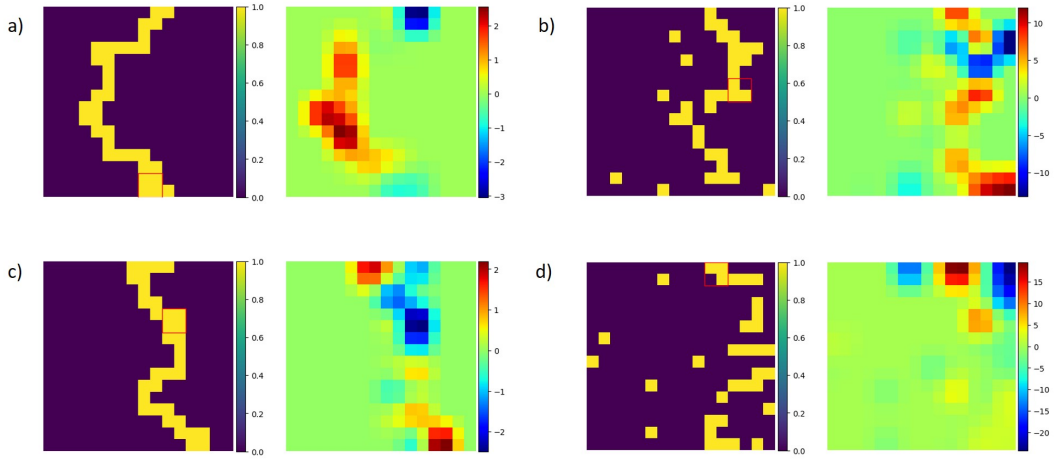

Figure 4.21: a) c) Attention maps for the percolated state, b) d) attention maps for the non-percolated state. All attention maps are w.r.t the highlighted patch in red.

order correlations to distinguish between the 2 classes of percolation data. We now explore what these 2^{nd} order correlations are by looking at the 2^{nd} order attention map Fig:4.21.

It is unclear what particular 2^{nd}-order terms the transformer learns, but we can see that the transformer attention map focuses on the percolation path. We plot the mean of this attention map for a set of snapshots from each class Fig:4.22. This tells us that the transformer finds discriminating features from the 2^{nd}-order attention map.

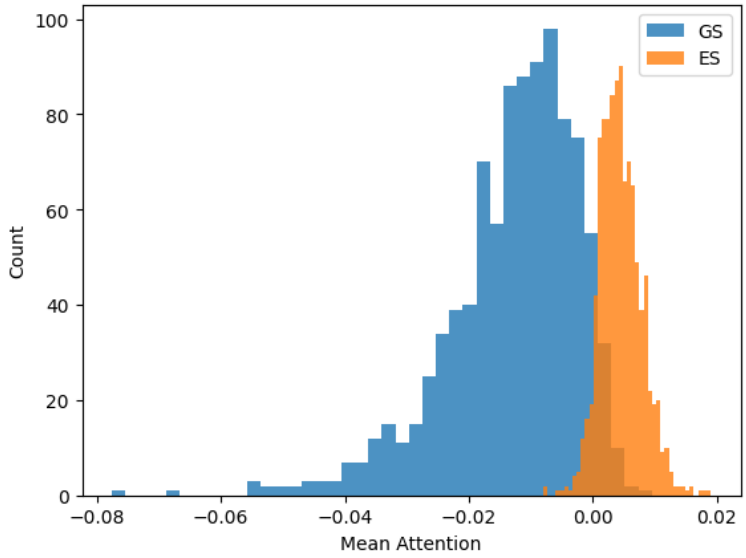

Figure 4.22: Mean of attention for both percolated (denoted by GS) and non-percolated (denoted by ES)

4.6.1 Note

It is important to note that the order of correlations given by the regularization paths is not 100% robust against changing hyperparameters. For e.g., in our testing, we found the transformer focused on 2^{nd} order correlators for classifying Ising LGT and by inspecting the weights, we found that it is indeed possible to construct a 2^{nd} order gauge constraint-like term. Also, we found 4^{th} order correlation for the cooper pair; that is, it could locate all the pairs on the surface to make the classification. This shows the strength of the transformer in computing higher-order long-range correlations. Thus, we have to be careful when interpreting such results.

Chapter 5

Neural Quantum States

5.1 Overview

Neural Quantum States (NQS) represent a novel approach in the field of condensed matter physics, leveraging the strength of artificial neural networks to model complex quantum systems. Carleo and Troyer's [20] seminal paper on neural quantum states using the Restricted Boltzmann Machine (RBM) opened the field of neural networks in solving many-body quantum systems. This approach has gained significant attention for its potential to solve problems that are intractable for traditional numerical methods. A neural network with a sufficient number of parameters and non-linear activation functions can be treated as a universal function approximator [49, 50, 51]. Since then, there have been implementations using Feed Forward Neural Networks (FFNN) [52, 53, 54, 55], Recurrent Neural Networks (RNN) [56, 57, 58, 59], Convolutional Neural Networks (CNN) [21, 60, 61, 62] and Auto encoders [63, 64, 60]. In this thesis, we use transformers as a variational ansatz generator. Previous works include but are not limited to [65, 66, 67, 68, 69].

Consider a system whose wave functions are parameterized by θ in configuration basis σ. The expectation value of any observable $\hat{O}$ is calculated as

$$\begin{aligned}
\langle \hat{O} \rangle &= \frac{\langle \Psi_\theta | \hat{O} | \Psi_\theta \rangle}{\langle \Psi_\theta | \Psi_\theta \rangle} \\
&= \frac{\sum_{\sigma,\sigma'} \langle \Psi_\theta | \sigma \rangle \langle \sigma | \hat{O} | \sigma' \rangle \langle \sigma' | \Psi_\theta \rangle}{\sum_\sigma |\langle \Psi_\theta | \sigma \rangle|^2} \\
&= \frac{\sum_\sigma \langle \Psi_\theta | \sigma \rangle \langle \sigma | \Psi_\theta \rangle \left(\sum_{\sigma'} \langle \sigma | \hat{O} | \sigma' \rangle \langle \sigma' | \Psi_\theta \rangle \right)}{\sum_\sigma |\langle \Psi_\theta | \sigma \rangle|^2} \\
&= \frac{\sum_\sigma |\langle \Psi_\theta | \sigma \rangle|^2 \left(\sum_{\sigma'} \langle \sigma | \hat{O} | \sigma' \rangle \langle \sigma' | \Psi_\theta \rangle \right)}{\sum_\sigma |\langle \Psi_\theta | \sigma \rangle|^2}.
\end{aligned} \tag{5.1}$$

We can see the above equation comprising of the following two terms,

$$P(\sigma) = \frac{|\langle \Psi_\theta | \sigma \rangle|^2}{\sum_\sigma |\langle \Psi_\theta | \sigma \rangle|^2} \tag{5.2}$$

$$O_{\text{loc}}(\sigma) = \sum_{\sigma'} \frac{\langle \sigma | \hat{O} | \sigma' \rangle \langle \sigma' | \Psi_\theta \rangle}{\langle \sigma | \Psi_\theta \rangle} \tag{5.3}$$

where $P(\sigma)$ denotes the probability amplitude of the wave function for a given configuration σ, and O_{loc} is the local estimator of the observable $\hat{O}$. We can approximate this quantum mechanical expectation value by sampling a sufficiently large number of samples $(\sigma^0, \sigma^1, ...\sigma^N)$from the probability distribution $P(\sigma)$ and can be expressed as

$$\langle \hat{O} \rangle \approx \frac{1}{N} \sum_{i=1}^{N} O_{\text{loc}}(\sigma^{(i)}) \tag{5.4}$$

where N is the number of samples. These samples are the snapshots we used in Chapter 3 for phase classification.

The idea of NQS is to generate appropriate variational wave function Ψ_θ and variationally reach $\langle \hat{O} \rangle$. We use transformer neural network to generate these variational anzast.

5.1.1 Variational Monte Carlo

Monte Carlo methods are a broad class of computational algorithms that estimate numerical outcomes through repeated random sampling. These methods are particularly useful in for solving problems that may be deterministic in principle but are too complex for analytical solutions.

For any trial wave function ψ_T, the energy expectation is calculated as follows

$$E_T = \frac{\langle \psi_T | \hat{H} | \psi_T \rangle}{\langle \psi_T | \psi_T \rangle}. \tag{5.5}$$

The true ground state energy E_0 of the Hamiltonian $\hat{H}$ will always be less than E_T.

$$E_0 \leq \frac{\langle \psi_T | \hat{H} | \psi_T \rangle}{\langle \psi_T | \psi_T \rangle}. \tag{5.6}$$

To obtain a reliable approximation of the ground state within the bounds of current computational power, it is essential that we parameterize the trial wave function using appropriate variational parameters. First, we construct a trial wave function $\psi_T = \psi_T(\sigma, \theta)$. for a many body configuration σ parameterised by m variational parameters $\theta = (\theta_1, \theta_2, ...\theta_m)$. The ex-

pectation value of the Hamiltonian is given by

$$E[H] = \langle H \rangle = \frac{\int d\sigma \, \Psi_T^*(\sigma,\theta) H(\sigma) \Psi_T(\sigma,\theta)}{\int d\sigma \, \Psi_T^*(\sigma,\theta) \Psi_T(\sigma,\theta)}. \tag{5.7}$$

We evaluate this integral using the Metropolis Hasting Algorithm. The parameters are updated according to Stochastic Reconfiguration, which is touched upon in the next section.

Algorithm 1 Variational Monte Carlo (VMC) Algorithm

1: Initialize wavefunction $\Psi_T(\sigma)$.
2: Initialize total energy accumulator $E_{\text{total}} = 0$.
2: **for** $i = 1$ to M **do**
3: Propose a spin exchange $\sigma \to \sigma'$.
4: Compute Metropolis acceptance ratio $A(\sigma \to \sigma') = min(1, \left|\frac{\Psi_T(\sigma';\vec{\theta})}{\Psi_T(\sigma;\vec{\theta})}\right|^2)$.
5: Generate a random number $r \in [0,1]$.
5: **if** $r < A(\sigma \to \sigma')$ **then**
6: Accept the move: $\sigma = \sigma'$.
7: Compute the local energy $E_L(\sigma) = \frac{\hat{H}\Psi_T(\sigma;\vec{\theta})}{\Psi_T(\sigma;\vec{\theta})}$.
8: Accumulate the energy: $E_{\text{total}} = E_{\text{total}} + E_L(\sigma)$.
9: **end loop**
10: Compute the average energy: $E_{\text{approx}} = \frac{E_{\text{total}}}{M}$.

5.1.2 Stochastic Reconfiguration

Introduced by [70], stochastic reconfiguration (SR) is a stochastic iterative method for energy minimization using variational Monte Carlo. Unlike standard optimization methods that use gradient descent, which can be inefficient due to the complex, high-dimensional landscapes typical in quantum problems, SR takes into account the curvature of the parameter space.

The goal is to find the ground state energy of a quantum system by iteratively updating the parameters of a trial wave function. This involves calculating the energy gradient concerning the parameters and the S matrix, which represents the metric of the parameter space.

Consider a variational wavefunction Ψ_θ parameterized by a set of parameters θ The expectation value of the Hamiltonian H is given by:

$$E(\theta) = \frac{\langle \Psi_\theta | H | \Psi_\theta \rangle}{\langle \Psi_\theta | \Psi_\theta \rangle} \tag{5.8}$$

We define an operator $\hat{O}_k$ at each iteration and parameter θ_k, which is diagonal in the configuration basis.

$$\hat{O}_k = \frac{\partial_{\theta_k} \psi_\theta}{\psi_\theta} \tag{5.9}$$

The gradient of the energy with respect to the parameters θ is:

$$F(\theta) = \frac{\partial E(\theta)}{\partial \theta} \tag{5.10}$$

Stochastic Reconfiguration updates the parameters θ according to:

$$\Delta\theta = -S^{-1}(\theta)F(\theta) \tag{5.11}$$

$S(\theta)$ is defined as:

$$S_{ij}(\theta) = \Re\left[\frac{\langle\Psi_\theta|\hat{O}_i^\dagger\hat{O}_j|\Psi_\theta\rangle}{\langle\Psi_\theta|\Psi_\theta\rangle} - \frac{\langle\Psi_\theta|\hat{O}_i^\dagger|\Psi_\theta\rangle\langle\Psi_\theta|\hat{O}_j|\Psi_\theta\rangle}{\langle\Psi_\theta|\Psi_\theta\rangle^2}\right] \tag{5.12}$$

Here, $\hat{O}_i$ and $\hat{O}_j$ are the operators corresponding to the partial derivatives of Ψ_θ $(\log(\Psi(\theta))$ with respect to θ_i and θ_j, and $\Re$ denotes the real part.

Algorithm 2 Stochastic Reconfiguration (SR) Algorithm

1: Perform a VMC simulation to collect M samples of configurations $\{\sigma_0, \sigma_1, ...\sigma_m\}$.
1: **for** each sample $i = 1$ to M **do**
2: Compute the derivatives of the wavefunction $\nabla_\theta \log\Psi_T(\sigma_i;\vec{\theta})$.
3: Accumulate $F(\vec{\theta})$ and $S(\vec{\theta})$ using E_L and $\nabla_\theta \log\Psi_T$.
4: Average $F(\vec{\theta})$ and $S(\vec{\theta})$ over M samples.
5: Solve the linear system $S(\vec{\theta})\Delta\vec{\theta} = -F(\vec{\theta})$ for $\Delta\vec{\theta}$.
6: Update the parameters: $\vec{\theta} \leftarrow \vec{\theta} + \eta\Delta\vec{\theta}$.

The variational Monte Carlo and optimization are done using Netket[71], and the transformer is written in Jax [72].

5.2 ViT NQS

Here, we implement ViT with talking multi-head attention, which allows all heads to share the information between them. This is achieved by having a linear projection of all the heads, resulting in the mixing of information from these heads. The input configurations are patched and projected to the hidden dimension space in the Embeddings block in Fig: 5.1 (a), then it goes through the multi-head attention and finally linearly projected and summed over all the patches to get $\log(\Psi)$ after applying the $\log(\cosh(.))$ non-linear activation function. Working with the logarithm of the wave function gives more stability for gradient propagation. We use relative positional embedding [25].

$$\text{Attention}(Q,K,V) = \left(\frac{QK^T}{\sqrt{d_k}} * F_p\right)V \tag{5.13}$$

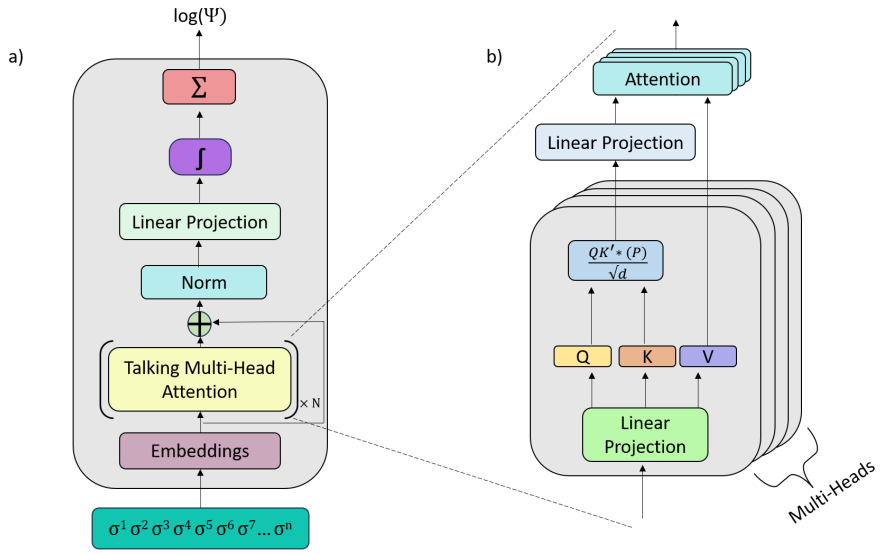

Figure 5.1: a) Schematic of the ViT NQS (The activation function $(\int)$ we use is log(cosh(.)), b) talking multi-head attention block

Where F_p is the Toeplitz matrix parametrized by scoring function f_θ. $(F_p)_{ij} = f_\theta(j-i)$, where $f_\theta(j-i)$ models the how the i^{th} position relate to the j^{th} and can be written as,

$$f_\theta(k) = \sum_{s=1}^{S} \alpha_s \exp(-|\beta_s|(k-\gamma_s)^2) \tag{5.14}$$

where $k = (i-j)$ and $\alpha_s, \beta_s, \gamma_s$ are trainable parameters for the kernel s.

5.3 Heisenberg AFM 1D

The one dimensional Heisenberg Hamiltonian is defined as

$$H = J \sum_{i=1}^{N} \left(S_i^x S_{i+1}^x + S_i^y S_{i+1}^y + S_i^z S_{i+1}^z \right) \tag{5.15}$$

S_i^α $(\alpha = x, y, z)$ are the spin operators at site i. J is the exchange coupling constant, which quantifies the strength and type (ferromagnetic for $J < 0$ or antiferromagnetic for $J > 0$) of the interaction between neighbouring spins. N is the total number of spins in the chain, and the periodic boundary condition is assumed ($S_{N+1} = S_1$).

5.3.1 Results

We use a hidden dim $d = 32$ and number of heads $h = 16$ for $L = 14, 16, 22$. For $L = 24$, $d = 48$ and $h = 24$. Patch size is 2 for the above lattice. The NQS ground state energy converges to the exact diagonalization results with relative error $\varepsilon \approx 0.01\%$ (Fig: 5.2). The relative error is given by $\varepsilon = \frac{|E_{nqs} - E_{exact}|}{E_{exact}}$.

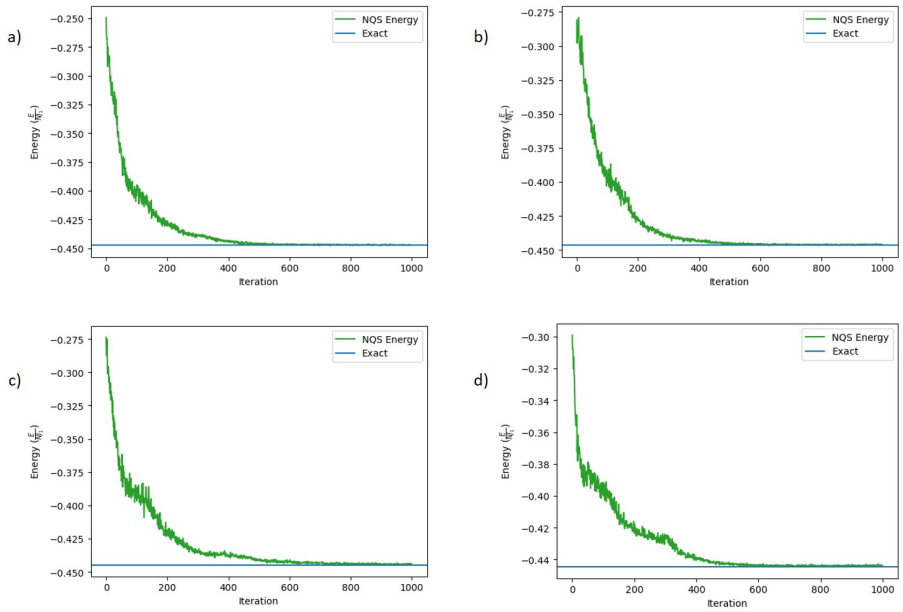

Figure 5.2: The ground state energy for lattice size a) L=14, b) L=16, c) L= 22 , d) L=24.

5.4 $J_1 - J_2$ Heisenberg 1D

The Hamiltonian for the 1D $J_1 - J_2$ Heisenberg chain is defined as:

$$H = J_1 \sum_i \vec{S}_i \cdot \vec{S}_{i+1} + J_2 \sum_i \vec{S}_i \cdot \vec{S}_{i+2} \tag{5.16}$$

The spin operator at site i, denoted as $\vec{S}_i$. This operator is a vector comprising: S_i^x, S_i^y, and S_i^z. The model is defined through two primary coupling constants, J_1 and J_2. The J_1 term quantifies the interaction energy between spins that are directly adjacent, acting as the coupling constant for nearest neighbour (NN) spin interactions. Antiferromagnetic coupling is indicated by a positive J_1, whereas ferromagnetic coupling is suggested by a negative J_1. J_2 represents the coupling constant for next-nearest neighbour (NNN) spin interactions, capturing the interaction energy between spins separated by two sites. The J_2 coupling adds frustration to the system if its interactions compete with those of J_1.

An important observable that we calculate is the structure factor, which is defined as

$$\chi(\vec{q}) = \frac{1}{L} \sum_{ij} \langle \sigma_i^z . \sigma_j^z \rangle e^{i\vec{q}(i-j)} \tag{5.17}$$

5.4.1 Results

The transformer uses $d = 32$, $h = 16$ and patch size =4 for lattice size $L = 16$. In the evaluation of the structure factor (Eqn: 5.17), we compute $\chi(\vec{q} = \pi)$.

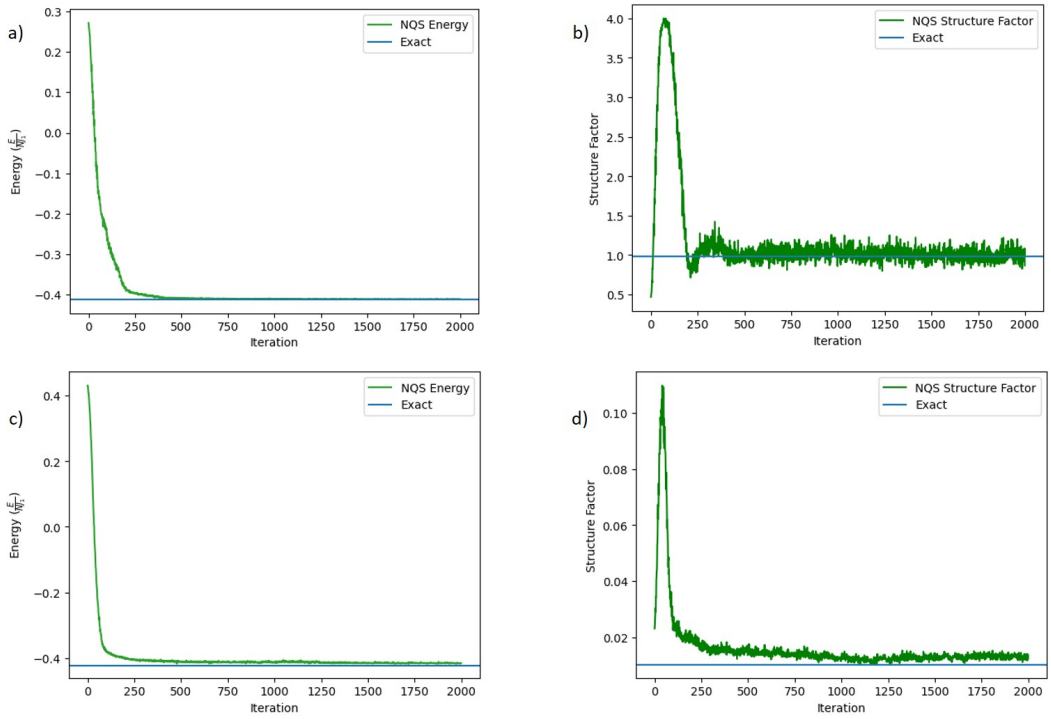

Figure 5.3: Results for system size L= 16 a),b) Energy and structure factor for $J_2/J_1 = 0.2$, c),d) Energy and structure factor for $J_2/J_1 = 0.8$

Fig 5.3 shows the results for a chain of length 16. While the transformer performs extremely well in the lower frustration limit, it has a reduced accuracy when the system is highly frustrated. For $J_2/J_1 = 0.8$ we have the relative error $\varepsilon \approx 1.4\%$ while $J_2/J_1 = 0.2$ have a better convergence in energy with $\varepsilon \approx 0.08\%$ For a slightly larger system of 40 spins (Fig: 5.4),

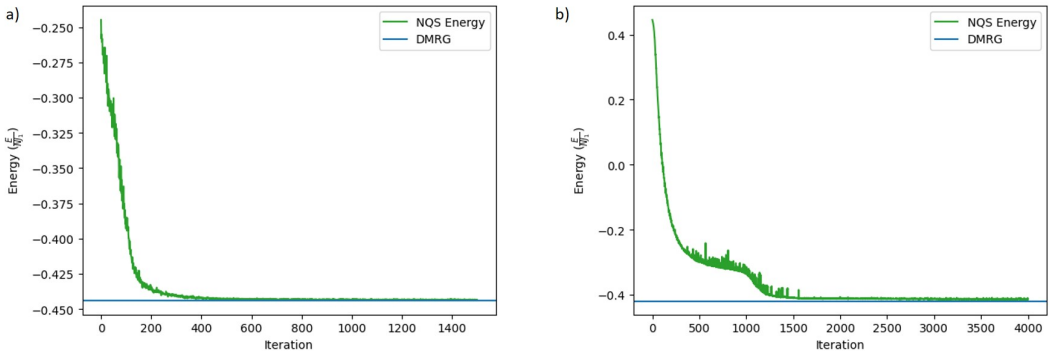

Figure 5.4: Energy for system size L= 40 a) For $J_2/J_1 = 0$, b) for $J_2/J_1 = 0.8$. The DMRG results are from [65].

we get similar convergence as previously. With $\varepsilon \approx 1.6\%$ for the highly frustrated case and $\varepsilon \approx 0.07\%$ for no next nearest neighbour coupling. We use $d = 32, h = 16$ and patch size= 10.

For system size L $= 8$ and $J_2/J_1 = 0.8$ (Fig: 5.5), the transformer finds the ground state of highly frustrated regime with excellent convergence $\varepsilon \approx 0.004\%$. We use $d = 16, h = 8$ and patch size= 2.

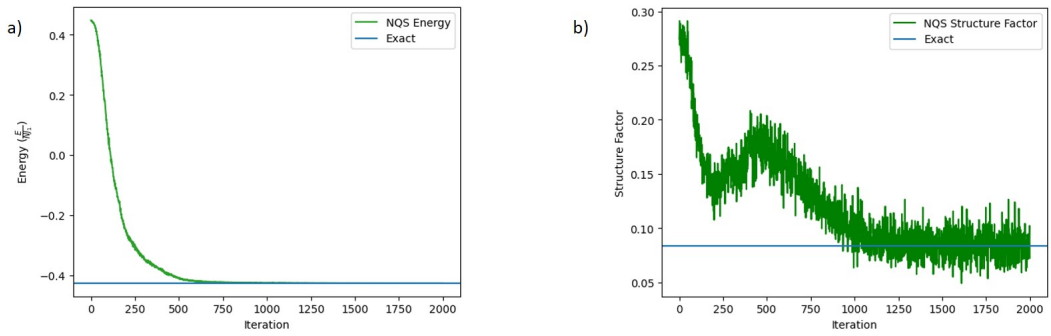

Figure 5.5: a) Ground state energy for system size L=8 with $J_2/J_1 = 0.8$, b) structure factor for the same system.

5.5 Heisenberg AFM 2D

The two-dimensional Heisenberg Hamiltonian is defined as

$$H = J \sum_{\langle i,j \rangle} \left(S_i^x S_j^x + S_i^y S_j^y + S_i^z S_j^z \right) \tag{5.18}$$

S_i^α $(\alpha = x, y, z)$ are the spin operators at lattice site i. J is the exchange coupling constant. J > 0 for antiferromagnetic coupling, while J < 0 for ferromagnetic coupling. The sum $\sum_{\langle i,j \rangle}$ runs over nearest-neighbour pairs of spins.r

5.5.1 Results

For the $4X4$ lattice, we use $d = 48$ and h=24, with patch size =2 and for $6x6$ lattice we use $d = 128$, h=64 and patch size =3. The number of parameters we use are less than their respective Hilbert space dimension, although of the same order. Fig 5.6 shows the ground state energy

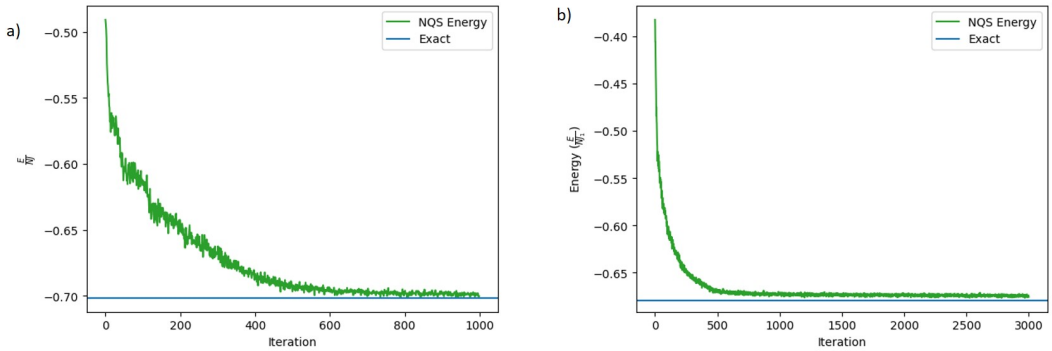

Figure 5.6: Ground state energy for a) L= 4X4, b) L = 6X6

obtained through ViT NQS with reference to the exact diagonalization results [73]. While $4X4$ shows relatively good convergence ($\varepsilon \approx 0.2\%$), $6X6$ lattice has an error of $\varepsilon \approx 0.5\%$.

Chapter 6

Conclusions and Outlook

Artificial intelligence, or, to be more specific, the application of deep neural networks in condensed matter physics problems, holds significant potential in discovering and probing fundamental physics. It has already been proven that these methods can be powerful tools in exploring domains where the traditional methods come short. With this great power comes great opaqueness in interpretability. This thesis has focused on making deep neural networks more "scientific" by exploring ways to shed light on what and why the model is able to classify certain physical states. We were successful in obtaining various higher-order correlators that are crucial for discriminating the phases. We also find long-range dependencies in synthetic cooper pair data, which traditional CNN models would have difficulty considering.

We also explored the transformer in its ability to parameterize many-body wave functions and reach good approximations for the ground state energy. Thus, we demonstrate the success of adopting transformer neural networks in the study of quantum many-body systems in the interpretability of phases and parametrizing the wave functions.

Future directions could include a general transformer architecture that could output the most relevant order correlations. The current thesis explored this, but the transformer was able to classify with orders of correlations that are different from standard theory for the model (for eg: Z_2 ising gauge theory). The ability of the transformer to find physically irrelevant correlations and classify them happens to be a big bottleneck in interpretability with regard to correlation functions. Regarding the NQS, many things could be done, including the time evolution of states and improving the architecture itself for better convergence, especially in the frustrated regime.

Bibliography

[1] S. R. White, "Density matrix formulation for quantum renormalization groups," *Phys. Rev. Lett.*, vol. 69, pp. 2863–2866, Nov 1992.

[2] F. Verstraete, D. Porras, and J. I. Cirac, "Density matrix renormalization group and periodic boundary conditions: A quantum information perspective," *Phys. Rev. Lett.*, vol. 93, p. 227205, Nov 2004.

[3] A. J. Daley, C. Kollath, U. SchollwĂ¶ck, and G. Vidal, "Time-dependent density-matrix renormalization-group using adaptive effective hilbert spaces," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2004, p. P04005, apr 2004.

[4] G. Vidal, "Efficient classical simulation of slightly entangled quantum computations," *Phys. Rev. Lett.*, vol. 91, p. 147902, Oct 2003.

[5] J. I. Cirac and F. Verstraete, "Renormalization and tensor product states in spin chains and lattices," *Journal of Physics A: Mathematical and Theoretical*, vol. 42, p. 504004, dec 2009.

[6] M. C. Bañuls, M. B. Hastings, F. Verstraete, and J. I. Cirac, "Matrix product states for dynamical simulation of infinite chains," *Phys. Rev. Lett.*, vol. 102, p. 240603, Jun 2009.

[7] M. Levin and C. P. Nave, "Tensor renormalization group approach to two-dimensional classical lattice models," *Phys. Rev. Lett.*, vol. 99, p. 120601, Sep 2007.

[8] F. Shenavarmasouleh and H. Arabnia, *DRDr: Automatic Masking of Exudates and Microaneurysms Caused by Diabetic Retinopathy Using Mask R-CNN and Transfer Learning*, pp. 307–318. 01 2021.

[9] W. Zhu, C. Liu, W. Fan, and X. Xie, "Deeplung: Deep 3d dual path nets for automated pulmonary nodule detection and classification," 2018.

[10] S. Bauer, C. May, D. Dionysiou, G. Stamatakos, P. BĂL'chler, and M. Reyes, "Multiscale modeling for image analysis of brain tumor studies," *IEEE Transactions on Biomedical Engineering*, vol. 59, pp. 25–29, 01 2012.

[11] J. B. Heaton, N. G. Polson, and J. H. Witte, "Deep learning in finance," 2018.

[12] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PLOS ONE*, vol. 12, pp. 1–24, 07 2017.

[13] O. B. Sezer, M. Ozbayoglu, and E. Dogdu, "A deep neural-network based stock trading system based on evolutionary optimized technical analysis parameters," *Procedia Computer Science*, vol. 114, pp. 473–480, 2017. Complex Adaptive Systems Conference with Theme: Engineering Cyber Physical Systems, CAS October 30 â€“ November 1, 2017, Chicago, Illinois, USA.

[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[15] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Ĺ˝Ădek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, p. 583â€"589, August 2021.

[16] K. Preuer, G. Klambauer, F. Rippmann, S. Hochreiter, and T. Unterthiner, "Interpretable deep learning in drug discovery," 2019.

[17] R. Ă–zĂ§elik, D. van Tilborg, J. JimĂ©nez-Luna, and F. Grisoni, "Structure-based drug discovery with deep learning," 2022.

[18] J. Zhou and O. Troyanskaya, "Predicting effects of noncoding variants with deep learning-based sequence model," *Nature Methods*, vol. 12, pp. 931–934, Sept. 2015. Funding Information: This work was primarily supported by US National Institutes of Health (NIH) grants R01 GM071966 and R01 HG005998 to O.G.T. This work was supported in part by the US National Science Foundation (NSF) CAREER award (DBI-0546275), NIH award T32 HG003284 and NIH grant P50 GM071508. O.G.T. is supported by the Genetic Networks program of the Canadian Institute for Advanced Research (CIFAR). We acknowledge the TIGRESS high-performance computer center at Princeton University for computational resource support. We are grateful to all Troyanskaya laboratory members for valuable discussions. Publisher Copyright: © 2015 Nature America, Inc. All rights reserved.

[19] J. Carrasquilla and R. G. Melko, "Machine learning phases of matter," *Nature Physics*, vol. 13, p. 431â€"434, Feb. 2017.

[20] G. Carleo and M. Troyer, "Solving the quantum many-body problem with artificial neural networks," *Science*, vol. 355, no. 6325, pp. 602–606, 2017.

[21] X. Liang, W.-Y. Liu, P.-Z. Lin, G.-C. Guo, Y.-S. Zhang, and L. He, "Solving frustrated quantum many-particle models with convolutional neural networks," *Phys. Rev. B*, vol. 98, p. 104426, Sep 2018.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[23] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2016.

[24] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," 2018.

[25] U. Wennberg and G. E. Henter, "The case for translation-invariant self-attention in transformer-based language models," in *Annual Meeting of the Association for Computational Linguistics*, 2021.

[26] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[27] A. Tanaka and A. Tomiya, "Detection of phase transition via convolutional neural networks," *Journal of the Physical Society of Japan*, vol. 86, p. 063001, June 2017.

[28] E. van Nieuwenburg, Y.-H. Liu, and S. Huber, "Learning phase transitions by confusion," *Nature Physics*, vol. 13, p. 435â€“439, Feb. 2017.

[29] K. Ch'ng, N. Vazquez, and E. Khatami, "Unsupervised machine learning account of magnetic transitions in the hubbard model," *Phys. Rev. E*, vol. 97, p. 013306, Jan 2018.

[30] P. Huembeli, A. Dauphin, and P. Wittek, "Identifying quantum phase transitions with adversarial neural networks," *Phys. Rev. B*, vol. 97, p. 134109, Apr 2018.

[31] S. J. Wetzel, "Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders," *Physical Review E*, vol. 96, Aug. 2017.

[32] L. Wang, "Discovering phase transitions with unsupervised learning," *Phys. Rev. B*, vol. 94, p. 195105, Nov 2016.

[33] W. Hu, R. R. P. Singh, and R. T. Scalettar, "Discovering phases, phase transitions, and crossovers through unsupervised machine learning: A critical examination," *Phys. Rev. E*, vol. 95, p. 062122, Jun 2017.

[34] P. Broecker, J. Carrasquilla, R. G. Melko, and S. Trebst, "Machine learning quantum phases of matter beyond the fermion sign problem," *Scientific Reports*, vol. 7, Aug. 2017.

[35] J.-L. Wynen, E. Berkowitz, S. Krieg, T. Luu, and J. Ostmeyer, "Machine learning to alleviate hubbard-model sign problems," *Physical Review B*, vol. 103, Mar. 2021.

[36] D. Carvalho, N. A. García-Martínez, J. L. Lado, and J. Fernández-Rossier, "Real-space mapping of topological invariants using artificial neural networks," *Phys. Rev. B*, vol. 97, p. 115453, Mar 2018.

[37] J. F. Rodriguez-Nieva and M. S. Scheurer, "Identifying topological order through unsupervised machine learning," *Nature Physics*, vol. 15, p. 790â€“795, May 2019.

[38] N. L. Holanda and M. A. R. Griffith, "Machine learning topological phases in real space," *Physical Review B*, vol. 102, Aug. 2020.

[39] E. Greplova, A. Valenti, G. Boschung, F. SchÄ¤fer, N. LÄ¶rch, and S. D. Huber, "Unsupervised identification of topological phase transitions using predictive models," *New Journal of Physics*, vol. 22, p. 045003, Apr. 2020.

[40] N. L. Holanda and M. A. R. Griffith, "Machine learning topological phases in real space," *Phys. Rev. B*, vol. 102, p. 054107, Aug 2020.

[41] P. Ponte and R. G. Melko, "Kernel methods for interpretable machine learning of order parameters," *Phys. Rev. B*, vol. 96, p. 205146, Nov 2017.

[42] K. Liu, J. Greitemann, and L. Pollet, "Learning multiple order parameters with interpretable machines," *Phys. Rev. B*, vol. 99, p. 104410, Mar 2019.

[43] P. Suchsland and S. Wessel, "Parameter diagnostics of phases and phase transition learning by neural networks," *Phys. Rev. B*, vol. 97, p. 174435, May 2018.

[44] N. C. Costa, W. Hu, Z. J. Bai, R. T. Scalettar, and R. R. P. Singh, "Principal component analysis for fermionic critical points," *Phys. Rev. B*, vol. 96, p. 195138, Nov 2017.

[45] J. Greitemann, K. Liu, and L. Pollet, "Probing hidden spin order with interpretable machine learning," *Phys. Rev. B*, vol. 99, p. 060404, Feb 2019.

[46] C. Miles, A. Bohrdt, R. Wu, C. S. Chiu, M. Xu, G. Ji, M. Greiner, K. Q. Weinberger, E. A. Demler, and E.-A. Kim, "Correlator convolutional neural networks as an interpretable architecture for image-like quantum matter data," *Nature Communications*, vol. 12, 2020.

[47] J. C. Halimeh, L. Homeier, C. Schweizer, M. Aidelsburger, P. Hauke, and F. Grusdt, "Stabilizing lattice gauge theories through simplified local pseudogenerators," *Phys. Rev. Res.*, vol. 4, p. 033120, Aug 2022.

[48] M. Holten, L. Bayha, K. Subramanian, S. Brandstetter, C. Heintze, P. Lunt, P. M. Preiss, and S. Jochim, "Observation of cooper pairs in a mesoscopic two-dimensional fermi gas," *Nature*, vol. 606, p. 287â€“291, June 2022.

[49] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems (MCSS)*, vol. 2, pp. 303–314, Dec. 1989.

[50] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[51] T. Kim and T. Adali, "Approximation by fully complex multilayer perceptrons," *Neural Comput.*, vol. 15, p. 1641â€“1666, jul 2003.

[52] Z. Cai and J. Liu, "Approximating quantum many-body wave functions using artificial neural networks," *Phys. Rev. B*, vol. 97, p. 035116, Jan 2018.

[53] K. Choo, G. Carleo, N. Regnault, and T. Neupert, "Symmetries and many-body excitations with neural-network quantum states," *Phys. Rev. Lett.*, vol. 121, p. 167204, Oct 2018.

[54] K. Çeven, M. O. Oktel, and A. Keleş, "Neural-network quantum states for a two-leg bose-hubbard ladder under magnetic flux," *Phys. Rev. A*, vol. 106, p. 063320, Dec 2022.

[55] Z. Zhu, M. Mattheakis, W. Pan, and E. Kaxiras, "Hubbardnet: Efficient predictions of the bose-hubbard model spectrum with deep neural networks," *Phys. Rev. Res.*, vol. 5, p. 043084, Oct 2023.

[56] M. Hibat-Allah, M. Ganahl, L. Hayward, R. Melko, and J. Carrasquilla, "Recurrent neural network wave functions," *Physical Review Research*, vol. 2, 06 2020.

[57] C. Roth, "Iterative retraining of quantum spin models using recurrent neural networks," 2020.

[58] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014.

[59] H. Lange, F. Doschl, J. Carrasquilla, and A. Bohrdt, "Neural network approach to quasiparticle dispersions in doped antiferromagnets," 2023.

[60] J.-Q. Wang, R.-Q. He, and Z.-Y. Lu, "Variational optimization of the amplitude of neural-network quantum many-body ground states," 2023.

[61] A. Chen and M. Heyl, "Efficient optimization of deep neural quantum states toward machine precision," 2023.

[62] C. Fu, X. Zhang, H. Zhang, H. Ling, S. Xu, and S. Ji, "Lattice convolutional networks for learning ground states of quantum many-body systems," 2022.

[63] A. Rocchetto, E. Grant, S. Strelchuk, G. Carleo, and S. Severini, "Learning hard quantum distributions with variational autoencoders," *npj Quantum Information*, vol. 4, June 2018.

[64] I. A. Luchnikov, A. Ryzhov, P.-J. Stas, S. N. Filippov, and H. Ouerdane, "Variational autoencoder reconstruction of complex many-body physics," *Entropy*, vol. 21, no. 11, 2019.

[65] L. L. Viteritti, R. Rende, and F. Becca, "Transformer variational wave functions for frustrated quantum spin systems," *Physical Review Letters*, vol. 130, June 2023.

[66] Y.-H. Zhang and M. Di Ventra, "Transformer quantum state: A multipurpose model for quantum many-body problems," *Physical Review B*, vol. 107, Feb. 2023.

[67] I. von Glehn, J. S. Spencer, and D. Pfau, "A self-attention ansatz for ab-initio quantum chemistry," 2023.

[68] R. Rende, F. Gerace, A. Laio, and S. Goldt, "What does self-attention learn from masked language modelling?," 2024.

[69] R. Rende, L. L. Viteritti, L. Bardone, F. Becca, and S. Goldt, "A simple linear algebra identity to optimize large-scale neural network quantum states," 2023.

[70] S. Sorella, "Wave function optimization in the variational monte carlo method," *Phys. Rev. B*, vol. 71, p. 241103, Jun 2005.

[71] F. Vicentini, D. Hofmann, A. SzabĂł, D. Wu, C. Roth, C. Giuliani, G. Pescia, J. Nys, V. Vargas-CalderĂłn, N. Astrakhantsev, and G. Carleo, "NetKet 3: Machine Learning Toolbox for Many-Body Quantum Systems," *SciPost Phys. Codebases*, p. 7, 2022.

[72] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[73] "2d heisenberg exact result l=6." https://github.com/orgs/netket/discussions/838.