Eur. Phys. J. C (2024) 84:192
https://doi.org/10.1140/epjc/s10052-024-12558-3

THE EUROPEAN PHYSICAL JOURNAL C

Regular Article - Theoretical Physics

The expressivity of classical and quantum neural networks on entanglement entropy

Chih-Hung Wu[1,a], **Ching-Che Yen**[2,b]

[1] Department of Physics, University of California, Santa Barbara, CA 93106, USA
[2] MediaTek Inc., Hsinchu, Taiwan

Received: 12 November 2023 / Accepted: 15 February 2024 / Published online: 26 February 2024

Abstract Analytically continuing the von Neumann entropy from Rényi entropies is a challenging task in quantum field theory. While the n-th Rényi entropy can be computed using the replica method in the path integral representation of quantum field theory, the analytic continuation can only be achieved for some simple systems on a case-by-case basis. In this work, we propose a general framework to tackle this problem using classical and quantum neural networks with supervised learning. We begin by studying several examples with known von Neumann entropy, where the input data is generated by representing $\mathrm{Tr}\,\rho_A^n$ with a generating function. We adopt KerasTuner to determine the optimal network architecture and hyperparameters with limited data. In addition, we frame a similar problem in terms of quantum machine learning models, where the expressivity of the quantum models for the entanglement entropy as a partial Fourier series is established. Our proposed methods can accurately predict the von Neumann and Rényi entropies numerically, highlighting the potential of deep learning techniques for solving problems in quantum information theory.

1 Introduction

The *von Neumann entropy* is widely regarded as an effective measure of quantum entanglement, and is often referred to as *entanglement entropy*. The study of entanglement entropy has yielded valuable applications, particularly in the context of quantum information and quantum gravity (see [1,2] for a review). However, the analytic continuation from the *Rényi entropies* to von Neumann entropy remains a challenge in quantum field theory for general systems. We tackle this problem using both classical and quantum neural networks to examine their expressive power on entanglement entropy and the potential for simpler reconstruction of the von Neumann entropy from Rényi entropies.

Quantum field theory (QFT) provides an efficient method to compute the n-th Rényi entropy with integer $n > 1$, which is defined as [3]

$$S_n(\rho_A) \equiv \frac{1}{1-n} \ln \mathrm{Tr}(\rho_A^n). \tag{1}$$

The computation is done by replicating the path integral representation of the reduced density matrix ρ_A by n times. This step is non-trivial; however, we will be mainly looking at examples where explicit analytic expressions of the Rényi entropies are available, especially in two-dimensional conformal field theories (CFT_2) [4–7]. Then upon analytic continuation of $n \to 1$, we have the von Neumann entropy

$$S(\rho_A) = \lim_{n\to 1} S_n(\rho_A). \tag{2}$$

The continuation can be viewed as an independent problem from computing the n-th Rényi entropy. Although the uniqueness of $S(\rho_A)$ from the continuation is guaranteed by Carlson's theorem with a sub-Hagedorn density of states, analytic expressions in closed forms are currently unknown for most cases.

Furthermore, while $S_n(\rho_A)$ are well-defined in both integer and non-integer n, determining it for a set of integer values $n > 1$ is not sufficient. To obtain the von Neumann entropy, we must also take the limit $n \to 1$ through a *space* of real $n > 1$. The relationship between the Rényi entropies and the von Neumann entropy is therefore complex, and the required value of n for a precise numerical approximation of $S(\rho_A)$ is not clear.

Along this line, we are motivated to adopt an alternative method proposed in [8], which would allow us to study the connection between higher Rényi entropies and von Neumann entropy "accumulatively." This method relies on defin-

[a] e-mail: chih-hungwu@physics.ucsb.edu (corresponding author)
[b] e-mail: johnson.yan@mediatek.com

ing a generating function that manifests as a Taylor series

$$G(w;\rho_A)=\sum_{k=1}^{\infty}\frac{\tilde{f}(k)}{k}w^k,\quad \tilde{f}(k)=\mathrm{Tr}[\rho_A(1-\rho_A)^k]. \quad (3)$$

Summing over k explicitly yields an absolutely convergent series that approximates the von Neumann entropy with increasing accuracy as $w\to 1$. This method has both numerical and analytical advantages, where we refer to [8] for explicit examples. Note that the accuracy we can achieve in approximating the von Neumann entropy depends on the truncation of the partial sum in k, which is case-dependent and can be difficult to evaluate. It becomes particularly challenging when evaluating the higher-order Riemann–Siegel theta function in the general two-interval case of CFT_2 [8], which remains an open problem.

On the other hand, deep learning techniques have emerged as powerful tools for tackling the analytic continuation problem [9–14], thanks to their universal approximation property. The universal approximation theorem states that artificial neural networks can approximate any continuous function under mild assumptions [15], where the von Neumann entropy is no exception. A neural network is trained on a dataset of known function values, with the objective of learning a latent manifold that can approximate the original function within the known parameter space. Once trained, the model can be used to make predictions outside the space by extrapolating the trained network. The goal is to minimize the prediction errors between the model's outputs and the actual function values. In our study, we frame the supervised learning task in two distinct ways: the first approach involves using densely connected neural networks to predict von Neumann entropy, while the second utilizes sequential learning models to extract higher Rényi entropies.

Instead of using a static "define-and-run" scheme, where the model structure is defined beforehand and remains fixed throughout training, we have opted for a dynamic "define-by-run" approach. Our goal is to determine the optimal model complexity and hyperparameters based on the input validation data automatically. To achieve this, we employ KerasTuner [16] with Bayesian optimization, which efficiently explores the hyperparameter space by training and evaluating different neural network configurations using cross-validation. KerasTuner uses the results to update a probabilistic model of the hyperparameter space, which is then used to suggest the next set of hyperparameters to evaluate, aiming to maximize expected performance improvement.

A similar question can be explicitly framed in terms of quantum machine learning, where a trainable quantum circuit can be used to emulate neural networks by encoding both the data inputs and the trainable weights using quantum gates. This approach bears many different names [17–22], but we will call it a *quantum neural network*. Unlike classical neural networks, quantum neural networks are defined through a series of well-defined unitary operations, rather than by numerically optimizing the weights for the non-linear mapping between targets and data. This raises a fundamental question for quantum computing practitioners: can *any unitary operation* be realized, or is there a particular characterization for the *learnable function class*? In other words, is the quantum model universal in its ability to express any function with the given data input? Answering these questions will not only aid in designing future algorithms, but also provide deeper insights into how quantum models achieve universal approximation [23,24].

Recent progress in quantum neural networks has shown that data-encoding strategies play a crucial role in their expressive power. The problem of data encoding has been the subject of extensive theoretical and numerical studies [25–28]. In this work, we build on the idea introduced in [29,30], which demonstrated the expressivity of quantum models as partial Fourier series. By rewriting the generating function for the von Neumann entropy in terms of a Fourier series, we can similarly establish the expressivity using quantum neural networks. However, the Gibbs phenomenon in the Fourier series poses a challenge in recovering the von Neumann entropy. To overcome this, we reconstruct the entropy by expanding the Fourier series into a basis of Gegenbauer polynomials.

The structure of this paper is as follows. In Sect. 2, we provide a brief overview for the analytic continuation of the von Neumann entropy from Rényi entropies within the framework of QFT. In addition, we introduce the generating function method that we use throughout the paper. In Sect. 3, we use densely connected neural networks with KerasTuner to extract the von Neumann entropy for several examples where analytic expressions are known. In Sect. 4, we employ sequential learning models for extracting higher Rényi entropies. Sect. 5 is dedicated to studying the expressive power of quantum neural networks in approximating the von Neumann entropy. In Sect. 6, we summarize our findings and discuss possible applications of our approach. Appendix A is devoted to the details of rewriting the generating function as a partial Fourier series, while Appendix B addresses the Gibbs phenomenon using Gegenbauer polynomials.

2 Analytic continuation of von Neumann entropy from Rényi entropies

Let us discuss how to calculate the von Neumann entropy in QFTs [31–34]. Suppose we start with a QFT on a d-dimensional Minkowski spacetime with its Hilbert space specified on a Cauchy slice Σ of the spacetime. Without loss of generality, we can divide Σ into two disjoint sub-regions $\Sigma=A\cup A^c$. Here A^c denotes the complement sub-region of A. Therefore, the Hilbert space also factorizes into the

tensor product $\mathcal{H}_\Sigma = \mathcal{H}_A \otimes \mathcal{H}_{A^c}$. We then define a reduced density matrix ρ_A from a pure state on Σ, which is therefore mixed, to capture the entanglement between the two regions. The von Neumann entropy $S(\rho_A)$ allows us to quantify this entanglement

$$S(\rho_A) \equiv -\operatorname{Tr}(\rho_A \ln \rho_A) = \frac{\text{Area}(\partial A)}{\epsilon^{d-2}} + \cdots . \tag{4}$$

Along with several nice properties, such as the invariance under unitary operations, complementarity for pure states, and a smooth interpolation between pure and maximally mixed states, it is therefore a fine-grained measure for the amount of entanglement between A and A^c. The second equality holds for field theory, where we require a length scale ϵ to regulate the UV divergence encoded in the short-distance correlations. The leading-order divergence is captured by the area of the entangling surface ∂A, a universal feature of QFTs [35].[1]

There have been efforts to better understand the structure of the entanglement in QFTs, including free theory [36], heat kernels [37,38], CFT techniques [39] and holographic methods based on AdS/CFT [40,41]. But operationally, computing the von Neumann entropy analytically or numerically is still a daunting challenge for generic interacting QFTs. For a review, see [1].

Path integral provides a general method to access $S(\rho_A)$. The method starts with the Rényi entropies [3]

$$S_n(\rho_A) = \frac{1}{1-n} \ln \operatorname{Tr} \rho_A^n, \tag{5}$$

for real $n > 1$. As previously mentioned, obtaining the von Neumann entropy via analytic continuation in n with $n \to 1$ requires two crucial steps. An analytic form for the n-th Rényi entropy must be derived from the underlying field theory in the first place, and then we need to perform analytic continuation toward $n \to 1$. These two steps are independent problems and often require different techniques. We will briefly comment on the two steps below.

Computing $\operatorname{Tr} \rho_A^n$ is not easy; therefore, the replica method enters. The early form of the replica method was developed in [34], and was later used to compute various examples in CFT_2 [4–7], which can be compared with holographic ones [42]. The idea behind the replica method is to consider an orbifold of n copies of the field theory to compute $\operatorname{Tr} \rho_A^n$ for positive integers n. The computation reduces to evaluating the partition function on a n-sheeted Riemann surface, which can be alternatively computed by correlation functions of twist operators in the n copies. For more details on the construction in CFTs, see [4–7]. If we are able to compute $\operatorname{Tr} \rho_A^n$ for any positive integer $n \geq 1$, we have

$$S(\rho_A) = \lim_{n\to 1} S_n(\rho_A) = -\lim_{n\to 1} \frac{\partial}{\partial n} \operatorname{Tr} \rho_A^n. \tag{6}$$

This is computable for special states and regions, such as ball-shaped regions for the vacuum of the CFT_d. However, in CFT_2, due to its infinite-dimensional symmetry being sufficient to fix lower points correlation functions, we are able to compute $\operatorname{Tr} \rho_A^n$ for several instances.

The analytic continuation in $n \to 1$ is more subtle. Ensuring the existence of a unique analytic extension away from integer n typically requires the application of the Carlson's theorem. This theorem guarantees the uniqueness of the analytic continuation from Rényi entropies to the von Neumann entropy, provided that we can find some locally holomorphic function $\mathcal{S}_\nu$ with $\nu \in \mathbb{C}$ such that $\mathcal{S}_n = S_n(\rho)$ for all integers $n > 1$ with appropriate asymptotic behaviors in $\nu \to \infty$. Then we have unique $S_\nu(\rho) = \mathcal{S}_\nu$ [43,44]. Carlson's theorem addresses not only the problem of unique analytic continuation but also the issue of continuing across non-integer values of the Rényi entropies.

There are other methods to evaluate $S(\rho_A)$ in the context of string theory and AdS/CFT; see for examples [45–50]. In this work, we would like to focus on an effective method outlined in [8] that is suitable for numerical considerations. In [8], the following generating function is used for the analytic continuation in n with a variable z

$$\begin{aligned} G(z;\rho_A) &\equiv -\operatorname{Tr}\left(\rho_A \ln \frac{1-z\rho_A}{1-z}\right) \\ &= \sum_{n=1}^{\infty} \frac{z^k}{k}\Big(\operatorname{Tr}(\rho_A^{k+1}) - 1\Big). \end{aligned} \tag{7}$$

This manifest Taylor series is absolutely convergent in the unit disc with $|z| < 1$. We can analytically continue the function from the unit disc to a holomorphic function in $\mathbb{C}\backslash[1,\infty)$ by choosing the branch cut of the logarithm to be along the positive real axis. The limit $z \to -\infty$ is within the domain of holomorphicity and is exactly where we obtain the von Neumann entropy

$$S(\rho_A) = \lim_{z\to-\infty} G(z;\rho_A). \tag{8}$$

However, a more useful form can be obtained by performing a Möbius transformation to a new variable $w = \frac{z}{z-1}$

$$G(w;\rho_A) = -\operatorname{Tr}\Big(\rho_A \ln\{1 - w(1-\rho_A)\}\Big). \tag{9}$$

It again manifests as a Taylor series

$$G(w;\rho_A) = \sum_{k=1}^{\infty} \frac{\tilde{f}(k)}{k} w^k, \tag{10}$$

[1] While in CFT_2, the leading divergence for a single interval A of length ℓ in the vacuum state on an infinite line is a logarithmic function of the length, this is the simplest example we will consider later.

where

$$\tilde{f}(k) = \mathrm{Tr}[\rho_A(1-\rho_A)^k] = \sum_{m=0}^{k} \frac{(-1)^m k!}{m!(k-m)!} \mathrm{Tr}\,(\rho_A^{m+1}). \tag{11}$$

We again have a series written in terms of $\mathrm{Tr}\,\rho_A^n$, and it is absolutely convergent in the unit disc $|w| < 1$. The convenience of using w is that by taking $w \to 1$, we have the von Neumann entropy

$$S(\rho_A) = \lim_{w\to 1} G(w;\rho_A) = \sum_{k=1}^{\infty} \frac{\tilde{f}(k)}{k}. \tag{12}$$

This provides an exact expression of $S(\rho_A)$ starting from a known expression of $\mathrm{Tr}\,\rho_A^n$. Numerically, we can obtain an accurate value of $S(\rho_A)$ by computing a partial sum in k. The method guarantees that by summing to sufficiently large k, we approach the von Neumann entropy with increasing accuracy.

However, a difficulty is that we need to sum up $k \sim 10^3$ terms to achieve precision within 10^{-3} in general [8]. It will be computationally costly for certain cases with complicated $\mathrm{Tr}\,\rho_A^n$. Therefore, one advantage the neural network framework offers is the ability to give accurate predictions with only a limited amount of data, making it a more efficient method.

In this paper, we focus on various examples from CFT_2 with known analytic expressions of $\mathrm{Tr}\,\rho_A^n$ [6], and we use the generating function $G(w;\rho_A)$ to generate the required training datasets for the neural networks.

3 Deep learning von Neumann entropy

This section aims to utilize deep neural networks to predict the von Neumann entropy via a supervised learning approach. By leveraging the gradient-based learning principle of the networks, we expect to find a non-linear mapping between the input data and the output targets. In the analytic continuation problem from the n-th Rényi entropy to the von Neumann entropy, such a non-linear mapping naturally arises. Accordingly, we consider $S_n(\rho_A)$ (equivalently $\mathrm{Tr}\,\rho_A^n$ and the generating function) as our input data and $S(\rho_A)$ as the target function for the training process. As supervised learning, we will consider examples where analytic expressions of both sides are available. Ultimately, we will employ the trained models to predict the von Neumann entropy across various physical parameter regimes, demonstrating the efficacy and robustness of the approach.

The major advantage of using deep neural networks lies in that they improve the accuracy of the generating function for computing the von Neumann entropy. As we mentioned, the accuracy of this method depends on where we truncate the partial sum, and it often requires summing up a large k in (12), which is numerically difficult. In a sense, it requires knowing much more information, such as those of the higher Rényi entropies indicated by $\mathrm{Tr}\,\rho_A^n$ in the series. Trained neural networks are able to predict the von Neumann entropy more accurately given much fewer terms in the input data. We can even predict the von Neumann entropy for other parameter spaces without resorting to any data from the generating function.

Furthermore, the non-linear mappings the deep neural networks uncover can be useful for investigating the expressive power of neural networks on the von Neumann entropy. Additionally, they can be applied to study cases where analytic continuations are unknown and other entanglement measures that require analytic continuations.

In the following subsections, we will give more details on our data preparation and training strategies, then we turn to explicit examples as demonstrations.

3.1 Model architectures and training strategies

Generating suitable training datasets and designing flexible deep learning models are empirically driven. In this subsection, we outline our strategies for both aspects.

Data preparation

To prepare the training datasets, we consider several examples with known $S(\rho_A)$. We use the generating function $G(w;\rho)$, which can be computed from $\mathrm{Tr}\,\rho_A^n$ for each example. This is equivalent to computing the higher Rényi entropies with different choices of physical parameters since the "information" available is always $\mathrm{Tr}\,\rho_A^n$. However, note that all the higher Rényi entropies are distinct information. Therefore, adopting the generating function is preferable to using $S_n(\rho_A)$ itself, as it approaches the von Neumann entropy with increasing accuracy, making the comparison more transparent.

We generate $N = 10{,}000$ input datasets for a fixed range of physical parameters, where each set contains $k_{\max} = 50$ terms in (12); their corresponding von Neumann entropies will be the targets. We limit the amount of data to mimic the computational cost of using the generating function. We shuffle the input datasets randomly and then split the data into 80% for training, 10% for validation, and 10% as the test datasets. Additionally, we use the trained neural networks to make predictions on another set of 10,000 test datasets with a different physical parameter regime and compare them with the correct values as a non-trivial test for each example.

Model design

To prevent overfitting and enhance the generalizability of our model, we have employed a combination of techniques in the design of neural networks. ReLU activation function is

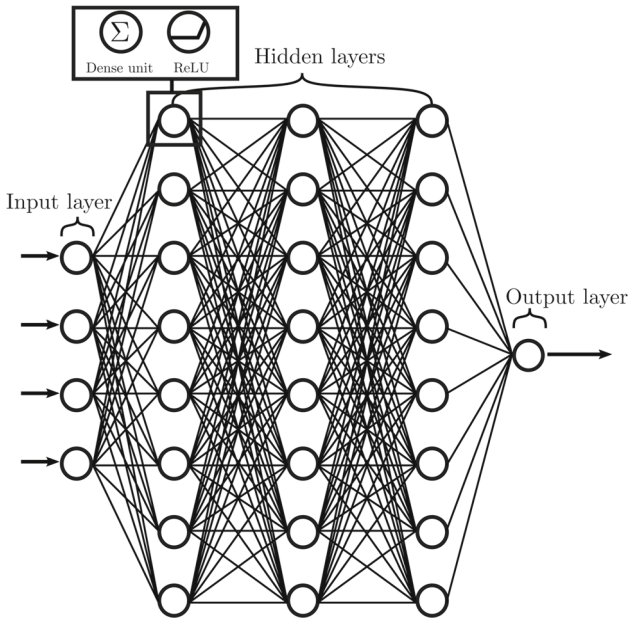

Fig. 1 An architecture of 3 densely connected layers, where each layer has 8 units. The final output layer is a single Dense unit with a unique output corresponding to the von Neumann entropy

used throughout the section. We adopt Adam optimizer [51] in the training process with mean square error (MSE) as the loss function.

We consider a neural network consisting of a few hidden Dense layers with varying numbers of units in TensorFlow-Keras [52,53]. In this case, each neuron in a layer receives input from all the neurons in the previous layer. The Dense connection allows the model to find non-linear relations between the input and output, which is the case for analytic continuation. The final layer is a Dense layer with a single unit that outputs a unique value for each training dataset, which is expected to correspond to the von Neumann entropy. As an example, we show a neural network with 3 hidden Dense layers, each with 8 units, in Fig. 1.

To determine the optimal setting of our neural networks, we employ KerasTuner [16], a powerful tool that allows us to explore different combinations of model complexity, depth, and hyperparameters for a given task. An illustration of the KerasTuner process can be found in Fig. 2. We use Bayesian optimization, and adjust the following designs and hyperparameters:

- We allow a maximum of 4 Dense layers. For each layer, we allow variable units in the range of 16 to 128 with a step size of 16. The number of units for each layer will be independent of each other.
- We allow BatchNormalization layers after the Dense layers as a Boolean choice to improve generalization and act as a regularization.
- A final dropout with log sampling of a dropout rate in the range of 0.1 to 0.5 is added as a Boolean choice.
- In the Adam optimizer, we only adjust the learning rate with log sampling from the range of 3×10^{-3} to 9×10^{-3}. All other parameters are taken as default values in TensorFlow-Keras. We also use the AMSGrad [54] variant of this algorithm as a Boolean choice.

We deploy the KerasTuner for 100 trials with 2 executions per trial and monitor the validation loss with EarlyStopping of patience 8. Once the training is complete, since we will not be making any further hyperparameter changes, we no longer evaluate performance on the validation data. A common practice is to initialize new models using the best model designs found by KerasTuner while also including the validation data as part of the training data. Indeed, we select the top 5 best designs and train each one 20 times with EarlyStopping of patience 8. We pick the one with the smallest relative errors from the targets among the 5×20 models as our final model. We set the batch size in both the KerasTuner and the final training to be 512.

In the following two subsections, we will examine examples from CFT_2 with $\text{Tr}\,\rho_A^n$ and their corresponding von Neumann entropies $S(\rho_A)$ [4–8]. These instances are distinct and worth studying for several reasons. They have different mathematical structures and lack common patterns in their derivation from the field theory side, despite involving the evaluation of certain partition functions. Moreover, the analytic continuation for each case is intricate, providing strong evidence for the necessity of independent model designs.

3.2 Entanglement entropy of a single interval

Throughout the following, we will only present the analytic expression of $\text{Tr}\,\rho_A^n$ since it is the only input of the generating function. We will also keep the UV cut-off ϵ explicit in the formula.

Single interval

The simplest example corresponds to a single interval A of length ℓ in the vacuum state of a CFT_2 on an infinite line. In this case, both the analytic forms of $\text{Tr}\,\rho_A^n$ and $S(\rho_A)$ are known [4], where $S(\rho_A)$ reduces to a simple logarithmic function that depends on ℓ. We have the following analytic form with a central charge c

$$\text{Tr}\,\rho_A^n = \left(\frac{\ell}{\epsilon}\right)^{\frac{c}{6}\left(\frac{1}{n}-n\right)}, \tag{13}$$

that defines $G(w;\rho_A)$. The corresponding von Neumann entropy is given by

$$S(\rho_A) = \frac{c}{3}\ln\frac{\ell}{\epsilon}. \tag{14}$$

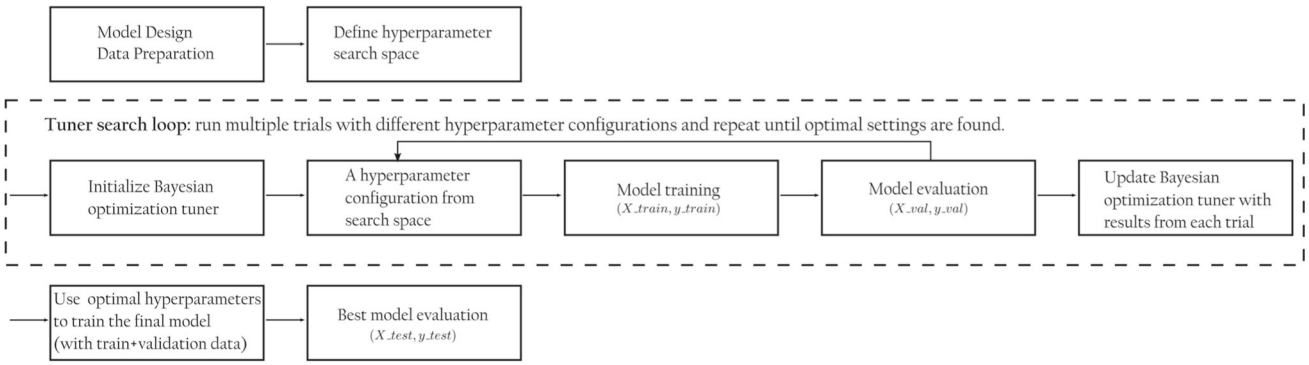

Fig. 2 Flowchart illustrating the steps of KerasTuner with Bayesian optimization. Bayesian optimization is a method for finding the optimal set of designs and hyperparameters for a given dataset, by iteratively constructing a probabilistic model from a prior distribution for the objective function and using it to guide the search. Once the tuner search loop is complete, we extract the best model in the final training phase by including both the training and validation data

We fixed the central charge $c = 1$ and the UV cutoff $\epsilon = 0.1$ when preparing the datasets. We generated 10,000 sets of data for the train-validation-test split from $\ell = 1$ to 50, with an increment of $\Delta\ell = 5 \times 10^{-3}$ between each step up to $k = 50$ in $G(w; \rho_A)$. To further validate our model, we generated an additional 10,000 test datasets for the following physical parameters: $\ell = 51$ to 100 with $\Delta\ell = 5 \times 10^{-3}$. For a density plot of the data distribution with respect to the target von Neumann entropy, see Fig. 3.

Figure 4 illustrates that the process outlined in the previous subsection effectively minimizes the relative errors in predicting the test data to a very small extent. Moreover, the model's effectiveness is further confirmed by its ability to achieve similarly small relative errors when predicting the additional test datasets. The accuracy of the model's predictions for the two test datasets significantly surpasses the approximate entropy obtained by summing the first 50 terms of the generating function, as can be seen in Fig. 5. We emphasize that in order for the generating function to achieve the same accuracy as the deep neural networks, we generally need to sum $k \geq 400$ from (12) [8]. This applies to all the following examples.

In this example, the von Neumann entropy is a simple logarithmic function, making it relatively straightforward for the deep learning models to decipher. However, we will now move on to a more challenging example.

Single interval at finite temperature and length

We extend the single interval case to finite temperature and length, where $\mathrm{Tr}\,\rho_A^n$ becomes a complicated function of the inverse temperature $\beta = T^{-1}$ and the length ℓ. The analytic expression of the Rényi entropies was first derived in [55] for a two-dimensional free Dirac fermion on a circle from bosonization. We can impose periodic boundary conditions that correspond to finite size and finite temperature. For simplicity, we set the total spatial size L to 1, and use ℓ to denote the interval length. In this case we have [55]

$$\mathrm{Tr}\,\rho_A^n = \prod_{k=-\frac{n-1}{2}}^{\frac{n-1}{2}} \left| \frac{2\pi\epsilon\eta(\tau)^3}{\theta_1(\ell|\tau)} \right|^{\frac{2k^2}{n^2}} \frac{|\theta_\nu(\frac{k\ell}{n}|\tau)|^2}{|\theta_\nu(0|\tau)|^2}, \tag{15}$$

where ϵ is a UV cutoff. We study the case of $\nu = 3$, which is the Neveu–Schwarz (NS-NS) sector. We then have the following Dedekind eta function $\eta(\tau)$ and the Jacobi theta functions $\theta_1(z|\tau)$ and $\theta_3(z|\tau)$

$$\eta(\tau) \equiv q^{\frac{1}{24}} \prod_{n=1}^{\infty}(1-q^n), \tag{16}$$

$$\theta_1(z|\tau) \equiv \sum_{n=-\infty}^{n=\infty} (-1)^{n-\frac{1}{2}} e^{(n+\frac{1}{2})^2 i\pi\tau} e^{(2n+1)\pi i z}, \tag{17}$$

$$\theta_3(z|\tau) \equiv \sum_{n=-\infty}^{n=\infty} e^{n^2 i\pi\tau} e^{2n\pi i z}. \tag{18}$$

Previously, the von Neumann entropy after analytically continuing (15) was only known in the high- and low-temperature regimes [55]. In fact, only the infinite length or zero temperature pieces are universal. However, the analytic von Neumann entropy for all temperatures was recently worked out by [56–58], which we present below

$$S(\rho_A) = \frac{1}{3}\log\frac{\sigma(\ell)}{\epsilon} + 4i\ell \times \int_0^\infty dq \frac{\zeta(iq\ell + 1/2 + i\beta/2) - \zeta(1/2) - \zeta(i\beta/2)}{e^{2\pi q} - 1}. \tag{19}$$

Here σ and ζ are the Weierstrass sigma function and zeta function with periods 1 and $i\beta$, respectively. We can see clearly that the analytic expressions for both $\mathrm{Tr}\,\rho_A^n$ and $S(\rho_A)$ are rather different compared to the previous example.

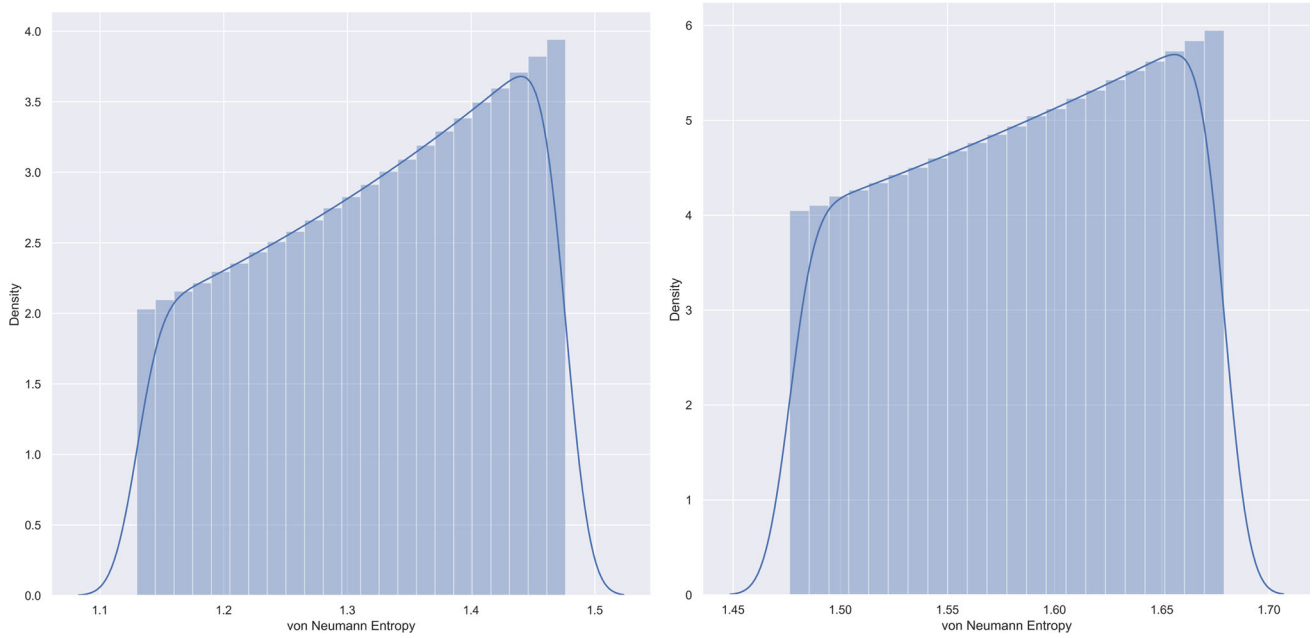

Fig. 3 The distribution of the data for the case of a single interval, where we plot density as a function of the von Neumann entropy computed by (14) with varying ℓ. The left plot represents the 10,000 datasets for the train-validation-test split, while the right plot corresponds to the additional 10,000 test datasets with a different physical parameter regime. The blue curves represent the kernel density estimate for a smoothed estimate of the data distribution

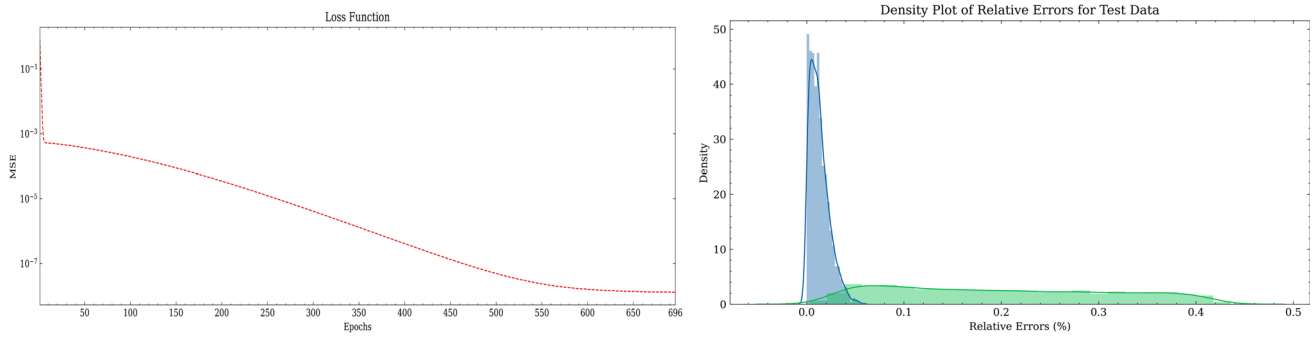

Fig. 4 Left: The MSE loss function as a function of epochs. We monitor the loss function with EarlyStopping, where the minimum loss is achieved at epoch 410 with loss $\approx 10^{-7}$ for this instance. Right: The density plot of relative errors between the model predictions and targets. Note that the blue color corresponds to the test datasets from the initial train-validation-test split, while the green color is for the additional test datasets. We can see clearly that for both datasets, we have achieved high accuracy with relative errors $\lesssim 0.30\%$

In preparing the datasets, we fixed the interval length $\ell = 0.5$ and the UV cutoff $\epsilon = 0.1$. We generated 10,000 sets of data for train-validation-test split from $\beta = 0.5$ to 1.0, with an increment of $\Delta\beta = 5 \times 10^{-5}$ between each step up to $k = 50$ in $G(w; \rho_A)$. Since β corresponds to the inverse temperature, this is a natural parameter to vary as the formula (19) is valid for all temperatures. To further validate our model, we generated 10,000 additional test datasets for the following physical parameters: $\beta = 1.0$ to 1.5 with $\Delta\beta = 5 \times 10^{-5}$. A density plot of the data with respect to the von Neumann entropy is shown in Fig. 6. As shown in Figs. 7 and 8, our model demonstrates its effectiveness in predicting both test datasets, providing accurate results for this highly non-trivial example.

3.3 Entanglement entropy of two disjoint intervals

We now turn to von Neumann entropy for the union of two intervals on an infinite line. In this case, several analytic expressions can be derived for both Rényi and von Neumann entropies. The theory we will consider is a CFT_2 for a free boson with central charge $c = 1$, and the von Neu-

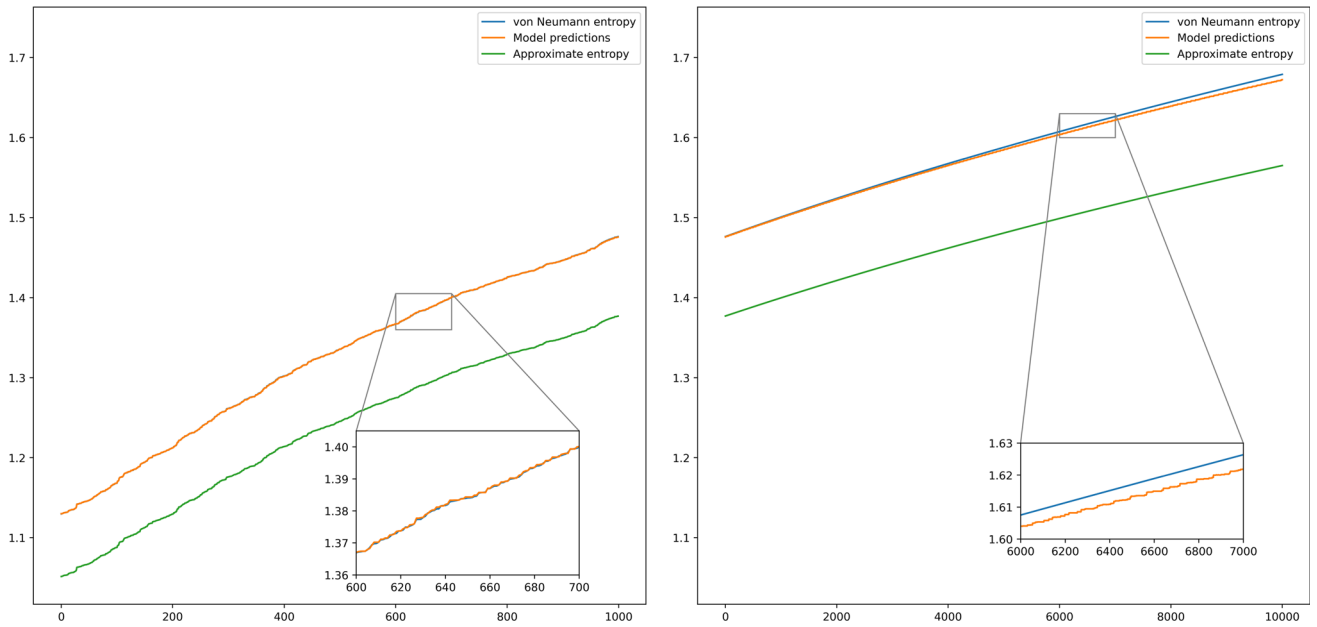

Fig. 5 We plot the predictions from the model with the analytic von Neumann entropy computed by (14) for the 1000 test datasets (left) from the training-validation-test split and the additional 10,000 test datasets (right), with the same scale on both figures. The y-axis represents the numerical value of the entropies, while the x-axis denotes the index of samples from the test datasets. We have re-ordered the samples with increasing entropy values. The correct von Neumann entropy overlaps with the model's predictions precisely. We have also included the approximate entropy by summing over $k = 50$ terms in the generating function

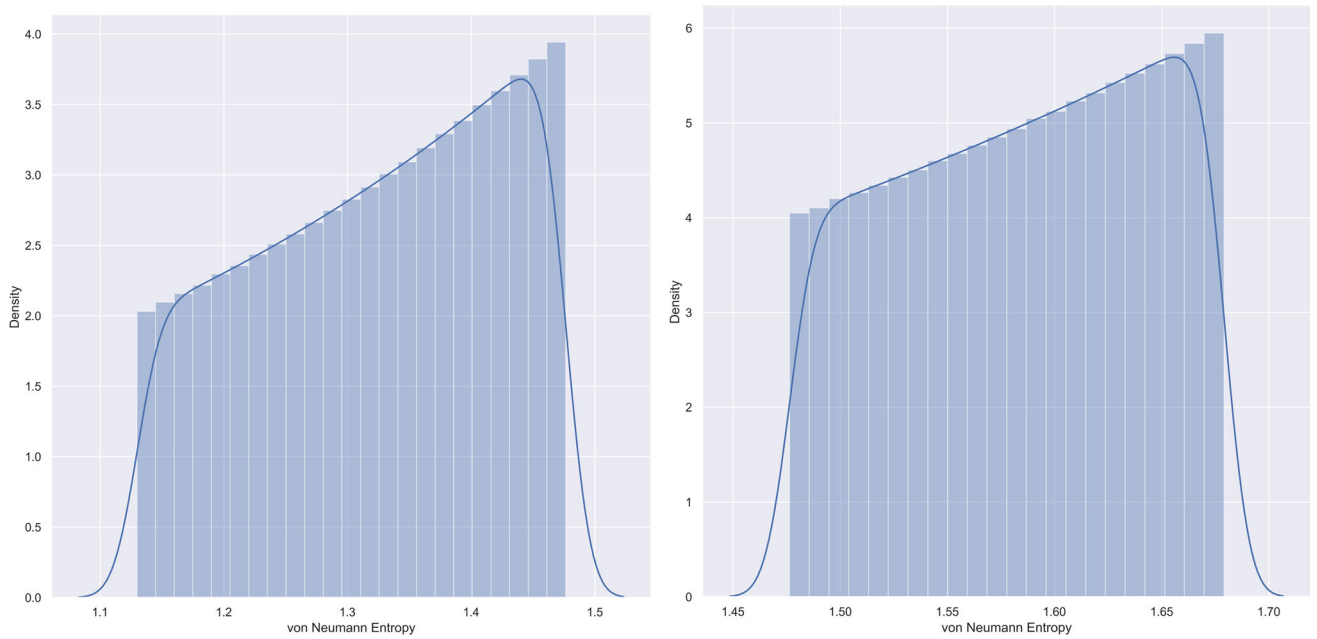

Fig. 6 The distribution of the two test datasets for the case of a single interval at finite temperature and length, where we plot density as a function of the von Neumann entropy computed by (19) with varying β. The blue curves represent the kernel density estimate for a smoothed estimate of the data distribution

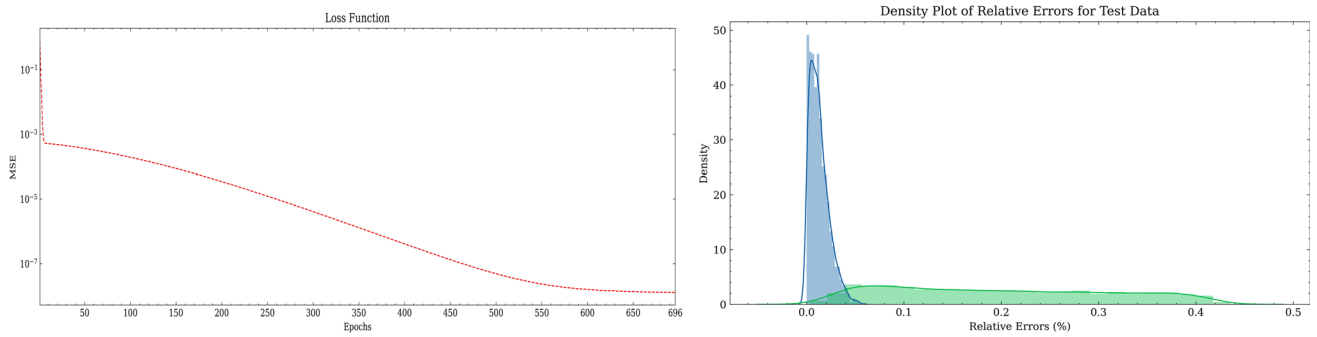

Fig. 7 Left: The MSE loss function as a function of epochs. The minimum loss close to 10^{-8} is achieved at epoch 86 for this instance. Right: The relative errors between the model predictions and targets for the two test datasets, where we have achieved high accuracy with relative errors $\lesssim 0.6\%$

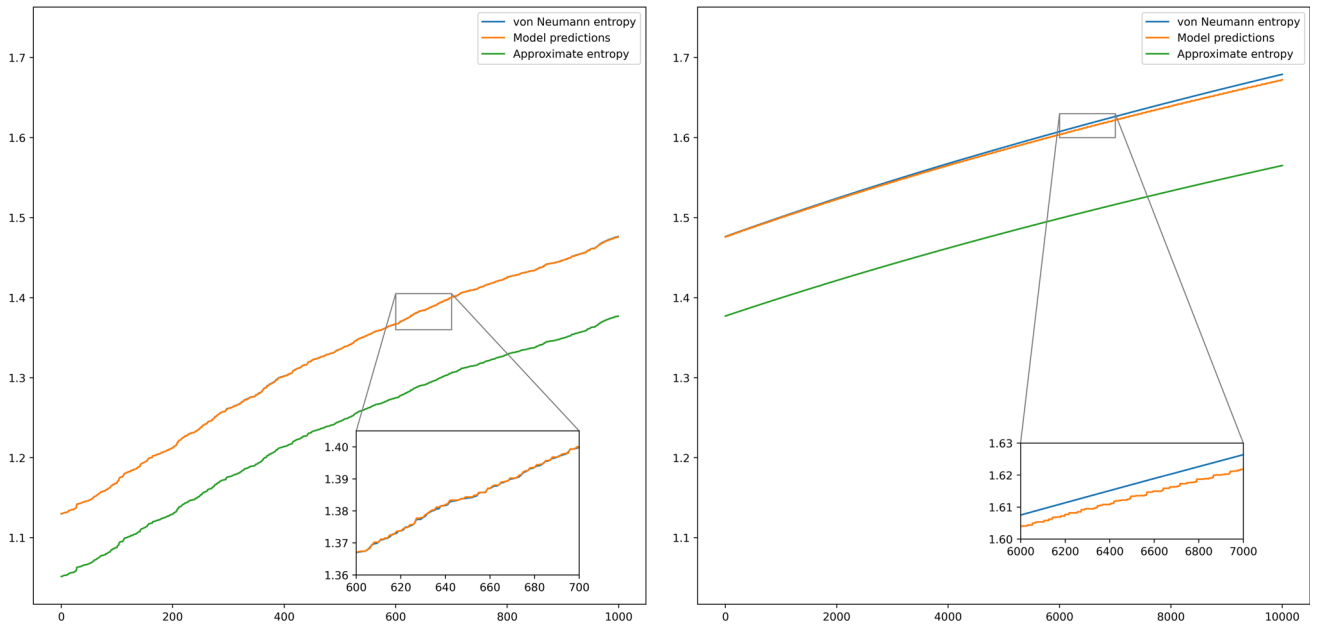

Fig. 8 We plot the predictions from the model with the analytic von Neumann entropy computed by (19) for the two test datasets. Again, the approximate entropy by summing over $k = 50$ terms in the generating function is included. Note that in order to achieve the same accuracy from the generating function, it requires at least $k \approx 700$ terms in this case

mann entropy will be distinguished by two parameters, a cross-ratio x and a universal critical exponent η. The latter is proportional to the square of the compactification radius.

To set up the system, we define the union of the two intervals as $A \cup B$ with $A = [x_1, x_2]$ and $B = [x_3, x_4]$. The cross-ratio is defined to be

$$x = \frac{x_{12}x_{34}}{x_{13}x_{24}}, \quad x_{ij} = x_i - x_j. \tag{20}$$

With the definition, we can write down the generating function for two intervals in a free boson CFT with finite x and η [5]

$$\mathrm{Tr}(\rho^n) = c_n \left(\frac{\epsilon^2 x_{13}x_{24}}{x_{12}x_{34}x_{14}x_{23}} \right)^{\frac{1}{6}(n-\frac{1}{n})} \mathcal{F}_n(x, \eta), \tag{21}$$

where ϵ is a UV cutoff and c_n is a model-dependent coefficient [6] that we set to $c_n = 1$ for simplicity. An exact expression for $\mathcal{F}_n(x, \eta)$ is given by

$$\mathcal{F}_n(x, \eta) = \frac{\Theta(0|\eta\Gamma)\Theta(0|\Gamma/\eta)}{[\Theta(0|\Gamma)]^2}, \tag{22}$$

for integers $n \geq 1$. Here $\Theta(z|\Gamma)$ is the Riemann–Siegel theta function defined as

$$\Theta(z|\Gamma) \equiv \sum_{m \in \mathbb{Z}^{n-1}} \exp[i\pi m^t \cdot \Gamma \cdot m + 2\pi i m^t \cdot z], \tag{23}$$

where Γ is a $(n-1) \times (n-1)$ matrix with elements

$$\Gamma_{rs} = \frac{2i}{n} \sum_{k=1}^{n-1} \sin\left(\frac{\pi k}{n}\right) \beta_{k/n} \cos\left[\frac{2\pi k}{n}(r-s)\right], \tag{24}$$

and

$$\beta_y = \frac{F_y(1-x)}{F_y(x)}, \qquad F_y(x) \equiv {}_2F_1(y, 1-y; 1; x), \tag{25}$$

where ${}_2F_1$ is the hypergeometric function. A property of this example is that (22) is manifestly invariant under $\eta \leftrightarrow 1/\eta$.

The analytic continuation towards the von Neumann entropy is not known, making it impossible to study this example directly with supervised learning. Although the Taylor series of the generating function guarantees convergence towards the true von Neumann entropy for sufficiently large values of k in the partial sum, evaluating the higher-dimensional Riemann–Siegel theta function becomes increasingly difficult. For efforts in this direction, see [59,60]. However, we will revisit this example in the next section when discussing the sequence model.

However, there are two limiting cases where analytic perturbative expansions are available, and approximate analytic continuations of the von Neumann entropies can be obtained. The first limit corresponds to small values of the cross-ratio x, where the von Neumann entropy has been computed analytically up to second order in x. The second limit is the decompactification limit, where we take $\eta \to \infty$. In this limit, there is an approximate expression for the von Neumann entropy.

Two intervals at small cross-ratio

Let us consider the following expansion of $\mathcal{F}_n(x,\eta)$ at small x for some $\eta \neq 1$

$$\mathcal{F}_n(x,\eta) = 1 + \left(\frac{x}{4n^2}\right)^{\alpha} s_2(n) + \left(\frac{x}{4n^2}\right)^{2\alpha} s_4(n) + \cdots, \tag{26}$$

where we can look at the first order contribution with

$$s_2(n) \equiv \mathcal{N}\frac{n}{2}\sum_{j=1}^{n-1} \frac{1}{[\sin(\pi j/n)]^{2\alpha}}. \tag{27}$$

The coefficient α for a free boson is given by $\alpha = \min[\eta, 1/\eta]$. $\mathcal{N}$ is the multiplicity of the lowest dimension operators, where for a free boson we have $\mathcal{N} = 2$. Up to this order, the analytic von Neumann entropy is given by

$$S(\rho_{AB}) = \frac{1}{3}\ln\left(\frac{x_{12}x_{34}x_{14}x_{23}}{\epsilon^2 x_{13}x_{24}}\right) - \mathcal{N}\left(\frac{x}{4}\right)^{\alpha} \frac{\sqrt{\pi}\Gamma(\alpha+1)}{4\Gamma\left(\alpha+\frac{3}{2}\right)} - \cdots. \tag{28}$$

We can set up the numerics by taking $|x_{12}| = |x_{34}| = r$, and the distance between the centers of A and B to be L, then the cross-ratio is simply

$$x = \frac{x_{12}x_{34}}{x_{13}x_{24}} = \frac{r^2}{L^2}. \tag{29}$$

Similarly we can express $|x_{14}| = L + r = L(1+\sqrt{x})$ and $|x_{23}| = L - r = L(1-\sqrt{x})$. This would allow us to express everything in terms of x and L.

For the datasets, we fixed $L = 14, \alpha = 0.5$, and $\epsilon^2 = 0.1$. We generated 10,000 sets of data for train-validation-test split from $x = 0.05$ to 0.1, with an increment of $\Delta x = 5 \times 10^{-6}$ between each step up to $k = 50$ in $G(w;\rho_A)$. To further validate our model, we generated 10,000 additional test datasets for the following physical parameters: $x = 0.1$ to 0.15 with $\Delta x = 5 \times 10^{-6}$. A density plot of the data with respect to the von Neumann entropy is shown in Fig. 9. We refer to Figs. 10 and 11 for a clear demonstration of the learning outcomes.

The study up to second order in x using the generating function method is available in [8], as well as through the use of holographic methods [61]. Additionally, an analytic continuation toward the von Neumann entropy up to second order in x for general CFT_2 can be found in [62]. Although this is a subleading correction, it can also be approached using our method.

Two intervals in the decompactification limit

There is a different limit that can be taken other than the small cross-ratio, where an approximate analytic Rényi entropies can be obtained. This is called the decompactification limit where we take $\eta \to \infty$, then for each fixed value of x we have $\mathcal{F}(x,\eta)$ as

$$\mathcal{F}_n(x,\eta) = \left[\frac{\eta^{n-1}}{\prod_{k=1}^{n-1} F_{k/n}(x) F_{k/n}(1-x)}\right]^{\frac{1}{2}}, \tag{30}$$

where ${}_2F_1$ is the hypergeometric function. Equation (30) is invariant under $\eta \leftrightarrow 1/\eta$, so we will instead use the result with $\eta \ll 1$

$$\mathcal{F}_n(x,\eta) = \left[\frac{\eta^{-(n-1)}}{\prod_{k=1}^{n-1} F_{k/n}(x) F_{k/n}(1-x)}\right]^{\frac{1}{2}}. \tag{31}$$

In this case, the exact analytic continuation of the von Neumann entropy is not known, but there is an approximate result following the expansion with $\eta \ll 1$

$$S(\rho_{AB}) \simeq S^W(\rho_{AB}) + \frac{1}{2}\ln\eta - \frac{D_1'(x) + D_1'(1-x)}{2} + \cdots, \tag{32}$$

with $S^W(\rho_{AB})$ being the von Neumann entropy computed from the Rényi entropies without the special function $\mathcal{F}_n(x,\eta)$ in (21). Note that

$$D_1'(x) = -\int_{-i\infty}^{i\infty} \frac{dz}{i} \frac{\pi z}{\sin^2(\pi z)} \ln F_z(x). \tag{33}$$

This approximate von Neumann entropy has been well tested in previous studies [5,8], and we will adopt it as the target values in our deep learning models.

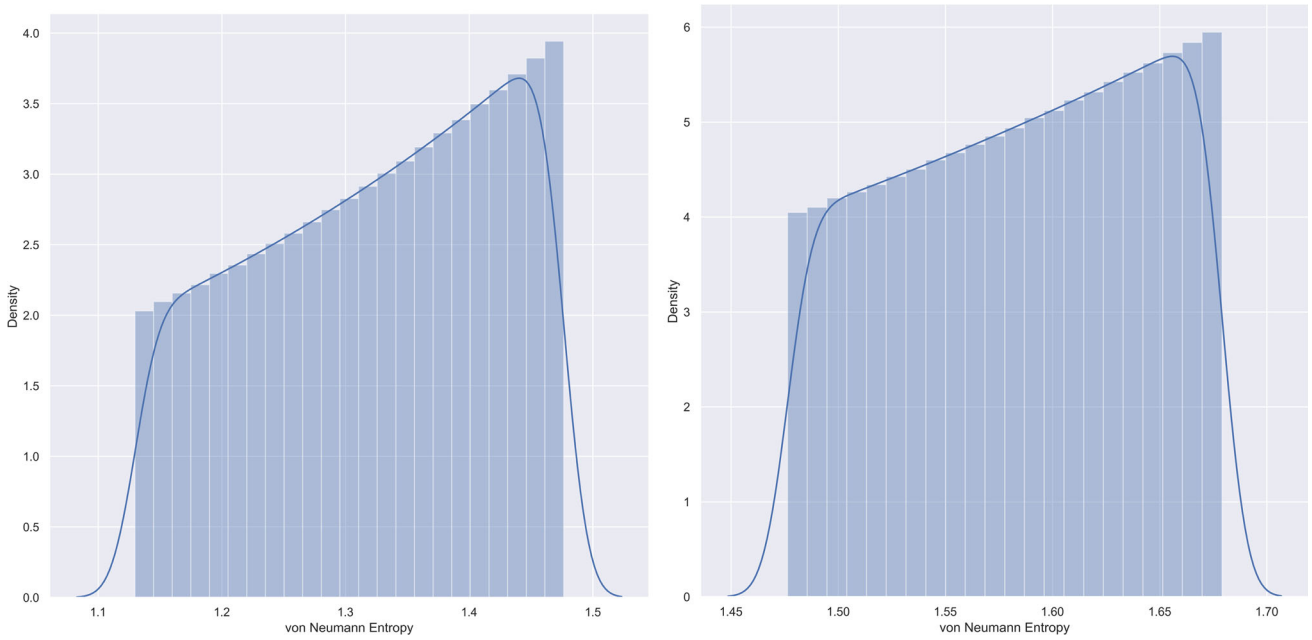

Fig. 9 The distribution of the two test datasets for the case of two intervals at small cross-ratio, where we plot density as a function of the von Neumann entropy computed by (28) with varying x. The blue curves represent the kernel density estimate for a smoothed estimate of the data distribution

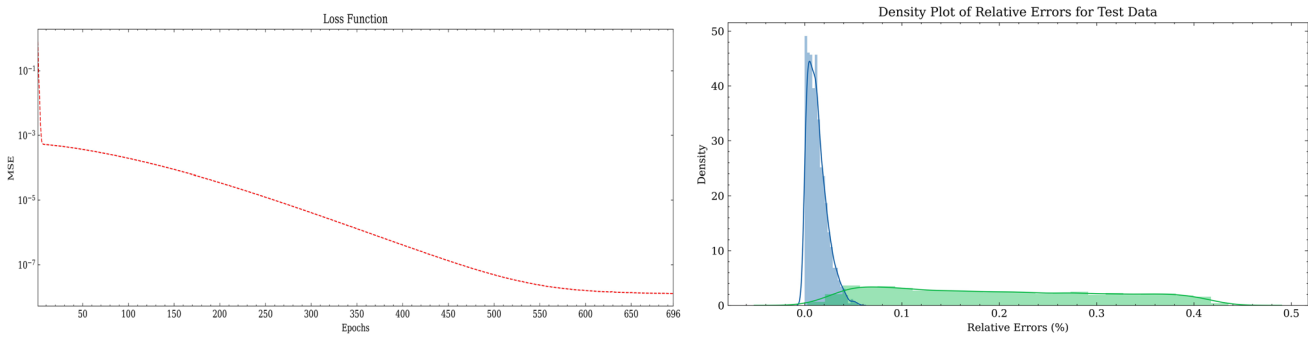

Fig. 10 Left: The MSE loss function as a function of epochs. The minimum loss close to 10^{-8} is achieved at epoch 696 for this instance. Right: The relative errors between the model predictions and targets for the two test datasets, where we have achieved high accuracy with relative errors $\lesssim 0.03\%$

For the datasets, we fixed $L = 14$, $x = 0.5$ and $\epsilon^2 = 0.1$. We generated 10,000 sets of data for train-validation-test split from $\eta = 0.1$ to 0.2, with an increment of $\Delta\eta = 10^{-5}$ between each step up to $k = 50$. To further validate our model, we generated 10,000 additional test datasets for the following physical parameters: $\eta = 0.2$ to 0.3 with $\Delta\eta = 10^{-5}$. A density plot of the data with respect to the von Neumann entropy is shown in Fig. 12. We again refer to Figs. 13 and 14 for a clear demonstration of the learning outcomes.

We have seen that deep neural networks, when treated as supervised learning, can achieve accurate predictions for the von Neumann entropy that extends outside the parameter regime in the training phase. However, the potential for deep neural networks may go beyond this.

As we know, the analytic continuation must be worked out on a case-by-case basis (see the examples in [4–7]) and may even depend on the method we use [8]. Finding general patterns in the analytic continuation is still an open question. Although it remains ambitious, the non-linear mapping that the neural networks uncover would allow us to investigate the expressive power of deep neural networks for the analytic continuation problem of the von Neumann entropy.

Our approach also opens up the possibility of using deep neural networks to study cases where analytic continuations are unknown, such as the general two-interval case. Furthermore, it may enable us to investigate other entanglement measures that follow similar patterns or require analytic continuations. We leave these questions as future tasks.

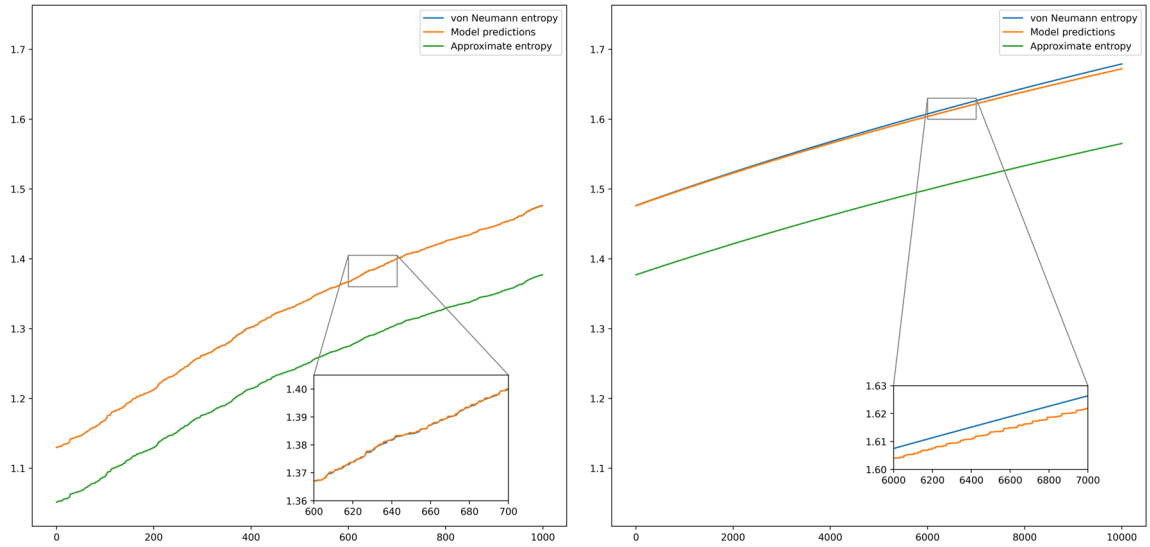

Fig. 11 We plot the predictions from the model with the analytic von Neumann entropy computed by (28) for the two test datasets. We also include the approximate entropy by summing over $k = 50$ terms in the generating function. Note that in order to achieve the same accuracy from the generating function, it requires at least $k \approx 800$ terms in this case

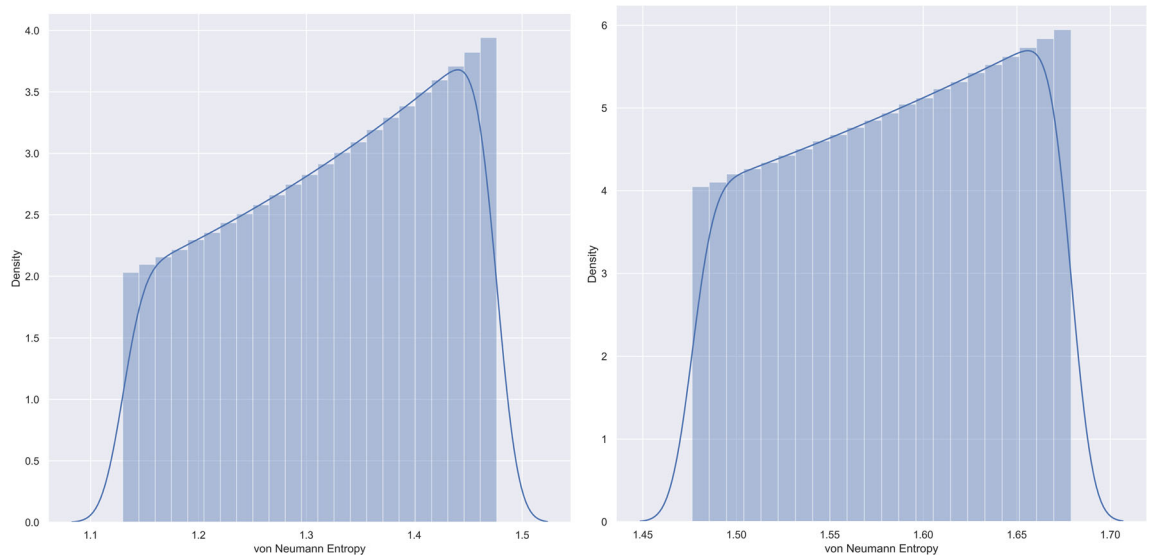

Fig. 12 The distribution of the two test datasets for the case of two intervals in the decompactification limit, where we plot density as a function of the von Neumann entropy computed by (32) with varying η. The blue curves represent the kernel density estimate for a smoothed estimate of the data distribution

4 Rényi entropies as sequential deep learning

In this section, we focus on higher Rényi entropies using sequential learning models. Studying higher Rényi entropies that depend on $\mathrm{Tr}\,\rho_A^n$ is equivalent to studying the higher-order terms in the Taylor series representation of the generating function (12). There are a few major motivations. Firstly, although the generating function can be used to compute higher-order terms, it becomes inefficient for more complex examples. Additionally, evaluating $\mathrm{Tr}\,\rho_A^n$ in (21) for the general two-interval case involves the Riemann–Siegel theta function, which poses a challenge in computing higher Rényi entropies [8,59,60]. On the other hand, all higher Rényi entropies should be considered independent and cannot be obtained in a linear fashion. They can all be used to predict the von Neumann entropy, but in the Taylor series expansion (12), knowing higher Rényi entropies is equivalent to knowing a more accurate von Neumann entropy. As we cannot simply extrapolate the series, using a sequential learning

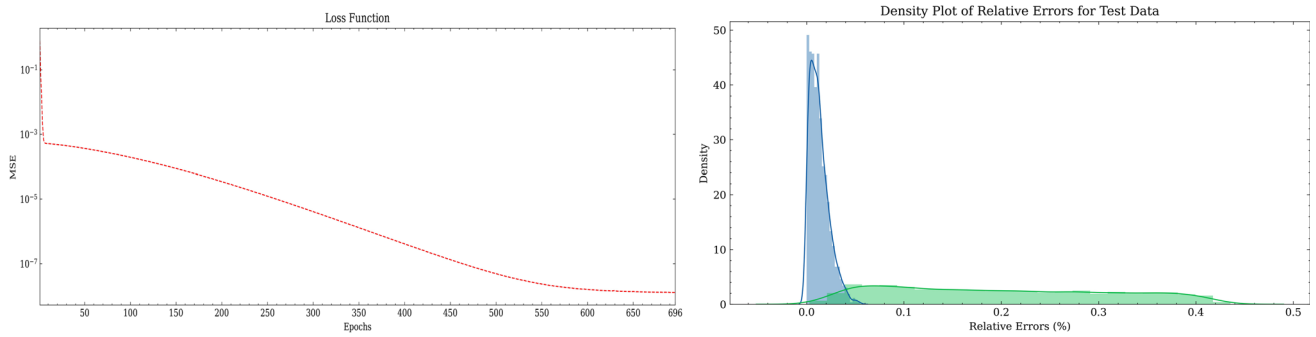

Fig. 13 Left: The MSE loss function as a function of epochs. The minimum loss at around 10^{-7} is achieved at epoch 132 for this instance. Right: The relative errors between the model predictions and targets for the two test datasets, where we have achieved high accuracy with relative errors $\lesssim 0.4\%$

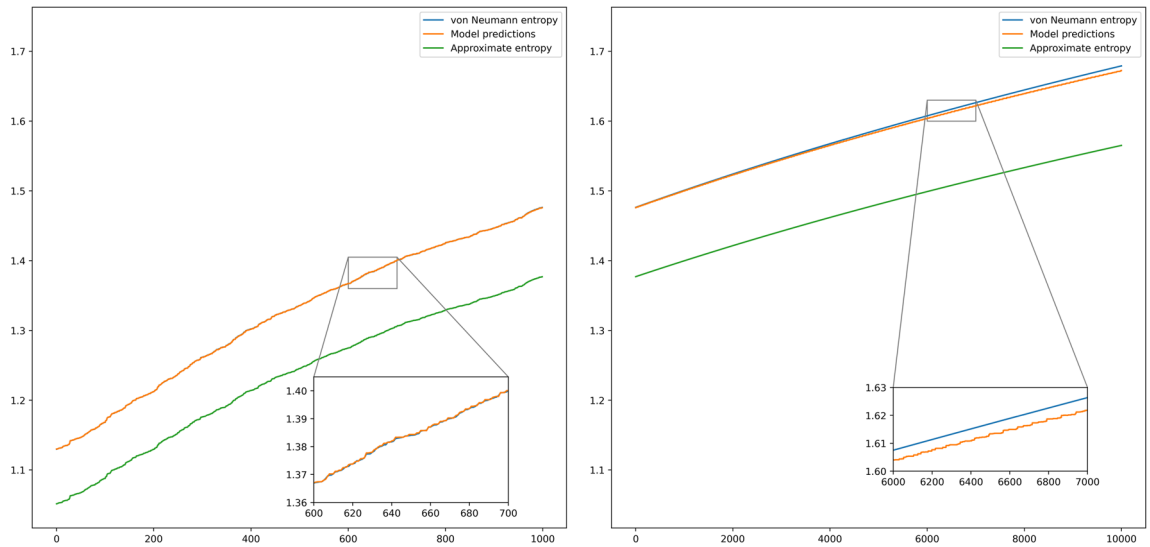

Fig. 14 We plot the predictions from the model with the analytic von Neumann entropy computed by (32) for the two test datasets. We also include the approximate entropy by summing over $k = 50$ terms in the generating function. Note that in order to achieve the same accuracy from the generating function, it requires at least $k \approx 400$ terms in this case

approach is a statistically robust way to identify underlying patterns.

Recurrent neural networks (RNNs) are a powerful type of neural network for processing sequences due to their "memory" property [63]. RNNs use internal loops to iterate through sequence elements while keeping a state that contains information about what has been observed so far. This property allows RNNs to identify patterns in a sequence regardless of their position in the sequence. To train an RNN, we initialize an arbitrary state and encode a rank-2 tensor of size (steps, input features), looping over multiple steps. At each step, the networks consider the current state at k with the input, and combine them to obtain the output at $k + 1$, which becomes the state for the next iteration.

RNNs incorporate both feedforward networks and *back-propagation through time* (BPTT) [64,65], with "time" representing the steps k in our case. The networks connect the outputs from a fully connected layer to the inputs of the same layer, referred to as the hidden states. These inputs receive the output values from the previous step, with the number of inputs to a neuron determined by both the number of inputs to the layer and the number of neurons in the layer itself, known as *recurrent connections*. Computing the output involves iteratively feeding the input vector from one step, computing the hidden states, and presenting the input vector for the next step to compute the new hidden states.

RNNs are useful for making predictions based on sequential data, or "sequential regression," as they learn patterns from past steps to predict the most probable values for the next step.

4.1 Model architectures and training strategies

In this subsection, we discuss the methodology of treating the Rényi entropies (the Taylor series of the generating function) as sequence models.

Data preparation

To simulate the scenario where $k_{\max}$ in the series cannot be efficiently computed, we generate $N = 10{,}000$ datasets for different physical parameters, with each dataset having a maximum of $k_{\max} = 50$ steps in the series. We also shuffle the N datasets since samples of close physical parameters will have most of their values in common. Among the N datasets, we only take a fraction $p < N$ for the train-validation-test split. The other fraction $q = N - p$ will all be used as test data for the trained model. This serves as a critical examination of the sequence models we find. The ideal scenario is that we only need small p datasets while achieving accurate performance for the q datasets.

Due to the rather small number of steps available, we are entitled to adopt the SimpleRNN structure in TensorFlow-Keas[2] instead of the more complicated ones such as LSTM or GRU networks [67,68].

We also need to be careful about the train-validation-test splitting process. In this type of problem, it is important to use validation and test data that is more recent than the training data. This is because the objective is to predict the next value given the past steps, and the data splitting should reflect this fact. Furthermore, by giving more weight to recent data, it is possible to mitigate the vanishing gradient (memory loss) problem that can occur early in the BPTT. In this work, the first 60% of the steps ($k = 1{-}30$) are used for training, the middle 20% ($k = 31{-}40$) for validation, and the last 20% ($k = 41{-}50$) for testing.

We split the datasets in the following way: for a single dataset from each step, we use a fixed number of past steps,[3] specified by ℓ, to predict the next value. This will create (steps $- \ell$) sequences from each dataset, resulting in a total of (steps $- \ell$) $\times$ p sequences for the p datasets in the train-validation-test splitting. Using a fixed sequence length ℓ allows the network to focus on the most relevant and recent information for predicting the next value, while also simplifying the input size and making it more compatible with our network architectures. We take $p = 1000$, $q = 9000$, and $\ell = 5$. An illustration of our data preparation strategy is shown in Fig. 15.

Model design

After the pre-processing of data, we turn to the model design. Throughout the section, we use the ReLU activation function and Adam optimizer with MSE as the loss function.

In KerasTuner, we employ Bayesian optimization by adjusting a few crucial hyperparameters and designs. We summarize them in the following list:

- We introduce one or two SimpleRNN layers, with or without recurrent dropouts. The units of the first layer range from 64 to 256 with a step size of 16. If a second layer is used, the units range from 32 to 128 with a step size of 8. Recurrent dropout is applied with a dropout rate in the range of 0.1 to 0.3 using log sampling.
- We take LayerNormalization as a Boolean choice to enhance the training stability, even with shallow networks. The LayerNormalization is added after the SimpleRNN layer if there is only one layer; in between the two layers if there are two SimpleRNN layers.
- We allow a Dense layer with units ranging from 16 to 32 and a step size of 8 as an optional regressor after the recurrent layers.
- A final dropout with log sampling of a dropout rate in the range of 0.2 to 0.5 is added as a Boolean choice.
- In the Adam optimizer, we only adjust the learning rate with log sampling from the range of 10^{-5} to 10^{-4}. All other parameters are taken as the default values in TensorFlow-Keras. We take the AMSGrad [54] variant of this algorithm as a Boolean choice.

The KerasTuner is deployed for 300 trials with 2 executions per trial. During the process, we monitor the validation loss using EarlyStopping of patience 8. Once the best set of hyperparameters and model architecture are identified based on the validation data, we initialize a new model with the same design, but with both the training and validation data. This new model is trained 30 times while monitoring the training loss using EarlyStopping of patience 10. The final predictions are obtained by averaging the results of the few cases with close yet overall smallest relative errors from the targets. The purpose of taking the average instead of picking the case with minimum loss is to smooth out possible outliers. We set the batch size in both the KerasTuner and the final training to be 2048.

We will also use the trained model to make predictions on the q test data and compare them with the correct values as validation for hitting the benchmark.

[2] SimpleRNN suffers from the vanishing gradient problem when learning long dependencies [66]. Even using ReLU, which does not cause a vanishing gradient, back-propagation through time with weight sharing can still lead to a vanishing gradient across different steps. However, since the length of the sequence is small due to the limited maximum steps available in our case, we have found that SimpleRNN generally performs better than its variants.

[3] We could also include as many past steps as possible, but we have found it less effective. This can be attributed to our choice of network architectures and the fact that we have rather short maximum steps available.

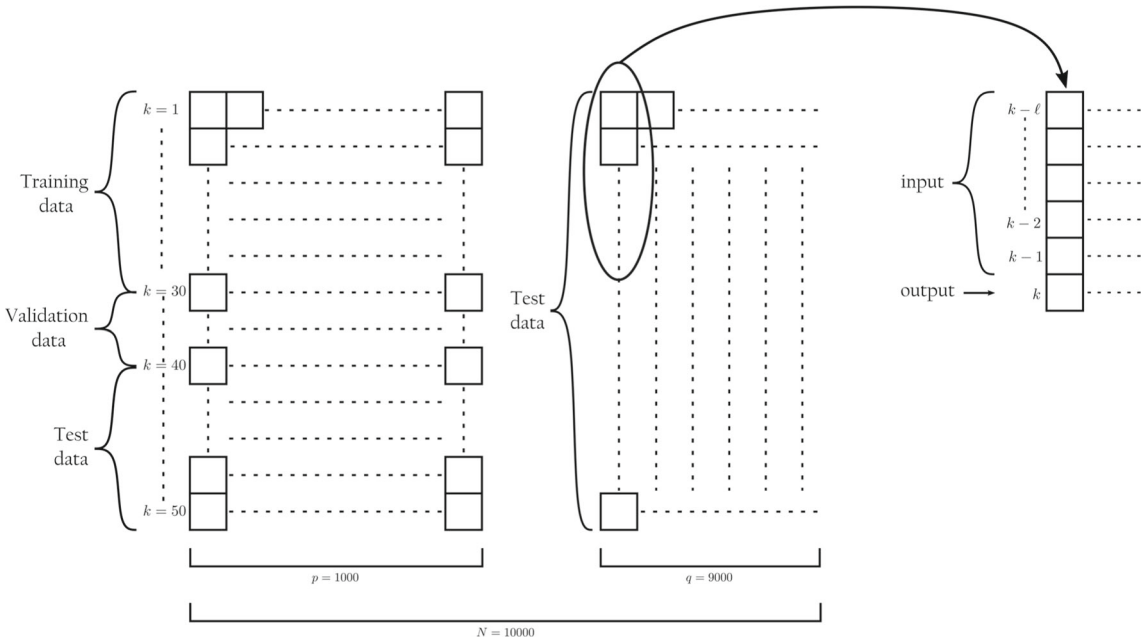

Fig. 15 Data preparation process for the sequential models. A total of N datasets are separated into two parts: the p datasets are for the initial train-validation-test split, while the q datasets are treated purely as test datasets. The zoomed-in figure on the right hand side illustrates how a single example sequence is generated, where we have used a fixed number of past steps $\ell = 5$. Note that for the additional q test datasets, a total of $(\text{steps} - \ell) \times q = 405000$ sequences are generated

4.2 Examples of the sequential models

The proposed approach will be demonstrated using two examples. The first example is a simple representative case of a single interval (13); while the second is a more challenging case of the two-interval at decompactification limit (32), where the higher-order terms in the generating function cannot be efficiently computed. Additionally, we will briefly comment on the most non-trivial example of the general two-interval case.

Single interval

In this example, we have used the same N datasets for the single interval as in Sect. 3.2. Following the data splitting strategy we just outlined, it is worth noting that the ratio of training data to the overall dataset is relatively small. We have plotted the losses of the three best-performing models, as well as the density plot of relative errors for the two test datasets in Fig. 16. Surprisingly, even with a small ratio of training data, we were able to achieve small relative errors on the additional test datasets.

Two intervals in the decompactification limit

Again, we have used the same N datasets for the two intervals in the $\eta \to \infty$ limit as in Sect. 3.3. In Fig. 17, we have plotted the losses of the four best-performing models and the density plot of relative errors for the two test datasets. In this example, the KerasTuner identified a relatively small learning rate, which led us to truncate the training at a maximum of 1500 epochs since we had achieved the required accuracy. In this case, the predictions are of high accuracy, essentially without outliers.

Let us briefly address the most challenging example discussed in this paper, which is the general two-interval case (21) where the analytic expression for the von Neumann entropy is not available. In this example, only $\text{Tr}\,\rho_A^n$ is known, and since it involves the Riemann–Siegel theta function, computing the generating function for large k in the partial sum becomes almost infeasible. Therefore, the sequential learning models we have introduced represent the most viable approach for extracting useful information in this case.

Since only $k_{\max} \approx 10$ can be efficiently computed from the generating function in this case, we have much shorter steps for the sequential learning models. We have tested the above procedure with $N = 10{,}000$ datasets and $k_{\max} = 10$, however, we could only achieve an average of 5% relative errors. This is not a generalizable outcome. Improvements may come from a larger dataset with a longer training time, or reducing the required datasets via certain cross-validation techniques, which we leave as a future task.

In general, sequential learning models offer a potential solution for efficiently computing higher-order terms in the generating function. To extend our approach to longer sequences beyond the $k_{\max}$ steps, we can treat the problem as self-supervised learning. However, this may require a more delicate model design to prevent error propagation. Nonethe-

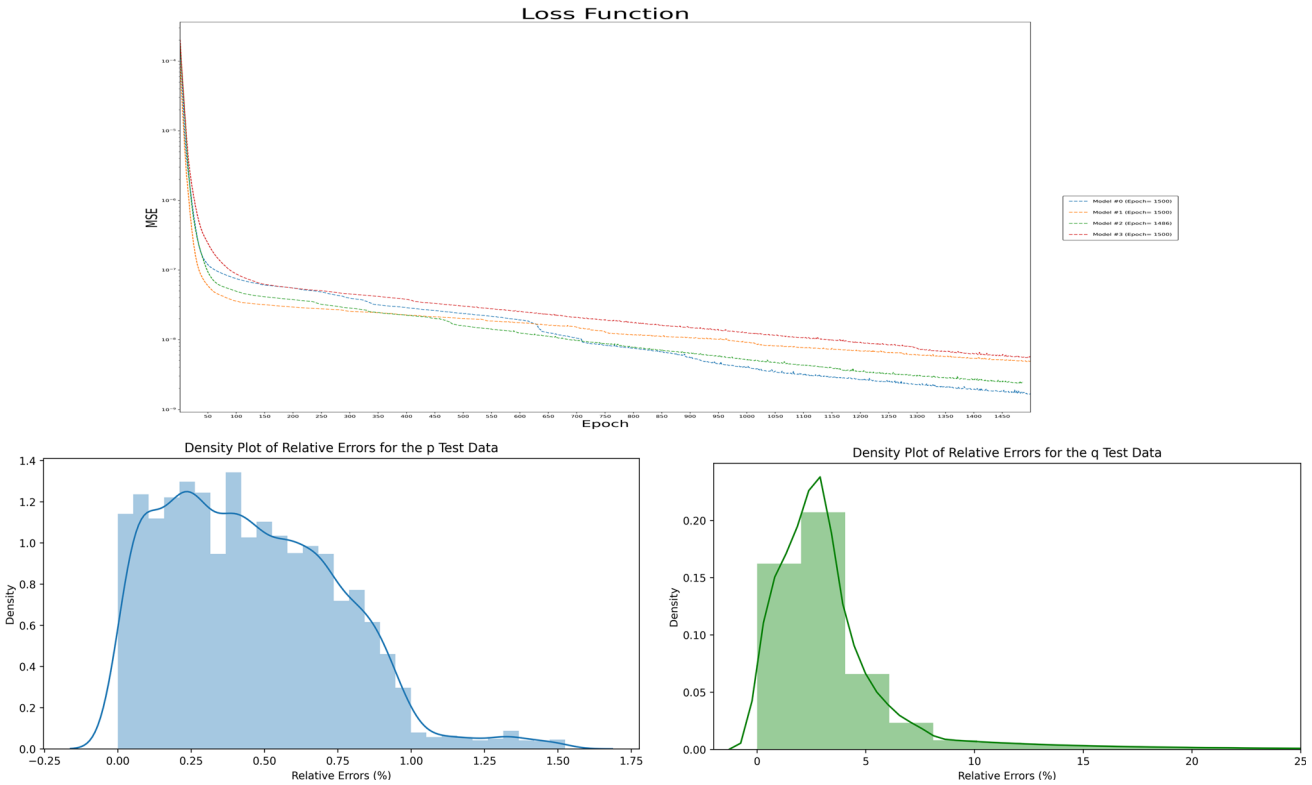

Fig. 16 Top: The loss function for the best 3 models as a function of epochs. We monitor the loss function with EarlyStopping, where the epochs of minimum losses at around 10^{-8} for different models are specified in the parentheses of the legend. Bottom: The density plots as a function of relative errors for the two test datasets. The relative errors for the p test datasets are concentrated at around 1%; while for the additional q test datasets, they are concentrated at around 2.5% with a very small ratio of outliers

less, exploring longer sequences can provide a more comprehensive understanding of the behavior of von Neumann entropy and its relation to Rényi entropies.

5 Quantum neural networks and von Neumann entropy

In this section, we explore a similar supervised learning task by treating the quantum circuits as models that map data inputs to predictions, which influences the expressive power of quantum circuits as function approximations. The purpose of the study is different from the previous cases with classical neural networks, where instead of a generalizable quantum model, we are exploring the expressivity of quantum circuits on the von Neumann entropy.

5.1 Fourier series from variational quantum machine learning models

We will focus on a specific function class that a quantum neural network can explicitly realize, namely a simple Fourier-type sum [29,30]. Before linking it to the von Neumann entropy, we shall first give an overview of the seminal works in [30].

Consider a general Fourier-type sum in the following form

$$f_{\theta_i}(\vec{x}) = \sum_{\vec{\omega}\in\Omega} c_{\vec{\omega}}(\theta_i) e^{i\vec{\omega}\cdot\vec{x}}, \tag{34}$$

with the frequency spectrum specified by $\Omega \subset \mathbb{R}^N$. Note that $c_{\vec{\omega}}(\theta_i)$ are the (complex) Fourier coefficients. We need to come up with a quantum model that can learn the characteristics of the sum by the model's control over the frequency spectrum and the Fourier coefficients, which are ultimately affected by some trainable parameters θ_i and the data input $\vec{x}$.

Now we define the quantum machine learning model as the following expectation value

$$f_{\theta_i}(x) = \langle 0|U^\dagger(x,\theta_i) M U(x,\theta_i)|0\rangle, \tag{35}$$

where $|0\rangle$ is taken to be some initial state of the quantum computer. We will see that the expressivity of the quantum circuit given by (35) manifests as a Fourier-type sum as in (34). The M will be the physical observable. Note that we have omitted writing the vector symbol and the hat on the operator, which should be clear from the context. The cru-

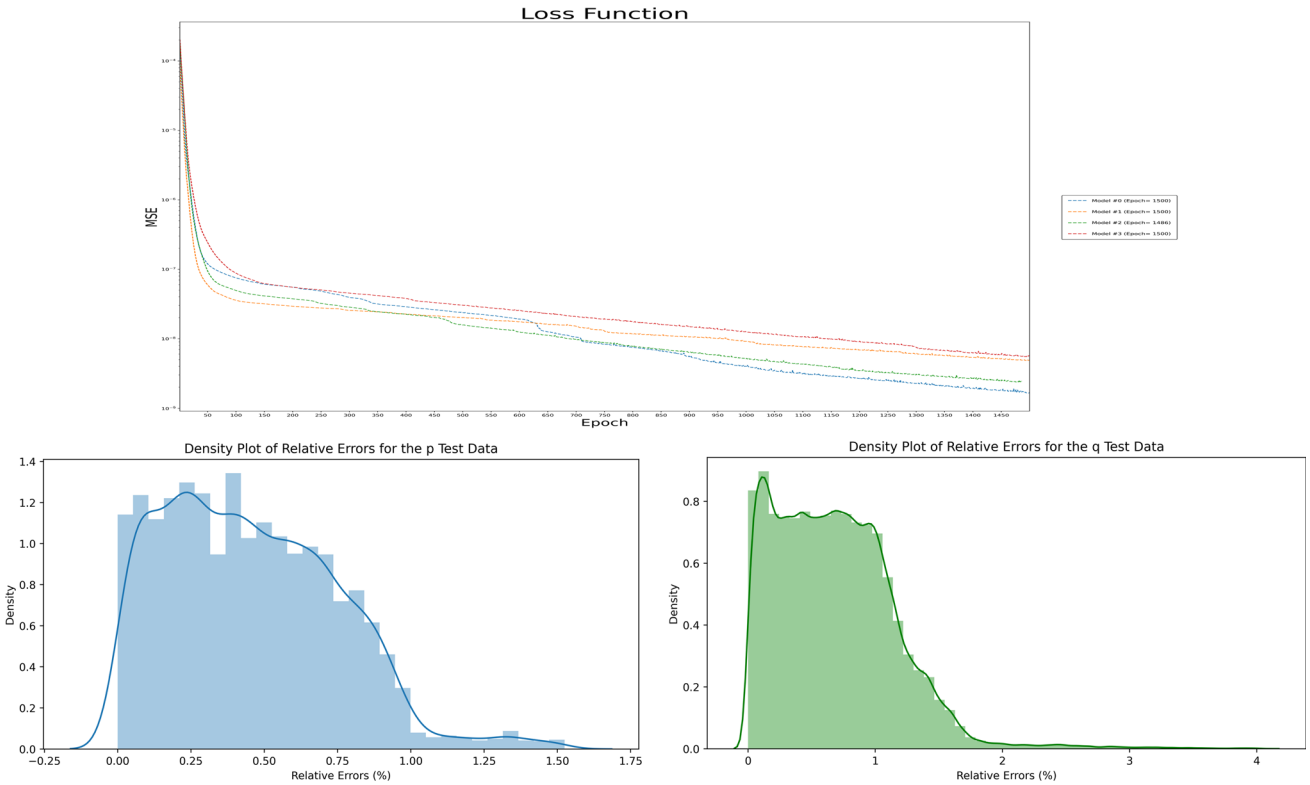

Fig. 17 Top: The loss function for the best 4 models as functions of epochs. We monitor the loss function with EarlyStopping. Bottom: The density plot as a function of relative errors for the two test datasets. The relative errors for the p test datasets are well within $\lesssim 1.5\%$; while for the additional q test datasets, they are well within $\lesssim 2\%$

cial component is $U(x, \theta_i)$, which is a quantum circuit that depends on the data input x and the trainable parameters θ_i with L layers. Each layer has a data-encoding circuit block $S(x)$, and the trainable circuit block $W(\theta_i)$. Schematically, it has the form

$$U(x, \theta_i) = W^{(L+1)}(\theta_i) S(x) W^{(L)}(\theta_i) \cdots W^{(2)}(\theta_i) S(x) W^{(1)}(\theta_i), \tag{36}$$

where we refer to Fig. 18 for a clear illustration.

Let us discuss the three major components of the quantum circuit in the following:

- The repeated data-encoding circuit block $S(x)$ prepares an initial state that encodes the (one-dimensional) input data x and is not trainable due to the absence of free parameters. It is represented by certain gates that embed classical data into quantum states, with gates of the form $g(x) = e^{-ixH}$, where H is the encoding Hamiltonian that can be any unitary operator. In this work, we use the Pauli X-rotation gate, and the encoding Hamiltonians in $S(x)$ will determine the available frequency spectrum Ω.
- The trainable circuit block $W(\theta_i)$ is parametrized by a set of free parameters $\theta_i = (\theta_1, \theta_2, \ldots)$. There is no special assumption made here and we can take these trainable blocks as arbitrary unitary operations. The trainable parameters will contribute to the coefficients c_ω.
- The final piece is the measurement of a physical observable M at the output. This observable is general, it could be local for each wire or subset of wires in the circuit.

Our goal is to establish that $f(x)$ can be written as a partial Fourier series [29,30]

$$f_{\theta_i}(x) = \langle 0 | U^\dagger(x, \theta_i) M U(x, \theta_i) | 0 \rangle = \sum_{n \in \Omega} c_n e^{inx}. \tag{37}$$

Note that here for simplicity, we have taken frequencies being integers $\Omega \subset \mathbb{Z}^N$. The training process goes as follows: we sample a quantum model with $U(x, \theta_i)$, and then define the mean square error as the loss function. To optimize the loss function, we need to tune the free parameters $\theta = (\theta_1, \theta_2, \ldots)$. The optimization is performed by a classical optimization algorithm that queries the quantum device, where we can treat the quantum process as a black box and only examine the classical data input and the measurement output. The output of the quantum model is the expectation value of a Pauli-Z measurement.

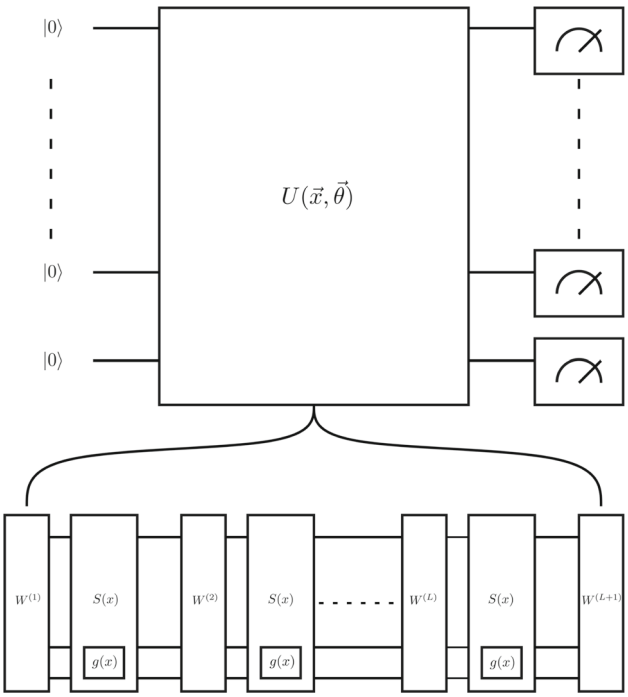

Fig. 18 Quantum neural networks with repeated data-encoding circuit blocks $S(x)$ (whose gates are of the form $g(x) = e^{-ixH}$) and trainable circuit blocks $W^{(i)}$. The data-encoding circuit blocks determine the available frequency spectrum for $\vec{\omega}$, while the remainder determines the Fourier coefficients $c_{\vec{\omega}}$

We use the single-qubit Pauli rotation gate as the encoding $g(x)$ [30]. The frequency spectrum Ω is determined by the encoding Hamiltonians. Two scenarios can be considered to determine the available frequencies: the *data reuploading* [69] and the *parallel encodings* [70] models. In the former, we repeat r times of a Pauli rotation gate in sequence, which means we act on the same qubit, but with multiple layers $r = L$; whereas in the latter, we perform similar operations in parallel on r different qubits. but with a single layer $L = 1$. These models allow quantum circuits to access increasingly rich frequencies, where $\Omega = \{-r, \ldots, -1, 0, 1, \ldots, r\}$ with a spectrum of integer-valued frequencies up to degree r. This will correspond to the maximum degree of the partial Fourier series we want to compute.

From the discussion above, one can immediately derive the maximum accessible frequencies of such quantum models [30]. But in practice, if the degree of the target function is greater than the number of layers (for example, in the single qubit case), the fit will be much less accurate.[4] Increasing the value of L typically requires more training epochs to converge at the same learning rate. For our demonstrations later, we will focus on target Fourier series up to degree 4, with both data reuploading ($L = 4$ layers) and parallel encodings models ($r = 4$ qubits).

This is relevant to a more difficult question of how to control the Fourier coefficients in the training process, given that all the blocks $W^{(i)}(\theta_i)$ and the measurement observable contribute to "every" Fourier coefficient. However, these coefficients are functions of the quantum circuit with limited degrees of freedom. This means that a quantum circuit with a certain structure can only realize a subset of all possible Fourier coefficients, even with enough degrees of freedom. While a systemic understanding is not yet available, a simulation exploring which Fourier coefficients can be realized can be found in [30]. In fact, it remains an open question whether, for asymptotically large L, a single qubit model can approximate any function by constructing arbitrary Fourier coefficients.

5.2 The generating function as a Fourier series

Given the framework of the quantum model and its relation to a partial Fourier series, a natural question arises as to whether the entanglement entropy can be realized within this setup. To approach this question, it is meaningful to revisit the generating function for the von Neumann entropy

$$G(z;\rho_A) \equiv -\operatorname{Tr}\left(\rho_A \ln\frac{1-z\rho_A}{1-z}\right) = \sum_{k=1}^{\infty}\frac{f(k)}{k}z^k, \tag{38}$$

as a manifest Taylor series. The goal is to rewrite the generating function in terms of a partial Fourier series. Therefore, we would be able to determine whether the von Neumann and Rényi entropies are the function classes that the quantum neural network can describe. Note that we will only focus on small-scale tests with a low depth or width of the circuit, as the depth or width of the circuit will correspond exactly to the orders that can be approximated in the Fourier series.

But we cannot simply convert either the original generating function or its Taylor series form to a Fourier series. By doing so, it will generally involve special functions in ρ_A, for which we will be unable to specify in terms of $\operatorname{Tr}\rho_A^n$. Therefore, it is essential to have an expression of the Fourier series that allows us to compute the corresponding Fourier coefficients at different orders using $\operatorname{Tr}\rho_A^n$, for which we know the analytic form from CFTs.

This can indeed be achieved, see Appendix A for a detailed derivation. The Fourier series representation of the generating function on an interval $[w_1, w_2]$ with period $T = w_2 - w_1$ is given by

$$G(w;\rho) = \frac{a_0}{2} + \sum_{n=1}^{\infty}\left\{\sum_{m=0}^{\infty}\frac{\tilde{f}(m)}{m}C_{cos}(n,m)\cos\left(\frac{2\pi n w}{T}\right)\right.$$

[4] Certain initial weight samplings may not even converge to a satisfactory solution. This is relevant to the barren plateau problem [71] generically present in variational quantum circuits with a random initialization, similar to the classical vanishing gradient problem.

$$+\sum_{m=0}^{\infty}\frac{\tilde{f}(m)}{m}C_{sin}(n,m)\sin\left(\frac{2\pi nw}{T}\right)\Bigg\}, \tag{39}$$

where C_{cos} and C_{sin} are some special functions defined as

$$\begin{aligned}&C_{cos}(n,m)\\&=\frac{2}{(m+1)T}\Bigg[{}_pF_q\left(\frac{m+1}{2};\frac{1}{2},\frac{m+3}{2};-\frac{n^2\pi^2t_2^2}{T^2}\right)t_2^{m+1}\\&\quad-{}_pF_q\left(\frac{m+1}{2};\frac{1}{2},\frac{m+3}{2};-\frac{n^2\pi^2t_2^2}{T^2}\right)t_1^{m+1}\Bigg],\end{aligned} \tag{40}$$

$$\begin{aligned}&C_{sin}(n,m)\\&=\frac{4n\pi}{(m+2)T^2}\Bigg[{}_pF_q\left(\frac{m+2}{2};\frac{3}{2},\frac{m+4}{2};-\frac{n^2\pi^2t_2^2}{T^2}\right)t_2^{m+2}\\&\quad-{}_pF_q\left(\frac{m+2}{2};\frac{3}{2},\frac{m+4}{2};-\frac{n^2\pi^2t_1^2}{T^2}\right)t_1^{m+2}\Bigg],\end{aligned} \tag{41}$$

with ${}_pF_q$ being the generalized hypergeometric function. Note also that

$$\tilde{f}(m)\equiv\sum_{k=0}^{m}\frac{(-1)^{2m-k+1}m!}{k!(m-k)!}\,\mathrm{Tr}\,(\rho_A^{k+1}). \tag{42}$$

Similarly, the zeroth order Fourier coefficient is given by

$$\begin{aligned}a_0&=\sum_{m=0}^{\infty}\frac{\tilde{f}(m)}{m}C_{cos}(0,m)\\&=\sum_{m=0}^{\infty}\frac{\tilde{f}(m)}{m}\frac{2(w_2^{m+1}-w_1^{m+1})}{(m+1)T}.\end{aligned} \tag{43}$$

Note that summing to $m=10$ suffices our purpose, while the summation in n corresponds to the degree of the Fourier series. Note that the complex-valued Fourier coefficients c_n to be used in our simulation can be easily reconstructed from the expression. Therefore, the only required input for evaluating the Fourier series is $\tilde{f}(m)$, with $\mathrm{Tr}\,\rho_A^{k+1}$ explicitly given. This is exactly what we anticipated and allows for a straightforward comparison with the Taylor series form.

Note the interval for the Fourier series is not arbitrary. We will take the interval $[w_1,w_2]$ to be $[-1,1]$, which is the maximum interval where the Fourier series (39) is convergent. Furthermore, we expect that as $w\to 1$ from (39), we arrive at the von Neumann entropy, that is

$$S(\rho_A)=\lim_{w\to 1}G(w;\rho_A). \tag{44}$$

However, as we can see in Fig. 19, there is a rapid oscillation near the end points of the interval for the Fourier series. The occurrence of such "jump discontinuity" is a generic feature for the approximation of discontinuous or non-periodic functions using Fourier series known as the *Gibbs phenomenon*. This phenomenon poses a serious problem in recovering accurate values of the von Neumann entropy because we are taking the limit to the boundary point $w\to 1$. We will return to this issue in Sect. 5.4.

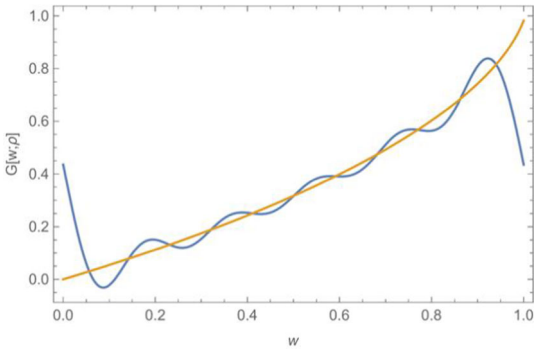

Fig. 19 Gibbs phenomenon for the Fourier series near the end point for $w\to 1$. We take the single interval example where the yellow curve represents the generating function as a Taylor series, and the blue curve is the Fourier series approximation of the generating function

5.3 The expressivity of the quantum models on the entanglement entropy

In this subsection, we will demonstrate the expressivity of the quantum models of the partial Fourier series with examples from CFTs. We will focus on two specific examples: a single interval and two intervals at small cross-ratio. While these examples suffice for our purpose, it is worth noting that once the Fourier series representation is derived using the expression in (39), all examples with a known analytic form of $\mathrm{Tr}\,\rho_A^n$ can be studied.

The demonstration is performed using Pennylane [72]. We have adopted the Adam optimizer with a learning rate 0.005 and batch size of 100, where MSE is the loss function. Note that we have chosen a smaller learning rate compared to [30] and monitor with EarlyStopping. For the two examples we study, we have considered both the serial (data reuploading) and parallel (parallel encodings) models for the training. Note that in the parallel model, we have used the StronglyEntanglingLayers in Pennylane with itself of 3 user-defined layers. In each case, we start by randomly initializing a quantum model with 300 sample points to fit the target function

$$f(x)=\sum_{n=-k}^{n=k}c_ne^{inx}. \tag{45}$$

where the complex-valued Fourier coefficients are calculated from the real coefficients in (39). We have chosen $k=4$ with prescribed physical parameters in the single- and two-interval examples. Therefore, we will need r in the serial and parallel models to be larger than $k=4$. We have executed multiple trials from each case, where we include the most successful results with maximum relative errors controlled in $\lesssim 3\%$ in Figs. 20, 21, 22, 23.

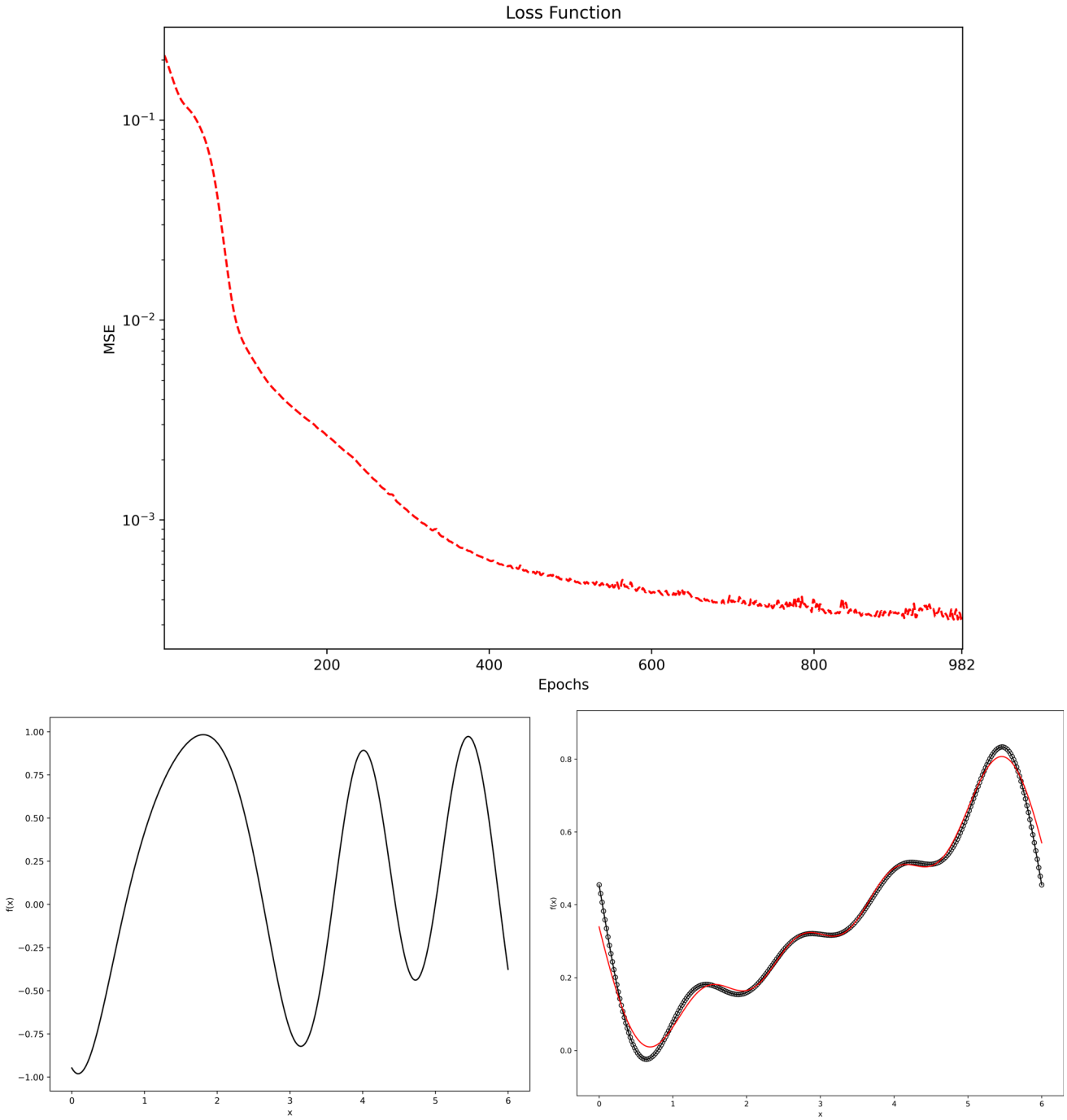

Fig. 20 A random serial quantum model trained with data samples to fit the target function of the single interval case. Top: the MSE loss function as a function of epochs, where the minimum loss is achieved at epoch 982. Bottom left: a random initialization of the serial quantum model with $r = 6$ sequential repetitions of Pauli encoding gates. Bottom right: the circles represent the 300 data samples of the single interval Fourier series with $\ell = 2$ and $\epsilon = 0.1$ for (14). The red curve represents the quantum model after training

As observed from Figs. 20, 21, 22, 23, a rescaling of the data is necessary to achieve precise matching between the quantum models and the Fourier spectrum of our examples. This rescaling is possible because the global phase is unobservable [30], which introduces an ambiguity in the data-encoding. Consider our quantum model

$$f_\theta(x) = \langle 0|U^\dagger(x,\theta)MU(x,\theta)|0\rangle = \sum_{\omega\in\Omega} c_\omega(\theta)e^{i\omega x}, \quad (46)$$

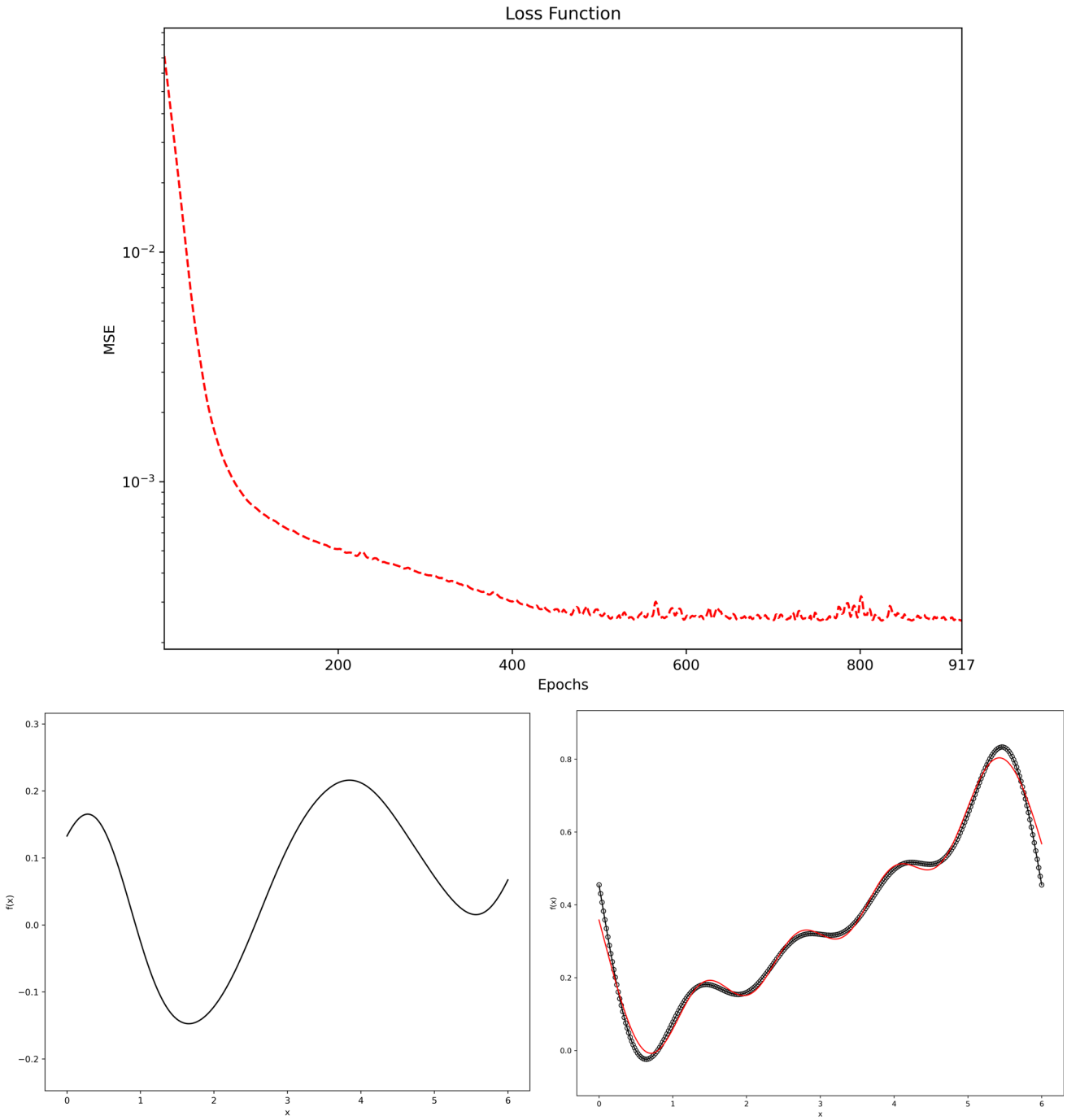

Fig. 21 A random parallel quantum model for the single interval case. Top: the loss function achieves minimum loss at epoch 917. Bottom: a random initialization of the quantum model with $r = 5$ parallel repetitions of Pauli encoding gates that has achieved a good fit

where we consider the case of a single qubit $L = 1$, then

$$U(x) = W^{(2)} g(x) W^{(1)}. \tag{47}$$

Note that the frequency spectrum Ω is determined by the eigenvalues of the data-encoding Hamiltonians, which is given by the operator

$$g(x) = e^{-ixH}. \tag{48}$$

H has two eigenvalues (λ_1, λ_2), but we can rescale the energy spectrum to $(-\gamma, \gamma)$ as the global phase is unobservable (e.g. for Pauli rotations, we have $\gamma = \frac{1}{2}$). We can absorb γ from the eigenvalues of H into the data input by re-scaling with

$$\tilde{x} = \gamma x. \tag{49}$$

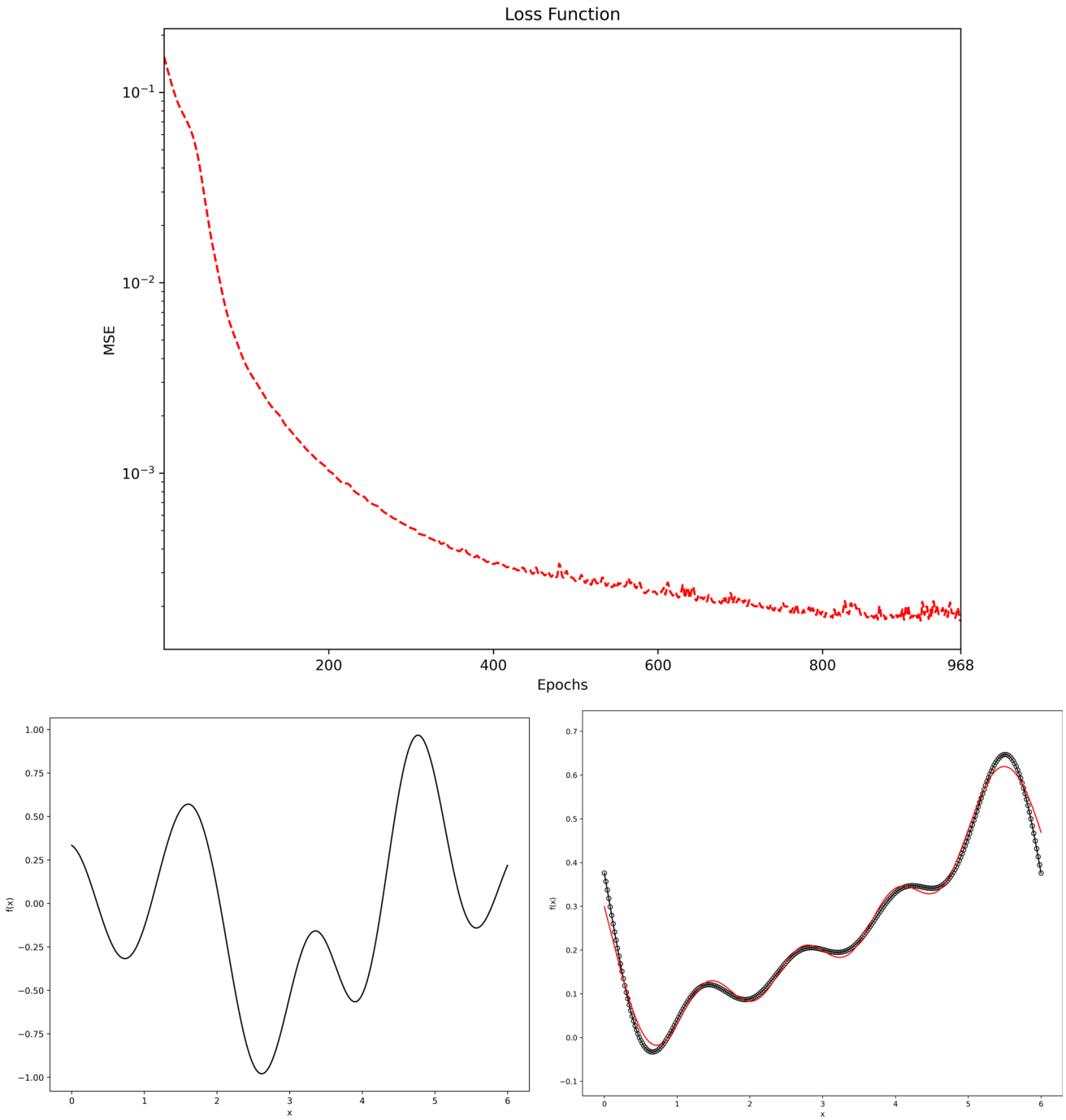

Fig. 22 A random serial quantum model trained with data samples to fit the target function of the two-interval system with a small cross-ratio. Top: the loss function achieves minimum loss at epoch 968. Bottom left: a random initialization of the serial quantum model of $r = 6$ sequential repetitions of Pauli encoding gates. Bottom right: the circles represent the 300 data samples of the two-interval Fourier series with $x = 0.05$, $\alpha = 0.1$, and $\epsilon = 0.1$ for (28). The red curve represents the quantum model after training

Therefore, we can assume the eigenvalues of H to be some other values. Specifically, we have chosen $\gamma = 6$ in the training, where the interval in x is stretched from $[0, 1]$ to $[0, 6]$, as can be seen in Figs. 20, 21, 22, 23.

We should emphasize that we are not re-scaling the original target data, but instead, we are re-scaling how the data is encoded. Effectively, we are re-scaling the frequency of the quantum model itself. The intriguing part is that the global phase shift of the operator acting on a quantum state cannot

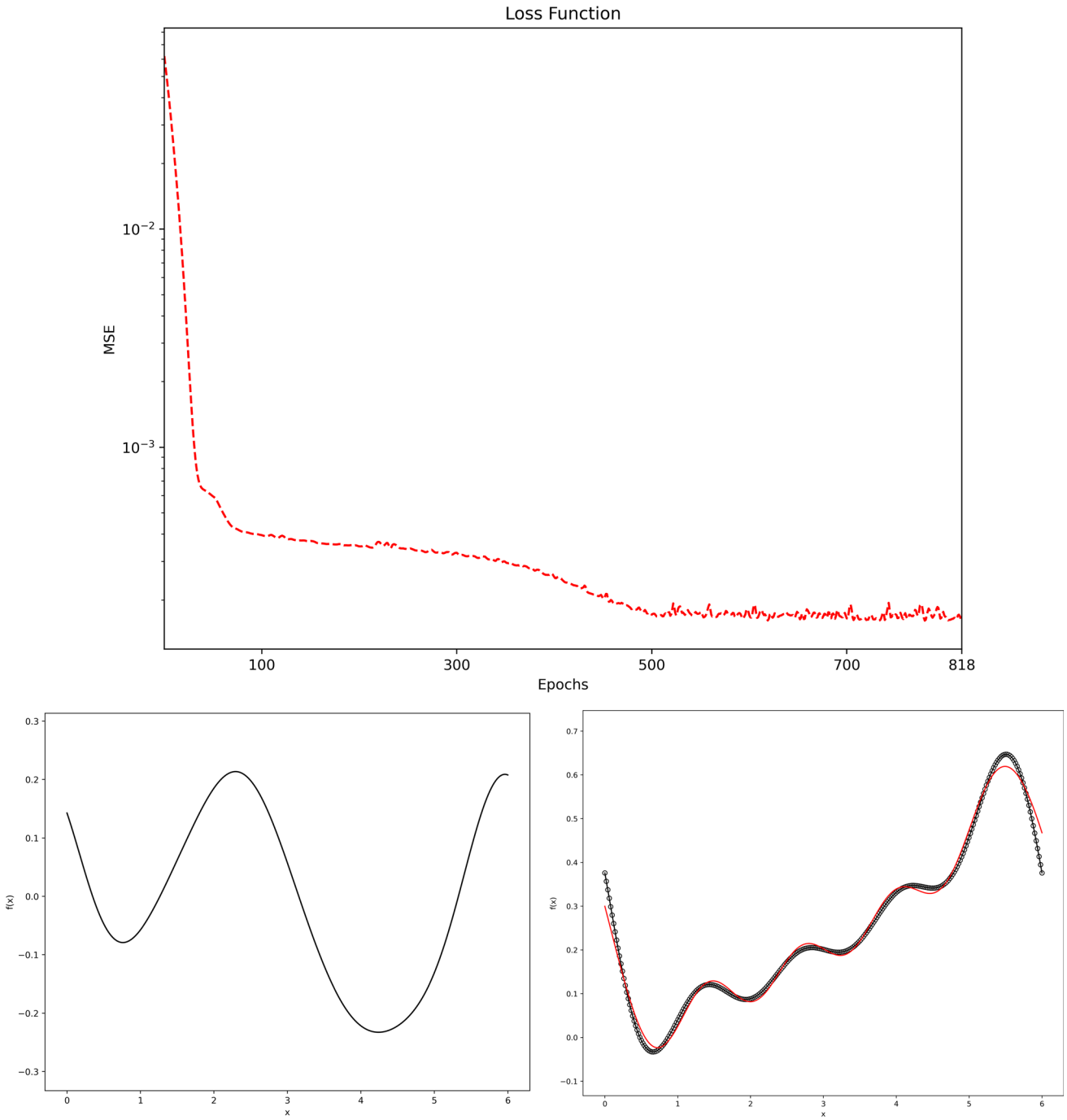

Fig. 23 A random parallel quantum model for the two-interval case. Top: the loss function achieves minimum loss at epoch 818. Bottom: a random initialization of the quantum model with $r = 5$ parallel repetitions of Pauli encoding gates that has achieved a good fit

be observed, yet it affects the expressive power of the quantum model. This can be understood as a pre-processing of the data, which is argued to extend the function classes of the quantum model that can represent [30].

This suggests that one may consider treating the re-scaling parameter γ as a trainable parameter [69]. This would turn the scaling into an adaptive "frequency matching" process, potentially increasing the expressivity of the quantum model. Here we only treat γ as a tunable hyperparameter. The scaling does not need to match with the data, but finding an appropriate scaling parameter is crucial for model training.

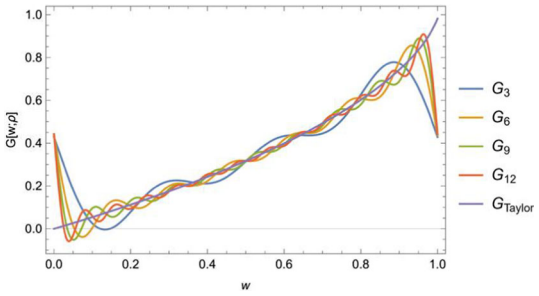

Fig. 24 We have plotted the single interval example with $L = 2$ and $\epsilon = 0.1$ for (14). Here the legends G_N refer to the Fourier series of the generating function to degree N, by summing up to $m = 10$ in (39). G_{Taylor} refers to the Taylor series form (12) of the generating function by summing up to $k = 100$

5.4 Recovering the von Neumann entropy

So far, we have managed to rewrite the generating function into a partial Fourier series $f_N(w)$ of degree N, defined on the interval $w \in [-1, 1]$. By leveraging variational quantum circuits, we have been able to reproduce the Fourier coefficients of the series accurately. In principle, with appropriate data-encoding and re-scaling strategies, increasing the depth or width of the quantum models would enable us to capture the series to any arbitrary degree N. Thus, the expressivity of the Rényi entropies can be established in terms of quantum models. However, a crucial problem remains, that is, we need to recover the von Neumann entropy under the limit $w \to 1$

$$\lim_{w\to 1} G(w;\rho_A) = S(\rho_A), \tag{50}$$

where the limiting point is exactly at the boundary of the interval that we are approximating. However, as we can see clearly from Fig. 24, taking such a limit naïvely gives a very inaccurate value compared to the true von Neumann entropy. This effect does not diminish even by increasing N to achieve a better approximation of the series when compared to its Taylor series form, as shown in Fig. 24. This is because the Fourier series approximation is always oscillatory at the endpoints, a general feature known as the *Gibbs phenomenon* for the Fourier series when approximating discontinuous or non-periodic functions.

A priori, a partial Fourier series of a function $f(x)$ is a very accurate way to reconstruct the point values of $f(x)$, as long as $f(x)$ is smooth and periodic. Furthermore, if $f(x)$ is analytic and periodic, then the partial Fourier series f_N would converge to $f(x)$ exponentially fast with increasing N. However, $f_N(x)$ in general is not an accurate approximation of $f(x)$ if $f(x)$ is either discontinuous or non-periodic. Not only the convergence is slow, there is an overshoot near the boundary of the interval. There are many different ways to understand this phenomenon. Broadly speaking, the difficulty lies in the fact that we are trying to obtain accurate local information from the global properties of the Fourier coefficients defined via an integral over the interval, which seems to be inherently impossible.

Mathematically, the occurrence of the Gibbs phenomenon can be easily understood in terms of the oscillatory nature of the Dirichlet kernel, which arises when the Fourier series is written as a convolution. Explicitly, the Fourier partial sum can be written as

$$s_n(x) = \frac{1}{\pi}\int_{-\pi}^{\pi} f(\xi) D_n(\xi - x) d\xi, \tag{51}$$

where the Dirichlet kernel $D_n(x)$ is given by

$$D_n(x) = \frac{\sin\,(n+\frac{1}{2})x}{2\sin\frac{x}{2}}. \tag{52}$$

This function oscillates between positive and negative values. The behavior is therefore responsible for the appearance of the Gibbs phenomenon near the jump discontinuities of the Fourier series at the boundary.

Therefore, our problem can be accurately framed as follows: given the $2N + 1$ Fourier coefficients $\hat{f}_k$ of our generating function (39) for $-N \le k \le N$, with the generating function defined in the interval $w \in [-1, 1]$, we need to reconstruct the point value of the function at the limit $w \to 1$. The point value of the generating function at this limit exactly corresponds to the von Neumann entropy. Especially, we need the reconstruction to converge exponentially fast with N to the correct point value of the generating function, that is

$$\lim_{w\to 1} |G(w;\rho_A) - f_N(w)| \le e^{-\alpha N}, \quad \alpha > 0. \tag{53}$$

This is for the purpose of having a realistic application of the quantum model, where currently the degree N we can approximate for the partial Fourier series is limited by the depth or the width of the quantum circuits.

We are in need of an operation that can diminish the oscillations, or even better, to completely remove them. Several filtering methods have been developed to ameliorate the oscillations, including the non-negative and decaying Fejér kernel, which smooths out the Fourier series over the entire interval, or the introduction of Lanczos σ factor, which locally reduces the oscillations near the boundary. For a comprehensive discussion on the Gibbs phenomenon and these filtering methods, see [73]. However, we emphasize that none of these methods are satisfying, as they still cannot recover accurate point values of the function $f(x)$ near the boundary.

Therefore, we need a more effective method to remove the Gibbs phenomenon completely. Here we will adopt a powerful method by re-expanding the partial Fourier series into a basis of Gegenbauer polynomials.[5] This is a method

[5] Note that other methods exist based on periodically extending the function to give an accurate representation within the domain of inter

developed in the 1990s by a series of seminal works [75–80], we also refer to [81,82] for more recent reviews.

The Gegenbauer expansion method allows for accurate representation, within exponential accuracy, by only summing a few terms from the Fourier coefficients. Given an analytic and non-periodic function $f(x)$ on the interval $[-1, 1]$ (or a sub-interval $[a, b] \subset [-1, 1]$) with the Fourier coefficients

$$\hat{f}_k = \frac{1}{2}\int_{-1}^{1} f(x)e^{-ik\pi x}dx, \tag{54}$$

and the partial Fourier series

$$f_N(x) = \sum_{k=-N}^{N} \hat{f}_k e^{ik\pi x}. \tag{55}$$

The following Gegenbauer expansion represents the original function we want to approximate with the Fourier information

$$S_{N,M}(x) = \sum_{n=0}^{M} g^{\lambda}_{n,N} C^{\lambda}_n(x), \tag{56}$$

where $g^{\lambda}_{n,N}$ is the Gegenbauer expansion coefficients and $C^{\lambda}_n(x)$ are the Gegenbauer polynomials.[6] Note that we have the following integral formula for computing $g^{\lambda}_{n,N}$

$$\frac{1}{h^{\lambda}_n}\int_{-1}^{1}(1-x^2)^{\lambda-\frac{1}{2}}e^{in\pi x}C^{\lambda}_n(x)dx = \Gamma(\lambda)\left(\frac{2}{\pi k}\right)^{\lambda} i^n(n+\lambda)J_{n+\lambda}(\pi k), \tag{59}$$

then

$$g^{\lambda}_{n,N} = \delta_{0,n}\hat{f}(0) + \Gamma(\lambda)i^n(n+\lambda) \times \sum_{k=-N,k\neq 0}^{N} J_{n+\lambda}(\pi k)\left(\frac{2}{\pi k}\right)^{\lambda}\hat{f}_k, \tag{60}$$

Footnote 5 continued
est, which involves reconstructing the function based on Chebyshev polynomials [74]. However, we do not explore this method in this work.

[6] The Gegenbauer expansion coefficients $g^{\lambda}_{n,N}$ are defined with the partial Fourier series $f_N(x)$ as

$$g^{\lambda}_{n,N} = \frac{1}{h^{\lambda}_n}\int_{-1}^{1}(1-x^2)^{\lambda-\frac{1}{2}} f_N(x)C^{\lambda}_n(x)dx, \quad 0 \le n \le M. \tag{57}$$

For $\lambda \geq 0$, the Gegenbauer polynomial of degree n is defined to satisfy

$$\int_{-1}^{1}(1-x^2)^{\lambda-\frac{1}{2}}C^{\lambda}_k(x)C^{\lambda}_n(x)dx = 0, \quad k \neq n. \tag{58}$$

We refer to Appendix B for a more detailed account on the properties of the Gegenbauer expansion.

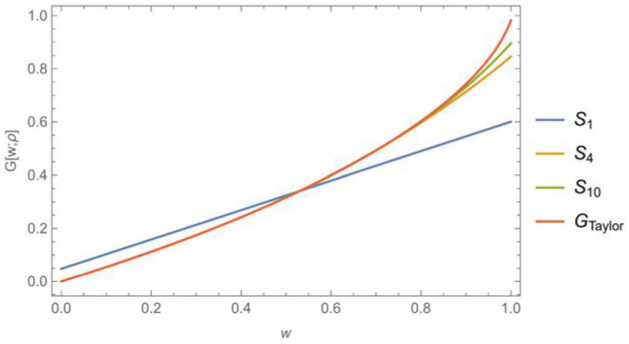

Fig. 25 Gegenbauer expansion constructed from the Fourier information. Here S_M refers to the Gegenbauer polynomials of order M. Note that we set $\beta\epsilon = 0.25$, then $\lambda = M = 0.25N$. Therefore, in order to construct the polynomials of order M, we need the information of the Fourier coefficients to order $N = 4M$

where we only need the Fourier coefficients $\hat{f}_k$.

In fact, the Gegenbauer expansion is a two-parameter family of functions, characterized by λ and M. It has been shown that by setting $\lambda = M = \beta\epsilon N$ where $\epsilon = (b-a)/2$ and $\beta < \frac{2\pi e}{27}$ for the Fourier case, the expansion can achieve exponential accuracy with N. Note that M will determine the degrees of the Gegenbauer polynomials, and as such, we should allow the degrees of the original Fourier series to grow with M. For a clear demonstration of how the Gegenbauer expansion approaches the generating function from the Fourier data, see Fig. 25. We will eventually be able to reconstruct the point value of the von Neumann entropy near $w \to 1$ with increasing order in the expansion. A more precise statement regarding the exponential accuracy can be found in Appendix B. This method is indeed a process of reconstructing local information from global information with exponential accuracy, thereby effectively removing the Gibbs phenomenon.

Given that the Gegenbauer reconstruction from the Fourier data is always possible, establishing the expressivity of quantum neural networks directly for the Gagenbauer polynomials is an open question worth pursuing.

6 Discussion

In this paper, we have considered a novel approach of using classical and quantum neural networks to study the analytic continuation of von Neumann entropy from Rényi entropies. We approach the analytic continuation problem in a way suitable to deep learning techniques by rewriting $\mathrm{Tr}\,\rho_A^n$ in the Rényi entropies in terms of a generating function that manifests as a Taylor series (12). We show that our deep learning models achieve this goal with a limited number of Rényi entropies.

Instead of using a static model design for the classical neural networks, we adopt the KerasTuner in finding the optimal model architecture and hyperparameters. There are two supervised learning scenarios: predicting the von Neumann entropy given the knowledge of Rényi entropies using densely connected neural networks, and treating higher Rényi entropies as sequential deep learning using RNNs. In both cases, we have achieved high accuracy in predicting the corresponding targets.

For the quantum neural networks, we frame a similar supervised learning problem as a mapping from inputs to predictions. This allows us to investigate the expressive power of quantum neural networks as function approximators, particularly for the von Neumann entropy. We study quantum models that can explicitly realize the generating function as a partial Fourier series. However, the Gibbs overshooting hinders the recovery of an accurate point value for the von Neumann entropy. To resolve this issue, we re-expand the series in terms of Gegenbauer polynomials, which leads to exponential convergence and improved accuracy.

Several relevant issues and potential improvements arise from our approach:

- It is crucial to choose the appropriate architectures before employing KerasTuner, for instances, densely connected layers in Sect. 3 and RNNs in Sect. 4. Because these architectures are built for certain tasks a priori. KerasTuner only serves as an effective method to determine the optimal complexity and hyperparameters for model training. However, since the examples from CFT_2 have different analytic structures for both the von Neumann and Rényi entropies, it would be interesting to explore how the different hyperparameters correlate with each example.
- Despite being efficient, the parameter spaces we sketched in Sects. 3.1 and 4.1 that the KerasTuner searches are not guaranteed to contain the optimal setting, and there could be better approaches.
- We can generate datasets by fixing different physical parameters, such as temperature for (19) or cross-ratio x for (28). While we have considered the natural parameters to vary, exploring different parameters may offer more representational power. It is possible to find a Dense model that provides feasible predictions in all parameter ranges, but may require an ensemble of models.
- Regularization methods, such as K-fold validation, can potentially reduce the model size or datasets while maintaining the same performance. It would be valuable to determine the minimum datasets required or whether models with low complexity still have the same representational power for learning entanglement entropy.
- On the other hand, training the model with more data and resources is the most effective approach to improve the model's performance. One can also scale up the search process in the KerasTuner or use ensemble methods to combine the models found by it.
- For the quantum neural networks, note that our approach does not guarantee convergence to the correct Fourier coefficients, as we outlined in Sect. 5.1. On the other hand, not all the trainable parameters will contribute to all the Fourier coefficients, where theoretical understanding is lacking. It may be beneficial to investigate various pre-processing or data-encoding strategies to improve the approximation of the partial Fourier series with a high degree r that generally requires more training parameters [83–86].

There are also future directions that are worth exploring that we shall comment on briefly:

- **Mutual information:** We can extend our study to mutual information for two disjoint intervals A and B, which is an entanglement measure related to the von Neumann entropy defined as

$$I(A:B) \equiv S(\rho_A) + S(\rho_B) - S(\rho_{A\cup B}). \tag{61}$$

  In particular, there is a conjectured form of the generating function in [8], with $\text{Tr}\,\rho_A^n$ being replaced by $\text{Tr}\,\rho_A^n\,\text{Tr}\,\rho_B^n/\,\text{Tr}\,\rho_{A\cup B}^n$. It is worth exploring the expressivity of classical and quantum neural networks using this generating function, particularly as mutual information allows eliminating the UV-divergence and can be compared with some realistic simulations, such as spin-chain models [87].
- **Self-supervised learning for higher Rényi entropies:** Although we have shown that RNN architecture is effective in the sequence learning problem in Sect. 4, it is worth considering other architectures that could potentially offer better performance. For instance, a time-delay neural network, depthwise separable convolutional neural network, or a Transformer may be appropriate for certain types of data. These architectures may be worth exploring in extending the task of extracting higher Rényi entropies as self-supervised learning, particularly for examples where analytic continuation is not available.
- **Other entanglement measures from analytic continuation:** There are other important entanglement measures, say, relative entropy or entanglement negativity that may require analytic continuation and can be studied numerically based on neural networks. We may also consider entanglement entropy or entanglement spectrum that can be simulated in specific models stemming from condensed matter or holographic systems.
- **Expressivity of classical and quantum neural networks:** We have studied the expressivity of classical

and neural networks for the von Neumann and Rényi entropies, with the generating function as the medium. This may help us in designing good generating functions for other entanglement measures suitable for neural networks. It is also worth understanding whether other entanglement measures are also in the function classes that the quantum neural networks can realize.

Acknowledgements We thank Xi Dong for his encouragement of this work. C-H.W. was supported in part by the U.S. Department of Energy under Grant No. DE-SC0023275, and the Ministry of Education, Taiwan. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-19-1-0360.

Data Availability Statement This manuscript has no associated data or the data will not be deposited. [Authors' comment: The data is available on GitHub at the same repository below in Code Availability].

Code availability The source code for both our classical and quantum models is publicly available on GitHub at the following repository: (https://github.com/Chih-HungWu/ML-von-Neumann-Entropy). The implementation is written in Python and is based on the TensorFlow-Keras framework [52,53] for the classical deep learning models studied in Sects. 3 and 4, and on the PennyLane framework [72] for the quantum models studied in Sect. 5. We have also included the Mathematica files used to generate the datasets based on the generating function method introduced in Sect. 2, along with our datasets. One can easily reproduce our results using their own choices of datasets with great generalizability; please refer to the documentation for further instructions.

Funded by SCOAP[3].

Appendix A Fourier series representation of the generating function

Suppose there is a Fourier series representation of the generating function from (7)

$$G(z;\rho_A)=\sum_{n=-\infty}^{\infty}c_n e^{inz}. \tag{A1}$$

The idea is that we want to compute the Fourier coefficients given only the information about $G(z;\rho)$ or $\mathrm{Tr}\,\rho_A^n$. We can compute the complex-valued Fourier coefficients c_n using real-valued coefficients a_n and b_n for a general period T where

$$G(z;\rho_A)=\frac{a_0}{2}+\sum_{n=1}^{\infty}a_n\cos\left(\frac{2\pi nz}{T}\right)+b_n\sin\left(\frac{2\pi nz}{T}\right). \tag{A2}$$

Note that

$$a_n=\frac{2}{T}\int_{z_1}^{z_2}G(z;\rho_A)\cos\left(\frac{2\pi nz}{T}\right)dz, \tag{A3}$$

$$b_n=\frac{2}{T}\int_{z_1}^{z_2}G(z;\rho_A)\sin\left(\frac{2\pi nz}{T}\right)dz, \tag{A4}$$

where we only need to compute the two Fourier coefficients using the generating function of $\mathrm{Tr}\,\rho_A^n$. However, the above integrals are hard to evaluate in general. Instead, we will show that both a_n and b_n can be written as the following series

$$a_n=\sum_{m=0}^{\infty}\frac{G(0;\rho)^{(m)}}{m!}C_{cos}(n,m), \tag{A5}$$

$$b_n=\sum_{m=0}^{\infty}\frac{G(0;\rho)^{(m)}}{m!}C_{sin}(n,m). \tag{A6}$$

where $C_{cos}(n,m)$ and $C_{sin}(n,m)$ involve certain special functions. The definitions of $G(0;\rho_A)^{(m)}$ starts from the following generating function in terms of w from (9)

$$G(w;\rho_A)=-\mathrm{Tr}\,(\rho_A\ln[1-w(1-\rho_A)]), \tag{A7}$$

where the m-th derivative with $w\to 0$

$$\begin{aligned}&G(0;\rho_A)^{(m)}\\&=-\mathrm{Tr}[(-1)^{m+1}(m-1)!\rho_A(\rho_A-1)^m]\\&=-(m-1)!\sum_{k=0}^{m}\frac{(-1)^{2m-k+1}m!}{k!(m-k)!}\mathrm{Tr}\,(\rho_A^{k+1}).\end{aligned} \tag{A8}$$

Note that we have to define for $m=0$ such that

$$G(0;\rho_A)^{(0)}=-\mathrm{Tr}(\rho_A\ln 1)=0. \tag{A9}$$

Then we have the Fourier series representation of the generating function on an interval $[w_1,w_2]$ with period $T=w_2-w_1$ given by

$$\begin{aligned}G(w;\rho_A)=&\frac{a_0}{2}\\&+\sum_{n=1}^{\infty}\left\{\sum_{m=0}^{\infty}\frac{\tilde{f}(m)}{m}C_{cos}(n,m)\cos\left(\frac{2\pi nw}{T}\right)\right.\\&\left.+\sum_{m=0}^{\infty}\frac{\tilde{f}(m)}{m}C_{sin}(n,m)\sin\left(\frac{2\pi nw}{T}\right)\right\},\end{aligned} \tag{A10}$$

where we have defined

$$\tilde{f}(m) \equiv -\sum_{k=0}^{m} \frac{(-1)^{2m-k+1} m!}{k!(m-k)!} \mathrm{Tr}\,(\rho_A^{k+1}). \tag{A11}$$

with manifest $\mathrm{Tr}\,\rho_A^{k+1}$ appearing in the expression.

Now we need to work out $C_{cos}(n,m)$ and $C_{sin}(n,m)$. First, let us consider in general

$$a_n = \frac{2}{T}\int_{t_1}^{t_2} f(t)\cos\left(\frac{2\pi nt}{T}\right)dt, \tag{A12}$$

where we have written $G(w;\rho_A)$ as $f(t)$ for simplicity. We can write down the Taylor series of both pieces

$$f(t) = \sum_{j=0}^{\infty} \frac{f^{(j)}(0)}{j!} t^j \tag{A13}$$

$$\cos\left(\frac{2\pi nt}{T}\right) = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k)!}\left(\frac{2\pi nt}{T}\right)^{2k}, \tag{A14}$$

Consider the following function

$$\begin{aligned} T_{cos}(t) &\equiv f(t)\cos\left(\frac{2\pi nt}{T}\right) \\ &= \left[\sum_{j=0}^{\infty} \frac{f^{(j)}(0)}{j!} t^j\right]\left[\sum_{k=0}^{\infty} \frac{(-1)^k}{(2k)!}\left(\frac{2\pi nt}{T}\right)^{2k}\right], \end{aligned} \tag{A15}$$

then let us collect the terms in orders of t

$$\begin{aligned} T_{cos}(t) &= f(0) + f^{(1)}(0)t \\ &+ \left(\frac{1}{2}f^{(2)}(0) - 2f(0)\left(\frac{\pi n}{T}\right)^2\right)t^2 \\ &+ \left(\frac{1}{6}f^{(3)}f(0) - 2f^{(1)}(0)\left(\frac{\pi n}{T}\right)^2\right)t^3 \\ &+ \left(\frac{1}{24}f^{(4)}(0) - f^{(2)}\left(\frac{\pi n}{T}\right)^2 + \frac{2}{3}f(0)\left(\frac{\pi n}{T}\right)^4\right)t^4 \\ &+ \cdots, \end{aligned} \tag{A16}$$

then the integral becomes

$$\begin{aligned} \int_{t_1}^{t_2} T_{cos}(t)dt &= f(0)(t_2-t_1) + \frac{1}{2}f^{(1)}(0)(t_2^2-t_1^2) \\ &+ \frac{1}{3}\left(\frac{1}{2}f^{(2)}(0) - 2f(0)\left(\frac{\pi n}{T}\right)^2\right)(t_2^3-t_1^3) \\ &+ \frac{1}{4}\left(\frac{1}{6}f^{(3)}f(0) - 2f^{(1)}(0)\left(\frac{\pi n}{T}\right)^2\right)(t_2^4-t_1^4) \\ &+ \frac{1}{5}\left(\frac{1}{24}f^{(4)}(0) - f^{(2)}\left(\frac{\pi n}{T}\right)^2\right. \\ &\left.+ \frac{2}{3}f(0)\left(\frac{\pi n}{T}\right)^4\right)(t_2^5-t_1^5) \\ &+ \cdots. \end{aligned} \tag{A17}$$

Now we want to re-order this expression, where we collect terms in terms of $f^{(m)}(0)$

$$\begin{aligned} &\int_{t_1}^{t_2} T_{cos}(t)dt \\ &= f(0)\left((t_2-t_1) - \frac{2}{3}\left(\frac{\pi n}{T}\right)^2(t_2^3-t_1^3)\right. \\ &\left.+ \frac{2}{15}\left(\frac{\pi n}{T}\right)^4(t_2^5-t_1^5) + \cdots\right) \\ &+ f^{(1)}(0)\left(\frac{1}{2}(t_2^2-t_1^2) - \frac{1}{2}\left(\frac{\pi n}{T}\right)^2(t_2^4-t_1^4) + \cdots\right) \\ &+ f^{(2)}(0)\left(\frac{1}{24}(t_2^4-t_1^4) + \cdots\right) + \cdots. \end{aligned} \tag{A18}$$

After multiplying a factor of $2/T$, this can be written as

$$a_n = \frac{2}{T}\int_{t_1}^{t_2} T_{cos}(t)dt = \sum_{m=0}^{\infty} \frac{f^{(m)}(0)}{m!} C_{cos}(n,m), \tag{A19}$$

where

$$\begin{aligned} &C_{cos}(n,m) \\ &= \sum_{p=0}^{\infty}\left[\frac{(-1)^p 2^{(2p+1)} n^{2p}\pi^{2p}}{(2p+m+1)(2p)!T^{2p+1}}\right. \\ &\left.\times\left(t_2^{(2p+m+1)} - t_1^{(2p+m+1)}\right)\right] \\ &= \frac{2}{(m+1)T} \\ &\times\left[{}_pF_q\left(\frac{m+1}{2};\frac{1}{2},\frac{m+3}{2};-\frac{n^2\pi^2t_2^2}{T^2}\right)t_2^{m+1}\right. \\ &\left.- {}_pF_q\left(\frac{m+1}{2};\frac{1}{2},\frac{m+3}{2};-\frac{n^2\pi^2t_2^2}{T^2}\right)t_1^{m+1}\right]. \end{aligned} \tag{A20}$$

Next, we consider the case for $C_{sin}(n,m)$, where we need to work out

$$b_n = \frac{2}{T}\int_{t_1}^{t_2} f(t)\sin\left(\frac{2\pi nt}{T}\right)dt, \tag{A21}$$

again, we know

$$\sin\left(\frac{2\pi nt}{T}\right) = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)!}\left(\frac{2\pi nt}{T}\right)^{(2k+1)}, \tag{A22}$$

then we define

$$\begin{aligned} T_{sin}(t) &\equiv f(t)\sin\left(\frac{2\pi nt}{T}\right) \\ &= \left[\sum_{j=0}^{\infty} \frac{f^{(j)}(0)}{j!} t^j\right]\left[\sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)!}\left(\frac{2\pi nt}{T}\right)^{2k+1}\right], \end{aligned} \tag{A23}$$

with the only difference being the denominator $(2k)! \to (2k+1)!$ and the power of $\frac{2\pi nt}{T}$ becomes $2k+1$. Then

$$\begin{aligned} C_{sin}(n,m) &= \sum_{p=0}^{\infty}\Bigg[\frac{(-1)^p 2^{(2p+2)} n^{2p+1}\pi^{2p+1}}{(2p+m+2)(2p+1)!T^{2p+2}} \\ &\quad\times\left(t_2^{(2p+m+2)} - t_1^{(2p+m+2)}\right)\Bigg] \\ &= \frac{4n\pi}{(m+2)T^2} \\ &\quad\times\Bigg[{}_pF_q\left(\frac{m+2}{2};\frac{3}{2},\frac{m+4}{2};-\frac{n^2\pi^2t_2^2}{T^2}\right)t_2^{m+2} \\ &\quad- {}_pF_q\left(\frac{m+2}{2};\frac{3}{2},\frac{m+4}{2};-\frac{n^2\pi^2t_1^2}{T^2}\right)t_1^{m+2}\Bigg]. \end{aligned} \tag{A24}$$

Appendix B The Gegenbauer polynomials and the Gibbs phenomenon

In the appendix, we discuss briefly the definition and properties of the Gegenbauer polynomials used to remove the Gibbs phenomenon in Sect. 5.4.

The Gegenbauer polynomials $C_n^\lambda(x)$ of degree n for $\lambda \geq 0$ are defined by the integral

$$\int_{-1}^{1}(1-x^2)^{\lambda-\frac{1}{2}}C_k^\lambda(x)C_n^\lambda(x)dx = 0, \quad k \neq n. \tag{B25}$$

with the following normalization

$$C_n^\lambda(1) = \frac{\Gamma(n+2\lambda)}{n!\Gamma(2\lambda)}. \tag{B26}$$

Note the polynomials are not orthonormal, the norm of $C_n^\lambda(x)$ is

$$\int_{-1}^{1}(1-x^2)^{\lambda-\frac{1}{2}}(C_n^\lambda(x))^2dx = h_n^\lambda, \tag{B27}$$

where

$$h_n^\lambda = \pi^{\frac{1}{2}}C_n^\lambda(1)\frac{\Gamma(\lambda+\frac{1}{2})}{\Gamma(\lambda)(n+\lambda)}. \tag{B28}$$

Given a function $f(x)$ defined on the interval $[-1,1]$ (or a sub-interval $[a,b] \subset [-1,1]$), the corresponding Gegenbauer coefficients $\hat{f}^\lambda(l)$ are given by

$$\hat{f}^\lambda(l) = \frac{1}{h_n^\lambda}\int_{-1}^{1}(1-x^2)^{\lambda-\frac{1}{2}}f(x)C_l^\lambda(x)dx, \tag{B29}$$

then the truncated Gegenbauer expansion up to the first $m+1$ terms is

$$f_m^\lambda(x) = \sum_{l=0}^{m}\hat{f}^\lambda(l)C_l^\lambda(x). \tag{B30}$$

Here we will sketch briefly how the Gegenbauer expansion leads to a resolution of the Gibbs phenomenon as we discussed in Sect. 5.4. In fact, one can prove that there is an exponential convergence between the function $f(x)$ we want to approximate and the m-th degree Gegenbauer polynomials. We will only sketch the idea behind the proof, and we refer the readers to the review in [80] for the details.

One can establish exponential convergence by demonstrating that the errors for the N-th Fourier coefficient, expanded into Gegenbauer polynomials, can be made exponentially small. Let us call the $f_N^m(x)$ the expansion of $f_N(x)$ into m-th degree Gegenbauer polynomials and $f^m(x)$ the expansion of $f(x)$ into m-th degree Gegenbauer polynomials. Then we have the following relation, where the approximation of $f(x)$ by $f_N^m(x)$ is obviously bounded by the error between $f(x)$ and $f^m(x)$ and the error between $f^m(x)$ and $f_N^m(x)$

$$\begin{aligned} ||f(x) - f_N^m(x)|| \leq\ & ||f(x) - f^m(x)|| \\ & + ||f^m(x) - f_N^m(x)||. \end{aligned} \tag{B31}$$

On the right hand side of the inequality, we call the first norm as the *regularization error*, while the second norm as the *truncation error*. Note that we take the norm to be the maximum norm over the interval $[-1,1]$. To be more precise, we can write the truncation error as

$$||f^m - f_N^m|| = \max_{-1\leq x\leq 1}\left|\sum_{k=0}^{m}(\hat{f}_k^\lambda - \hat{g}_k^\lambda)C_k^\lambda(x)\right|, \tag{B32}$$

where we take $\hat{f}_k^\lambda$ to be the unknown Gegenbauer coefficients of the function $f(x)$. If both λ and m grow linearly with N, this error is shown to be exponentially small. On the other hand, the regularization error can be written as

$$||f - f^m|| = \max_{-1\leq 1}\left|f(x) - \sum_{k=0}^{m}\hat{f}_k^\lambda C_k^\lambda(x)\right|. \tag{B33}$$

It can also be shown that this error is exponentially small for $\lambda = \gamma m$ with a positive constant γ. Since both the regularization and truncation errors can be made exponentially small with the prescribed conditions, the Gegenbauer expansion achieves uniform exponential accuracy and removes the Gibbs phenomenon from the Fourier data.

References

1. T. Faulkner, T. Hartman, M. Headrick, M. Rangamani, B. Swingle, in *2022 Snowmass Summer Study* (2022)
2. R. Bousso, X. Dong, N. Engelhardt, T. Faulkner, T. Hartman, S.H. Shenker, D. Stanford, Snowmass White Paper: Quantum Aspects of Black Holes and the Emergence of Spacetime (2022). arXiv:2201.03096 [hep-th]
3. A. Rényi, In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions*

to the Theory of Statistics (University of California Press, 1961), pp. 547–561. http://projecteuclid.org/euclid.bsmsp/1200512181
4. P. Calabrese, J.L. Cardy, Entanglement entropy and quantum field theory. J. Stat. Mech. **0406**, P06002 (2004). https://doi.org/10.1088/1742-5468/2004/06/P06002. arXiv:hep-th/0405152
5. P. Calabrese, J. Cardy, E. Tonni, Entanglement entropy of two disjoint intervals in conformal field theory. J. Stat. Mech. **0911**, P11001 (2009). https://doi.org/10.1088/1742-5468/2009/11/P11001. arXiv:0905.2069 [hep-th]
6. P. Calabrese, J. Cardy, Entanglement entropy and conformal field theory. J. Phys. A **42**, 504005 (2009). https://doi.org/10.1088/1751-8113/42/50/504005. arXiv:0905.4013 [cond-mat.stat-mech]
7. P. Calabrese, J. Cardy, E. Tonni, Entanglement entropy of two disjoint intervals in conformal field theory II. J. Stat. Mech. **1101**, P01021 (2011). https://doi.org/10.1088/1742-5468/2011/01/P01021. arXiv:1011.5482 [hep-th]
8. E. D'Hoker, X. Dong, C.H. Wu, An alternative method for extracting the von Neumann entropy from Rényi entropies. JHEP **01**, 042 (2021). https://doi.org/10.1007/JHEP01(2021)042. arXiv:2008.10076 [hep-th]
9. H. Yoon, J.H. Sim, M.J. Han, Analytic continuation via domain knowledge free machine learning. Phys. Rev. B **98**(24), 245101 (2018)
10. R. Fournier, L. Wang, O.V. Yazyev, Q. Wu, Artificial neural network approach to the analytic continuation problem. Phys. Rev. Lett. **124**(5) (2020). https://doi.org/10.1103/physrevlett.124.056401
11. X. Xie, F. Bao, T. Maier, C. Webster, Analytic continuation of noisy data using Adams Bashforth ResNet (2019). arXiv preprint arXiv:1905.10430
12. T. Song, R. Valenti, H. Lee, Analytic continuation of the self-energy via machine learning techniques (2020). arXiv preprint arXiv:2007.13610
13. D. Huang, Yf. Yang, Learned optimizers for analytic continuation. Phys. Rev. B **105**(7), 075112 (2022)
14. K.W. Sun, F. Wang, Neural network analytic continuation for Monte Carlo: improvement by statistical errors (2023). arXiv preprint arXiv:2302.11317
15. G. Cybenko, Approximation by superpositions of a sigmoidal function. Math. Control Signals Syst. (MCSS) **2**(4), 303–314 (1989). https://doi.org/10.1007/BF02551274
16. T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi, et al. Kerastuner (2019). https://github.com/keras-team/keras-tuner
17. J.R. McClean, J. Romero, R. Babbush, A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms. N. J. Phys. **18**(2), 023023 (2016). https://doi.org/10.1088/1367-2630/18/2/023023
18. J. Romero, A. Aspuru-Guzik, Variational quantum generators: generative adversarial quantum machine learning for continuous distributions. Adv. Quantum Technol. **4**(1), 2000003 (2021)
19. K. Mitarai, M. Negoro, M. Kitagawa, K. Fujii, Quantum circuit learning. Phys. Rev. A **98**(3) (2018). https://doi.org/10.1103/physreva.98.032309
20. E. Farhi, H. Neven, Classification with quantum neural networks on near term processors (2018). arXiv preprint arXiv:1802.06002
21. J.R. McClean, S. Boixo, V.N. Smelyanskiy, R. Babbush, H. Neven, Barren plateaus in quantum neural network training landscapes. Nat. Commun. **9**(1) (2018). https://doi.org/10.1038/s41467-018-07090-4
22. M. Benedetti, E. Lloyd, S. Sack, M. Fiorentini, Parameterized quantum circuits as machine learning models. Quantum Sci. Technol. **4**(4), 043001 (2019). https://doi.org/10.1088/2058-9565/ab4eb5
23. A. Pérez-Salinas, D. López-Núñez, A. García-Sáez, P. Forn-Díaz, J.I. Latorre, One qubit as a universal approximant. Phys. Rev. A **104**(1), 012405 (2021)
24. T. Goto, Q.H. Tran, K. Nakajima, Universal approximation property of quantum machine learning models in quantum-enhanced feature spaces. Phys. Rev. Lett. **127**(9), 090506 (2021)
25. B.Y. Gan, D. Leykam, D.G. Angelakis, Fock state-enhanced expressivity of quantum machine learning models. EPJ Quantum Technol. **9**(1), 16 (2022)
26. C.C. Chen, M. Watabe, K. Shiba, M. Sogabe, K. Sakamoto, T. Sogabe, On the expressibility and overfitting of quantum circuit learning. ACM Trans. Quantum Comput. **2**(2), 1–24 (2021)
27. S. Shin, Y. Teo, H. Jeong, Exponential data encoding for quantum supervised learning (2022). arXiv preprint arXiv:2206.12105
28. M.C. Caro, E. Gil-Fuster, J.J. Meyer, J. Eisert, R. Sweke, Encoding-dependent generalization bounds for parametrized quantum circuits. Quantum **5**, 582 (2021)
29. F.J. Gil Vidal, D.O. Theis, Input redundancy for parameterized quantum circuits. Front. Phys. **8**, 297 (2020)
30. M. Schuld, R. Sweke, J.J. Meyer, Effect of data encoding on the expressive power of variational quantum-machine-learning models. Phys. Rev. A **103**(3) (2021). https://doi.org/10.1103/physreva.103.032430
31. R.D. Sorkin, In *10th International Conference on General Relativity and Gravitation*, vol. 2, pp. 734–736 (1984)
32. L. Bombelli, R.K. Koul, J. Lee, R.D. Sorkin, A quantum source of entropy for black holes. Phys. Rev. D **34**, 373–383 (1986). https://doi.org/10.1103/PhysRevD.34.373
33. M. Srednicki, Entropy and area. Phys. Rev. Lett. **71**, 666–669 (1993). https://doi.org/10.1103/PhysRevLett.71.666. arXiv:hep-th/9303048
34. C. Holzhey, F. Larsen, F. Wilczek, Geometric and renormalized entropy in conformal field theory. Nucl. Phys. B **424**, 443–467 (1994). https://doi.org/10.1016/0550-3213(94)90402-2. arXiv:hep-th/9403108
35. E. Witten, APS Medal for Exceptional Achievement in Research: Invited article on entanglement properties of quantum field theory. Rev. Mod. Phys. **90**(4), 045003 (2018). https://doi.org/10.1103/RevModPhys.90.045003. arXiv:1803.04993 [hep-th]
36. H. Casini, M. Huerta, Entanglement entropy in free quantum field theory. J. Phys. A **42**, 504007 (2009). https://doi.org/10.1088/1751-8113/42/50/504007. arXiv:0905.2562 [hep-th]
37. S.N. Solodukhin, Entanglement entropy, conformal invariance and extrinsic geometry. Phys. Lett. B **665**, 305–309 (2008). https://doi.org/10.1016/j.physletb.2008.05.071. arXiv:0802.3117 [hep-th]
38. M.P. Hertzberg, F. Wilczek, Some calculable contributions to entanglement entropy. Phys. Rev. Lett. **106**, 050404 (2011). https://doi.org/10.1103/PhysRevLett.106.050404. arXiv:1007.0993 [hep-th]
39. V. Rosenhaus, M. Smolkin, Entanglement entropy: a perturbative calculation. JHEP **12**, 179 (2014). https://doi.org/10.1007/JHEP12(2014)179. arXiv:1403.3733 [hep-th]
40. R.C. Myers, A. Sinha, Holographic c-theorems in arbitrary dimensions. JHEP **01**, 125 (2011). https://doi.org/10.1007/JHEP01(2011)125. arXiv:1011.5819 [hep-th]
41. H. Liu, M. Mezei, A refinement of entanglement entropy and the number of degrees of freedom. JHEP **04**, 162 (2013). https://doi.org/10.1007/JHEP04(2013)162. arXiv:1202.2070 [hep-th]
42. T. Faulkner, The entanglement Renyi entropies of disjoint intervals in AdS/CFT (2013). arXiv:1303.7221 [hep-th]
43. R.P. Boas, *Entire Functions* (Academic Press, New York, 1954)
44. E. Witten, Open strings on the Rindler horizon. JHEP **01**, 126 (2019). https://doi.org/10.1007/JHEP01(2019)126. arXiv:1810.11912 [hep-th]
45. A. Dabholkar, Strings on a cone and black hole entropy. Nucl. Phys. B **439**, 650–664 (1995). https://doi.org/10.1016/0550-3213(95)00050-3. arXiv:hep-th/9408098

46. E. Witten, Open strings on the Rindler horizon. JHEP **01**, 126 (2019). https://doi.org/10.1007/JHEP01(2019)126. arXiv:1810.11912 [hep-th]
47. C.A. Agon, M. Headrick, D.L. Jafferis, S. Kasko, Disk entanglement entropy for a Maxwell field. Phys. Rev. D **89**(2), 025018 (2014). https://doi.org/10.1103/PhysRevD.89.025018. arXiv:1310.4886 [hep-th]
48. A. Lewkowycz, J. Maldacena, Generalized gravitational entropy. JHEP **08**, 090 (2013). https://doi.org/10.1007/JHEP08(2013)090. arXiv:1304.4926 [hep-th]
49. C. Akers, G. Penington, Leading order corrections to the quantum extremal surface prescription. JHEP **04**, 062 (2021). https://doi.org/10.1007/JHEP04(2021)062. arXiv:2008.03319 [hep-th]
50. X. Dong, X.L. Qi, M. Walter, Holographic entanglement negativity and replica symmetry breaking. JHEP **06**, 024 (2021). https://doi.org/10.1007/JHEP06(2021)024. arXiv:2101.11029 [hep-th]
51. D.P. Kingma, J. Ba, Adam: a method for stochastic optimization (2014). arXiv preprint arXiv:1412.6980
52. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng. TensorFlow: large-scale machine learning on heterogeneous systems (2015). Software available from https://www.tensorflow.org/
53. F. Chollet et al., Keras. https://keras.io (2015)
54. S.J. Reddi, S. Kale, S. Kumar, On the convergence of adam and beyond (2019). arXiv preprint arXiv:1904.09237
55. T. Azeyanagi, T. Nishioka, T. Takayanagi, Near extremal black hole entropy as entanglement entropy via AdS(2)/CFT(1). Phys. Rev. D **77**, 064005 (2008). https://doi.org/10.1103/PhysRevD.77.064005. arXiv:0710.2956 [hep-th]
56. D. Blanco, A. Garbarz, G. Pérez-Nadal, Entanglement of a chiral fermion on the torus. JHEP **09**, 076 (2019). https://doi.org/10.1007/JHEP09(2019)076. arXiv:1906.07057 [hep-th]
57. P. Fries, I.A. Reyes, Entanglement and relative entropy of a chiral fermion on the torus. Phys. Rev. D **100**(10), 105015 (2019). https://doi.org/10.1103/PhysRevD.100.105015. arXiv:1906.02207 [hep-th]
58. D. Blanco, T.F. Chase, J. Laurnagaray, G. Pérez-Nadal, Rényi entropies of the massless Dirac field on the torus. Phys. Rev. D **105**(4), 045014 (2022). https://doi.org/10.1103/PhysRevD.105.045014. arXiv:2112.14237 [hep-th]
59. B. Deconinck, M. Heil, A. Bobenko, M. Van Hoeij, M. Schmies, Computing Riemann theta functions. Math. Comput. **73**(247), 1417–1442 (2004). arXiv:nlin/0206009 [nlin.SI]
60. J. Frauendiener, C. Jaber, C. Klein, Efficient computation of multidimensional theta functions. J. Geom. Phys. **141**, 147–158 (2019). arXiv:1701.07486 [nlin.SI]
61. T. Barrella, X. Dong, S.A. Hartnoll, V.L. Martin, Holographic entanglement beyond classical gravity. JHEP **09**, 109 (2013). https://doi.org/10.1007/JHEP09(2013)109. arXiv:1306.4682 [hep-th]
62. E. Perlmutter, Comments on Renyi entropy in AdS_3/CFT_2. JHEP **05**, 052 (2014). https://doi.org/10.1007/JHEP05(2014)052. arXiv:1312.5740 [hep-th]
63. D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986). https://doi.org/10.1038/323533a0
64. D. Rumelhart, G. Hinton, R. Williams, S.D.I.f.C.S. University of California, *Learning Internal Representations by Error Propagation*. ICS report (Institute for Cognitive Science, University of California, San Diego, 1985). https://books.google.com/books?id=Ff9iHAAACAAJ
65. P.J. Werbos, Generalization of backpropagation with application to a recurrent gas market model. Neural Netw. **1**(4), 339–356 (1988). https://doi.org/10.1016/0893-6080(88)90007-X. https://www.sciencedirect.com/science/article/pii/089360808890007X
66. Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994). https://doi.org/10.1109/72.279181
67. S. Hochreiter, J. Schmidhuber, Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
68. K. Cho, B. van Merriënboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: encoder–decoder approaches, pp. 103–111 (2014). https://doi.org/10.3115/v1/W14-4012. https://aclanthology.org/W14-4012
69. A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, J.I. Latorre, Data re-uploading for a universal quantum classifier. Quantum **4**, 226 (2020)
70. P. Rebentrost, M. Mohseni, S. Lloyd, Quantum support vector machine for big data classification. Phys. Rev. Lett. **113**(13), 130503 (2014)
71. J.R. McClean, S. Boixo, V.N. Smelyanskiy, R. Babbush, H. Neven, Barren plateaus in quantum neural network training landscapes. Nat. Commun. **9**(1), 4812 (2018)
72. V. Bergholm et al., PennyLane: automatic differentiation of hybrid quantum-classical computations (2018). arXiv:1811.04968 [quant-ph]
73. A.J. Jerri, The Gibbs phenomenon in Fourier analysis, splines and wavelet approximations. Math. Appl. **446** (1998)
74. D. Huybrechs, On the Fourier extension of nonperiodic functions. SIAM J. Numer. Anal. **47**(6), 4326–4355 (2010)
75. D. Gottlieb, C.W. Shu, A. Solomonoff, H. Vandeven, On the Gibbs phenomenon I: Recovering exponential accuracy from the Fourier partial sum of a nonperiodic analytic function. J. Comput. Appl. Math. **43**(1–2), 81–98 (1992)
76. D. Gottlieb, C.W. Shu, Resolution properties of the Fourier method for discontinuous waves. Comput. Methods Appl. Mech. Eng. **116**(1–4), 27–37 (1994)
77. D. Gottlieb, C.W. Shu, On the Gibbs phenomenon III: recovering exponential accuracy in a sub-interval from a spectral partial sum of a piecewise analytic function. SIAM J. Numer. Anal. **33**(1), 280–290 (1996)
78. D. Gottlieb, C.W. Shu, On the Gibbs phenomenon. IV. Recovering exponential accuracy in a subinterval from a Gegenbauer partial sum of a piecewise analytic function. Math. Comput. **64**(211), 1081–1095 (1995)
79. D. Gottlieb, C.W. Shu, On the Gibbs phenomenon V: Recovering exponential accuracy from collocation point values of a piecewise analytic function. Numer. Math. **71**(4), 511–526 (1995)
80. D. Gottlieb, C.W. Shu, On the Gibbs phenomenon and its resolution. SIAM Rev. **39**(4), 644–668 (1997)
81. A. Gelb, S. Gottlieb, *The resolution of the Gibbs phenomenon for Fourier spectral methods. Advances in The Gibbs Phenomenon* (Sampling Publishing, Potsdam, 2007)
82. S. Gottlieb, J.H. Jung, S. Kim, A review of David Gottlieb's work on the resolution of the Gibbs phenomenon. Commun. Comput. Phys. **9**(3), 497–519 (2011)
83. J. Liu, F. Tacchino, J.R. Glick, L. Jiang, A. Mezzacapo, Representation learning via quantum neural tangent kernels. PRX Quantum **3**(3), 030323 (2022). https://doi.org/10.1103/PRXQuantum.3.030323. arXiv:2111.04225 [quant-ph]
84. J. Liu, Z. Lin, L. Jiang, Laziness, Barren plateau, and noise in machine learning (2022). arXiv:2206.09313 [cs.LG]
85. X. Wang, J. Liu, T. Liu, Y. Luo, Y. Du, D. Tao, Symmetric pruning in quantum neural networks (2022). arXiv:2208.14057 [quant-ph]
86. J. Liu, K. Najafi, K. Sharma, F. Tacchino, L. Jiang, A. Mezzacapo, Analytic theory for the dynamics of wide quantum neural networks.

Phys. Rev. Lett. **130**(15), 150601 (2023). https://doi.org/10.1103/PhysRevLett.130.150601. arXiv:2203.16711 [quant-ph]

87. S. Furukawa, V. Pasquier, J. Shiraishi, Mutual information and compactification radius in a c = 1 critical phase in one dimension. Phys. Rev. Lett. **102**, 170602 (2009). https://doi.org/10.1103/PhysRevLett.102.170602. arXiv:0809.5113 [cond-mat.stat-mech]