Large Language Model Scaling Laws for Neural Quantum States in Quantum Chemistry

Oliver Knitter[1,2,*]   **Dan Zhao**[2,3]   **Stefan Leichenauer**[2]   **Shravan Veerapaneni**[1]

[1]**University of Michigan**   [2]**SandboxAQ**   [3]**New York University**
*knitter@umich.edu, currently at IonQ

Abstract

Scaling laws have been used to describe how large language model (LLM) performance scales with model size, training data size, or amount of computational resources. Motivated by the fact that neural quantum states (NQS) has increasingly adopted LLM-based components, we seek to understand NQS scaling laws, thereby shedding light on the scalability and optimal performance–resource trade-offs of NQS ansatze. In particular, we identify scaling laws that predict the performance, as measured by absolute error and V-score, for transformer-based NQS as a function of problem size in second-quantized quantum chemistry applications. By performing analogous compute-constrained optimization of the obtained parametric curves, we find that the relationship between model size and training time is highly dependent on loss metric and ansatz, and does not follow the approximately linear relationship found for language models.

1   Introduction

Electronic ground state calculations are a fundamental problem in *ab initio* quantum chemistry. As explicit calculation of the ground truth scales exponentially with molecule size, heuristic algorithms based on the variational principle have been devised including Hartree-Fock, coupled cluster methods [2], and tensor network methods such as DMRG [23].

The incorporation of LLM components into NQS opens these ansatze up to systematic scalability analysis using scaling laws [15], mathematical relationships that describe how the computational requirements of a neural network model scale with the size of the underlying problem. For NLP, scaling laws have helped reveal how the performance of specific models grows with access to greater training data and compute resources. These laws help predict the limits of model capabilities, optimize the allocation of resources, and guide development of new architectures.

Applying this broad analysis of NQS performance across different problem types is largely made possible by the introduction of V-Score, [28], a problem-agnostic and model-agnostic metric analyzing performance of any algorithm designed to find the minimal eigenstate of a Hamiltonian. With it, the application of these scaling laws to NQS ansatze may prove fruitful, allowing us to systematically assess the efficiency and effectiveness of different neural architectures for representing quantum states within an application of such practical interest. Understanding the scaling behavior of autoregressive NQS ansatze is particularly important as it would:

- Identify optimal neural architectures balancing accuracy, computational cost, and scalability. This is essential in quantum chemistry, where solving even fundamental problems requires ansatze that are sufficiently expressive at favorable model sizes.

- Illuminate the strengths and weaknesses of NQS as an overall paradigm, highlighting specific areas where improvements in model design could lead to substantial gains in performance.

- Gauge the broader applicability of these models for addressing larger molecular systems, more complex chemical environments, or general tasks beyond ground state energies.

Contributions. We estimate and analyze Chinchilla-like scaling laws [12] for autoregressive NQS models in solving/estimating electronic ground states. Drawing parallels with LLM scaling laws, we examine whether similar principles govern the efficiency and accuracy of deep learning architectures in the NQS domain. We focus on analyzing three specific autoregressive ansatze, based on three well-known architectures: MADE [30], transformers [29], and retentive networks (RetNets) [17].

Our analysis focuses on comparing the performance and costs of these architectures to better understand how they scale with model size and training time, relative to problem size. Inspired by similar estimation frameworks for LLMs [15], we introduce general floating point operation (FLOP) estimates for NQS problems as a proxy for compute costs. Our goal is to provide guidance for optimizing NQS performance in quantum chemistry by appropriately prioritizing the use of available compute resources for a given problem and molecule size.

Our paper is organized as follows. Section 3 gives an overview of the general problem formulation. Section 3.2 describes the NQS paradigm and the autoregressive ansatze under consideration. We detail the use of scaling laws in LLMs and how we apply them in our ansatz experiments for NQS in section 4. We summarize our findings in section 5 and conclude with a discussion of our findings and future directions in section 6.

2 Related Work

Recent advances have used neural networks trained via variational Monte Carlo [6] as an alternative variational ansatz. Known as neural quantum states (NQS), this method offers a potentially scalable solver for electronic ground state calculations [7] with many variants based on autoregressive ansatze [1, 3, 30] such as the transformer architecture [29] among others [17] as autoregressive ansatze, given the inherent expressiveness and utility of transformers within large language models (LLMs). However, further exploration is still necessary to understand how well these architectures perform and scale as NQS ansatze.

Scaling law analyses [10, 15] empirically study how LLM performance can scale with model or data size, computational resources, etc. The eponymous Chinchilla scaling laws [12] estimated that doubling the size of a LLM requires a doubling in the number of training tokens for optimality. Since then, similar efforts have studied scaling behavior in transfer learning settings for downstream tasks [14], the influence on scaling behavior of inductive biases in model architectures [26], effects of scaling on smaller language model trade-offs [13], the potential impact of synthetic data on scaling and model collapse [8], how data-constrained language models scale with repeated data [21], and scaling-informed optimization of model/data sizes when accounting for inference demand [22].

Scaling properties of NQS have been studied in time-evolved quantum states [18] focusing on training unnormalized RBM ansatze [6] and variations of the autoregressive NAQS ansatz [24] under different Ising Hamiltonians. Our analysis is fundamentally similar with a few distinct caveats. We consider more sophisticated ansatz architectures, designed for greater expressiveness with respect to size. Furthermore, we are interested in exploring the scaling properties of autoregressive ansatze applied to a specific practical use case. While [18] indicates NQS ansatze likely require exponentially many parameters to model all possible states of an exponentially large quantum system, the required expressiveness for practical applications may be much lower.

3 Problem

3.1 Preliminaries

Calculating the energy of a molecule requires knowledge of its wavefunction, represented by a unit vector $|\psi\rangle$ in a complex Hilbert space, and its electronic Hamiltonian: a Hermitian operator H_e on this space encoding the total kinetic and potential energy of the molecule. Its ground state energy corresponds to the minimal spectral value of H_e. Under the Born–Oppenheimer approximation, the relatively heavy atomic nuclei are treated as classical point charges, leaving the electrons to be modeled as quantum particles [5].
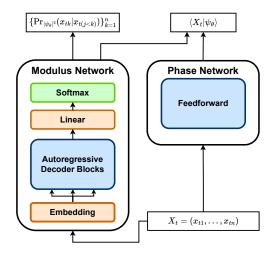
Figure 1: A general autoregressive NQS ansatz $|\psi_\theta\rangle$. The model takes a qubit configuration x and returns either the ansatz entry $\langle x|\psi_\theta\rangle$ or conditional probabilities $\mathrm{Prob}(x_j \mid x_{k<j})$. The modulus network is an autoregressive model that directly contributes to both outputs, while the phase network is typically an MLP only contributing to the ansatz entry.

Given a set of orthonormal spin-orbitals $\{\phi_j\}_{j=1}^n$, second quantization and the Jordan–Wigner transformation together map H_e to a Hamiltonian H in an n-qubit state space [20], whose computational basis corresponds with occupancy configurations, known as *Slater determinants*, of the n spin-orbitals. The smallest eigenvalue of H, also called the ground state energy, approximates the energy of the true molecular ground state. We consider H as a real linear combination of Pauli strings:

$$H = \sum_j \omega_j \left(\sigma_1^j \otimes \cdots \otimes \sigma_n^j \right), \quad \sigma_k^j \in \{I, X, Y, Z\} \tag{1}$$

where the set $\{I, X, Y, Z\}$ denotes the four Pauli matrices. These strings form an orthonormal basis for the 4^n-dimensional space of n-qubit Hamiltonians; the construction of H ensures that the number of terms in (1) is $O(n^4)$ [20]. To find the ground state, we consider the expectation value of H with respect to $|\psi_\theta\rangle$, an n-qubit variational ansatz:

$$L(\theta) = \langle\psi_\theta| H |\psi_\theta\rangle \tag{2}$$

By the Rayleigh–Ritz principle, the only minimum for (2) is the ground state energy of H, in which case $|\psi_\theta\rangle$ represents an associated eigenvector. The goal is then to variationally optimize θ to find this ground state.

3.2 Autoregressive Neural Quantum States

Neural quantum states [6] is a classical deep learning architecture for solving variational many-body problems using a neural network ansatz. Over the past few years, serious interest has manifested toward addressing electronic ground state problems with NQS [7], particularly using autoregressive ansatze [1, 17, 29, 30]; we exclusively consider the autoregressive formulation in this paper. An n-qubit NQS ansatz is a neural network that maps bit strings of length n to complex numbers. For the purposes of training, we consider these outputs as the complex logarithm of the ansatz:

$$\log \langle x|\psi_\theta\rangle = \log \psi_\theta(x) : \{0,1\}^n \to \mathbb{C} \tag{3}$$

Autoregressive ansatze encode these values in two networks: a phase net ϕ_θ returning a single number, and a modulus net returning an autoregressive sequence of nested conditional probabilities:

$$\log p_\theta^j(x_1, \ldots, x_j) = \log \left(\mathrm{Pr}\left(x_j = 1 | x_{k<j}; \theta\right)\right)$$

The individual network outputs are collected into the ansatz output

$$\log \langle x|\psi_\theta\rangle = \frac{1}{2} \sum_{j=1}^n p_\theta^j(x_{k\leq j}) + i\phi_\theta(x) \tag{4}$$

3

This specific construction ensures the ansatz always encodes a normalized quantum state, which is significant because every state induces a probability distribution on the set of qubit spin configurations,

$$\Pr(x;\theta) = |\langle x|\psi_\theta\rangle|^2$$

and we may rewrite the expectation (2) as a stochastic loss function with respect to this distribution:

$$L(\theta) = \mathbb{E}_{x\sim|\psi_\theta|^2}\left[l_\theta(x)\right], \quad l_\theta(x) = \frac{\langle x|\,H\,|\psi_\theta\rangle}{\langle x|\psi_\theta\rangle} \tag{5}$$

The value $l_\theta(x)$ in (5) is called the *local energy* of H at x. The autoregressive ansatz construction makes efficient sampling from $|\psi_\theta|^2$ possible, through use of the conditional probabilities generated by the modulus network to sample spins qubit-by-qubit. For each sample x, the local energy is constructed by multiplying all nonzero entries in row x of H with the corresponding entries of $|\psi_\theta\rangle$, then adding the results. In practice, the set of unique samples is stored and a list of frequencies monitors how often each is sampled with replacement [30]. Explicit constraints are built into the network to ensure the state distribution only samples from physically plausible Slater determinants: the full Hilbert space considers all orbital occupancy configurations, but most do not correspond with the correct number of electrons in the molecule. As in (5), we can rewrite the gradient of the expectation as a stochastic estimator over the state distribution:

$$\nabla_\theta L(\theta) = 2\,\mathrm{Re}\,\mathbb{E}_{x\sim|\psi_\theta|^2}\left[(l_\theta(x) - b)\,\nabla_\theta \log\langle\psi_\theta|x\rangle\right] \tag{6}$$

The baseline b is arbitrary but typically chosen as $b = L(\theta)$ for purposes of variance reduction.

3.3 Types of Autoregressive NQS Ansatze

Model	Sequential Forward FLOP Estimate	Full Ansatz Training FLOP Estimate
MADE	$3N$	$BT\big((1.5n + 3M + 10 - 3\log B)N_{\mathrm{mod}} + 2(M+3)N_{\mathrm{ph}}\big)$
RetNet	$n_{\mathrm{seq}}\left(2N + 4n_{\mathrm{b}}n_{\mathrm{seq}}d_{\mathrm{attn}}\right)$ (Parallel)	$BT\big(((M+4)n - 2\log B)N_{\mathrm{mod}} + 2(M+3)N_{\mathrm{ph}}$
	$n_{\mathrm{seq}}\left(2N + 5n_{\mathrm{b}}(d_{\mathrm{attn}})^2\right)$ (Recurrent)	$+n_{\mathrm{b}}d_{\mathrm{m}}\times(2.5d_{\mathrm{m}}((M+1)n - 2\log B) + 3n^2))$
Transformer	$n_{\mathrm{seq}}\left(2N + 4n_{\mathrm{b}}n_{\mathrm{seq}}d_{\mathrm{attn}}\right)$	$BT\big(((M+4)n - \frac{4}{3}\log B)N_{\mathrm{mod}} + 2(M+3)N_{\mathrm{ph}}$
		$+n_{\mathrm{b}}d_{\mathrm{m}}((M+3.5)n^2 - \frac{8}{9}\log B - \frac{2}{3}(\log B)^2)\big)$

Table 1: Estimates of the total FLOP counts required both to perform one forward pass of MADE, RetNet, and transformer on an input sequence, based on parameter count $N = N_{\mathrm{mod}} + N_{\mathrm{phase}}$, sequence length n_{seq}, number of decoder blocks n_{b}, and the attention/retention dimension d_{attn}; and to train a corresponding NQS ansatz, based on qubit number n, training steps T, average number B of unique samples processed, and unique bit flip terms M in the problem Hamiltonian. RetNet and Transformer estimates require the number n_{b} of blocks and the internal dimension d_{m}.

Figure 1 depicts a general autoregressive NQS ansatz, highlighting both modulus and phase networks. The three ansatze examined in this work all follow this form, using a simple feedforward network to model the phase, which is joined by a sequence of autoregressive decoder blocks modeling the conditional probabilities. They differ only in the type of decoder used: in this paper, we consider the MADE [30], RetNet [17], and transformer [29] architectures. Reserving a more detailed description of them for the appendix, we emphasize here that each model enforces the autoregressive constraint on its outputs in a distinct way. Fundamentally, MADE [9] follows a basic feedforward architecture; a fixed binary mask is applied to the model weights in each forward pass, severing connections between neurons so as to ensure the autoregressive property. This makes MADE easy to sample from and train in a parallelizable way, at the expense of making it a fundamentally less expressive architecture relative to model size.

Transformers [27] calculate all pairwise dot product similarities within each input sequence. Attention can be made autoregressive through input masking, and is highly parallelizable, though at the expense of introducing a quadratic computational cost with respect to input sequence length. RetNets [25] are recurrent architectures, which are innately autoregressive and perform forward passes in linear time. RetNets compress into a mathematically equivalent parallel form analogous to the transformer; utilizing these two forms during inference and training respectively allows RetNet to utilize both the time complexity advantage of recurrent networks and the greater ease of training of transformers. We

describe in table 1 leading term estimates for the FLOP count required by each of these models to perform a single forward pass on a single input sequence, from which it is possible to extrapolate FLOP count estimations for an entire NQS training run. We provide detailed derivations of these estimates in Appendix A.4.

4 Methodology

We introduce a general framework for FLOP count estimation to quantify the computational cost of NQS. Drawing on the Chinchilla laws [12], we present a parametric scaling law for NQS and describe the specific experimental setup of our analysis.

4.1 FLOP Estimates for NQS

Typically, supervised learning only requires one forward and one backward pass through the model for each batch sample at each gradient update step. NQS, however, does not have training data, instead using the model to first generate a batch of sample spins from $x \sim |\psi_\theta|^2$, and then calculate local energies. For each sample x, this requires not only one forward pass to obtain $\langle x|\psi_\theta \rangle$, but many more to calculate the numerator $\langle x|H|\psi_\theta \rangle$. Each term P_i in (1) defines a specific bit flip sequence: applying this flip to x yields the index x' of the term $\langle x'|\psi_\theta \rangle$ that corresponds with a nonzero entry of P_i. Each bit flip adds an additional forward pass to the compute cost. Before training, all unique bit flip sequences are identified between the terms of H, eliminating redundant forward passes [29]. Nonetheless, a significant number of forward passes remains, though none of them contribute to the gradient calculation, yielding some compute cost savings depending on the ansatz.

For an NQS problem, we estimate the training FLOP count based on the number $n = 2n_{\text{seq}}$ of qubits, the average unique sample batch size B at each iteration, the single-token FLOP count F_{mod} of the modulus network, the single-sequence FLOP count F_{ph} of the phase network, the number M of unique bit flip patterns within H, and the training steps T. We estimate the sampling FLOP count as

$$\text{FLOP}_{\text{Sample}} \approx F_{\text{mod}}T \left(\sum_{m=0}^{\lfloor \log_4 B \rfloor - 1} 4^m + B \left(\frac{n}{2} - \lfloor \log_4 B \rfloor \right) \right) \tag{7}$$

For ease of approximation, we assume at each step the sampling process starts from the four configurations of a single orbital and branches into B unique samples in approximately $\log_4(B)$ steps. The remaining steps are needed to identify the full spin configuration for each sample. This assumption is an overestimate, since the sampling process is less costly if it does not branch this quickly. Then, the FLOP count to construct and backpropagate through (5) is approximately

$$\text{FLOP}_{\text{Loss}} \approx BT \left((M + 3) \left(\frac{n}{2} F_{\text{mod}} + F_{\text{ph}} \right) \right) \tag{8}$$

For each sample at each time step, (8) covers M forward passes for the local energy numerator, one for the denominator, and then backpropagation—approximately equivalent to two forward passes. We ignore the FLOP count for assembling the model outputs into loss values as they contribute no leading terms to the overall count. We also note that F_{mod} is implicitly context-dependent even within eqs. (7) and (8); for each ansatz, F_{mod} may vary based on the corresponding phase of training.

From derivations provided in section A.4, we arrive at the ansatz-specific FLOP count estimates for MADE, RetNet, and transformer shown in table 1. For the latter two ansatze, we set $d_{\text{attn}} = d_{\text{m}}$. In all cases, we consider separately the number of parameters $N = N_{\text{mod}} + N_{\text{ph}}$ for the modulus and phase networks, respectively, ignoring all non-leading terms. KV caching can reduce some of the transformer compute cost, for a considerable increase in memory. Accomplishing this cost savings would also require cache sharing between separate forward passes to an atypical extent. Though this paper performs FLOP-constrained scaling law analysis using rougher estimates given in table 4, which have been simplified for ease of analysis, we introduce these three estimate formulas here as general resources for assessing the compute cost of the MADE, transformer, and RetNet ansatze.

4.2 Parametric Scaling Law for NQS

We now derive NQS scaling laws to identify predictive relationships between accuracy/performance and several key attributes of NQS: model size, compute budget, problem size, and the number of unique samples used for training. In practice, the search space grows too quickly with problem size to allow for sampling from all feasible spins under a fixed compute budget, so we truncate the number of unique samples generated at each sampling run. Since these samples contribute to the loss function analogously to training data, the total number obtained in training serves as a proxy for the number of tokens used in traditional scaling law analysis.

Appendix A.6 gives a brief overview of the Chinchilla scaling law [12] establishing basic power law relationships between the final training loss L, model size N, and number D of tokens used. To adapt this curve to NQS, we must consider that any scaling analysis predicting the general behavior of an ansatz across a variety electronic ground state calculations must account for performance differences caused by training on different molecules. Increasing problem size, while keeping all other hyperparameters equal, is expected to both increase computational cost and decrease accuracy for any ansatz. To help the scaling law account for variation in problem size, we propose incorporating, in addition to the number N of model parameters (per thousand), a new term SF equaling the average fraction of the total search space that is sampled throughout training. For a molecular system with n spin-orbitals, N_E electrons, and a singlet ground state—where the electrons are equally divided among the up and down spins—the search space size is $\binom{0.5n}{0.5N_E}^2$.

This calculation is analogous for molecules with nonzero spin, though all molecules used for analysis in section 4.3 have either singlet or triplet ground states. For a given NQS run, we then obtain SF by dividing the average number B of unique samples generated at each iteration by the search space size. Since SF incorporates information about the number of unique samples seen at each optimization step, we multiply it by the number T of optimization steps to give

$$D' = T \times SF$$

a size-controlled analog to the variable D from [12].

Since different molecules have different energies, we modify the loss $L(N, D')$: absolute error between the NQS result and the ground truth, obtainable for moderate problem sizes via full configuration interaction (FCI), is a suitable measure since an error tolerance within 0.00159 Hartree is a widely accepted standard of chemical accuracy. Unfortunately, absolute error is not viable at scale since FCI calculations quickly become intractable. As such, we also consider the V-Score [28], a recent performance metric for assessing variational Hamiltonian eigensolvers. For an n-qubit Hamiltonian H and ansatz $|\psi_\theta\rangle$, the V-Score is given by

$$\text{V-Score} = n \frac{\langle\psi_\theta|H^2|\psi_\theta\rangle - \langle\psi_\theta|H|\psi_\theta\rangle^2}{\left(\langle\psi_\theta|H|\psi_\theta\rangle - \omega_0\right)^2} \tag{9}$$

where ω_0 is the weight value from (1) corresponding with identity operator. Both (2) and its variance—the numerator in (9)—may be estimated by the sample mean and variance of (5). V-Score was developed as a proxy measure for relative error, and it is both dimensionless and invariant with respect to global energy shifts. These properties make V-Score a natural candidate for scaling law analysis. Altogether, we consider two modifications of the Chinchilla curve for assessing the performance of NQS models for electronic ground state calculations:

$$\text{Error}_{\text{Abs}}(N, D') = A_0 + \frac{A_1}{N^{\alpha_1}} + \frac{A_2}{D'^{\alpha_2}}, \quad \text{V-Score}(N, D') = A_0 + \frac{A_1}{N^{\alpha_1}} + \frac{A_2}{D'^{\alpha_2}} \tag{10}$$

4.3 Experimental Setup

We apply each ansatz to each of the molecules in table 2, performing a coarse ablation varying model size—both modulus and phase networks—optimization steps, and the maximum number of unique samples generated per step. We perform all training with Adam, annealing the learning rate along a cosine schedule from 2.5×10^{-3} to 5×10^{-8}, with a linear warmup over the first 4% of steps. Following the same practice from [17], we exponentially increase the number of non-unique samples at each step from 10^4 to 10^{12}, and for the first 90% of training, we only recompute the loss every 10

Molecule	n	N_E	Determinants	Paulis	M	FCI
H_2O	14	10	441	1,390	190	-75.0155
N_2	20	14	14,400	2,591	446	-107.6602
O_2	20	16	2,025	2,879	530	-147.7502
H_2S	22	18	3,025	9,558	1,524	-394.3546
PH_3	24	18	48,400	24,369	4,333	-338.6984
LiCl	28	20	1,002,001	24,255	4,508	-460.8496
Li_2O	30	14	41,409,225	20,558	3,810	-87.8927

Table 2: Molecules whose second quantized (in the the STO-3G basis set) Hamiltonians were used for analysis. We list the corresponding number of spin-orbitals, electrons, Slater determinants in the search space, Pauli basis terms, and unique bit flip patterns among the Pauli terms. FCI ground truth values from [29] are also shown.
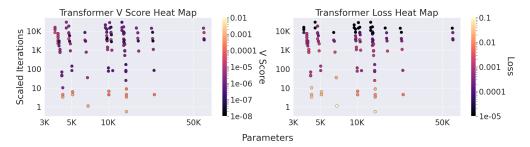

Figure 2: Heat maps showing the final V-score and loss values for each data point collected for the transformer ansatz, as a function of both model parameters N and scaled iterations D'. Since the variance and loss values comprising the V-score are stochastic estimates, we depict 10^{-8} as the lower limit of V-score accuracy, while absolute loss values are truncated at 10^{-5} to reflect desired thresholds of chemical accuracy.

steps. We also incorporate variational neural annealing (VNA) [11] to improve accuracy. We vary the number of training steps between 12,500 and 50,000, the model dimensions of RetNet and transformer between 8 and 64—the internal feedforward dimension always equals 4x this dimension—and the hidden state dimensions of the MADE and phase networks between 8 and 128. For both RetNet and transformer, we utilized 1 or 2 decoder blocks. The maximum number of unique samples generated at each step was varied between 1,000 and 16,000.

We follow [17] in choosing the final loss value of each training run. We fit Eq. (10) to our data as in [12], replacing zero absolute error values with 10^{-5} to avoid issues with the log-loss and using Adam alongside a cosine-annealed learning rate for 50,000 steps, with initial parameters taken from ranges given for the original Chinchilla law [4, 12].

5 Results

We collected 335 datapoints applying the three ansatze to the problems in table 2: 120 for MADE, 109 for RetNet, and 106 for transformer. We observe in fig. 2 the distribution of data points collected for transformer, to which those of MADE and RetNet are qualitatively similar and given in section A.7. These heat maps display the V-scores and absolute loss values as functions of the variables N and D' as defined in section 4.2: the figures qualitatively indicate a correlation between N, D', and both dependent variables. Similar trends are observable within the data sets for both the tranformer and RetNet ansatze. Reserving some discussion of these other models for the appendix, we emphasize how similar the RetNet behavior in fig. 5 is to that of the transformer shown here. In general, the data point distributions of the two networks are directly comparable, since RetNet and transformer ansatze of equivalent dimension have nearly identical numbers of parameters; likewise, both models train similarly under equivalent setups [17].

In both cases, we observe a pronounced stratification of the data point distribution with respect to N. A similar stratification is visible in fig. 4, but that data set succeeds in covering a wider range of parameter sizes. This difference is due to the fact that the architecture of RetNets and transformers is
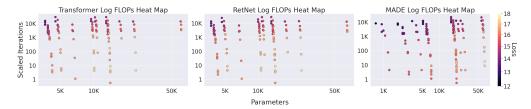
Figure 3: Heat maps showing the logarithm of the FLOP count estimates for all data points collected with each of the three ansatze. In all three cases, FLOP count does not decrease monotonically with respect to scaled iterations, indicating the latter is not merely a proxy for problem size.

less explicitly dependent on problem size than that of MADE. Furthermore, in testing, we maintained a fixed ratio of 8 hidden dimensions per attention/retention head inside each transformer and RetNet.

V-score	A_0	A_1	A_2	α_1	α_2	Log-R^2
MADE	9.37×10^{-11}	2.58×10^{-5}	5.53×10^{-2}	1.459	2.828	0.68
RetNet	6.71×10^{-7}	0.626	4.39×10^{-3}	9.370	2.451	0.40
Transformer	1.34×10^{-6}	0.197	7.25×10^{-3}	7.952	2.371	0.53

Absolute Error	A_0	A_1	A_2	α_1	α_2	Log-R^2
MADE	4.12×10^{-9}	0.033	0.106	2.224	0.757	0.65
RetNet	1.13×10^{-4}	5.26×10^{-4}	0.111	0.452	1.179	0.49
Transformer	2.83×10^{-9}	0.720	0.039	5.274	0.637	0.62

Table 3: Parametric curve parameters for MADE, RetNet, and transformer NQS ansatze, for both V-score and Loss data. R^2 are given on a log–log scale between model outputs and true loss values.

Following the training procedure from section 4.3, we have identified parametric curve coefficients, shown in table 3 for (10) for all three ansatze. These coefficients indicate that model size makes a greater impact, as represented by A_1 and α_1, on V-score for the transformer and RetNet models, although this preference is not as strong for the transformer. In contrast, model size appears to make a nearly negligible impact for MADE, and here we should rather focus on increasing training time as a means of improving performance. The transformer ansatz exhibits scaling behavior that is more similar to RetNet, preferring an increase in model size to an increase in training time.

Looking now to the coefficients for the absolute error values, we immediately observe that the baseline coefficient A_0 is smaller than the predetermined error tolerance for both MADE and transformer. This observation alone is not alarming, as all datasets contain several problem instances where the model achieved the exact tolerance. For the RetNet, a higher baseline coefficient may indicate the model is fundamentally less accurate than the other two, but such a distinction is not practically meaningful since the baseline still lies within chemical accuracy. Furthermore, these curves indicate that model size impacts final accuracy less for the RetNet, so one should focus on increasing training time; in contrast, it appears more advantageous to increase model size for the transformer.

5.1 FLOP Analysis

In fig. 3, we show FLOP estimates, on a logarithmic scale, for each of the entries in our three datasets. These estimates come from the equations outlined in section 4.1, and provide a coarse, yet still meaningful, understanding of the compute time needed to produce these results. Qualitatively, we observe for all three ansatze that when controlling for scaled training time D', an increase in model size N correlates with an increase in compute time, which makes intuitive sense.

Conversely, D' scales inversely with compute requirements; this observation is also intuitive, since larger problem sizes and search spaces should obviously correlate with larger compute costs: for a fixed training time, a larger problem size will generally incur a greater compute cost and lower D'. As a result, one may consider the argument that D' only serves to predict the V-score and absolute error values in our data to the extent that smaller problems are easier to learn than larger ones. As a potential rebuttal, and an indication that D' may reveal something more fundamental, we observe in fig. 3 for all ansatze that when keeping N fixed, an increase in D' appears to first correspond with an

increase in FLOPs before the general decrease: a similar reversal does not appear in either the V-score or absolute error data. This phenomenon indicates that the correlation between D' and FLOPs is likely weaker than that between D' and our two loss metrics, lending credence to the hypothesis that D' is truly a size-agnostic measure of training time.

5.2 Compute Constrained Scaling Analysis

Ansatz	Simplified Training FLOP Estimate
MADE	$3MSD'N$
RetNet	$\frac{1}{13}\left(15.5Mn + \frac{3n^2}{d_{\mathrm{m}}}\right)SD'N$
Transformer	$M\left(n + \frac{n^2}{12d_{\mathrm{m}}}\right)SD'N$

Table 4: Simplified FLOP estimates for constrained scaling law optimization. These are derived from the estimates given in table 1 and are intended to prioritize ease of computation at the expense of a decrease in accuracy. Here S refers to the search space size of the NQS problem.

The FLOP estimates in table 1, while already ignoring many non-leading contributions, are meant to reflect the primary contributions of all relevant factors of an NQS training run on compute cost. These complicated formulas contrast with the simple $6ND$ cost formula assumed by [15] for language modeling, which allows for a simple constrained optimization comparing performance relative to compute cost. To perform a similar analysis here, we introduce further simplified compute estimates in table 4, with detailed discussion of their derivation in section A.5. Future analyses that consider N_{mod} and N_{ph} as separate variables may benefit from the more accurate estimates of table 1.

$D' =$	MADE	RetNet	Transformer
V-score	$0.053 \times N^{0.516}$	$13.076 \times N^{3.823}$	$6.707 \times N^{3.354}$
Absolute Error	$0.889 \times N^{2.938}$	$0.005 \times N^{0.383}$	$2663.825 \times N^{8.279}$

Table 5: The empirical coefficients in table 3 and estimates in table 4 together define curves tracking the growth of D', with respect to N, along the efficient frontier determined by compute budget C.

Derivations made in section A.8 indicate that, for compute budget estimates of the form $C = kD'N$ for some problem-defined k, such as those given in table 4, the constrained optimization of (10) is entirely analogous to the process carried out in [12]. Consequently, we give in table 5 the optimal growth rate of D' as a function of N, for each loss metric and ansatz considered. These growth rates are highly metric and ansatz dependent, and more importantly, do not reflect the approximately linear curves identified in the Chinchilla scaling laws.

6 Conclusion

Our work applies Chinchilla-style LLM scaling laws, incorporating a new training time variable dependent on problem size, to MADE, transformer, and RetNet NQS ansatze in order to assess their performance on a selection of quantum chemistry problems of varying size. Using an ablation dataset, we fit this curve to predict both absolute error and V-score loss metrics for each ansatz. Together with a novel framework for estimating the compute cost of NQS, these results demonstrate how the performance of NQS at scale may be analyzed similarly to LLMs.

This line of research invites a more comprehensive ablation study, with greater variety of architectures, problem sizes and training setups. For instance, the model size N in (10) comprises both modulus and phase networks, but it may be illuminating to isolate the phase network's role. Varying the total number of non-unique Monte Carlo samples may also impact accuracy. A technique introduced in [29] efficiently approximates local energy calculations by neglecting Hamiltonian terms associated with unsampled states, and more recent work [19] demonstrates the extent of its potential. Analyzing the scaling properties of NQS under this new variation would help provide insight into the extent of its benefits, especially when considering NQS as a problem-agnostic method of dequantization [16].

Acknowledgments

References

[1] Thomas D Barrett, Aleksei Malyshev, and AI Lvovsky. Autoregressive neural-network wavefunctions for ab initio quantum chemistry. *Nature Machine Intelligence*, 4(4):351–358, 2022.

[2] Rodney J Bartlett and Monika Musiał. Coupled-cluster theory in quantum chemistry. *Reviews of Modern Physics*, 79(1):291–352, 2007.

[3] Elizabeth Bennewitz, Florian Hopfmueller, Bohdan Kulchytskyy, Juan Carrasquilla, and Pooya Ronagh. Neural error mitigation of near-term quantum simulations. *Nature Machine Intelligence*, 4(7):618–624, Jul 2022.

[4] Tamay Besiroglu, Ege Erdil, Matthew Barnett, and Josh You. Chinchilla scaling: A replication attempt. *arXiv preprint arXiv: 2404.10102*, 2024.

[5] Max Born and Robert Oppenheimer. Zur quantentheorie der molekeln. *Annalen der Physik*, 389(20):457–484, 1927.

[6] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

[7] Kenny Choo, Antonio Mezzacapo, and Giuseppe Carleo. Fermionic neural-network states for ab-initio electronic structure. *Nature Communications*, 11(1), May 2020.

[8] Yunzhen Feng, Elvis Dohmatob, Pu Yang, Francois Charton, and Julia Kempe. A tale of tails: Model collapse as a change of scaling laws. In *ICLR 2024 Workshop on Navigating and Addressing Data Problems for Foundation Models*, 2024.

[9] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International conference on machine learning*, pages 881–889. PMLR, 2015.

[10] Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer. *arXiv journal arXiv:2102.01293*, 2021.

[11] Mohamed Hibat-Allah, Estelle M Inack, Roeland Wiersema, Roger G Melko, and Juan Carrasquilla. Variational neural annealing. *Nature Machine Intelligence*, 3(11):952–961, 2021.

[12] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[13] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

[14] Berivan Isik, Natalia Ponomareva, Hussein Hazimeh, Dimitris Paparas, Sergei Vassilvitskii, and Sanmi Koyejo. Scaling laws for downstream task performance of large language models. *arXiv preprint arXiv:2402.04177*, 2024.

[15] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[16] Oliver Knitter, James Stokes, and Shravan Veerapaneni. Toward neural network simulation of variational quantum algorithms. *NeurIPS workshop on AI for Science: Progress and Promises*, 2022.

[17] Oliver Knitter, Dan Zhao, James Stokes, Martin Ganahl, Stefan Leichenauer, and Shravan Veerapaneni. Retentive neural quantum states: Efficient ansätze for ab initio quantum chemistry. *Machine Learning: Science and Technology*, 2025.

[18] Sheng-Hsuan Lin and Frank Pollmann. Scaling of neural-network quantum states for time evolution. *physica status solidi (b)*, 259(5):2100172, 2022.

[19] Aleksei Malyshev, Markus Schmitt, and A. I. Lvovsky. Neural quantum states and peaked molecular wave functions: Curse or blessing?, 2024.

[20] Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C Benjamin, and Xiao Yuan. Quantum computational chemistry. *Reviews of Modern Physics*, 92(1):015003, 2020.

[21] Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models. *arXiv preprint arXiv:2305.16264*, 2023.

[22] Nikhil Sardana, Jacob Portes, Sasha Doubov, and Jonathan Frankle. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws. In *Forty-first International Conference on Machine Learning*, 2024.

[23] Ulrich Schollwöck. The density-matrix renormalization group. *Reviews of modern physics*, 77(1):259–315, 2005.

[24] Or Sharir, Yoav Levine, Noam Wies, Giuseppe Carleo, and Amnon Shashua. Deep autoregressive models for the efficient variational simulation of many-body quantum systems. *Physical review letters*, 124(2):020503, 2020.

[25] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

[26] Yi Tay, Mostafa Dehghani, Samira Abnar, Hyung Won Chung, William Fedus, Jinfeng Rao, Sharan Narang, Vinh Q. Tran, Dani Yogatama, and Donald Metzler. Scaling laws vs model architectures: How does inductive bias influence scaling? In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[28] Dian Wu, Riccardo Rossi, Filippo Vicentini, Nikita Astrakhantsev, Federico Becca, Xiaodong Cao, Juan Carrasquilla, Francesco Ferrari, Antoine Georges, Mohamed Hibat-Allah, Masatoshi Imada, Andreas M. Läuchli, Guglielmo Mazzola, Antonio Mezzacapo, Andrew Millis, Javier Robledo Moreno, Titus Neupert, Yusuke Nomura, Jannes Nys, Olivier Parcollet, Rico Pohle, Imelda Romero, Michael Schmid, J. Maxwell Silvester, Sandro Sorella, Luca F. Tocchio, Lei Wang, Steven R. White, Alexander Wietek, Qi Yang, Yiqi Yang, Shiwei Zhang, and Giuseppe Carleo. Variational benchmarks for quantum many-body problems. *Science*, 386(6719):296–301, 2024.

[29] Yangjun Wu, Chu Guo, Yi Fan, Pengyu Zhou, and Honghui Shang. Nnqs-transformer: an efficient and scalable neural network quantum states approach for ab initio quantum chemistry. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–13, 2023.

[30] Tianchen Zhao, James Stokes, and Shravan Veerapaneni. Scalable neural quantum states architecture for quantum chemistry. *Machine Learning: Science and Technology*, 4(2):025034, jun 2023.

A Appendix

A.1 The MADE Ansatz

The first ansatz we analyze [30] utilizes MADE [9], a feedforward network designed as an autoencoder for probability distributions, as the decoder. MADE applies binary masks to its weights, severing key neural connections between the hidden layers to ensure that its outputs obey the autoregressive property. MADE does not require any feature embedding or positional encoding of the input spins, and is the simplest of the three architectures being examined. MADE was not the first autoregressive ansatz applied to electronic ground state calculations, but it serves as a precursor for the more sophisticated architectures discussed.

FLOP estimates for a MADE network are simple: a feedforward network with N parameters requires approximately $2N$ FLOPs to perform a forward pass on a single spin configuration, since most model weights correspond with one multiplication and one addition operation inside each forward pass; biases only correspond with one addition operation, but are heavily outnumbered among the total parameter count. We ignore the contribution of nonlinearities to the compute cost, alongside other non-leading FLOP contributions, throughout this work. These count estimates hold true for a MADE network, but the binary masking of model weights introduces an additional FLOP for each weight, so MADE requires $3N$ FLOPs to perform a forward pass on a single spin configuration.

A.2 NQS–Transformer

The transformer is a ubiquitous architecture that underpins many large language models. They process input sequences in parallel and generate autoregressive output, making them well-suited to learn the conditional probabilities of an NQS ansatz [3, 29]. Each transformer block comprises an attention block and a feedforward layer, interspersed with residual connections and layer normalizations. Attention forms the the key input processing apparatus, calculating a weighted context vector for each input token based on all pairwise contextual relationships within the input sequence. Tokens are embedded based on value and relative position, then projected to obtain batches of query, key, and value vectors, which are used to construct the context vectors:

$$\text{Attention}(X) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \text{ where } Q = XW_Q, \ K = XW_K, \ V = XW_V \quad (11)$$

Attention can easily learn correlations between tokens separated by significant relative distance, and a transformer typically utilizes several attention heads simultaneously—known as *multi-head attention*—to learn different types of contextual information [27]. Appropriate input masking during training and inference makes attention autoregressive. Being so naturally expressive, transformers have been shown to perform well as NQS ansatze [29]. In this context, the transformer operates on input sequences of spatial orbital occupancies. Each spatial orbital can contain two electrons, so these sequences have length $n/2$ for an n-qubit NQS problem.

FLOP estimates [15] indicate that for a transformer with N total parameters, n_b blocks, and attention dimension d_attn, a forward pass across n_seq tokens requires approximately $2N + 4n_\text{b}n_\text{seq}d_\text{attn}$ operations per token. FLOP estimates made for LLMs typically truncate this value to $2N$, but for NQS, the typically smaller ratio of model-to-problem size, coupled with the need for multiple forward passes to generate local energy values, necessitates considering the full FLOP estimate [16]. We emphasize here that the presence of n_seq inside this estimate implies that the transformer requires $O(n_\text{seq}^2)$ time to perform a forward pass an entire sequence.

A.3 RetNet NQS

The Retentive Network (RetNet) is a recurrent language model that compresses into an equivalent parallel form [25], which processes inputs and calculate gradients through backpropagation similarly to transformers. In this format, the RetNet block is essentially analogous to the transformer, except that multi-head attention is replaced by multi-scale retention, which compresses the repeated action of a linear recurrent model onto the embedded inputs:

$$\text{Retention}(X) = \left(QK^T \odot D\right)V, \text{ where } Q = (XW_Q)\odot\Theta, \ K = (XW_K)\odot\overline{\Theta}, \ V = XW_V. \tag{12}$$

Under retention, embedded tokens are still projected to queries, keys, and values, but the retention head itself encodes relative positional information Θ, along with applying its own weighted autoregressive mask D. These matrices are defined by

$$\Theta_j = e^{ij\theta}, \quad D_{jk} = \begin{cases} \gamma^{j-k}, & j \geq k \\ 0, & j < k \end{cases},$$ (13)

where θ is a real parameter vector and γ is a positive scalar. Different retention heads have different, fixed γ values to capture different types of contextual information. Consequently, retention heads are followed by group, not layer, normalization. RetNets train in parallel form but they perform retention recurrently during inference. Taking Q, K, and V as in (12), retention can be expressed as a recurrence on the rows of these three matrices:

$$S_t = \gamma S_{t-1} + (K_n)^T V_t, \; S_0 = 0$$
$$\text{Retention}(X_t) = Q_t S_t$$ (14)

Thus RetNets perform inference in $O(n_{\text{seq}})$ time, unlike the transformer's $O(n_{\text{seq}}^2)$. For a RetNet with $2N$ total parameters, n_b decoder blocks, retention dimension d_{attn}, and sequence length n_{seq}, the parallel RetNet requires $2N + 4n_b n_{\text{sec}} d_{\text{attn}}$ operations per token—same as transformer—to perform a forward pass. Recurrent RetNet, in contrast, requires $2N + 5n_b (d_{\text{attn}})^2$ operations per token [17]. While RetNet contains a few more parameters than an equivalent transformer, these results indicate that a recurrent RetNet scales more favorably for problems where $n_{\text{seq}} > 1.75 d_{\text{attn}}$. Though fundamentally less expressive than transformers, RetNets have shown comparable performance on key language modeling baselines [25]. Since NQS training incorporates output from multiple training and inference forward passes, its computational cost is dominated by these forward passes: the dual form of the RetNet makes it useful for improving compute efficiency of NQS, and RetNet ansatze are amenable to solving electronic ground state problems [17].

A.4 Deriving FLOP Estimates for MADE, RetNet, and Transformer Ansätze

Following the principles of estimation outlined in section 4.1, we now derive the different total FLOP count estimates shown in table 1. The MADE ansatz is simple to estimate, since it requires a full forward pass through the entire input sequence at all stages of training. Its FLOP estimate is given by

$$\text{FLOP}_M \approx 3N_{\text{mod}}T\left(\sum_{m=0}^{\lfloor \log_4 B \rfloor - 1} 4^m + B\left(\frac{n}{2} - \lfloor \log_4 B \rfloor\right)\right) + BT\left((M+3)(3N_{\text{mod}} + 2N_{\text{ph}})\right)$$

$$\approx 3N_{\text{mod}}T\left(\frac{B}{3} + B\left(\frac{n}{2} - \log B\right)\right) + BT\left((3M+9)(N_{\text{mod}} + 2(M+3)N_{\text{ph}})\right)$$

$$\approx BT\left(\left(\frac{3n+2}{2} + 3M + 9 - 3\log B\right)N_{\text{mod}} + 2(M+3)N_{\text{ph}}\right).$$ (15)

We now proceed with RetNets, as in contrast with transformers, calculating the total FLOP estimate remains relatively straightforward: we use the recurrent form of RetNet for (7) and the M passes in (8), and the parallel form for the rest, which gives us

$$\text{FLOP}_R \approx BT\left(\frac{1}{3} - \log(B) + \frac{(M+1)n}{2}\right)F_{\text{rec}} + BT\left((M+3)F_{\text{ph}} + \frac{3n}{2}F_{\text{par}}\right)$$

$$\approx BT\left((2N_{\text{mod}} + 5d_{\text{m}}^2 n_b)\left(\frac{(M+1)n}{2} - \log(B)\right)\right)$$
$$\quad + BT\left((2(M+3)N_{\text{ph}} + 3n(N_{\text{mod}} + nd_{\text{m}}n_b))\right)$$
$$\approx BT\left(((M+4)n - 2\log B)N_{\text{mod}} + 2(M+3)N_{\text{ph}}\right)$$
$$\quad + BT\left(n_b d_{\text{m}}\left(2.5 d_{\text{m}}((M+1)n - 2\log B) + 3n^2\right)\right).$$ (16)

Estimating the FLOPs of the transformer ansatz requires consideration of a few different factors. The contribution of the phase network remains unchanged, but unlike RetNets, F_{mod} implicitly depends

on n during sampling, though we may mitigate this effect to an extent by excluding dummy tokens from forward passes. In this case, we must make adjustments to the expression of (7) and get

$$\mathrm{FLOP_{S(T)}} \approx T \left(\sum_{m=0}^{\lfloor \log_4 B \rfloor - 1} (m+1)4^m \left(2N_{\mathrm{mod}} + 4(m+1)n_{\mathrm{b}}d_{\mathrm{m}} \right) + B \sum_{m=\lfloor \log_4 B \rfloor + 1}^{n/2} \left(2N_{\mathrm{mod}} + 4mn_{\mathrm{b}}d_{\mathrm{m}} \right) \right)$$

$$\approx T \left(2N_{\mathrm{mod}} \sum_{m=0}^{\lfloor \log_4 B \rfloor - 1} (m+1)4^m + n_{\mathrm{b}}d_{\mathrm{m}} \sum_{m=1}^{\lfloor \log_4 B \rfloor} m^2 4^m \right)$$

$$+ T \left(BN_{\mathrm{mod}} (n - 2\log B) + 4Bn_{\mathrm{b}}d_{\mathrm{m}} \sum_{m=\lfloor \log_4 B \rfloor + 1}^{n/2} m \right)$$

$$\approx BT \left(\left(n - \frac{4}{3}\log B - \frac{2}{9} \right) N_{\mathrm{mod}} + n_{\mathrm{b}}d_{\mathrm{m}} \left(\frac{n^2}{2} + \frac{20}{27} - \frac{8}{9}\log B - \frac{2}{3}(\log B)^2 \right) \right). \tag{17}$$

Estimating the loss count gives

$$\mathrm{FLOP_{L(T)}} \approx BT \left((M+3)nN_{\mathrm{mod}} + 2(M+3)N_{\mathrm{ph}} + n_{\mathrm{b}}d_{\mathrm{m}}(M+3)n^2 \right), \tag{18}$$

and through combining these two estimates, we obtain the transformer FLOP count estimate

$$\mathrm{FLOP_T} \approx BT \left(\left((M+4)n - \frac{4}{3}\log B \right) N_{\mathrm{mod}} + 2(M+3)N_{\mathrm{ph}} \right)$$

$$+ BT \left(n_{\mathrm{b}}d_{\mathrm{m}} \left((M+3.5)n^2 - \frac{8}{9}\log B - \frac{2}{3}(\log B)^2 \right) \right). \tag{19}$$

A.5 Derivation of Simplified FLOP Estimates

We now describe the simplification process yielding the rougher FLOP estimates given in table 4. Traditional LLM compute-constrained scaling law analysis presumes an approximate FLOP count of $6ND$ [15], where N is the model parameter count and D is the number of tokens used from training. This estimate is quite rough, based on the presumption that while the single-token forward pass FLOP count of a transformer is technically greater than $2N$, this difference is not significant in practice.

The NQS FLOP estimates in table 1 are far more elaborate, incorporating several additional terms that dependent on problem size—the qubit number n, number M of unique bit flip patterns within the terms of the Hamiltonian, which is roughly $O(n^4)$, and the sample batch size B. Furthermore, these formulas do not yet explicitly depend on N and D', the variables used in the NQS parametric scaling law curve from section 4.2. Accordingly, we must modify the NQS FLOP estimates both to depend on these two variables explicitly and to be more amenable to the simple constrained optimization performed in scaling law analysis. We give a detailed description for modifying the Transformer ansatz FLOP estimate, but those for RetNet and MADE follow similar principles.

Drawing inspiration from the general principles that lead to the $6ND$ estimate in [15], we first remove all remaining non-leading terms in (19). We also remove all terms dependent on $\log B$, since B is bounded above by the size of the search space: the discussion of search space size in section 4.2 and a basic asymptotic analysis of binomial coefficients based on Stirling's approximation together indicate that $\log B$ is $O(n)$, hence it is not one of the primary leading terms in the estimate. We then have a reduced estimate

$$\mathrm{FLOP_T} \approx BT \left(nMN_{\mathrm{mod}} + 2MN_{\mathrm{ph}} + n_{\mathrm{b}}d_{\mathrm{m}} \left(Mn^2 \right) \right). \tag{20}$$

Looking back to [15], under the model constraints enforced in section 4.3, $N_{\mathrm{mod}} \approx 12n_{\mathrm{b}}d_{\mathrm{m}}^2$, simplifying (20) further to

$$\mathrm{FLOP_T} \approx BT \left(M \left(n + \frac{n^2}{12d_{\mathrm{m}}} \right) N_{\mathrm{mod}} + 2MN_{\mathrm{ph}} \right). \tag{21}$$
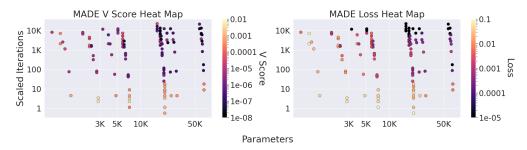
Figure 4: Heat maps showing V-score and absolute error for the MADE ansatz. We note that data points are far less stratified than in fig. 2

If we let S equal the size of the search space defined by the given NQS problem, and permit ourselves some further overestimate of the FLOP contribution from the phase network, then our reduced FLOP estimate becomes

$$\text{FLOP}_\text{T} \approx M \left(n + \frac{n^2}{12d_\text{m}} \right) SD'N. \tag{22}$$

A similar simplification process results in the other entries of table 4.

A.6 Brief Overview of Chinchilla Parametric Curve Fitting

A crucial aspect of assessing a model's performance with the Chinchilla scaling law analysis [12] involves fitting a parametric curve to a collection of data points representing separate training runs of the model: the curve predicts the final training loss value L attained by each instance as a function of model size N and number of tokens D processed by the model over the course of training. The contribution of each variable to the final loss is given via a power law functional form:

$$L(N, D) = A_0 + \frac{A_1}{N^{\alpha_1}} + \frac{A_2}{D^{\alpha_2}}, \tag{23}$$

with the final curve being determined through minimizing a Huber loss, taken with $\delta = 10^{-3}$, applied to the logarithms of the power law and target loss values and averaged across the entire dataset:

$$\min_{A_i, \alpha_i} \frac{1}{R} \sum_{j \in R} \text{Huber}_\delta \left(\log \left(A_0 + \frac{A_1}{N_j^{\alpha_1}} + \frac{A_2}{D_j^{\alpha_2}} \right) - \log L(N_j, D_j) \right) \tag{24}$$

Though the exact values of these regression parameters have been disputed [4] since the unveiling of the Chinchilla laws, the laws themselves remain a crucial foundation for scaling analysis in LLMs.

A.7 MADE and RetNet Result Discussion

We show in fig. 4 the equivalent heat maps for the MADE ansatz to those of the transformer from fig. 2. Observing that these data points are less stratified with respect to model size as can be seen for tranformer and RetNet, we emphasize here that the exact values of N come from both problem size and hidden layer dimension. This observation is technically true for all models discussed in this paper, but the effect is far less visible for the other models, which do not solely consists of feedforward components. The number of hidden layers was deliberately kept fixed and shallow to avoid introducing further sources of error as a result of the specified training hyperparameter configuration being insufficient to handle deeper networks. We do this during testing not only for both networks inside the MADE ansatz, but also for the phase networks, which are feedforward networks, of the RetNet and transformer ansatze.

In contrast, the data distribution shown in fig. 5 for the RetNet ansatz is almost identical to what can be seen in fig. 2, owing to the fact that RetNets and transformers are so architecturally similar. Of course, the exact V scores and absolute errors attained differ between the two models, but the general trends here corroborate the conclusion from [17] that the two models also train quite similarly as NQS ansatze. We can make a few additional general qualitative observations about the observed
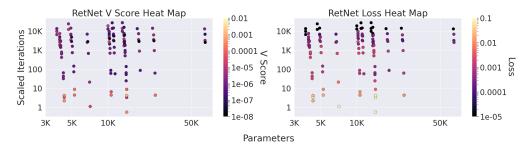
Figure 5: Heat maps showing V-score and absolute error values for the RetNet-based ansatz. We do note that the distribution of data points is more directly similar to fig. 2 than to what is shown in fig. 4.

data. For instance, it appears that, on average, a given combination of independent variables N and D' will yield a lower V-score with a RetNet ansatz than with a transformer, but at the same time, the transformer appears more likely to produce a lower absolute error than the RetNet. A similar comparison between the scaling performance of these architectures and the MADE ansatz is more difficult to make since MADE is a fundamentally different model, but one can observe in the data that RetNet and transformer ansatze show better performance, in terms of both V-score and absolute error, than the MADE ansatz using lower numbers of parameters.

A.8 Analysis of Compute-Constrained Scaling

Here we give, in moderate detail, the derivation of the efficient frontier obtained in [12], slightly modified for our purposes. Each of the FLOP estimates in table 4 is of the form $kD'N$, for some k specific to the choice of ansatz and the exact problem size. This allows for a similar constrained optimization to [12], allowing us to determine from the empirically obtained parametric curve constants the optimal training configuration relative to a fixed compute cost. Both parametric curves in (10) take the form

$$L(N, D') = A_0 + \frac{A_1}{N^{\alpha_1}} + \frac{A_2}{D'^{\alpha_2}} \tag{25}$$

for some loss function L. The method of Lagrange multipliers then says that for a fixed compute budget C, we must solve the system of equations

$$\alpha_1 A_1 N^{\alpha_1 - 1} = \lambda k D'$$
$$\alpha_2 A_2 D'^{\alpha_2 - 1} = \lambda k N$$
$$kD'N = C.$$

Substituting $D' = \frac{C}{kN}$ into the other two equations gives

$$\alpha_1 A_1 N^{\alpha_1} = \lambda C$$
$$\alpha_2 A_2 C^{\alpha_2 - 1} = \lambda k^{\alpha_2} N^{\alpha_2},$$

and substituting the first equation into the second then gives

$$\alpha_2 A_2 C^{\alpha_2} = \alpha_1 A_1 k^{\alpha_2} N^{\alpha_1 + \alpha_2}, \text{ hence } N = \left(\frac{\alpha_2 A_2}{\alpha_1 A_1}\right)^{\frac{1}{\alpha_1 + \alpha_2}} \left(\frac{C}{k}\right)^{\frac{\alpha_2}{\alpha_1 + \alpha_2}} \tag{26}$$

Likewise, substituting $N = \frac{C}{kD'}$ into the second equation gives $\alpha_2 A_2 D'^{\alpha_2} = \lambda C$, so

$$D' = \left(\frac{\alpha_1 A_1}{\alpha_2 A_2} N^{\alpha_1}\right)^{\frac{1}{\alpha_2}} = \left(\frac{\alpha_1 A_1}{\alpha_2 A_2}\right)^{\frac{1}{\alpha_1 + \alpha_2}} \left(\frac{C}{k}\right)^{\frac{\alpha_1}{\alpha_1 + \alpha_2}}. \tag{27}$$

These equations for the compute-optimal N and D' are clear analogs of those found in [12].

16