

Sampling Weights at Intermediate Temperatures is Optimal for Training Large Language Protein Models

L. Ghiringhelli 

Department of Physics, *Università degli Studi di Milano*, via Celoria 16, 20133 Milano, Italy

A. Zambon ^{*} and G. Tiana [†]

Department of Physics, *Università degli Studi di Milano* and *INFN*, via Celoria 16, 20133 Milano, Italy



(Received 31 March 2026; accepted 21 September 2026; published 14 October 2026)

Using a statistical mechanics framework, we investigate the parameter space of transformer models trained on protein sequence data. We sample the loss landscape at varying temperatures using Langevin dynamics to characterize the low-loss manifold and to understand the mechanisms underlying transformers' superior performance in protein structure prediction. We find that, at variance with networks not based on the attention mechanism, the lack of a first-order-like transition in the loss of the transformer produces a range of intermediate temperatures with good learning properties; this is true for both synthetic and natural protein sequences. We also show that the parameters of most layers are highly conserved at these temperatures if the dimension of the embedding is optimal, and we provide an operative way to find this dimension. Additionally, we show that the attention matrix is more predictive of the contact maps of the protein at higher temperatures and for higher dimensions of the embedding than those optimal for learning. Finally, we showed that the models sampled at intermediate temperatures can predict the free-energy variation upon mutation, better than models obtained through standard optimization techniques.

DOI: [10.1103/xvjl-qrwc](https://doi.org/10.1103/xvjl-qrwc)

I. INTRODUCTION

Large language models, based on the transformer architecture [1], have had a significant impact on a broad range of machine-learning tasks. In particular, they have affected the field of biophysics, enabling computers to predict the native conformation of proteins from the knowledge of their sequences alone [2,3]. Despite their significance, the comprehension of their intrinsic mechanisms remains significantly less understood than that of other types of feedforward networks.

In the case of simple feedforward networks, a large amount of information is now available about how they work, how they generalize from the training data, and how the space of solutions looks like [4–11].

Apart from the fact that it is more cumbersome than training feedforward networks with the same number of parameters but not based on the attention mechanism, less is known about the learning mechanism of transformers [12]. In one-layer systems, gradient optimization was shown to act in two stages, first learning causal relations by amplifying

attention weights between words that are distinctive in an example and suppressing those between common ones, and then refining the results logarithmically slowly [13,14].

Remarkably, it was proved analytically that a single transformer learns the couplings of a Potts model in a way that is equivalent to that obtained from pseudolikelihood learning [15]. Since Potts models describe well the coevolution of amino acids in proteins [16], transformers seem the ideal machine-learning tool to capture proximity relations in proteins.

As with the coevolution described by Potts models, transformers can be studied using the formalism of the canonical ensemble, thus treating them in the context of statistical mechanics. This approach has been shown to be effective in the study of simple feedforward networks. In these networks, the low-loss manifold has been found to be fully connected and to have a spiky shape [11].

In this work, we present results from sampling the parameters of a transformer to better understand how it works and why it efficiently retrieves spatial and chemical relationships between amino acids along the chain. We compare these results with those from a pooled feedforward architecture lacking an attention mechanism. We framed the problem of exploring the parameter space in terms of statistical mechanics to leverage the computational algorithms developed in this field, in particular, in connection with the canonical ensemble.

We sampled the space of parameters of the model with an algorithm based on the numerical solution of the Langevin equation, where the gradients are evaluated approximately on minibatches of training sequences to gain computational

^{*}Contact author: alessandro.zambon@unimi.it

[†]Contact author: guido.tiana@unimi.it

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

efficiency [17]. This algorithm relies on the control of the noise introduced by the approximation, adjusting in real time the masses and the friction coefficients of the Langevin equations to sample the canonical ensemble at the desired temperature. In this way, we combine the advantages of stochastic gradient descent, which is fast for optimization problems but does not sample specific probability distributions, and of Langevin equations, which samples Boltzmann's distribution but requires a lengthy calculation of the forces acting on the system.

Initially, we assigned a simple task to the network, which is learning a set of synthetic sequences that fold into a single structure. We chose this task because it is computationally efficient enough to collect statistics and draw robust conclusions about the system. This allows us to obtain information about how transformers function in general and their ability to predict protein structures. We then generalized our results by studying the same problem for datasets composed of sequences belonging to different protein families.

In the next sections, we show that transformers display an intermediate range of temperatures with optimal generalization properties, while pooled feedforward networks have a sharper transition from low to high temperatures that prevents this regime (Sec. III A for the case of synthetic sequences). Moreover, we demonstrate that the parameters of most layers are highly conserved among models sampled at the optimal temperature, provided that the embedding is properly dimensioned, and we suggest an operative way to find this ideal dimension (Sec. III B). Additionally, we show that the attention matrix is more predictive of the contact maps of the protein at temperatures and for dimensions of the embedding that are higher than those optimal for learning (Sec. III C). Finally, we obtain the same qualitative thermodynamic results for real sequence data and we demonstrate that, on average, sampling at intermediate temperatures can improve the generalization performance and the prediction of free-energy variation due to single-site mutations with respect to standard optimization techniques (Sec. III D).

II. DATASETS, MODELS, AND ALGORITHMS

A. The synthetic datasets

The task we addressed is a masked language modeling problem, for a synthetic dataset of amino acid sequences folding to the native structure of the acyl-coenzyme A binding protein (Fig. 1, pdb code: 2abd) [18].

The dataset was generated following the same strategy as in Ref. [19]: We defined the energy of a sequence $\mathbf{x}$ as

$$U(\mathbf{x}) = \sum_{ij} B_{x_i x_j} \Delta_{ij}, \quad (1)$$

where B_{xy} is a randomly generated symmetric interaction matrix between amino acids, extracted from a Gaussian distribution with zero average and unitary variance, while Δ_{ij} is a binary symmetric contact matrix, derived from the native conformation of the acyl-coenzyme A binding protein so that $\Delta_{ij} = 1$ if the corresponding amino acids x_i and x_j have a pair of heavy atoms closer than 3.5 Å, a standard threshold that accounts for atomic interactions. In the canonical ensemble

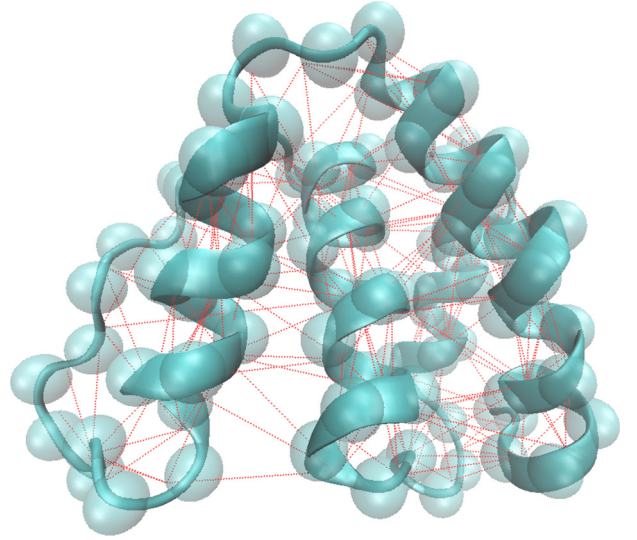


FIG. 1. The structure of acyl-coenzyme A binding protein (pdb code: 2abd), where the contacts between amino acids are indicated with dashed red lines.

framework, we generated 15 000 low-temperature ($T_s = 1.27$, in energy units; cf. Ref. [19]) sequences $\mathbf{x}$ of length $n = 86$ with an adaptive simulated tempering algorithm [20].

For each generated sequence $\mathbf{x}^\mu$, we selected approximately 15% of its amino acids and replaced them with a masking token. The true amino acids in the masked sites were gathered in a vector $\mathbf{y}^\mu$, which was kept fixed for each sequence. Then, we partitioned the whole dataset $\{(\mathbf{x}^\mu, \mathbf{y}^\mu)\}_{\mu=1}^{15\,000}$ into three subsets: train, validation, and test, composed of 1000, 1000, and 13 000 pairs, respectively.

B. The Pfam datasets

We addressed the same masked language modeling problem described in Sec. II A, using sequences belonging to ten different protein families, as defined in the Pfam database [21]. Each family was associated with a different dataset, where, as previously, we masked approximately 15% of the amino acids of each sequence and replaced them with a masking token. Each dataset was then partitioned into three subsets: train, validation, and test, corresponding to 70%, 15%, and 15% of the total, respectively. The generation process of each Pfam dataset, along with the list of the selected protein families, can be found in Appendix A.

C. The prediction models

We selected three different architectures to tackle the masked language modeling problems described in Secs. II A and II B. Two of these are transformers [1], made up of either one (TF₁) or four (TF₄) modules in series (upper panel in Fig. 2), and the third is a pooled feedforward network (FF) whose number of parameters (1.74×10^5) is approximately equal to that of the four-module transformer (1.69×10^5), and four times that of TF₁ (4.4×10^4 parameters).

All of these models follow a three-stage structure comprising an input construction stage, an encoder, and a final output projection layer. The input and output components

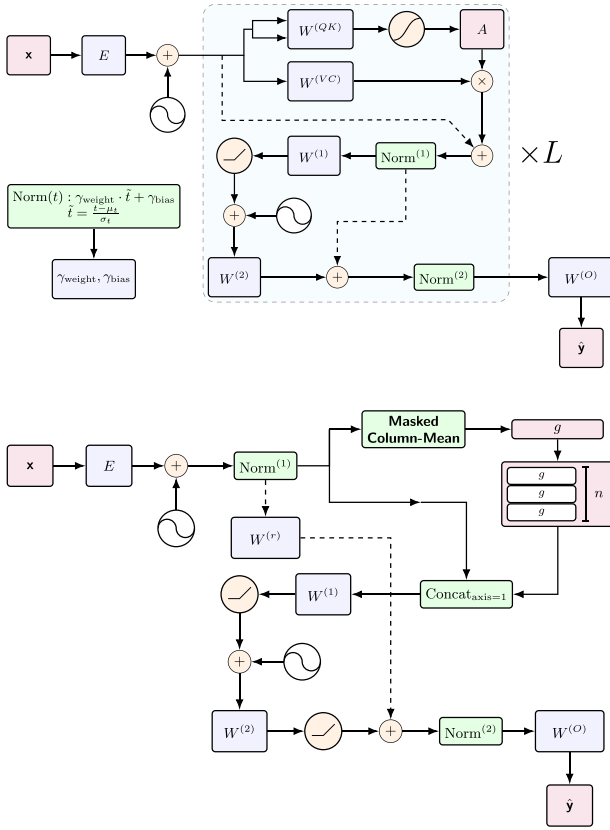


FIG. 2. The scheme of the L -module transformer (upper scheme) and of the pooled feedforward model (lower scheme) used in this work.

are the same for the two network families studied here, but the internal encoder differs between transformer-based and pooled feedforward networks.

The encoders of transformer-based models use the attention mechanism to efficiently capture long-range dependencies in the input and produce a context-aware representation of each input component.

In the pooled feedforward encoder, we modeled the architecture to resemble a transformer as closely as possible, replacing only the attention matrix with a pooling layer. This allows us to isolate the effects of the attention mechanism. Furthermore, each architecture is designed so that all possible permutation symmetries are eliminated without impairing the learning capabilities of the model. The detailed description of each neural network can be found in Appendix B.

For a given masked input, each model produces a matrix. Every line of the matrix represents the predicted probability distribution over the vocabulary tokens (in this case, amino acid types) for the corresponding site along the sequence. The quality of this prediction has been evaluated using the cross-entropy loss function and the error function (see Appendix C for mathematical details).

D. The sampling algorithm

The space of parameters of the system is sampled according to Boltzmann's probability, using the loss as the energy of the system and considering it in the canonical ensemble of

statistical mechanics. In fact, Boltzmann's distribution is the maximum-entropy distribution at fixed average loss; sampling the parameter space at different temperatures T allowed us to study the manifold corresponding to different loss values without biasing the exploration. Moreover, we could take advantage of the large set of theoretical and computational tools developed in the realm of statistical mechanics.

Operatively, we first minimized the loss of each network with the Adam optimizer (cf. Appendix D), producing two types of trained weight vectors, namely, early stopped (ES) and overtrained (OT). Then, we sampled the space of all the parameters that define the networks using a norm-constrained version (cf. Appendix E) of the pseudo-Langevin sampling algorithm, recently developed for high-dimensional parameter spaces like those of artificial neural networks [17]. It basically consists of solving the Langevin equations, using approximate forces evaluated on minibatches and adjusting the fictitious masses associated with the parameters in real time to guarantee that the noise term of the dynamical equations is thermalized at the correct temperature.

At each temperature, we could thus collect a large number of sets of parameters, corresponding to different prediction models. We used these sets to estimate average quantities over the sampled models (such as the average validation error) and probability distributions (such as the similarity distribution between the parameters of the models). Assuming the system is ergodic, the averages converge to the thermodynamic averages and the probability distributions to Boltzmann's.

III. RESULTS

We studied the properties of models sampled according to Boltzmann distribution at different temperatures when trained either with synthetic sequences (Secs. III A, III B, and III C) or with families of natural sequences (Sec. III D).

A. The generalization error is minimal at intermediate temperatures

We compared the sampling of the parameter space at different temperatures for the single-block transformer TF_1 , the four-block transformer TF_4 , and the pooled feedforward network FF , acting on the same data with the same task (cf. Sec. II C). The comparison is extended to the results obtained with the standard optimization of the parameters by gradient-descent techniques, which in the present context can be regarded as zero-temperature calculations. For each architecture type, the system is in the overparametrized regime, as clear from the fact that OT solutions and low-temperature weight vectors are associated with a training error equal or close to 0 (Fig. 12(b) and Table II in the Appendixes). The most apparent difference between transformers and pooled feedforward architectures is in the behavior of the average loss as a function of the temperature [Fig. 3(a)]. While the pooled feedforward network displays a sharp jump in the loss [corresponding to a sharp peak in the specific heat, Fig. 3(b)], reminiscent of a first-order phase transition, both transformers display a much smoother shape. For the purposes of comparison, we have normalized the temperature with respect to the temperature $T_{1/2}$ at the maximum in the specific heat (which

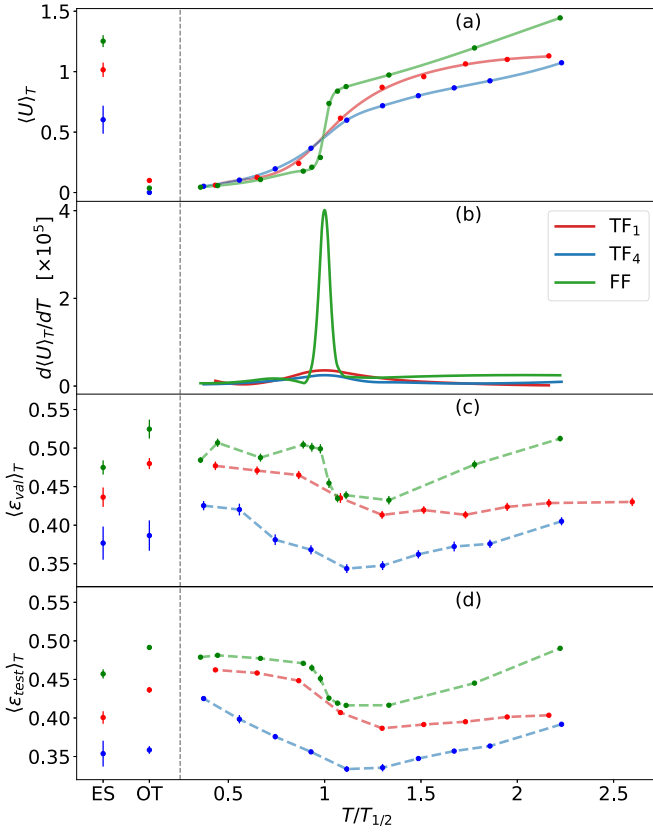


FIG. 3. Right of the line: The average value of the potential energy $\langle U \rangle_T$ [panel (a)], of the specific heat $d\langle U \rangle_T/dT$ [panel (b)], of the validation error $\langle \epsilon_{\text{val}} \rangle_T$ [panel (c)], and of the test error $\langle \epsilon_{\text{test}} \rangle_T$ [panel (d)] as functions of the rescaled temperature $T/T_{1/2}$ for the models TF_1 (red curves), TF_4 (blue curves), and FF (green curves). Left of the line: The same for ES solutions and OT solutions obtained with Adam. Circles denote numerical measurements, while solid lines represent spline interpolations. All points have an error bar that indicates the standard deviation.

is architecture dependent), in order to make the temperature scales of the different models comparable.

The accuracy of the prediction on the test set quantifies the ability of the model to generalize. The generalization accuracy for all models has a nonmonotonic shape, reaching a maximum right after the transition temperature [Fig. 3(d)]. This intermediate-temperature regime can be easily identified on the fly, looking at the error function computed on the validation set [Fig. 3(c)]. In this range of temperatures, the four-block transformer has the greatest generalization ability. In any case, the one-block transformer performs better than the feedforward architecture, despite the size difference between the two models (cf. Table I in Appendix B).

The OT weights of TF_1 and FF display loss, training, and generalization errors similar to the averages at low temperatures (Fig. 3), in agreement with the expectation that OT corresponds to zero temperature. On the contrary, the OT weights of TF_4 have much lower generalization error, while the loss is again what is expected at zero temperature. The ES weights show an average loss that is similar to that obtained from sampling at intermediate temperatures, and they consistently demonstrate higher generalization ability. However, the

average test error ϵ_{test} over the ES vectors is always larger than the average at the best temperature (by 0.01 at $T^{\text{best}} = 1.3 T_{1/2}$ for TF_1 , by 0.02 at $T^{\text{best}} = 1.1 T_{1/2}$ for TF_4 , and by 0.04 at $T^{\text{best}} = 1.1 T_{1/2}$ for FF). Moreover, the variability of the generalization error for different realizations of the ES minimization is much larger than that observed in the sampling at intermediate temperature [error bars in Fig. 3(d)].

B. Varying conservation of the parameters in the different layers

We then examined the variability of the parameters in the different layers of the networks during the sampling. For this purpose, we calculated the distribution of the mutual similarity

$$q_{t,t'}^l = \frac{\mathbf{w}^l(t) \cdot \mathbf{w}^l(t')}{\|\mathbf{w}^l(t)\| \|\mathbf{w}^l(t')\|} \quad (2)$$

of each layer l over the sampling, where $\mathbf{w}^l(t)$ is the vector containing the parameters of the layer l sampled at time t during a simulation at temperature T^{best} (see Sec. III A). We could compare the weights elementwise because the permutation symmetries of the networks had been broken by construction (see Appendix B).

In the one-block transformer, the distribution $\rho(q^l)$ of mutual similarity between pairs of models is either peaked around zero or very broad for most layers, indicating that the network can find several sets of parameters that maintain a sufficiently low loss (Fig. 14 in Appendix F). There are, however, exceptions. The parameters $\mathbf{y}_{\text{weight}}^{(1)}$ and $\mathbf{y}_{\text{weight}}^{(2)}$ that weight the rows in the normalization layers display similarity distribution strongly peaked toward 1, indicating that the values are well conserved throughout the sampling (red curves in the lower panels in Fig. 4). The parameters $\mathbf{W}^{(VC)}$ that control the value layer and $\mathbf{E}$ controlling the embedding are also conserved, although their distribution is more broadened than that of the normalization layers (Fig. 4, red curves in the middle panels). The parameters $\mathbf{W}^{(O)}$ controlling the output and the query-key parameters $\mathbf{W}^{(QK)}$ are peaked at intermediate values (Fig. 4, upper panels, red curves). Different layers of TF_1 are then conserved at varying degree at the optimal temperature.

In the case of a network made of four transformer blocks, the variability of the parameters in each block is very similar to that of the TF_1 system (the distributions for the first layer are shown as blue curves in Fig. 4). An exception is the set of value parameters $\mathbf{W}^{(VC)}$ that is peaked around 0 in the last three blocks (Fig. 14), while it is peaked around 1, as for the single-module architecture, only in the first block of the transformer.

The distribution of similarities in the layers of the pooled feedforward network FF (Fig. 4, green curves) is overall quite similar to that of the corresponding layers in the transformers, suggesting that their conservation is not a consequence of the attention mechanism.

The strong conservation of the weights $\mathbf{y}_{\text{weight}}^{(1)}$ and $\mathbf{y}_{\text{weight}}^{(2)}$ in all architectures facilitated the inspection of the values they assume during the sampling. For the TF_1 model, the distribution is bimodal (Fig. 5, solid red curves), with one peak centered around zero (amounting to approximately 60%

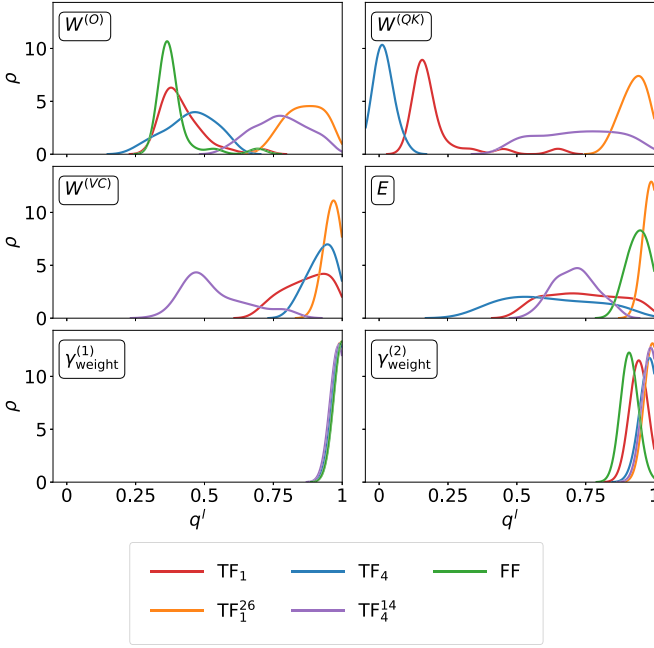


FIG. 4. The distributions ρ of the similarity q^l of parameters for some of the layers, obtained by comparing equilibrated configurations at temperature T^{best} for each architecture (identified by different colors). The complete set is displayed in Fig. 14 in Appendix F. The architectures labeled as TF_1^{26} and TF_4^{14} correspond to transformers of reduced dimensionality (see text).

of the elements for $\gamma_{\text{weight}}^{(1)}$ and 80% of the elements for $\gamma_{\text{weight}}^{(2)}$ and the other one around a positive value. Since these vectors have size equal to the embedding dimension $d = 64$ and they weight the information after the attention matrix, we hypothesized that the elements with value close to zero are essentially useless for the network's task.

Interestingly, the distribution of both sets of parameters has a single-peaked distribution for ES weights, centered at nonzero values. This is similar to the higher peak obtained through sampling for TF_1 (Fig. 5, dashed red curves), which suggests that our hypothesis would not apply to the parameters obtained from loss minimization.

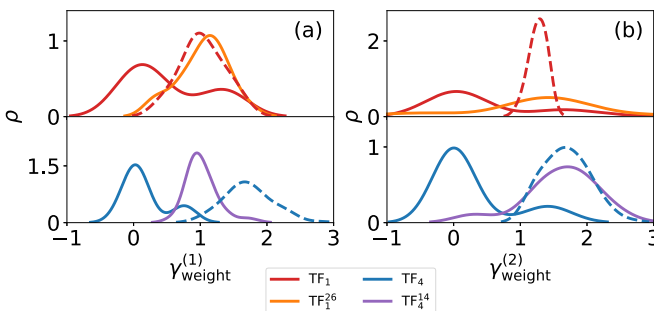


FIG. 5. The distribution ρ of the values of $\gamma_{\text{weight}}^{(1)}$ [panel (a)] and $\gamma_{\text{weight}}^{(2)}$ [panel (b)] obtained from the sampling at T^{best} of the original and reduced transformers (solid lines). The dotted red lines indicate the distribution of $\gamma_{\text{weight}}^{(1/2)}$ values obtained from the ES minimization. The values reported for the four-module transformers refer to the last module.

To verify the hypothesis, we trained a new network (details in Appendix D), where the dimension of the embedding is reduced to $d = 26$, corresponding to the elimination of the rows associated with values of $\gamma_{\text{weight}}^{(1)}$ close to zero. This new architecture will be referred to as TF_1^{26} (cf. Table I in Appendix B). We performed a sampling of the parameters at temperature $T^{\text{best}} \approx 1.5 T_{1/2}$ obtaining an average generalization error of $\langle \epsilon_{\text{test}} \rangle_{T^{\text{best}}} = 0.40 \pm 0.001$, which is comparable to that of the original larger model $\langle \epsilon_{\text{test}} \rangle_{T^{\text{best}}} = 0.38 \pm 0.002$. As expected, the values assumed by the new weight vectors $\gamma_{\text{weight}}^{(1/2)}$ during the sampling are now distributed according to a single-peaked distribution at nonzero values (Fig. 5, orange curves) and remain very conserved during the sampling (Fig. 4, orange curves in the lower panels).

Moreover, the per-layer similarities of the matrices E , $W^{(VC)}$, $W^{(QK)}$, and $W^{(O)}$ for the compact architecture TF_1^{26} are now all peaked toward higher values (Fig. 4, orange curves). This suggests that the high variability these layers had in the larger system was primarily due to their oversized dimensions. Summing up, in a single-block transformer whose embedding dimension is carefully tuned, only the layers $W^{(1)}$, $W^{(2)}$ after the attention and the biases $b^{(0)}$ of the output can vary quite freely along the sampling, while the other layers are essentially fixed.

This line of reasoning can also be applied to the TF_4 architecture. We examined the distributions of the values assumed by the vectors $\gamma_{\text{weight}}^{(1/2)}$ in the last transformer block, observing that they have qualitatively the same distribution we identified in the single-block model (Fig. 5, blue curves). Again, we trained a new network with a reduced embedding dimension $d = 14$ (called TF_4^{14}), corresponding to the number of nonzero values of $\gamma_{\text{weight}}^{(1)}$ in the last block. The sampling at $T^{\text{best}} \approx 1.7 T_{1/2}$ of the parameters of this reduced architecture shows a distribution $\rho(\gamma_{\text{weight}}^{(1/2)})$ that is unimodal and centered at nonzero values (Fig. 5, purple curves; the complete set of distributions of ρ is reported in Fig. 15 in Appendix F). The average generalization error is $\langle \epsilon_{\text{test}} \rangle_{T^{\text{best}}} = 0.36 \pm 0.002$, comparable to that of the TF_4 (i.e., $\langle \epsilon_{\text{test}} \rangle_{T^{\text{best}}} = 0.33 \pm 0.004$). The similar generalization ability occurs in spite of the huge difference in size between the two models (cf. Table I in Appendix B). For the different layers of TF_4^{14} , the similarity of the sampled parameters is overall smaller than that for the one-block architecture, but is anyhow larger, in general, than 0.5, with the exception of biases that can vary much from model to model (Fig. 4, purple curves).

C. The attention matrix is optimal at higher temperatures and larger embeddings

When applied to the structure prediction of proteins, a key property of transformers is that their attention matrix resembles the adjacency matrix of the input sequence [22].

For the one-block transformer TF_1 , the mean attention matrix (i.e., the attention matrix symmetrized and averaged over the test set sequences) for models sampled at intermediate temperatures ($T = 2T_{1/2}$) resembles the experimental contact map (cf. upper panels in Fig. 6 and Fig. 11 in Appendix F). However, the same is not true for the mean matrices associated with ES or OT vectors [cf. Figs. 13(a) and 13(b) in

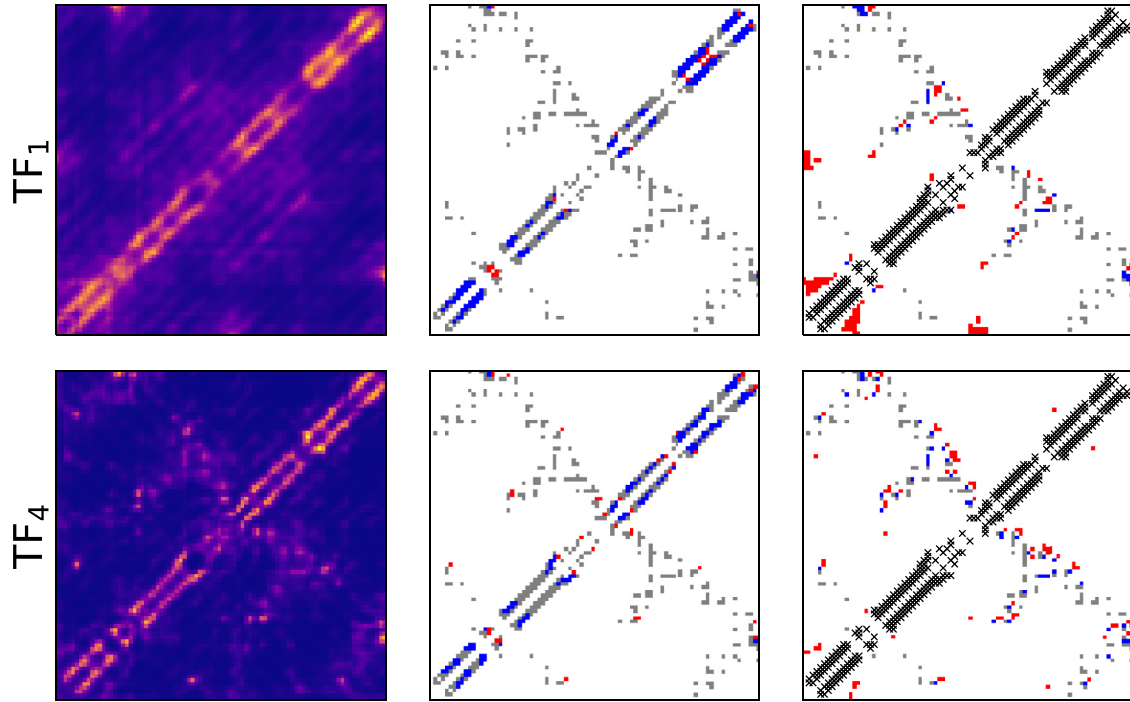


FIG. 6. Left column: The mean attention matrix, symmetrized and averaged over the test set sequences, for the TF_1 (upper row) and for the TF_4 (lower row) models, obtained during a sampling a temperature $T = 2 T_{1/2}$ and $T = 1.1 T_{1/2}$, respectively. Center column: The contact map obtained from the mean attention matrix using $P@L$; true positives are in blue, false positives in pink, and false negatives in gray. Right column: The contact map obtained from the mean attention matrix using $P@L_{NT}^1$.

Appendix F, respectively]. The mean attention matrix can be regarded as a self-averaging quantity, since individual models sampled at the same temperature produce mean matrices that are very similar to each other [e.g., Figs. 13(d), 13(f), and 13(g) in Appendix F].

The mean attention matrix of one of the blocks of the TF_4 model has a better similarity to the experimental contact map (lower panels in Fig. 6) than that of TF_1 . In this example, the map giving the improved agreement is that belonging to the third attention block of a model sampled at $T = 1.1 T_{1/2}$. The mean attention matrix can capture most contacts within alpha helices and many tertiary contacts. Most false-positive contacts are actually between residues that are adjacent to residues forming true contacts.

The mean attention matrices belonging to the other blocks are very dissimilar from the contact map of the protein [cf. Figs. 13(j), 13(k), and 13(l) in Appendix F]. In another independent sampling we conducted, the mean matrix that resembled the experimental contact map ranged from the third to the fourth. The mean attention matrix of the best-matching block seems self-averaging also in the TF_4 model [e.g., Figs. 13(m), 13(n), and 13(o) in Appendix F].

At variance with the TF_1 model, in the TF_4 also ES and OT give mean matrices reasonably predictive of the contact map, though they perform worse than those obtained through sampling [cf. Figs. 13(g), 13(h), 13(m), 13(n), and 13(o) in Appendix F].

A metric commonly used to assess the model capability of predicting protein contacts is the “precision at L” ($P@L$) [23], which is the fraction of correct contacts predicted from the

highest L elements of the attention matrix. In the present case, the metric is computed on the mean attention matrices, with $L = 86$ (i.e., the length of the protein sequence). The fraction of correctly predicted contacts in the case of very large language models ranges typically between 0.4 and 0.6 [24].

We calculated the $P@L$ metric by averaging over the models sampled at each temperature for both TF_1 and the best-matching block of TF_4 . We repeated the same calculation for the ES and OT solutions [Fig. 7(a)]. However, we observe that this metric does not recapitulate the intuition that TF_4 is more predictive than TF_1 , as suggested by Fig. 6. In fact, the $P@L$ prioritizes, in the list of the L top scores, contacts close to the diagonal, not providing a metric that quantifies the ability of the mean attention matrices to predict the contact map as a whole.

To solve this problem, we introduce a new metric, $P@L_{NT}^{\Delta}$, which takes into account the contacts between residues i and j only if $|j - i| > 7$. Additionally, it defines a predicted contact (i_p, j_p) as correct if it is within an interval Δ from a true contact (i_t, j_t) (i.e., if $|i_p - i_t| + |j_p - j_t| \leq \Delta$). According to the new metric, TF_4 is more predictive at all intermediate temperatures than TF_1 [Fig. 7(b)]. The overall lower values of $P@L_{NT}^1$ with respect to $P@L$ are due to the fact that the former is agnostic of the alpha helices, which are anyway easy to predict with other algorithms [25]. Increasing the value of Δ in the definition of $P@L_{NT}^{\Delta}$ does not change qualitatively the results, just shifting the curves upward (cf. Fig. 16 in Appendix F).

For the TF_1 and TF_4 architectures, the mean attention matrices obtained by sampling perform much better than those

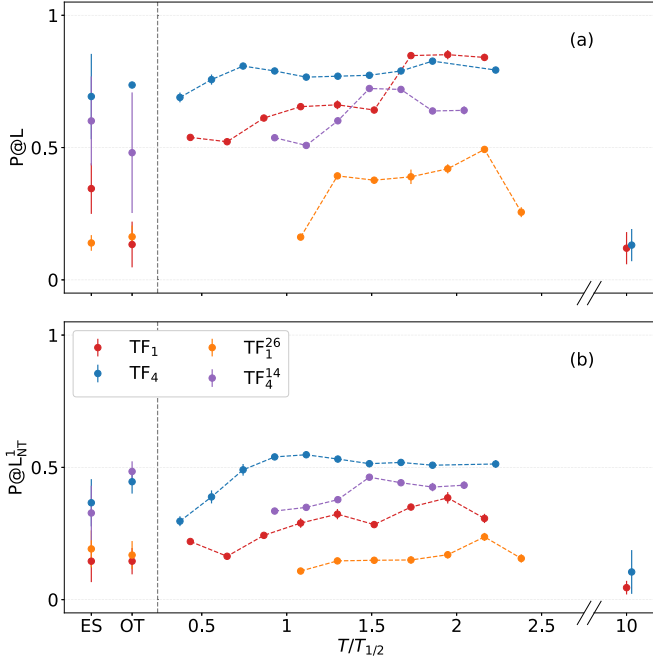


FIG. 7. The average and standard deviation of the P@L [panel (a)] and of the $P@L_{NT}^1$ [panel (b)] metrics for ES and OT vectors (left of the line) and for models obtained from sampling at different rescaled temperatures $T/T_{1/2}$, for each architecture type.

obtained from training [Fig. 7(b), blue and red points]. This improvement is smaller (for TF_1^{26}) or absent (for TF_4^{14}) for transformer with reduced embedding dimension.

Interestingly, there is not a straightforward correlation between generalization error and contact prediction capability [cf. Figs. 3(d) and 7(b)]. The values of $P@L_{NT}^1$ for the TF_4 model stay constantly high in the whole range of intermediate temperatures, whereas $\langle \epsilon_{test} \rangle_T$ increases shortly after $T_{1/2}$. For the TF_1 architecture, the best-matching mean attention matrices are found at temperatures consistently higher than $T_{1/2}$. We then conclude that the temperatures that provide predictive attention matrices do not match those at which the prediction of the protein sequence is optimal.

The models with a reduced embedding dimension d (i.e., TF_1^{26} and TF_4^{14}) exhibit a similar behavior. However, their performance is clearly worse [Fig. 7(b)]. Thus, while these minimal models can reproduce protein sequences as well as those with a larger embedding dimension, their ability to predict contact maps is much poorer. This further points to the fact that contact prediction (for which the network has not been trained) is not completely correlated with sequence prediction. Modifying the models to improve the training and the generalization error thus is not a good strategy to improve the predictivity of the attention matrix. This is also confirmed by the better performance of the OT models with respect to the ES solutions for almost every architecture [and especially for TF_4 and TF_4^{14} ; Fig. 7(b)].

D. Application to natural proteins

We then studied ten families of natural proteins, to validate the results obtained for the synthetic sequences (Sec. III C) in

the real-life case, where evolution is driven by constraints that are more complex than thermodynamic stability. For each protein family (see Sec. II B), we sampled the space of parameters of the four-block transformer TF_4 at different temperatures (Fig. 8, blue dots). As for the synthetic scenario, we observed a range of intermediate temperatures where the test error has a minimum. Moreover, in most cases the optimal temperature T^{best} is found just after the peak in specific heat ($T/T_{1/2} \approx 1$), as for synthetic proteins. There are some exceptions, namely, protein families PF00041, PF00096, and PF00153, where the best generalization performance is obtained at lower temperatures ($T/T_{1/2} \ll 1$).

We also compared the sampled averages with the results obtained with standard optimization of the parameters by gradient-descent techniques (Fig. 8, orange dots), as already done in Sec. III A for synthetic sequences. As a rule, the generalization capabilities of the network either are improved (as for PF00018, PF00072, and PF00096) or remain unchanged (for PF00013, PF00014, PF00076, PF00887, and PF13354) after sampling at T^{best} . The only exceptions are the protein families PF00041 and (especially) PF00153, where the sampling worsens the test error obtained after the training with Adam and early stopping.

To understand whether the dominant feature learned by the network is still thermodynamic stability, we investigated the ability of the trained and sampled models to predict the free-energy variation $\Delta\Delta G$ caused by single-site mutations within a specific sequence, which are known for some proteins as the result of protein-engineering experiments. For four such proteins (belonging to the families PF00014 [26], PF00018 [27], PF00041 [28], and PF00887 [29], respectively), we predicted the free-energy change $\Delta\Delta G^{pred}$ from the variation of the logarithmic output score [Eq. (C6)]. We then calculated the Spearman correlation coefficient between the experimental values ($\Delta\Delta G^{exp}$) and the predicted values ($\Delta\Delta G^{pred}$), obtained by either ES solutions or models sampled at T^{best} (Fig. 17). In three cases (Fig. 9), the correlation coefficient obtained from the sampled models is larger than 0.5, suggesting that the model takes the energetic properties of the protein into account. In the fourth case (PF00887), the results of the sampled models vary considerably, with some unable to capture the interactions between amino acids.

Finally, in three out of four cases, the distribution of the correlation coefficients for the sampled models is peaked at larger values than those obtained by gradient minimization. The only exception is the sequence belonging to the PF00041 protein family, which is also one of the two datasets over which the sampling slightly worsens the model performance in terms of test error (see Fig. 8).

IV. DISCUSSION AND CONCLUSIONS

Transformers are a widespread tool used for many tasks, most notably in large language models and protein structure prediction. They are extremely powerful and have surpassed multilayer perceptrons and recurrent networks in many areas, yet a deep understanding of how they work is lacking. In this regard, statistical mechanics has been useful in elucidating the working mechanisms of several artificial neural network architectures [4–11].

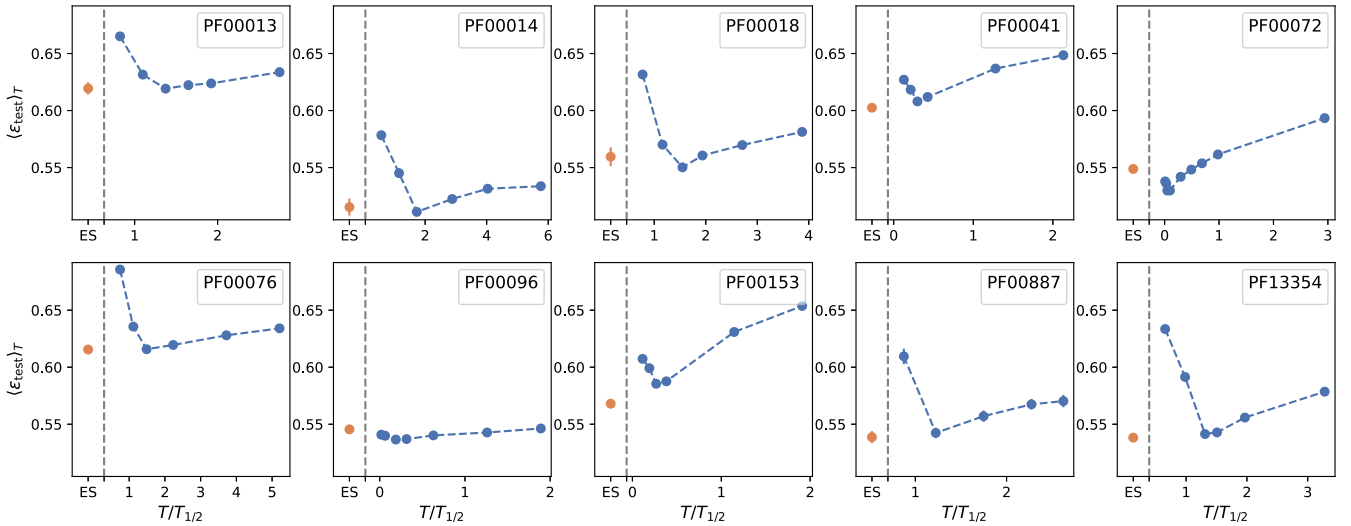


FIG. 8. For each panel, at the right of the gray dotted line, the average value of the test error $\langle \epsilon_{\text{test}} \rangle_T$ as a function of the rescaled temperature $T/T_{1/2}$ for the TF₄ model. At the left of the line, the same for ES solutions obtained with Adam. All points have an error bar that indicates the standard deviation.

In order to clarify how transformers function in protein structural prediction, we explored the parameter space of transformers that were trained to recognize protein sequences. Our goal is to understand why transformers are so efficient by exploring the low-loss manifold of the system. One way to do this is to sample the parameter space at a low temperature in the canonical ensemble. This transforms the problem into the realm of statistical mechanics, allowing us to use computational algorithms developed in this field over the last 50 years. Specifically, we employed a type of molecular dynamics in the parameter space driven by approximate Langevin equations, which are highly effective for sampling artificial neural networks [17].

We studied small transformers made of either one or four blocks for the simple task of learning the statistical properties of amino acid sequences. We used for training and test

either synthetic sequences, generated by energy minimization on a specific native conformation, having the only property of folding stably [19,30], and real sequences, belonging to ten different protein families. This allowed us to explore the space of parameters in detail under different conditions without being strongly limited by computational constraints. Of course, we cannot yet determine how our results scale for larger transformers and more ambitious tasks, such as learning the structure of all known proteins.

An interesting feature that emerges from our calculations is a wide range of intermediate temperatures at which the generalization capability of the transformer is particularly high, as testified by lower values in test error. Basically, the generalization error is a nonmonotonic function of the temperature for both synthetic and real proteins. At lower temperatures, the loss and the training error are smaller, but also the ability to generalize is smaller. In this respect, transformers behave quite differently from pooled feedforward networks with similar number of parameters and similar architecture, but lacking the attention mechanism. Pooled feedforward networks display a sharp transition in the average loss (and consequently in the training error) with respect to temperature from a low-temperature state with good training but bad generalization accuracy (i.e., the network interpolates the data), to a higher-temperature state where the training error increases drastically. This behavior seems to affect pooled feedforward networks independently of their detailed architecture and task [11]. On the contrary, in transformers the transition is smooth and does not appear to sharpen increasing the size of the network, generating a wide range of temperatures where the training error is still small enough and where generalization is more efficient.

This behavior explains why the typical large language models work so well despite being largely undertrained [31]. Due to the huge number of parameters, it would anyway be impossible to minimize their loss to a state corresponding to the low-temperature states observed in our system. Instead,

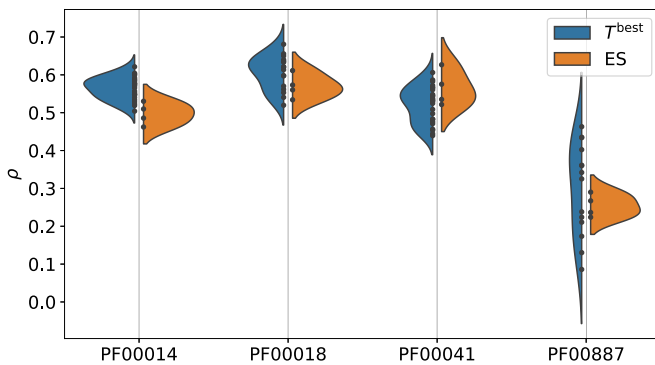


FIG. 9. The distribution of the Spearman correlation coefficient computed between experimental and predicted $\Delta\Delta G$, for models sampled at T^{best} (blue curves) and for early-stopped solutions (orange curves), for four sequences belonging to different protein families (from left to right, PF00014, PF00018, PF00041, and PF00887). The correlation coefficient associated with each model (either sampled or trained) is shown as a dot in the respective distribution.

their optimization approaches the wide intermediate-temperature region that is associated with better generalization properties.

From the sampling at intermediate temperatures, we also learn that most layers of the transformer can be highly variable from model to model. Similarly, the pooled feedforward network that we use as a control displays similar patterns; therefore, this behavior does not depend on the attention mechanism. A group of particularly conserved parameters are the ones associated with normalization layers, whose value distribution assumes a bimodal shape with one peak centered around zero during samplings in the intermediate-temperature range. Since these vectors have dimension d equal to the embedding size, the peak centered around higher values identifies the minimal dimension needed to obtain good performances on the masked-token prediction task.

In the minimal model obtained from decreasing the size of the embedding, the parameters of several layers become significantly more conserved along the sampling compared to those of the larger model. The matrices defining the embedding, the query/key, the value, the normalization, and the output have little freedom to vary in the one-block transformer, and somewhat more in the multiblock architecture. Only the biases are really free to vary.

The fact that the attention matrix is similar to the contact map of proteins is a useful by-product of transformers trained with amino acid sequences [22]. The sampling at intermediate temperatures produces attention matrices that are more similar to the experimental contact map than loss optimization. This is true for both transformer architectures we studied, although in the case of multiblocks one faces the problem of identifying *a priori* what block has the most informative attention matrix. Interestingly, transformers sampled at temperatures higher than those optimal for training and testing sequences produce attention matrices that are more similar to the experimental contact map. Overtrained solutions seem to perform better in the prediction of native contacts than early-stopped ones. The reduction of the dimension of the embedding that has little effect on the ability of the transformer to learn protein sequences decreases the ability to predict the contact map. The misalignment between conditions that favor sequence learning and those that favor the prediction of the contact map is not unexpected. As a matter of fact, the network is not trained to reproduce the latter. We can speculate that at higher temperatures or for larger embeddings, it is more difficult for the transformer to predict protein sequences out of noise, and this requires a more selective representation of the correlations between amino acids; the attention matrix is then expected to be cleaner and more similar to the contact map.

The results obtained with real protein families are as a rule in agreement with those obtained with synthetic sequences. For the former, we had also the possibility to compare the energetic effect of mutations with the experimental values, which are an independent set of data available for some of them. The fair match we obtained suggests not only that the sampled models are learning correctly the properties of the proteins, but also that they are dominated by thermodynamic requirements.

Finally, we suggest that the Langevin strategy we used in this study can be an efficient tool for optimization of large

language models, as it provides configurations with better predictive capabilities for both problems, i.e., masked-token prediction and native contact prediction. A proper tuning of the temperature in the intermediate-to-high range could help find models with better generalization properties.

ACKNOWLEDGMENT

We would like to thank R. Zecchina for his valuable discussions and suggestions.

DATA AVAILABILITY

The codes used for the sampling is available in Ref. [32]. The data generated by our sampling algorithm are available in Ref. [33].

APPENDIX A: THE PFAM DATASETS

The ten datasets of real protein sequences were obtained from the complete, annotated Pfam sequence alignment using the following procedure.

First, we identified all sequences belonging to the same selected protein family. From these sequences, we selected a representative one. We then removed all sites containing a gap in the reference sequence from each sequence in the alignment. Then, we discarded all those with more than 10% of their length composed of gaps after the removal of the sites. Finally, we clustered the remaining sequences using MMseqs2 [34] with a similarity threshold of 70%, and then selected a single representative sequence from each cluster.

In this way, we created protein datasets of fixed length (as in the case of the synthetic dataset), discarding sequences with a high proportion of gap sites and also sequences that were too similar to each other. The selected protein families and the code of the associated representative sequence can be found in Table III.

APPENDIX B: THE MODELS

The architectures considered in this study follow a common three-stage structure: an input construction stage, an encoder, and a final output projection layer. The input and output components are identical for the two network families studied here, while the internal encoder differs between transformer-based and pooled feedforward models.

1. Embedding

All models operate on sequences $\mathbf{x} = (x_1, \dots, x_n)$ of n discrete tokens drawn from a finite vocabulary. The latter is composed of 20 amino acid tokens and a special masking symbol added to represent missing or hidden tokens in the input sequence. Its cardinality is therefore $|V| = 21$.

In the input stage, each token $x_i \in \{1, \dots, |V|\}$ is represented by a one-hot vector $\mathbf{e}_{x_i} \in \{0, 1\}^{|V|}$ whose x_i th component is equal to 1 and all other components are 0, and then is mapped to a continuous vector representation through an embedding layer. Given an embedding matrix $E \in \mathbb{R}^{|V| \times d}$, the embedded representation of the token is obtained as

$$\mathbf{x}_i^{(E)} = E^\top \mathbf{e}_{x_i} \in \mathbb{R}^d, \quad (\text{B1})$$

producing a matrix sequence representation as $X^{(E)} \in \mathbb{R}^{n \times d}$, where n denotes the sequence length and d the embedding size. After the embedding, positional information is added through standard sinusoidal positional encoding [1],

$$P_{k,2r}^{(z)} = \sin\left(\frac{k}{10000^{2r/z}}\right), \quad P_{k,2r+1}^{(z)} = \cos\left(\frac{k}{10000^{2r/z}}\right), \quad (\text{B2})$$

with $P^{(z)} \in \mathbb{R}^{n \times z}$. This yields the common encoder input

$$\tilde{X} = X^{(E)} + P^{(d)}. \quad (\text{B3})$$

2. Compact transformer encoder

The first encoder considered in this work is a simplified compact transformer inspired by the standard architecture in Ref. [35].

Let $\tilde{X} \in \mathbb{R}^{n \times d}$ denote the input representation of a given encoder block. The self-attention mechanism computes interactions between embedded tokens through a bilinear form

$$A = \text{softmax}\left(\frac{\tilde{X}W^{(QK)}\tilde{X}^\top}{\sqrt{d}}\right) \in \mathbb{R}^{n \times n}, \quad (\text{B4})$$

where the softmax is applied rowwise, and where the matrix $W^{(QK)} \equiv W^{(Q)}(W^{(K)})^\top \in \mathbb{R}^{d \times d}$ represents the combined query-key transformation.

The attention weights are then used to aggregate token information as follows:

$$U^{(0)} = A\tilde{X}W^{(VC)} \in \mathbb{R}^{n \times d}, \quad (\text{B5})$$

with $W^{(VC)} \equiv W^{(V)}W^{(C)} \in \mathbb{R}^{d \times d}$ as the combined value-output projection matrix.

This contribution is added to the residual branch and normalized

$$U = \text{Norm}^{(1)}(\tilde{X} + U^{(0)}), \quad (\text{B6})$$

where “Norm” denotes layer normalization applied independently to each token representation. Given a generic matrix $V \in \mathbb{R}^{a \times b}$, the normalization is performed rowwise, i.e., independently for each vector $V_i \in \mathbb{R}^b$. For the i th row, we compute

$$\mu_i = \frac{1}{b} \sum_{j=1}^b V_{ij}, \quad \sigma_i^2 = \frac{1}{b} \sum_{j=1}^b (V_{ij} - \mu_i)^2, \quad (\text{B7})$$

and the normalization layer output is then

$$\text{Norm}(V)_{ij} = \frac{V_{ij} - \mu_i}{\sigma_i} (\gamma_{\text{weight}})_j + (\gamma_{\text{bias}})_j, \quad (\text{B8})$$

with $\gamma_{\text{weight}}, \gamma_{\text{bias}} \in \mathbb{R}^b$ being learnable vectors applied elementwise along the feature dimension.

The subsequent feedforward stage expands the representation to a higher-dimensional space

$$H = UW^{(1)} + \mathbf{b}^{(1)} \in \mathbb{R}^{n \times m}, \quad (\text{B9})$$

where m represents the size of the hidden feedforward layer, and where $W^{(1)} \in \mathbb{R}^{d \times m}$ and $\mathbf{b}^{(1)} \in \mathbb{R}^m$ are the associated weight matrix and bias vector, respectively.

After applying a nonlinearity, an additional positional contribution is introduced in the hidden representation

$$\tilde{H} = \text{ReLU}(H) + P^{(m)}, \quad (\text{B10})$$

where the function $\text{ReLU}(x) = \max(0, x)$ is applied elementwise.

The representation is then projected back to the model dimension

$$Z^{(0)} = \tilde{H}W^{(2)} + \mathbf{b}^{(2)} \in \mathbb{R}^{n \times d}, \quad (\text{B11})$$

where $W^{(2)} \in \mathbb{R}^{m \times d}$ and $\mathbf{b}^{(2)} \in \mathbb{R}^d$ are the weight matrix and the bias vector of the second hidden feedforward layer of size d .

Finally, the output of the encoder block is obtained through a second residual connection followed by a normalization

$$Z = \text{Norm}^{(2)}(U + Z^{(0)}) \in \mathbb{R}^{n \times d}, \quad (\text{B12})$$

which represents the contextualized token representation produced by the block.

The encoder described above corresponds to a single transformer block. To increase the expressive capacity of the model, multiple identical blocks can be composed sequentially. Let L denote the depth of the encoder, or the number of stacked blocks. Given an input representation, the output of each block is used as the input for the next block. Thus, the full encoder is obtained by composing L transformations. The stacked blocks share the same functional form, yet they have separate, trainable parameters.

3. Feedforward encoder

The second type of encoder can be interpreted as a feedforward network with an organization similar to that of a transformer. It has the same residual structure as the transformer block but replaces the attention mechanism with a simpler global context aggregation step. Rather than computing pairwise token interactions through an attention matrix, contextual information is introduced via a pooling operation over the sequence. This difference makes the model a useful baseline for isolating the contribution of self-attention and, more generally, of explicit contextualization mechanisms.

Starting from the shared input representation $\tilde{X}$ [see Eq. (B3)], the first step consists of applying layer normalization independently to each token representation,

$$U = \text{Norm}^{(1)}(\tilde{X}). \quad (\text{B13})$$

This normalization stabilizes the representation and ensures that feature magnitudes are comparable across positions. The next step is to construct a simple form of contextual information shared by the whole sequence, extracting a global summary from the visible tokens only.

More precisely, given a binary visibility mask $\chi \in \{0, 1\}^n$, where $\chi_i = 0$ for masked positions and 1 otherwise, we define the global context vector g as the columnwise average of the representations of the visible tokens,

$$g = \frac{1}{\sum_{i=1}^n \chi_i} \sum_{i=1}^n \chi_i U_i \in \mathbb{R}^d. \quad (\text{B14})$$

The vector g provides a coarse summary of the sequence based only on unmasked tokens, and it is then broadcast to all positions,

$$G = \mathbf{1}_n g^\top \in \mathbb{R}^{n \times d}. \quad (\text{B15})$$

The local representation U and the global broadcast context G are finally concatenated along the feature dimension,

$$S = [U \ G] \in \mathbb{R}^{n \times 2d}. \quad (\text{B16})$$

Through this construction, each token receives both its local features and a shared global summary of the sequence. This pooling mechanism provides a simple form of contextualization that replaces the pairwise interactions produced by the attention matrix in the transformer encoder [see Eq. (B4)].

Then, similarly to the transformer encoder, a two-layer feedforward transformation is applied. The first linear projection expands the representation to an internal dimension m ,

$$H = SW^{(1)} + \mathbf{b}^{(1)} \in \mathbb{R}^{n \times m}, \quad (\text{B17})$$

where $W^{(1)} \in \mathbb{R}^{2d \times m}$ and $\mathbf{b}^{(1)} \in \mathbb{R}^m$ are the weight matrix and bias vector of the first hidden layer.

A nonlinearity is applied and then a positional contribution is introduced in the hidden representation,

$$\tilde{H} = \text{ReLU}(H) + P^{(m)}. \quad (\text{B18})$$

The hidden representation is then processed by a second linear transformation within the internal space,

$$F = \text{ReLU}(\tilde{H}W^{(2)} + \mathbf{b}^{(2)}) \in \mathbb{R}^{n \times m}, \quad (\text{B19})$$

where once more $W^{(2)} \in \mathbb{R}^{m \times m}$ and $\mathbf{b}^{(2)} \in \mathbb{R}^m$ are the weight matrix and the bias vector of the second feedforward layer.

In parallel, a residual projection of the normalized input is computed,

$$R = UW^{(r)} + \mathbf{b}^{(r)} \in \mathbb{R}^{n \times m}, \quad (\text{B20})$$

with $W^{(r)} \in \mathbb{R}^{d \times m}$ and $\mathbf{b}^{(r)} \in \mathbb{R}^m$. The encoder output is then obtained by combining the two streams and applying a normalization layer

$$Z = \text{Norm}^{(2)}(R + F). \quad (\text{B21})$$

In contrast to the transformer architecture, this feedforward encoder is implemented using a single block and no stacking is considered. Therefore, no depth parameter L is introduced for this model.

4. Output

After the input construction stage, the encoder transforms the input representation into contextual features that capture dependencies between sequence positions. The encoder output is a sequence of hidden representations

$$Z \in \mathbb{R}^{n \times h}, \quad (\text{B22})$$

where h denotes the dimensionality of the internal representation. The precise transformation that produces Z depends on the specific encoder architecture described in the previous sections.

TABLE I. Model configurations and number of trainable parameters.

Encoder type	L	d	m	Parameters [$\times 10^3$]
Transformer	1	26	104	8
Transformer	1	64	256	44
Transformer	4	14	56	9
Transformer	4	64	256	169
Feedforward	–	64	320	174

In the final stage, the encoded representation is projected back into the vocabulary space through a linear transformation,

$$\hat{\mathbf{y}}_i = (W^{(o)})^\top Z_i + \mathbf{b}^{(o)}, \quad (\text{B23})$$

where $W^{(o)} \in \mathbb{R}^{h \times |V|}$ and $\mathbf{b}^{(o)} \in \mathbb{R}^{|V|}$. The resulting vector $\hat{\mathbf{y}}_i \in \mathbb{R}^{|V|}$ contains the model scores associated with each vocabulary token for the site i along the input sequence.

5. Details and hyperparameters of our models

All models share the same input embedding and output projection layers introduced in the previous section and differ only in the internal encoder. This makes it possible to compare architectures while keeping the input representation and prediction head fixed.

We consider two transformer-based models, obtained by stacking identical encoder blocks with depth $L \in \{1, 4\}$. In both cases, the embedding dimension is fixed to $d = 64$ and the internal feedforward width to $m = 4d = 256$. These two configurations allow us to compare a shallow and a deeper transformer while preserving the same block structure.

In addition, we consider two smaller transformer-based model, again with one module (with $d = 26$ and $m = 4d = 104$) and four modules (with $d = 14$ and $m = 4d = 56$), whose embedding size d is derived from the Boltzmann sampling of their respective larger counterparts (see Sec. III B).

As a baseline, we also consider a feedforward-based model. For this model, the dimensions are fixed to $d = 64$ and $m = 320$. The size of the first feedforward layer m is chosen so that the total number of trainable parameters is comparable to that of the bigger transformer (i.e., $L = 4$ and $d = 64$). In this way, the comparison isolates the effect of attention while keeping the overall parameter scale similar. The resulting model configurations are summarized in Table I.

Finally, it should be noted that both encoder architectures have slight modifications to their typical compositions. For example, in the transformer-based encoder, we merged the query and key matrices [see Eq. (B4)] and the value and output matrices [see Eq. (B5)]. Additionally, we applied an extra positional encoding to the output of the first hidden layer for both architectures [see Eqs. (B18) and (B10)]. These modifications were made to eliminate all possible permutation symmetries within the studied models, without strongly affecting their learning capabilities (see Table II).

APPENDIX C: OBSERVABLES

Let us consider a neural network model among the ones previously described (Appendix B 5) and let us call $\mathbf{w}$ the weight vector representing all its parameters. The quality of the prediction of the NN over a dataset $\mathcal{D}$ (as the ones defined in Secs. II A and II B) is evaluated using two quantities: the cross-entropy loss function and the error function.

For a given pair $(\mathbf{x}, \mathbf{y})$, let $\mathcal{M} \subset \{1, \dots, n\}$ denote the set of associated masked indices, and let $\hat{\mathbf{y}}_i \in \mathbb{R}^{|V|}$ be the vector of output scores for the masked position i (see Appendix B 4).

For the selected input-label pair, the cross-entropy loss is defined as

$$\ell_{\text{CE}}(\mathbf{w}|\mathbf{x}, \mathbf{y}) = -\frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \sum_{v=1}^{|V|} e_{y_i, v} \ln(\text{softmax}(\hat{\mathbf{y}}_i)_v), \quad (\text{C1})$$

where $e_{y_i} \in \{0, 1\}^{|V|}$ denotes the one-hot encoded vector representing the true amino acid at masked site i . Analogously, the global loss function is obtained by averaging over all the pairs, that is,

$$\mathcal{L}(\mathbf{w}|\mathcal{D}) = \frac{1}{P} \sum_{\mu=1}^P \ell_{\text{CE}}(\mathbf{w}|\mathbf{x}^\mu, \mathbf{y}^\mu), \quad (\text{C2})$$

which in our case coincides with the potential energy $U(\mathbf{w}|\mathcal{D})$ associated with the weight vector $\mathbf{w}$, as no regularization terms were added.

From the same $\hat{\mathbf{y}}_i$ vector, we can define the most probable amino acid at the masked position i as

$$\hat{c}_i = \text{argmax}_v [\text{softmax}(\hat{\mathbf{y}}_i)_v], \quad (\text{C3})$$

with $\hat{c}_i \in V$ being a token from the vocabulary. The masked-token accuracy evaluated on the $(\mathbf{x}, \mathbf{y})$ pair is then

$$a(\mathbf{w}|\mathbf{x}, \mathbf{y}) = \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mathbf{1}[\hat{c}_i = y_i]. \quad (\text{C4})$$

As before, the global error function can be computed by averaging over all the input-label couples in the dataset,

$$\epsilon(\mathbf{w}|\mathcal{D}) = \frac{1}{P} \sum_{\mu=1}^P [1 - a(\mathbf{w}|\mathbf{x}^\mu, \mathbf{y}^\mu)]. \quad (\text{C5})$$

Fianlly, in Sec. III D, we characterized the capability of a given model to predict the free-energy variation in a sequence after a single-site mutation, that is, $\Delta\Delta G^{\text{pred}}$. Let us consider a sequence $\mathbf{x}$ masked on site j , and a single-site mutation on the same site, that is, $x_j = \sigma \rightarrow \xi$. Then, the predicted free-energy variation deriving from such mutation is defined as follows:

$$\Delta\Delta G^{\text{pred}}(\mathbf{w}|\mathbf{x}, \sigma, \xi, j) = -\log \left(\frac{\text{softmax}(\hat{\mathbf{y}}_j)_\xi}{\text{softmax}(\hat{\mathbf{y}}_j)_\sigma} \right), \quad (\text{C6})$$

where, as before, $\hat{\mathbf{y}}_j$ is the vector of output scores for the site j given the masked sequence $\mathbf{x}$ as input.

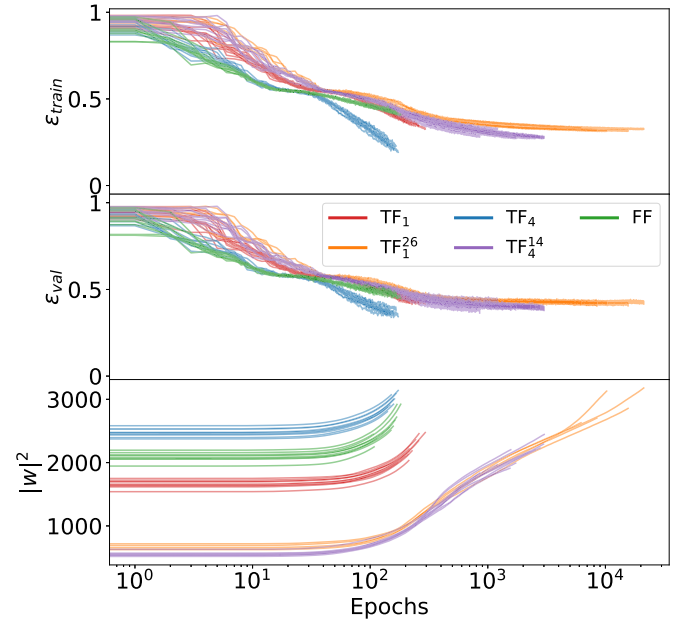


FIG. 10. The training error ϵ_{train} (upper panel), the validation error ϵ_{val} (middle panel), and the squared norm $\|\mathbf{w}\|^2$ (lower panel) as functions of the training epoch during several first-stage trainings for the TF_1 model (red curves), the TF_1^{26} model (orange curves), the TF_4 model (blue curves), the TF_4^{14} model (purple curves), and the FF model (green curves).

APPENDIX D: TRAINING ALGORITHMS

All models are trained using standard optimization techniques commonly adopted for transformer-based architectures [1]. Both the transformer-based and the feedforward models are optimized with Adam, using momentum parameters $(\beta_1, \beta_2) = (0.9, 0.98)$ and $\epsilon = 10^{-9}$. The learning rate follows the standard schedule, consisting of a linear warm-up phase followed by inverse square-root decay. At optimization step t , the learning rate is

$$\eta(t) = d^{-1/2} \min(t^{-1/2}, ts^{-3/2}), \quad (\text{D1})$$

where d denotes the embedding dimension and $s = 4000$ is the number of warm-up steps.

A dropout rate of $p_{\text{drop}} = 0.1$ is used throughout, following the placement adopted in the original transformer: Dropout is applied to the input embeddings, and to the outputs of attention and feedforward sublayers. No additional explicit regularization term is introduced during training.

We adopt a two-stage training protocol. In the first stage, multiple independent instances of each model are trained with early stopping based on validation cross-entropy. From the ensemble of trained networks, we compute the empirical distribution of the squared norm of the parameter vectors, $\|\mathbf{w}\|^2$, and estimate its mean value, with $\mathbf{w}$ denoting the full vector of trainable parameters of the network. For each architecture, the first stage is repeated over ten independent runs with different random initializations. The resulting training dynamics and norm evolution across independent runs for the synthetic dataset are shown in Fig. 10.

TABLE II. Results of the second-stage trainings for ES (✓) and OT (×) solutions, with the associated number of training epochs (Epochs) and their average performance on the train and test synthetic datasets evaluated through the error function ($\langle\epsilon_{\text{train}}\rangle$ and $\langle\epsilon_{\text{test}}\rangle$, respectively).

Encoder type	L	d	m	Parameters [$\times 10^3$]	ES	Epochs	$\langle\epsilon_{\text{train}}\rangle$	$\langle\epsilon_{\text{test}}\rangle$
Transformer	1	26	104	8	×	100k	0.31 ± 0.006	0.40 ± 0.006
Transformer	1	26	104	8	✓	10k	0.32 ± 0.006	0.39 ± 0.005
Transformer	1	64	256	44	×	500k	0.02 ± 0.003	0.43 ± 0.004
Transformer	1	64	256	44	✓	240	0.35 ± 0.02	0.40 ± 0.008
Transformer	4	14	56	9	×	100k	0.22 ± 0.014	0.36 ± 0.16
Transformer	4	14	56	9	✓	2k	0.40 ± 0.012	0.37 ± 0.006
Transformer	4	64	256	169	×	1k	0.00 ± 0.00	0.36 ± 0.005
Transformer	4	64	256	169	✓	170	0.22 ± 0.04	0.35 ± 0.016
Feedforward	–	64	320	174	×	2k	0.00 ± 0.00	0.50 ± 0.012
Feedforward	–	64	320	174	✓	170	0.42 ± 0.012	0.46 ± 0.006

In the second stage, we train three final reference models for each architecture type while explicitly monitoring $|\mathbf{w}|^2$. Training is stopped as soon as the squared norm reaches the mean value estimated in the first stage for the corresponding model class, to save the reached configuration (ES models). The ES solutions are then employed as starting points in the samplings at fixed norm performed using the constrained pseudo-Langevin algorithm (see Appendix E).

Additionally, for the synthetic dataset, each training procedure continues without any norm constraints up until both the cross-entropy function and the error function evaluated on the train set saturate (OT models).

The performances of trained models for each architecture are summed up in Table II (for the synthetic sequence datasets) and in Table III (separately for each protein family studied).

APPENDIX E: THE SAMPLING ALGORITHM

The pseudo-Langevin (pL) algorithm [17] is an approximate integrator of the Langevin stochastic differential equations of motion. It produces a dynamics in the phase space $(\mathbf{w}, \mathbf{p})$, with $\mathbf{p}$ being the momenta associated with the weight vector $\mathbf{w}$, which efficiently samples the Boltzmann equilibrium distribution at a certain temperature T .

We modified the pL sampling scheme, in order to perform a norm-constrained dynamics in the space of the weights of the neural network. Imposing a constant norm for the vector of parameters $\mathbf{w}$ at all times during integration can be translated into two equations, that is, one for the holonomic constraint

$$C(\mathbf{w}) = |\mathbf{w}|^2 - Q = 0 \quad (\text{E1})$$

and one for its time derivative

$$\dot{C}(\mathbf{w}) = 2\mathbf{w}\dot{\mathbf{w}} = 0 \Rightarrow \mathbf{w}\mathcal{M}^{-1}\mathbf{p} = 0, \quad (\text{E2})$$

where $\dot{\mathbf{w}}$ stands for the time derivative of the vector $\mathbf{w}$, which we have rewritten in terms of the corresponding momenta $\mathbf{p} = \mathcal{M}\dot{\mathbf{w}}$, with $\mathcal{M}$ being a fictitious diagonal mass matrix. The interpretation is obvious; that is, the momenta must be perpendicular at all times to the hypersphere norm vector.

During the numerical integration of the stochastic equations of motion, the unconstrained pseudo-Langevin update generally produces configurations that do not exactly satisfy this condition. To restore the constraint, the configuration must be projected back onto the constraint hypersphere of radius $\sqrt{Q}$. Let us call $(\mathbf{w}^*(t + \delta t), \mathbf{p}^*(t + \delta t))$ the unconstrained update of the current vectors $(\mathbf{w}(t), \mathbf{p}(t))$ through pL, with δt being the integration time step. The integration scheme

TABLE III. Results of the second-stage trainings for TF₄ ES solutions on each Pfam dataset (Pfam), with the associated number of training epochs (Epochs) and their average performance on the train and test sets evaluated through the error function ($\langle\epsilon_{\text{train}}\rangle$ and $\langle\epsilon_{\text{test}}\rangle$, respectively). Each protein family code is associated with the PDB code of the reference sequence used to generate the dataset.

Encoder type	L	d	m	Parameters [$\times 10^3$]	Pfam	PDB	Epochs	$\langle\epsilon_{\text{train}}\rangle$	$\langle\epsilon_{\text{test}}\rangle$
Transformer	4	64	256	169	PF00013	4QVZ	40	0.586 ± 0.004	0.619 ± 0.005
Transformer	4	64	256	169	PF00014	1PIT	50	0.491 ± 0.003	0.515 ± 0.007
Transformer	4	64	256	169	PF00018	1SHG	60	0.502 ± 0.007	0.560 ± 0.008
Transformer	4	64	256	169	PF00041	1FNF	70	0.565 ± 0.003	0.602 ± 0.0014
Transformer	4	64	256	169	PF00072	3CHY	100	0.544 ± 0.0005	0.549 ± 0.0007
Transformer	4	64	256	169	PF00076	1HA1	60	0.573 ± 0.004	0.616 ± 0.002
Transformer	4	64	256	169	PF00096	1SP1	30	0.545 ± 0.0008	0.546 ± 0.0009
Transformer	4	64	256	169	PF00153	1OKC	130	0.503 ± 0.005	0.568 ± 0.0009
Transformer	4	64	256	169	PF00887	2ABD	80	0.484 ± 0.028	0.539 ± 0.005
Transformer	4	64	256	169	PF13354	1HTZ	120	0.447 ± 0.005	0.538 ± 0.0011

is then modified as follows:

$$\begin{aligned} \mathbf{w}(t + \delta t) &= \mathbf{w}^*(t + \delta t) + \lambda \frac{\delta t^2}{2} \mathcal{M}^{-1} \nabla C(\mathbf{w}(t)), \\ \mathbf{p}(t + \delta t) &= \mathbf{p}^*(t + \delta t) \\ &\quad + \frac{c_1 \delta t}{2} [\lambda \nabla C(\mathbf{w}(t)) + \Lambda \nabla C(\mathbf{w}(t + \delta t))], \end{aligned} \quad (\text{E3})$$

where c_1 accounts for the dissipation caused by the viscosity of the medium, $\nabla C(\mathbf{w}) = 2\mathbf{w}$, and λ, Λ are two Lagrange multipliers that can be computed by imposing the validity of Eqs. (E1) and (E2) for $\mathbf{w}(t + \delta t)$ and $\mathbf{p}(t + \delta t)$, respectively.

First, we fix the norm of the updated weight vector by solving

$$|\mathbf{w}(t + \delta t)|^2 - Q = 0. \quad (\text{E4})$$

This leads to

$$\lambda^\pm = \frac{-S_1 \pm \sqrt{S_1^2 - S_3(S_2 - Q)}}{\delta t^2 S_3}, \quad (\text{E5})$$

where we defined

$$\begin{aligned} S_1 &= \mathbf{w}^*(t + \delta t) \mathcal{M}^{-1} \mathbf{w}(t), \\ S_2 &= |\mathbf{w}^*(t + \delta t)|^2, \\ S_3 &= |\mathcal{M}^{-1} \mathbf{w}(t)|^2. \end{aligned} \quad (\text{E6})$$

Among the two roots from Eq. (E5), we select as λ the one with the smallest absolute value, since it corresponds to the minimal correction compatible with the constraint.

Once the weight vector has been projected back to the correct hypersphere, we proceed to orthogonalize the momenta vector by solving

$$\mathbf{w}(t + \delta t) \mathcal{M}^{-1} \mathbf{p}(t + \delta t) = 0. \quad (\text{E7})$$

Analogously, the solution to this equation is

$$\Lambda = -\frac{S_4}{c_1 \delta t S_5}, \quad (\text{E8})$$

where we introduced

$$\begin{aligned} S_4 &= \mathbf{w}(t + \delta t) \mathcal{M}^{-1} [\mathbf{p}^*(t + \delta t) + \lambda c_1 \delta t \mathbf{w}(t + \delta t)], \\ S_5 &= \mathbf{w}(t + \delta t) \mathcal{M}^{-1} \mathbf{w}(t + \delta t). \end{aligned} \quad (\text{E9})$$

Finally, we should mention that a projection is also applied when the initial momentum is generated. Given an unconstrained initial draw $\mathbf{p}(t_0) \sim \mathcal{N}(0, \mathcal{M}T)$, we project it onto the tangent space defined by the starting parameters configuration $\mathbf{w}(t_0)$ by subtracting its normal component

$$\mathbf{p}(t_0) \leftarrow \mathbf{p}(t_0) - \frac{\mathbf{w}(t_0) \mathcal{M}^{-1} \mathbf{p}(t_0)}{|\mathbf{w}(t_0)|^2} \mathcal{M} \mathbf{w}(t_0), \quad (\text{E10})$$

which satisfies Eq. (E2).

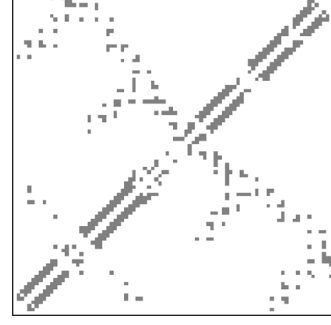


FIG. 11. The acyl-coenzyme A binding protein contact map.

APPENDIX F: SUPPORTING FIGURES

The detailed thermodynamic features of the synthetic sequences generated for acyl-coenzyme A binding protein (whose contact map is plotted in Fig. 11) are reported in Figs. 12–15. Those obtained for natural protein sequences are reported in Figs. 16 and 17.

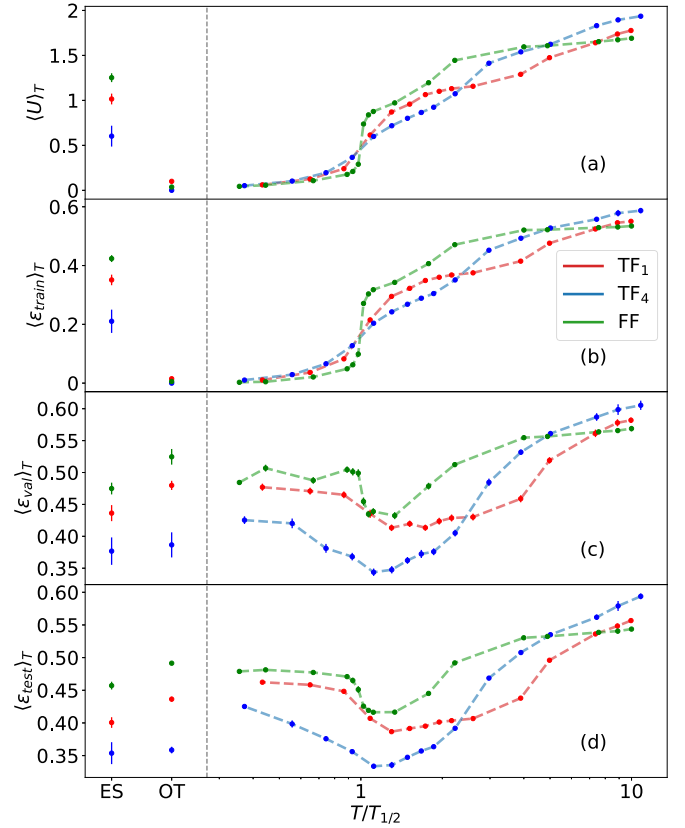


FIG. 12. Right of the line: The average value of the potential energy $\langle U \rangle_T$ [panel (a)], of the training error $\langle \epsilon_{\text{train}} \rangle_T$ [panel (b)], of the validation error $\langle \epsilon_{\text{val}} \rangle_T$ [panel (c)], and of the test error $\langle \epsilon_{\text{test}} \rangle_T$ [panel (d)] as functions of the rescaled temperature $T/T_{1/2}$ for the TF1 model (red curves), the TF4 model (blue curves), and the FF model (green curves). Left of the line: The average of the same observables and for the same models, for ES solutions and OT solutions obtained with Adam.

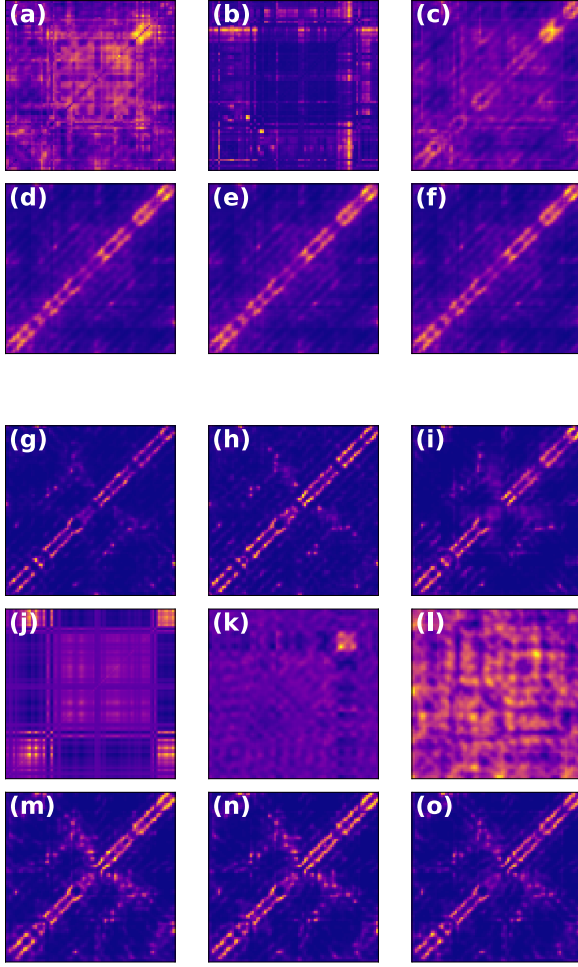


FIG. 13. Each panel shows a mean attention matrix, which is symmetrized and averaged over the test set sequences. The top six images are relative to the TF_1 architecture. Images (a), (b), and (c) are the mean attention matrices for an early-stopped vector, an overtrained vector, and a single model sampled at T^{best} . Images (d), (e), and (f) are the mean attention matrices for three different models sampled at $T = 2 T_{1/2}$. The remaining images are relative to the TF_4 architecture. Images (g), (h), and (i) are the best-block mean attention matrices for an early-stopped vector, an overtrained vector, and a single model sampled at T^{best} , respectively. Images (j), (k), and (l) are the mean attention matrices of the first, second, and fourth blocks of the TF_4 model sampled at $T = 1.1 T_{1/2}$. Images (m), (n), and (o) are the mean attention matrices of the third block of the TF_4 model for three different models sampled at $T = 1.1 T_{1/2}$.

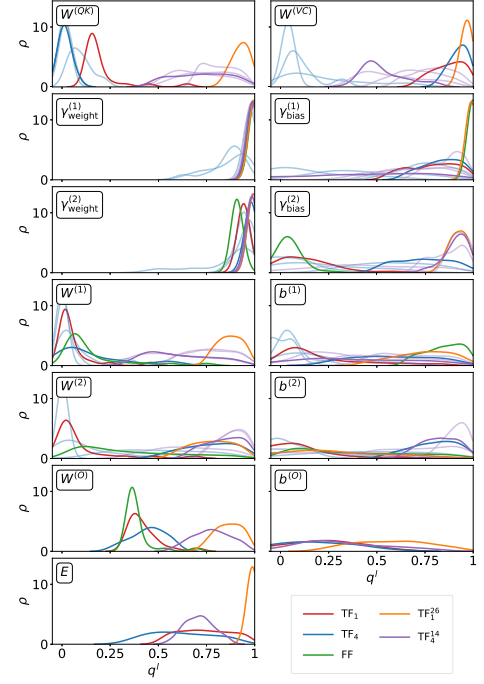


FIG. 14. The distributions ρ of the similarity q^l calculated between the weights of each layer. The similarity values are obtained by comparing equilibrated configurations at temperature T^{best} , different for each architecture, i.e., TF_1 (red curves), TF_1^{26} (orange curves), TF_4 (blue curves), TF_4^{14} (purple curves), and FF (green curves). For the transformer-based models with four modules, in the panels for $W^{(OK)}$, $W^{(VC)}$, $Y_{\text{weight}}^{(1)}$, $Y_{\text{weight}}^{(2)}$, $W^{(1)}$, $W^{(2)}$, and $b^{(2)}$, the solid lines represent the distribution computed between weights from the first block, whereas the three faded curves are relative to the weights from the second, third, and fourth blocks.

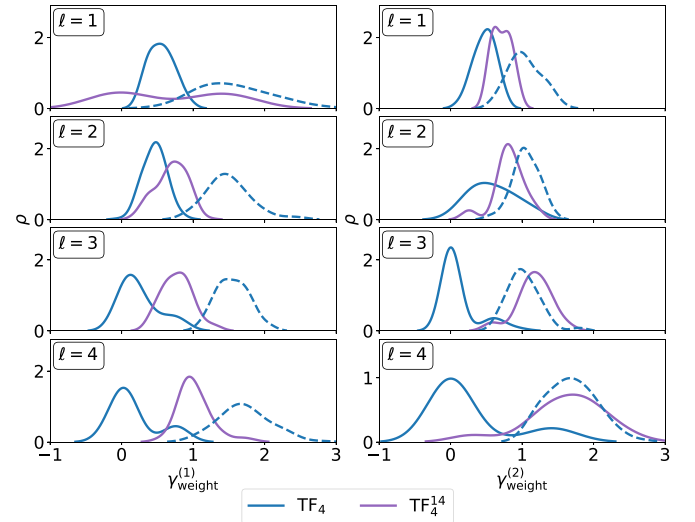


FIG. 15. Distribution ρ of the values of $\gamma_{\text{weight}}^{(1)}$ (left panels) and $\gamma_{\text{weight}}^{(2)}$ (right panels) for the four-block architectures TF_4 (blue curves) and TF_4^{14} (purple curves) in the different layers ($l = 1, 2, 3, 4$). The solid lines represent the results obtained through equilibrated simulations at T^{best} separately for each architecture. The dashed red lines represent the distribution of $\gamma_{\text{weight}}^{(1/2)}$ values obtained from the ES solutions for the TF_4 network only.

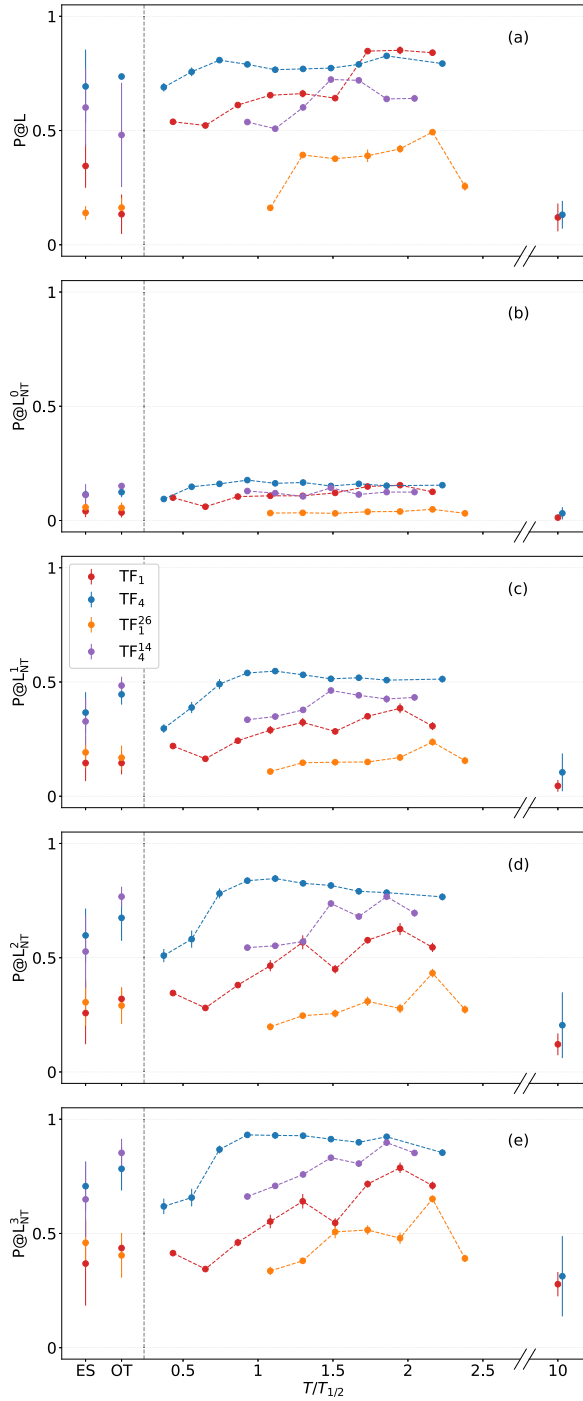


FIG. 16. The average and standard deviation of the P@L [panel (a)] and of the $P@L_{NT}^{\Delta}$ metrics for $\Delta = 0, 1, 2, 3$ [panels (b), (c), (d), and (e), respectively] for ES and OT vectors (left of the line) and for models obtained from sampling at different rescaled temperatures $T/T_{1/2}$, for each architecture type.

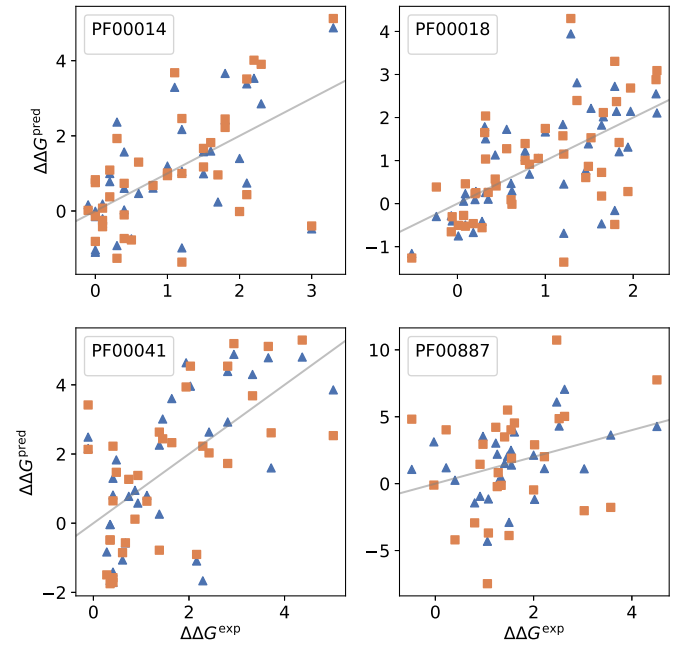


FIG. 17. Scatterplot of the experimental (x axis) against the predicted (y axis) $\Delta\Delta G$ for four protein sequences, each associated with a different protein family (PF00014, PF00018, PF00041, and PF00887). In each panel, each point represents a single-site mutation and the predicted free-energy variation $\Delta\Delta G^{\text{pred}}$ value is obtained using the early-stopped solution (orange squares) or the model sampled at T^{best} (blue triangles) with the best Spearman correlation coefficient (see Fig. 9) for that specific sequence.

- [1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, Attention is all you need, [arXiv:1706.03762](#).
- [2] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Židek, A. Potapenko, *et al.*, Highly accurate protein structure prediction with AlphaFold, *Nature (London)* **596**, 583 (2021).
- [3] Z. Lin, H. Akin, R. Rao, B. Hie, Z. Zhu, W. Lu, N. Smetanin, R. Verkuil, O. Kabeli, Y. Shmueli, A. D. Santos Costa, M. Fazel-Zarandi, T. Sercu, S. Candido, and A. Rives, Evolutionary-scale prediction of atomic level protein structure with a language model, [bioRxiv](#) (2022).
- [4] H. S. Seung, H. Sompolinsky, and N. Tishby, Statistical mechanics of learning from examples, *Phys. Rev. A* **45**, 6056 (1992).
- [5] R. Monasson and D. O’Kane, Domains of solutions and replica symmetry breaking in multilayer neural networks, *Europhys. Lett.* **27**, 85 (1994).
- [6] Y. Bahri, J. Kadmon, J. Pennington, S. S. Schoenholz, J. Sohl-Dickstein, and S. Ganguli, Statistical mechanics of deep learning, *Annu. Rev. Condens. Matter Phys.* **11**, 501 (2020).
- [7] C. Baldassi, A. Ingrosso, C. Lucibello, L. Saglietti, and R. Zecchina, Subdominant dense clusters allow for simple learning and high computational performance in neural networks with discrete synapses, *Phys. Rev. Lett.* **115**, 128101 (2015).
- [8] C. Baldassi, C. Borgs, J. T. Chayes, A. Ingrosso, C. Lucibello, L. Saglietti, and R. Zecchina, Unreasonable effectiveness of learning neural networks: From accessible states and robust ensembles to basic algorithmic schemes, *Proc. Natl. Acad. Sci. USA* **113**, E7655 (2016).
- [9] J. W. Rocks and P. Mehta, Memorizing without overfitting: Bias, variance, and interpolation in overparameterized models, *Phys. Rev. Res.* **4**, 013201 (2022).
- [10] S. Becker, Y. Zhang, and A. A. Lee, Geometry of energy landscapes and the optimizability of deep neural networks, *Phys. Rev. Lett.* **124**, 108301 (2020).
- [11] A. Zambon, E. M. Malatesta, G. Tiana, and R. Zecchina, Sampling the space of solutions of an artificial neural network, *Phys. Rev. E* **112**, 045303 (2025).
- [12] L. Liu, X. Liu, J. Gao, W. Chen, and J. Han, Understanding the difficulty of training transformers, [arXiv:2004.08249](#).
- [13] Y. Tian, Y. Wang, B. Chen, and S. Du, Scan and snap: Understanding training dynamics and token composition in 1-layer transformer, [arXiv:2305.16380](#).
- [14] E. Nichani, A. Damian, and J. D. Lee, How transformers learn causal structure with gradient descent, [arXiv:2402.14735](#).
- [15] R. Rende, F. Gerace, A. Laio, and S. Goldt, Mapping of attention mechanisms to a generalized Potts model, *Phys. Rev. Res.* **6**, 023057 (2024).
- [16] F. Morcos, A. Pagnani, B. Lunt, A. Bertolino, D. S. Marks, C. Sander, R. Zecchina, J. N. Onuchic, T. Hwa, and M. Weigt, Direct-coupling analysis of residue coevolution captures native contacts across many protein families, *Proc. Natl. Acad. Sci. USA* **108**, E1293 (2011).
- [17] A. Zambon, F. Caruso, R. Zecchina, and G. Tiana, Controlled Langevin dynamics for sampling of feedforward neural networks trained with minibatches, [arXiv:2603.15367](#).
- [18] B. B. Kragelund, C. V. Robinson, J. Knudsen, C. M. Dobson, and F. M. Poulsen, Folding of a four-helix bundle: Studies of acyl-coenzyme a binding protein, *Biochemistry* **34**, 7217 (1995).
- [19] G. Franco, M. Cagiada, G. Bussi, and G. Tiana, Statistical mechanical properties of sequence space determine the efficiency of the various algorithms to predict interaction energies and native contacts from protein coevolution, *Phys. Biol.* **16**, 046007 (2019).
- [20] G. Tiana and L. Sutto, Equilibrium properties of realistic random heteropolymers and their relevance for globular and naturally unfolded proteins., *Phys. Rev. E* **84**, 061910 (2011).
- [21] M. Punta, P. C. Coghill, R. Y. Eberhardt, J. Mistry, J. Tate, C. Boursnell, N. Pang, K. Forslund, G. Ceric, J. Clements, A. Heger, L. Holm, E. L. L. Sonnhammer, S. R. Eddy, A. Bateman, and R. D. Finn, The Pfam protein families database, *Nucl. Acids Res.* **40**, D290 (2012).
- [22] R. Rao, J. Meier, T. Sercu, S. Ovchinnikov, and A. Rives, Transformer protein language models are unsupervised structure learners, [bioRxiv](#) (2020).
- [23] R. Rao, J. Liu, R. Verkuil, J. Meier, J. F. Canny, P. Abbeel, T. Sercu, and A. Rives, MSA Transformer, in *Proceedings of the 38th International Conference on Machine Learning*, Vol. 139 (PMLR, 2021).
- [24] J. Vig, A. Madani, L. R. Varshney, C. Xiong, R. Socher, and N. F. Rajani, BERTology meets biology: Interpreting attention in protein language models, in *ICLR 2021 Conference* (ICLR, 2021), pp. 1–24.
- [25] L. J. McGuffin, K. Bryson, and D. T. Jones, The PSIPRED protein structure prediction server, *Bioinformatics* **16**, 404 (2000).
- [26] M.-H. Yu, J. S. Weissman, and P. S. Kim, Contribution of individual side-chains to the stability of BPTI examined by alanine-scanning mutagenesis, *J. Mol. Biol.* **249**, 388 (1995).
- [27] V. P. Grantcharova, D. S. Riddle, J. V. Santiago, and D. Baker, Important role of hydrogen bonds in the structurally polarized transition state for folding of the src SH3 domain, *Nat. Struct. Mol. Biol.* **5**, 714 (1998).
- [28] E. Cota, S. J. Hamill, S. B. Fowler, and J. Clarke, Two proteins with the same structure respond very differently to mutation: The role of plasticity in protein stability, *J. Mol. Biol.* **302**, 713 (2000).
- [29] F. M. Poulsen, B. B. Kragelund, P. Osmark, T. B. Neergaard, J. Schiødt, K. Kristiansen, and J. Knudsen, The formation of a native-like structure containing eight conserved hydrophobic residues is rate limiting in two-state protein folding of ACBP, *Nat. Struct. Biol.* **6**, 594 (1999).
- [30] E. I. Shakhnovich and A. M. Gutin, Engineering of stable and fast-folding sequences of model proteins., *Proc. Natl. Acad. Sci. USA* **90**, 7195 (1993).
- [31] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, *et al.*, Training compute-optimal large language models, [arXiv:2203.15556](#).
- [32] L. Ghiringhelli, A. Zambon, and G. Tiana, Github repository (2026), <https://github.com/guidotiana/PseudoLangevin>.

- [33] L. Ghiringhelli, A. Zambon, and G. Tiana, Supplementary material for: Sampling at intermediate temperatures is optimal for training large language models in protein structure prediction, UNIMI Dataverse, V1, 2026, https://doi.org/10.13130/RD_UNIMI/J1TOFK.
- [34] M. Steinegger and J. Söding, MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets, *Nat. Biotechnol.* **35**, 1026 (2017).
- [35] J. Thickstun, The transformer model in equations, 2020, <https://johnthickstun.com/docs/transformers.pdf>.