Extrapolation of Periodic Functions Using Binary Encoding of Continuous Numerical Values

Anonymous authors
Paper under double-blind review

Abstract

We report the discovery that binary encoding allows neural networks to extrapolate periodic functions beyond their training bounds. We introduce Normalized Base-2 Encoding (NB2E) as a method for encoding continuous numerical values and demonstrate that, using this input encoding, vanilla multi-layer perceptrons (MLP) successfully extrapolate diverse periodic signals without prior knowledge of their functional form. Internal activation analysis reveals that NB2E induces bit-phase representations, enabling MLPs to learn and extrapolate signal structure independently of position.

1 Introduction

The choice of input representation fundamentally shapes what neural networks are capable of learning through inductive bias. For coordinate-based learning, various encoding methods have been developed to enable networks to capture high-frequency features. Random Fourier features were proposed by Rahimi & Recht (2007) as a shift-invariant kernel method and adapted by Tancik et al. (2020) as a means of encoding for neural networks, whereby

$$\gamma(x) = [\cos(2\pi\omega_1 x), \sin(2\pi\omega_1 x), \cos(2\pi\omega_2 x), \sin(2\pi\omega_2 x), \ldots], \tag{1}$$

where $\boldsymbol{\omega}_i \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}), \quad i = 1, 2, \ldots, m.$

The random Fourier features of Equation 1 were further developed for use in neural radiance fields (NeRF, Mildenhall et al., 2020) using deterministic features. That is, for frequencies $\omega_i = 2^{i-1}$ where $i = 1, 2, \ldots, N$, fixed Fourier features are given by

$$\gamma(x) = [\sin(2\pi x), \cos(2\pi x), \sin(4\pi x), \cos(4\pi x), \sin(8\pi x), \cos(8\pi x), \ldots]. \tag{2}$$

Hereafter, to compare to our method and in recognition of its functional implementation as encoding coordinates, we will refer to this method as Fixed Fourier Encoding (FFE).

In the same paper that Rahimi & Recht (2007) introduced Fourier features, the authors also introduced random binning as a kernel method, with further analysis provided by Rahimi & Recht (2008). In random binning, the spatial domain is randomly partitioned (as opposed to the frequency domain with Fourier features), where a partition containing a given value is represented by unity and zero otherwise. In this manner, it is representing a continuous value in a discrete space. As FFE can be considered a deterministic form of random Fourier features, our concept can be considered a deterministic form of random spatial binning. That is, the near-exact representation of a continuous value with a binary vector.

Binary encoding provides a fundamentally different representation than FFE. Both can be viewed as multi-resolution decompositions in that FFE provides continuous sinusoidal basis functions and binary encoding provides discrete step functions at the same hierarchical frequency scales. The discrete nature of binary representation induces different learning dynamics in neural networks. We explore this through Normalized Base-2 Encoding (NB2E), which encodes continuous values in $[0, 1)$ using their binary representation as a vector in the form

$$\gamma(x) = [B_1, B_2, \ldots, B_N], \quad \text{where } x = \sum_{i=1}^{N} B_i \times 2^{-i},\ B_i \in \{0, 1\}. \tag{3}$$

In this paper, we report a surprising discovery: NB2E enables vanilla multi-layer perceptrons (MLPs) to extrapolate periodic functions beyond their training bounds, while FFE with identical frequencies and standard continuous numerical input both fail at this task. Through systematic investigation, we demonstrate that NB2E induces bit-phase internal representations that allow networks to learn periodic structure independently of position, enabling extrapolation across unseen regions of the input domain.

This paper is structured as follows. In Section 2 we will introduce NB2E and the limits thereof in Section 3. Our results are shown in Section 4, with subsections addressing (i) the comparison of NB2E to FFE and continuous input, (ii) sensitivity studies, (iii) examination of the activations and (iv) periodic activation functions. We provide a discussion in Section 5 and our conclusions in Section 6.

2 Normalized Base-2 Encoding

NB2E implementation first requires the normalization of the input data to a $[0, 1)$ scale, followed by an elementwise encoding of the base-2 representation. While normalization is not a strict requirement for binary encoding in general, we note that consistency of the meaning of each element of the binary vector is critical and normalization ensures that each continuous numerical value is encoded within the same context. We also found normalization to be more amenable to managing the 2^N boundaries of the encoding (discussed in Section 3). In the lower panel of Figure 1, we show an example NB2E representation of the number 0.987654321.

For all NB2E, the encoded vector of length N has a representation given by Equation 3. Clearly, the minimum N2BE representable value is 0, while the maximum representable value is $\sum_{i=1}^{N} 2^{-i}$, approaching a limit of unity with increasing N. Represented values can naturally be more precise as N increases. Selection of N should be customized to the needed precision of the application. We found diminishing value to increasing N beyond 32, with a minimum nonzero representable value of $\frac{1}{2^{32}} \approx 4.7 \times 10^{-10}$, but found $N = 48$, with a minimum nonzero representable value of $\frac{1}{2^{48}} \approx 3.6 \times 10^{-15}$, to provide an acceptable balance between representable precision and what is effectively noise. Any computational expense from varying N is negligible, as the size of our neural network (described in Section 4) remains the same, and the only modification to the number of parameters caused by N is in the weight and bias arrays between the input and the first layer.

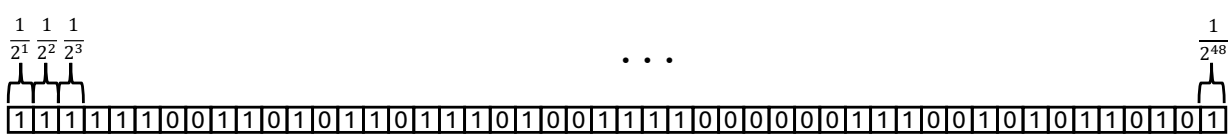

Figure 1: 48-element NB2E of the number 0.987654321. The given value is encoded into a vector of ones and zeros in the manner described by Equation 3 to be used as input to a neural network.

3 NB2E Limits

In order for a neural network to properly learn the meaning of the NB2E, each element of the encoded vector must have contained values of both zero and unity during training. The first element of the NB2E vector (see Figure 1) is $\frac{1}{2^1} = 0.5$, so will not be unity for any number less than 0.5. As such, the training data needs to extend beyond 0.5 in order to ensure the training of the first element of the encoding. That is, for input dataset X, normalized to X' with subsets X'_{train} and X'_{test},

$$0 \leq \min(X'_{\text{train}}) < 0.5 < \max(X'_{\text{train}}) < \min(X'_{\text{test}}) < \max(X'_{\text{test}}) < 1. \quad (4)$$

We emphasize that X' is normalized and therefore the size of the validation domain (where one would extrapolate) scales with the domain of the non-normalized dataset.

In order to expand the upper limit, effectively increasing the extrapolation horizon, normalization in NB2E need not be to the maximum value of the dataset. That is, for dataset X and normalizing value z, the normalized dataset X' is obtained by the elementwise division

$$X' = \frac{X}{z}, \quad \text{where } z < 2\max(X), \tag{5}$$

such that $\max(X') > 0.5$, abiding by Equation 4. This is an upper limit for z, however, and reasonable values should be much lower to ensure sufficient training. For example, if a dataset is a time series for a duration of ten years and the desired extrapolation is two years into the future, simply divide the dataset by twelve years, i.e. $z = 1.2\max(X)$. Train and test using $X' <= \frac{5}{6}$, then extrapolate to any value x' within $\frac{5}{6} < x' < 1$. The extrapolation is then valid as a prediction for two years into the future, where $x = x'z$.

The normalizing value, z, should be carefully chosen and informed by the domain of the data and desired prediction domain, with consideration that Equations 4 and 5 are limits rather than ideal values. We will examine the sensitivity of the model predictions to the maximum value of the training domain in Section 4.2, but will mention here that we suggest $\max(X'_{\text{train}}) >= 0.7$ and as a guiding principle for practical application.

4 Results

We tested the extrapolative ability of NB2E against FFE (described in Section 1, with mathematical representation given by Equation 2) and continuous numerical input. Our neural network (shown in Figure 2) is a simple Multi-Layer Perceptron (MLP) with five dense layers containing 512 neurons each and using exponential linear unit (ELU) activation (Clevert et al., 2016), with linear activation of a single neuron on the final layer. $L2$ regularization with a regularization factor of 10^{-4} is applied to each hidden layer to prevent overfitting. Note that we do not use periodic activation functions (Parascandolo et al., 2016; Sitzmann et al., 2020) which, while they have not been claimed to allow for extrapolation, do provide internal representation of complex signals implicitly, whereas we wanted to test the qualities of the encoding directly.

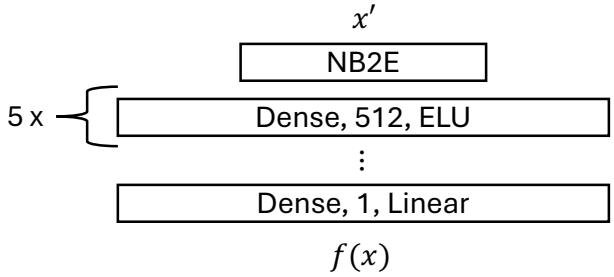

Figure 2: MLP structure. The NB2E is not part of the neural network, just noted as the encoding of the input. Replace NB2E with FFE for the appropriate examples, whereas for continuous numerical input there is no encoding. There are five dense layers of 512 neurons each with ELU activation and $L2$ regularization with a regularization factor of 10^{-4}, followed by a dense layer with a single neuron and linear activation.

For each case in X'_{train}, the NB2E of x' is given as input to the neural network, and $f(x)$ is given as a training target. We emphasize the previous sentence for clarity, noting that $f(x)$ is the value of the function at the non-normalized x. There are no dependencies trained as in sequence models, only the mapping from a single input to a single output per training sample. Training batches have inputs of shape $(N_{\text{batch}}, 48)$ for NB2E (see Section 2 for discussion of the size of the NB2E encoding), $(N_{\text{batch}}, 96)$ for FFE (which requires a sine and cosine term for each frequency, hence double the size of the NB2E, see Table 1), and $(N_{\text{batch}}, 1)$ for continuous numerical input. Outputs, in all cases, are of shape $(N_{\text{batch}}, 1)$. We found batches of size $N_{\text{batch}} = 1000$ to train quickly and effectively.

Bit	Frequency	NB2E	FFE
1	$2^0 = 1$	2^{-1}	$\sin(2\pi x), \cos(2\pi x)$
2	$2^1 = 2$	2^{-2}	$\sin(4\pi x), \cos(4\pi x)$
3	$2^2 = 4$	2^{-3}	$\sin(8\pi x), \cos(8\pi x)$
4	$2^3 = 8$	2^{-4}	$\sin(16\pi x), \cos(16\pi x)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
i	2^{i-1}	2^{-i}	$\sin(2^i\pi x), \cos(2^i\pi x)$

Table 1: NB2E bits and the equivalent FFE. Each NB2E bit i with positional value 2^{-i} corresponds to FFE at frequency 2^{i-1}, creating a hierarchical representation with identical frequencies but different encodings (discrete for NB2E vs. continuous for FFE).

4.1 Examples

We first tested a basic sine function where $\{x_i\}_{i=1}^{10000} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(0, 100)$ and $f(x) = \sin(x)$. We imposed randomness on x in order to avoid any suspicion that the MLP could learn patterns from regular intervals. We normalized and split the data into ∼70/30 train/test in ordinal space such that $0.0 \leq X'_{\text{train}} \leq 0.7 < X'_{\text{test}} < 1.0$. We acknowledge this split as somewhat arbitrary, but we note that our motivation was to maximize the space for demonstration of extrapolative ability, so we exceed the standard 80/20 split. Note that the split is *approximately* 70/30 as x_i is randomly drawn from a uniform distribution and we required a hard limit to the maximum value of the training data (i.e. 0.7) to evenly assess the qualities of the NB2E encoding. None of the functions we test here are periodic on a power of two, nor can the NB2E bit patterns fully repeat, as each complete bit pattern in X'_{test} is unique and not seen in X'_{train}.

We used the AdamW optimizer (Kingma & Ba, 2015; Loshchilov & Hutter, 2019) and a cosine annealing learning rate scheduler with warm restarts (Loshchilov & Hutter, 2017)[1]. The Mean Absolute Error (MAE) was our objective function in all cases, where each example was trained for 4,000 epochs. On a single NVIDIA Tesla V100 GPU, training completes in 3-5 minutes.

The results of our first test are shown in Figure 3. The predictions of the training data are in blue and test data in red. Note, again, that $0.0 \leq X'_{\text{train}} \leq 0.7 < X'_{\text{test}} < 1.0$. During training, the MLP never saw the encoding of any value greater than 0.7. The most accurate predictions are clearly made by the MLP trained with NB2E input, followed closely by FFE, which makes good predictions immediately after the training domain but appears to drift in phase and amplitude. The prediction from continuous numerical input fails catastrophically, unable to fit even the training data.

Encouraged by the result, we rendered our test more challenging. We combined two sine functions rather arbitrarily, making $f(x) = \sin(x) + 2.5\sin(3.7 + 1.4x)$, with $\{x_i\}_{i=1}^{10000} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(0, 400)$, normalized to [0,1). The results are shown in Figure 4. Not only could the neural network trained with NB2E learn the more complex function, but it was also able to maintain stunning accuracy in the extrapolation beyond the training bounds. FFE fits the training data nicely, but fails to extrapolate. Continuous numerical input cannot even fit the training data.

We then attempted a more complex function without sines or cosines, testing a composite of sawtooth and triangle functions with different periods, given in detail in Appendix A. We used $\{x_i\}_{i=1}^{10000} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(0, 200)$, normalized to [0,1). The results are shown in Figure 5. While there are occasional large residuals, the MLP trained with NB2E again learned and extrapolated the function better than either of the other methods. While there are occasional large residuals in the prediction, the prediction generally fits quite well. FFE again fits the training data nicely, but the extrapolation only vaguely resembles the signal. Continuous numerical input again fails completely.

[1] Warm restarts reset the learning rate to a higher value at a fixed schedule during training, which causes reduced performance temporarily while the model seeks a new minimum. While not optimal for all applications, this method can improve outcomes in the presence of false local minima.

We continued to complicate the function, searching for a level of complexity where the NB2E MLP would no longer be viable. We created a composite periodic signal from three different functions, given in detail in Appendix A. The results for this function are shown in Figure 6. The MLP with NB2E input again demonstrated the ability to learn the function and extrapolate it better than the other two methods. Residuals understandably increase in this example with the complexity of the signal, but are again substantially smaller than residuals from the other methods. FFE again nicely fits the training data but struggles to extrapolate, while continuous numerical input yet again fails completely.

4.2 Sensitivity Analysis

Investigating the limits imposed by Equation 4, we examined the sensitivity of NB2E in our examples from Section 4.1 to the maximum normalized value of the training domain. We split the normalized data set X' into X'_{train} and X'_{test} while keeping the number of data points and the number of cycles of the periodic signal constant in X'_{train}. For $X'_{\text{train}} < p < X'_{\text{test}}$, we tested examples at increments of 0.1 from $p = 0.1$ through $p = 0.7$, training each for 4,000 epochs. The results for the sine function are shown in Figure 7, with the split between X'_{train} and X'_{test} identified in each plot by the gray dashed vertical line. Predictions for X'_{train} are in blue and predictions for X'_{test} are in red, with green dashed vertical lines from $x = 0.125$ to $x = 0.875$ in increments of 0.125, corresponding to any location where any of the first three bits of the NB2E will change (refer back to Figure 1 to see that the first element represents $\frac{1}{2}$, the second element $\frac{1}{4}$, the third element $\frac{1}{8}$, ... , the last element $\frac{1}{2^N}$.).

It can be easily seen that the prediction fails for any $p <= 0.5$, as suggested by Equation 4. As we discussed in Section 3, valid prediction cannot be made unless the MLP has been trained with each element of the NB2E vector. Training with $X'_{\text{train}} < 0.5$ does not train the first element of the NB2E vector. So, as expected for examples $p = 0.5$, $p = 0.4$, and $p = 0.3$, the prediction falls apart at $x' = 0.5$. For $p = 0.2$, it fails at $x' = 0.25$ and even more so at $x' = 0.5$, corresponding to the second and first NB2E bits, respectively. For $p = 0.1$, it fails progressively in stages at $x' = 0.125$, $x' = 0.25$, and $x' = 0.5$, corresponding to the third, second, and first NB2E bits, respectively. Also note that even for the example $p = 0.6$, which complies with Equation 4, the prediction fails at the bit transition point $x' = 0.75$. But, in that particular instance, it can also be seen that the model had difficulty fitting the training data at $x' = 0.25$, indicating that representations of the second bit trained poorly, as well as potentially the third or fourth bit due to the smaller error at $x' = 0.6875 = \frac{1}{2} + \frac{1}{8} + \frac{1}{16}$. At $p = 0.7$, as with our initial example in Section 4.1, the model has clearly learned the signal and extrapolates nicely.

Interestingly, the failure is not catastrophic with $p < 0.5$, as the basic shape of the sine curve and its period is maintained. It is, rather, a phase shift. In the case of a maximum value of X_{train} less than any of the noted bit transitions, the model output resets to the learned outputs from the rest of the NB2E vector after that bit transition. For example, for $p = 0.3$, $p = 0.4$, and $p = 0.5$, the outputs beginning at $x' = 0.5$ are as if they are beginning at $x' = 0$. Since, in these cases, the model has not learned the meaning of the first bit, it uses its knowledge of the rest of the bit patterns to make a prediction, almost as if the first bit does not exist. So, we can see rather clearly that the conditions of Equation 4 are critical to using NB2E and, in practical application, the maximum value of X_{train} should be as high as possible to ensure proper training of all the bit interrelationships of the NB2E while still leaving space for testing and extrapolation.

We also examined the sensitivity of our method to the number of complete cycles of the periodic signal in the training data. Again using specifically the sine curve example from Figure 3, we held constant the number of data points in X'_{train} and X'_{test} at a ∼70/30 split and $X'_{\text{train}} < 0.7 < X'_{\text{test}}$, while altering the number of complete cycles of the sine curve. The results are shown in Figure 8, with, from top to bottom, 1-7 cycles. The top two panels clearly show that neither one nor two cycles is sufficient for our method to learn anything resembling the sine curve. It begins to learn the periodicity and the range of the signal from 3-5 cycles with various bit-learning inefficiencies causing sudden jumps in residuals, then refines it to near-perfection by 7 cycles. More cycles continue to improve the refinement, but are visually indistinguishable.

We also investigated the sensitivity of the NB2E method to the number of data points. In our examples from Section 4, we used 10,000 data points in each case. The NB2E representation requires that the MLP learn interrelationships of every element of the input vector between each other as well as with the output

value. The more data points the MLP has seen, the more complex these internal representations can be. A dataset with fewer data points requires the MLP to learn broader generalities of these relationships. The examination of sensitivity to the size of data set is, then, an examination of the limits of generality in the MLP's internal representations. The results of our analysis are shown in Figure 9, which we provide as a plot of the sine curve example from Figure 3 with 250 to 2,000 data points in increments of 250 in each panel, from top to bottom. Rather surprisingly, the prediction of the test set still resembles a sine curve in the top panel with only 250 data points. The predictions, of course, improve as more data points are added. We provide only the figure of the sine curve, but we did examine each of our examples and found that the sensitivity is highly dependent on the data itself, but in most cases very practical even with small datasets.

Considering applicability of the NB2E method to noisy real-world signals, we also examined the ability of the MLP to extrapolate a sine curve with added random noise. We used the same format as the original sine curve example from Section 4.1, $\{x_i\}_{i=1}^{10000} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(0, 100)$, normalized to [0,1), and $f(x) = \sin(x)$. Again, the train/test split was ∼70/30 such that $0.0 \leq X'_{\text{train}} \leq 0.7 < X'_{\text{test}} < 1.0$. We applied Gaussian noise with zero mean and standard deviation in the range 0.25 to 2.0 in increments of 0.25. The results are shown in Figure 10. It can be seen that, despite the noise, the MLP with NB2E encoding is able to recover the sine curve, though the quality of the extrapolated signal understandably decreases with increasing noise. Note that in X'_{train}, we compare the prediction to the noisy signal whereas in X'_{test} we compare the prediction to the clean sine curve, resulting in a change in the nature of the residuals.

4.3 Activation Analysis

The success of NB2E at extrapolating periodic functions, especially in contrast to the equivalent FFE, raises the question of how the input is being processed by the MLP. We specifically examined the sine curve example for the most straightforward analysis. To answer this question, we examined the activations of each hidden layer using Uniform Manifold Approximation and Projection (UMAP, McInnes et al., 2018), projecting the 512-D activations to the 3-D space. We then clustered the UMAP projections using the SCIKIT-LEARN (Pedregosa et al., 2011) implementation of Density-Based Spatial Clustering of Applications with Noise (DBSCAN, Ester et al., 1996) in order to perform additional analysis on the contents of each group. We found clear structure to the internal activations, which we plot in Figure 11 by phase (left column), position (center column), and clustered group (right column) for each hidden layer of the MLP.

It can be seen in this figure that the activations of the NB2E MLP form an interestingly elegant structure. Separate groupings exist primarily distinguished by phase of the sine curve, while each phase-group spans the much of the positional space. Through the layers, the structure of the activations becomes more complex, initially forming rough groupings that stretch and intertwine as the shape of the manifold changes. A closer analysis of the contents of the clustered groups shows that each has a unique signature of three specific bits: 5, 6, and 7.

We examine this phenomenon closer in Figure 12. The upper panel shows the range of bits 5, 6, and 7 for each clustered group, the lower-left panel shows the distributions by phase of each group, and the lower-right panel shows the bit values for each group. In the top and lower-left panels, it can be seen that, with some overlap, the combination of these three bits near-uniquely represents phase of the signal. Note that there is a slight downward angle to the range of the determinative bits in the top panel, which indicates that these are not just groupings of phase, but also of bit values, as the MLP is unaware of the value of x'. So, for the phase-overlap regions of the group distributions, the internal representation of phase is not enough for valid predictions and the MLP also has to represent bit values in conjunction with phase, or bit-phase. Despite the fact that the neural network has never seen the NB2E of any point $x' > 0.7$, those points are still properly placed in the manifold, which suggests that the internal representations are not based on position. In grouping data points by phase, the MLP is learning an internal representation similar to a Fourier decomposition. Yet, it is a more complicated representation than a Fourier decomposition, because it also must know bit values and be able to represent a trend in their progression. We can then hypothesize that, with extrapolation, the MLP is not learning to extrapolate in positional space, but rather projecting the learned bit-phase-space onto the unlearned positional space.

The results of the analysis of the sine function activations naturally raise the question of internal representations in the case of composite signals. In the case of the sine function, the MLP generally learned to associate bit patterns with phase. So, with composite signals, does the MLP learn a single phase representation of the composite signal or does it learn to separate the signals? To examine this, we performed the same analysis on the example from Figure 4, the composite sine function.

While the groupings of the sine function activations were stable throughout all five layers of the MLP, the composite sine function activations are dynamic. We show the 3D UMAP projections of the activations in Figure 13. The first column shows the projections by phase of the first signal of the composite, $f(x) = \sin(x)$ (referred to hereafter and in the plots as 'Period 1', $P = 2\pi$), the second column shows the projections by phase of the second signal of the composite, $f(x) = 2.5\sin(3.7 + 1.4x)$ (referred to hereafter and in the plots as 'Period 2', $P = \frac{2\pi}{1.4} \approx 4.49$), the third column shows the projections by position x', and the fourth column shows the DBSCAN clustered groups. Layers are shown top to bottom, 1-5 in integer increments. In the first layer, the UMAP projection of the activations is spherical, generally organized by the phase of Period 1. In the second layer, they form two clear groups. An examination of the contents of these groups showed that they interestingly separate only by the value of bit 7. This is a perfect separation with no overlap. Bit 7 obviously does not separate the two signals, so the organization of the activations in this manner clearly has value to the following layers. As another interesting phenomenon, the two periods seem to be represented orthogonally within the two groups. The groupings expand in subsequent layers with interleaved structure such that we can no longer trust that DBSCAN is properly identifying *all* of the independent groups with such a complicated structure. However, we do find in the fifth layer that four groups are identified, each containing two distinct bit patterns for bits 4 and 7. We show the bit signatures and phase distributions of the fifth layer in Figure 14, with the groups broken down by subgroups with unique bit 4 and 7 patterns and indicating the fraction of the group with each unique signature.

We can see from the plots of x' vs. phase for each group in the upper panel of Figure 14, where 'Period 1' is the darker shade and 'Period 2' is the lighter shade of each color, that, with the composite signal, the MLP is learning to internally represent activations in terms of both phases. Clear bands of lighter vs. darker colored scatter points indicate the phase groupings. We can also see that these bands are diagonal, shifting the phase grouping with position, which we interpret to mean that the MLP has learned a 'trend' in the progression of bit positions, as it is unaware of the value of x'. This trend was present in the activations of the sine curve as well (see the top panel of Figure 11), though much less exaggerated.

We note from the bottom left panel of Figure 14 that the learned phase distributions for Period 1 are nearly identical for groups 1/4 and groups 2/3. All groups resemble Gaussian distributions. These distributions account for the dark bands that we see in the subplots of the upper panel of Figure 14. The distributions of Period 2, in the lower middle panel of Figure 14, also resemble Gaussian distributions but are much broader in phase, with groups 1/3 and 2/4 having similar distributions, in contrast to Period 1.

We conclude that, with composite periodic signals, NB2E induces implicit joint representations of bit-phase, as the activations are clearly multi-dimensional functions of the composite phases and bit patterns. Extrapolation, then, is achieved through an internal representation of periodic signals in bit-phase space rather than in positional space. Projection of the generalized bit-phase representations onto the positional axis allows for positional extrapolation within the limits of the NB2E representation.

4.4 Periodic Activations

Earlier in this section, we briefly mentioned that we were not using periodic activation functions in order to assess the quality of the encoding directly. We wanted to avoid implicit representations of the signals, which are induced by sinusoidal activation as demonstrated in sinusoidal representation networks (SIREN, Sitzmann et al., 2020). However, our conclusions regarding the underlying mechanism of NB2E MLPs raised the question of whether the implicit signal representations of sinusoidal activation would complement the strengths of NB2E.

To address this question, we ran our experiments again, changing only the activation from the ELU function to the sine function. We compare each in Figure 15, with ELU activations in the upper panels and sine activations in the lower panels. We did not attempt a new sensitivity analysis or activation analysis with

the periodic activations. However, from Figure 15, we can conclude that a MLP with NB2E input and sinusoidal activations does indeed improve the prediction noticeably for our first three examples, without much effect in the fourth. Residuals are plotted on the same scale for each example to make the difference visually perceptible. We emphasize that we changed nothing about the MLP except for the activations for this comparison. Though our analysis of the periodic activation functions combined with NB2E given here is minimal, we conclude that the sine function is clearly the superior activation for use with NB2E and we suspect that there is additional benefit to be gained by fine-tuning the MLP or improving the structure in conjunction with sinusoidal activation.

5 Discussion

NB2E demonstrates the unique ability to induce internal bit-phase representations that enable MLPs to extrapolate periodic functions beyond their training bounds. This capability is distinct from existing approaches: symbolic regression (Schmidt & Lipson, 2009; Martius & Lampert, 2016; Udrescu & Tegmark, 2020) discovers explicit functional forms through search, Physics-informed Neural Networks (PINNs, Raissi et al., 2019) leverage known governing equations, and Neural ODEs (Chen et al., 2018) model temporal derivatives but do not inherently capture periodicity. To our knowledge, NB2E is the first method that enables extrapolation of unknown periodic functions through input encoding alone, without architectural modifications or prior knowledge of the functional form.

It is also important to note the characteristic that predictions from a MLP with NB2E input are equally valid through the entire extrapolation domain. Compare this to sequence-based models, which tend to accumulate error with each additional step in the prediction horizon. Using NB2E, there is no change in the nature of the input in the domain [0.5, 1.0), i.e. it is a binary vector of a fixed length where the training data contained both 0 and 1 in each vector element throughout the course of training. Given this input representation, a prediction at the upper bound of the extrapolation domain is fundamentally no different from a prediction at the lower bound.

Expanding on this concept, another valuable characteristic of the NB2E MLP predictions are the continuity thereof. Sequence-based methods make predictions at fixed intervals, whereas predictions with NB2E can be made for the entire set of representable numbers less than unity. This novel feature of NB2E extrapolation lends itself to use with *irregular* time series data. Sequence prediction models tend to ignore the actual value of the progression axis (generally time or position) by assuming regular sampling. In this manner, these models learn dependencies on *order* rather than any relationship with the axis. This severely limits any prediction to the regular sampling cadence (in applications where regularity is inherent, a regular cadence is of course a feature rather than a limitation). Prediction at *any* value of time or position, however, as provided by NB2E, has the potential to enhance prediction models with a continuous extrapolation domain.

Consider, in particular, the representation of time, from research into which our initial development of NB2E was derived. Although we framed our examples in this work in terms of position, x, the axis could just as easily be time, t. While cyclical time such as seasonality is representable by a sinusoid and calendar dates are representable categorically, an effective representation of linear time remains elusive. In ML applications with time-series data, the actual value of time tends to be ignored as discussed in the previous paragraph regarding regular sampling, where the actual value of time becomes irrelevant. Representation of a time-series as a 1D shape (see, e.g., Kostov et al., 2025; Powell et al., 2021, for astrophysical applications) is another way to ignore the numerical value of time. Even the Time2Vec (Kazemi et al., 2019) method of time embedding is not a representation of time as much as it is a means for learning temporal features. Time can be considered as more of a label than a feature and, as such, requires different treatment, ideally needing to be a *magnitude-free representation of magnitude*. Such a representation is provided by NB2E. We speculate that, in sequence applications where the value of time has a relationship with the output, the NB2E representation of time could prove to be powerful as an input feature of a sequence prediction model.

Expanding on the concept of *magnitude-free representation of magnitude* that NB2E provides, we emphasize that it encodes continuous numerical values in a way that preserves order and relationships without absolute scale. Each bit represents relative position in a hierarchical decomposition rather than an absolute value. This creates a coordinate system where "distance" between values is encoded through shared bit patterns,

not through numerical differences that networks must learn to interpret. This principle may extend beyond just a representation of time. Any continuous quantity requiring both precise representation and scale-independent learning could benefit from similar binary hierarchical encodings. The success of NB2E with extrapolation suggests that, when generalization must occur independently of absolute position, discrete positional encodings with hierarchical structure may outperform continuous alternatives.

The contrast with Fourier encoding is instructive. In our tests, both NB2E and FFE provide multi-scale decompositions at identical frequencies, yet NB2E enables extrapolation while FFE does not. While we leave the theoretical foundation of this phenomenon to future work, we emphasize that this suggests the geometry of representation space (discrete versus continuous) may matter more than the frequency content for certain learning tasks.

The prediction limits that we described in Section 3 as well as additional limitations found in the sensitivity analysis in Section 4.2 are substantial constraints on the use of NB2E. Due to the necessity of learning the relationships between each element of the encoding, a neural network trained with NB2E input can only predict forward to the next power of two greater than the maximum input value of the training data. In Section 3 we discussed a method of normalization to render this limitation much less obstructive. We also suggest that future work could establish methods of working with a base-2 input vector in terms of relative rather than absolute positions. That is, to learn relationships between relative positional differences (as in Shaw et al., 2018, for transformer positional encoding) in the vector such that powers of two do not form boundaries in the prediction space.

6 Conclusions

We have demonstrated that binary encoding enables neural networks to extrapolate periodic functions beyond their training bounds, which is a capability not previously shown with any coordinate encoding method. Using NB2E, vanilla MLPs successfully extrapolate diverse periodic signals from a simple sine function to complex composite functions, while FFE and continuous inputs both fail at this task despite identical network architectures and training procedures.

Through systematic empirical characterization, we established the requirements and limitations of this approach: extrapolation requires training on several complete cycles with the training domain extending beyond the 0.5 threshold in normalized space, while predictions remain valid to a limit of unity. While this seems rather constraining, we identified in Section 3 that normalization need not be to the maximum value of the dataset, leaving ample room for predictive extrapolation. The NB2E method demonstrates robustness to noise and sparse data, with prediction quality maintained throughout the continuous extrapolation domain rather than degrading with discretized distance as in sequence-based approaches.

Our activation analysis reveals that NB2E induces bit-phase representations, where networks learn how bit patterns and phase interact rather than simply mapping positions to outputs. This position-independent learning mechanism enables extrapolation by projecting learned bit-phase dynamics onto unseen positional coordinates. For composite signals, the NB2E MLP represents multiple coupled phases simultaneously, all within a simple feedforward architecture.

While we have characterized when and how this capability emerges empirically, the theoretical question of why discrete encodings fundamentally change the learning dynamics remains open. Understanding the mathematical principles underlying bit-phase learning and formalizing the conditions under which binary representations enable extrapolation are important directions for future work. Additionally, extensions to multi-dimensional inputs and integration with modern architectures beyond vanilla MLPs offer promising avenues for expanding this capability.

References

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, vol-

ume 31, pp. 6571–6583, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *International Conference on Learning Representations (ICLR)*, 2016. URL https://arxiv.org/abs/1511.07289.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD '96)*, pp. 226–231. AAAI Press, 1996.

Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: Learning a vector embedding of time. In *International Conference on Machine Learning (ICML) Workshop on Time Series*, 2019. doi: 10.48550/arXiv.1907.05321.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. doi: 10.48550/arXiv.1412.6980.

Veselin B. Kostov, Brian P. Powell, Aline U. Fornear, Marco Z. Di Fraia, Robert Gagliano, Thomas L. Jacobs, Julien S. de Lambilly, Hugo A. Durantini Luca, Steven R. Majewski, Mark Omohundro, Jerome Orosz, Saul A. Rappaport, Ryan Salik, Donald Short, William Welsh, Svetoslav Alexandrov, Cledison Marcos da Silva, Erika Dunning, Gerd Gühne, Marc Huten, Michiharu Hyogo, Davide Iannone, Sam Lee, Christian Magliano, Manya Sharma, Allan Tarr, John Yablonsky, Sovan Acharya, Fred Adams, Thomas Barclay, Benjamin T. Montet, Susan Mullally, Greg Olmschenk, Andrej Prša, Elisa Quintana, Robert Wilson, Hasret Balcioglu, Ethan Kruse, and The Eclipsing Binary Patrol Collaboration. The TESS Ten Thousand Catalog: 10,001 Uniformly Vetted and Validated Eclipsing Binary Stars Detected in Full-frame Image Data by Machine Learning and Analyzed by Citizen Scientists. *The Astrophysical Journal Supplement Series*, 279(2):50, August 2025. doi: 10.3847/1538-4365/ade2d8.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://arxiv.org/abs/1608.03983.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://arxiv.org/abs/1711.05101.

Georg Martius and Christoph H Lampert. Extrapolation and learning equations. *arXiv preprint arXiv:1610.02995*, 2016. doi: 10.48550/arXiv.1610.02995.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018. doi: 10.21105/joss.00861.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pp. 405–421. Springer, 2020. doi: 10.1007/978-3-030-58452-8_24.

Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. Taming the waves: sine as activation function in deep neural networks. In *International Conference on Learning Representations (ICLR) Workshop*, 2016. URL https://openreview.net/forum?id=Sks3zF9eg.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Brian P. Powell, Veselin B. Kostov, Saul A. Rappaport, Tamás Borkovits, Petr Zasche, Andrei Tokovinin, Ethan Kruse, David W. Latham, Benjamin T. Montet, Eric L. N. Jensen, Rahul Jayaraman, Karen A. Collins, Martin Mašek, Coel Hellier, Phil Evans, Thiam-Guan Tan, Joshua E. Schlieder, Guillermo Torres, Alan P. Smale, Adam H. Friedman, Thomas Barclay, Robert Gagliano, Elisa V. Quintana, Thomas L. Jacobs, Emily A. Gilbert, Martti H. Kristiansen, Knicole D. Colón, Daryll M. LaCourse, Greg Olmschenk,

Mark Omohundro, Jeremy D. Schnittman, Hans M. Schwengeler, Richard K. Barry, Ivan A. Terentev, Patricia Boyd, Allan R. Schmitt, Samuel N. Quinn, Andrew Vanderburg, Enric Palle, James Armstrong, George R. Ricker, Roland Vanderspek, S. Seager, Joshua N. Winn, Jon M. Jenkins, Douglas A. Caldwell, Bill Wohler, Bernie Shiao, Christopher J. Burke, Tansu Daylan, and Joel Villaseñor. TIC 168789840: A Sextuply Eclipsing Sextuple Star System. *The Astronomical Journal*, 161(4):162, April 2021. doi: 10.3847/1538-3881/abddb5.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, volume 20, pp. 1177–1184, 2007. URL `https://proceedings.neurips.cc/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf`.

Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems*, volume 21, pp. 1313–1320, 2008. URL `https://papers.nips.cc/paper/2008/hash/0efe32849d230d7f53049ddc4a4b0c60-Abstract.html`.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324 (5923):81–85, 2009. doi: 10.1126/science.1165893.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 464–468, 2018. doi: 10.18653/v1/N18-2074. URL `https://aclanthology.org/N18-2074/`.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7462–7473, 2020. doi: 10.48550/arXiv.2006.09661.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7537–7547, 2020. doi: 10.48550/arXiv.2006.10739.

Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020. doi: 10.1126/sciadv.aay2631.

A Composite Signal Functions

To create the function from Figure 5, we defined the following periodic functions:

$$\text{sawtooth}(x, p) = 2\left(\frac{x}{p} - \left\lfloor \frac{x}{p} + 0.5 \right\rfloor\right),$$

a centered sawtooth wave with period p and range $[-1, 1]$, and

$$\text{triangle}(x, p) = 2|\text{sawtooth}(x, p)| - 1,$$

a triangle wave with period p and range $[-1, 1]$. The composite function with arbitrarily selected parameters can be written compactly as:

$$f(x) = \text{sawtooth}(x, 3.1) + \text{triangle}(x, 5).$$

To create the function from Figure 6, we defined the following periodic functions:

$$\text{beat}(x, p_1, p_2, A) = \frac{A}{2}\left(\sin\left(\frac{2\pi x}{p_1}\right) + \sin\left(\frac{2\pi x}{p_2}\right)\right),$$

a beat frequency pattern with periods p_1 and p_2 and amplitude A,

$$\text{expdecay}(x, p, \lambda, A) = A \exp\left(-\lambda\left(\frac{x}{p} - \left\lfloor \frac{x}{p} \right\rfloor\right)\right),$$

an exponential decay that resets each period p with decay rate λ and amplitude A, and

$$\text{square}(x, p, A) = A \cdot \text{sgn}\left(\sin\left(\frac{2\pi x}{p}\right)\right),$$

a square wave with period p and amplitude A, where $\text{sgn}(\cdot)$ denotes the sign function. Given arbitrarily selected parameters, the composite function can be written as:

$$f(x) = \text{beat}(x, 2.1, 2.3, 4.0) + \text{expdecay}(x, 7.2, 9.7, 2.0) + \text{square}(x, 12.2, 0.7).$$

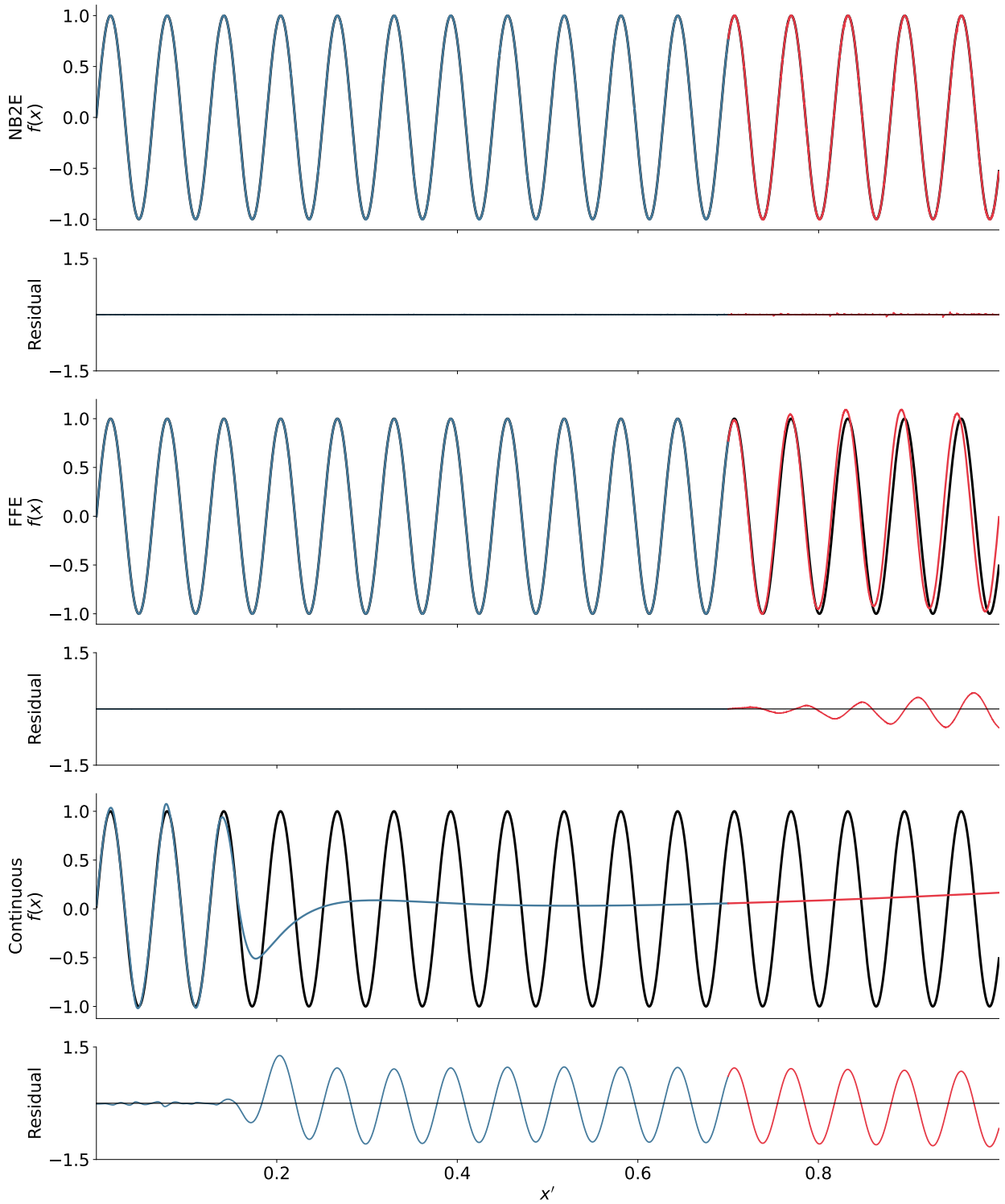

Figure 3: Results of training the MLP from Figure 2 with encoded input data for $f(x) = \sin(x)$. Predictions for the training data (blue) and test data (red) for (*top*) NB2E, (*middle*) FFE, and (*bottom*) continuous, with residuals plotted on the same scale for visual comparison. The true function is in black.

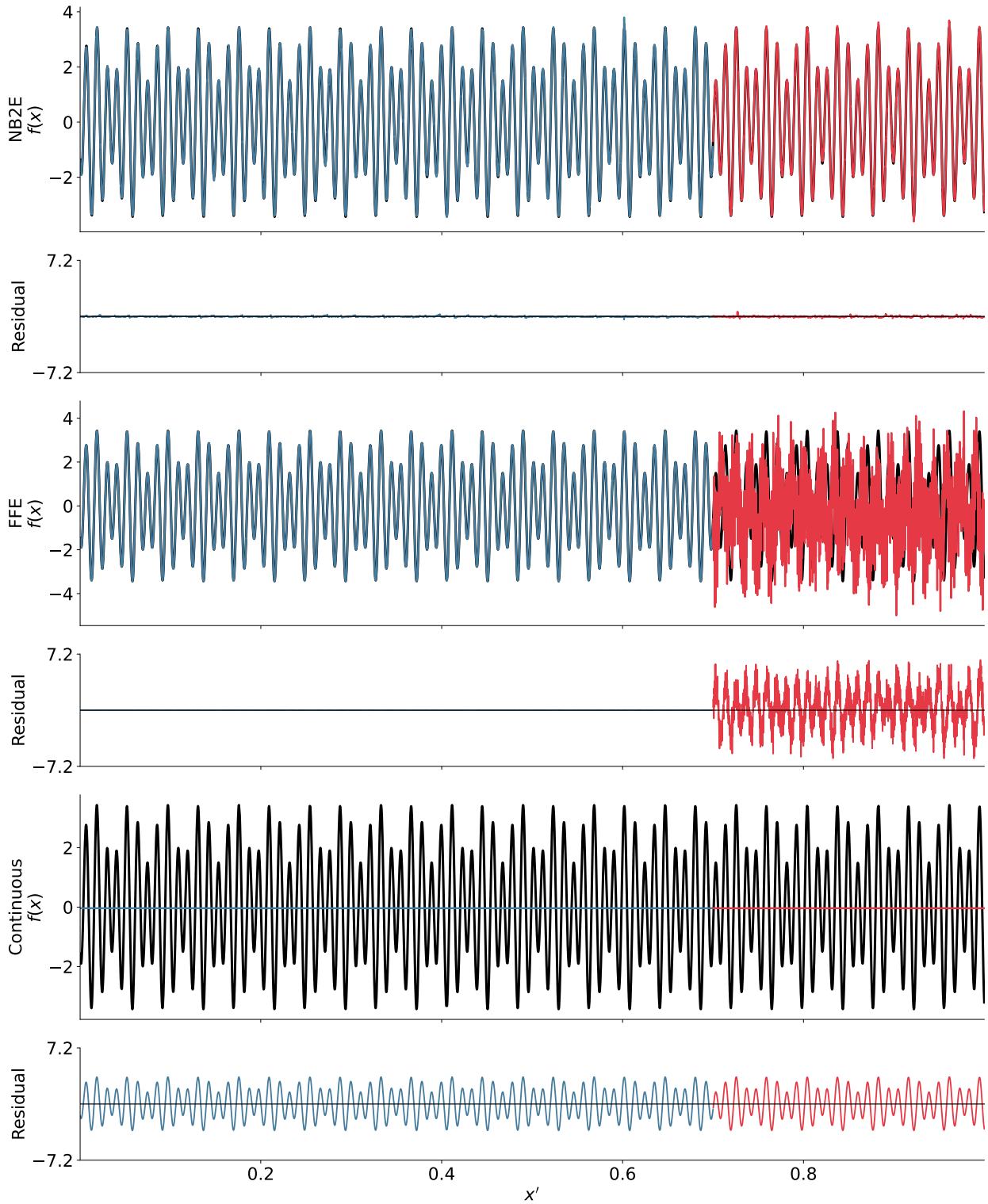

Figure 4: Same as Figure 3 but for the composite sine function $f(x) = \sin(x) + 2.5\sin(3.7 + 1.4x)$.

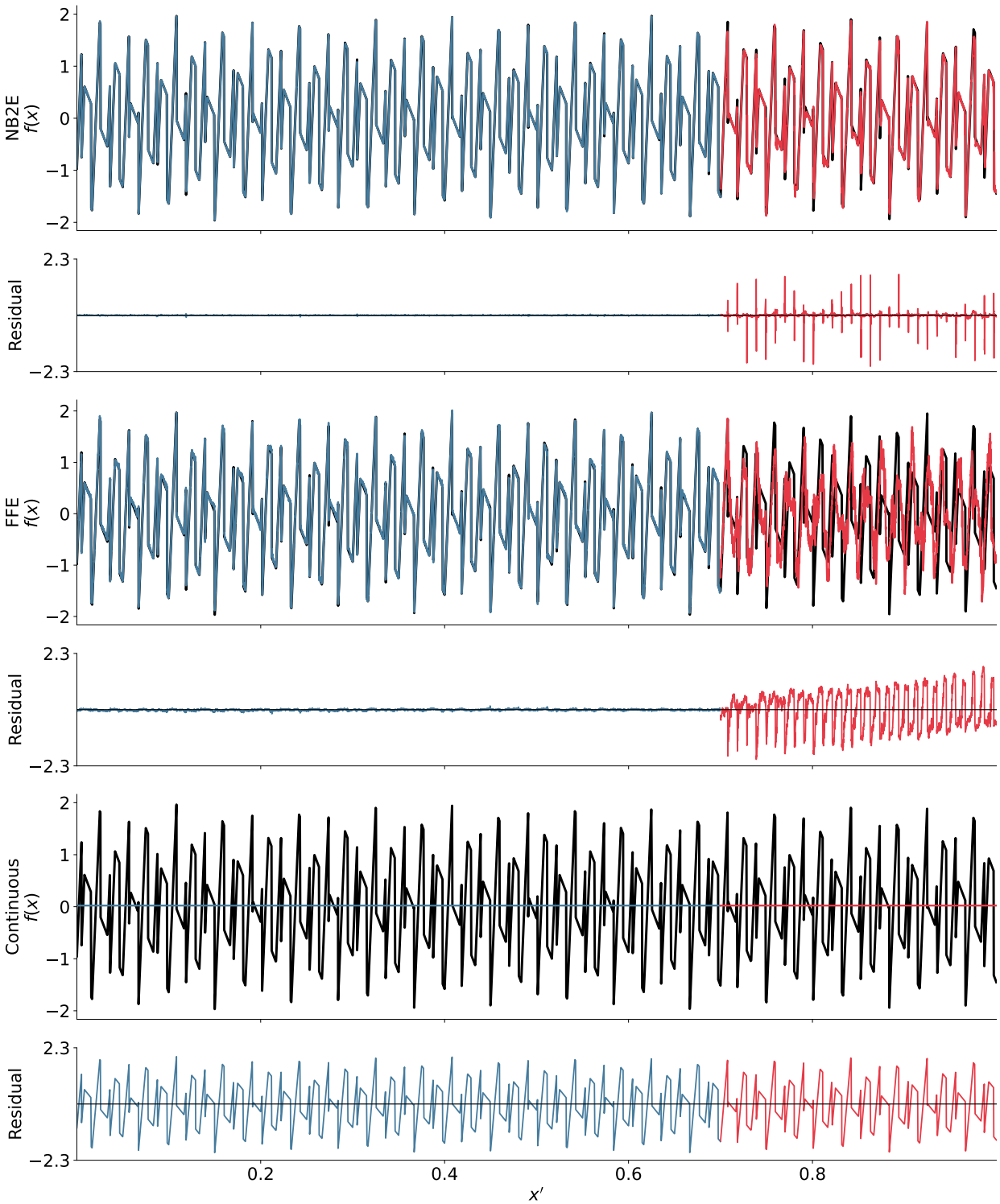

Figure 5: Same as Figures 3 and 4 but for the composite function given in Appendix A

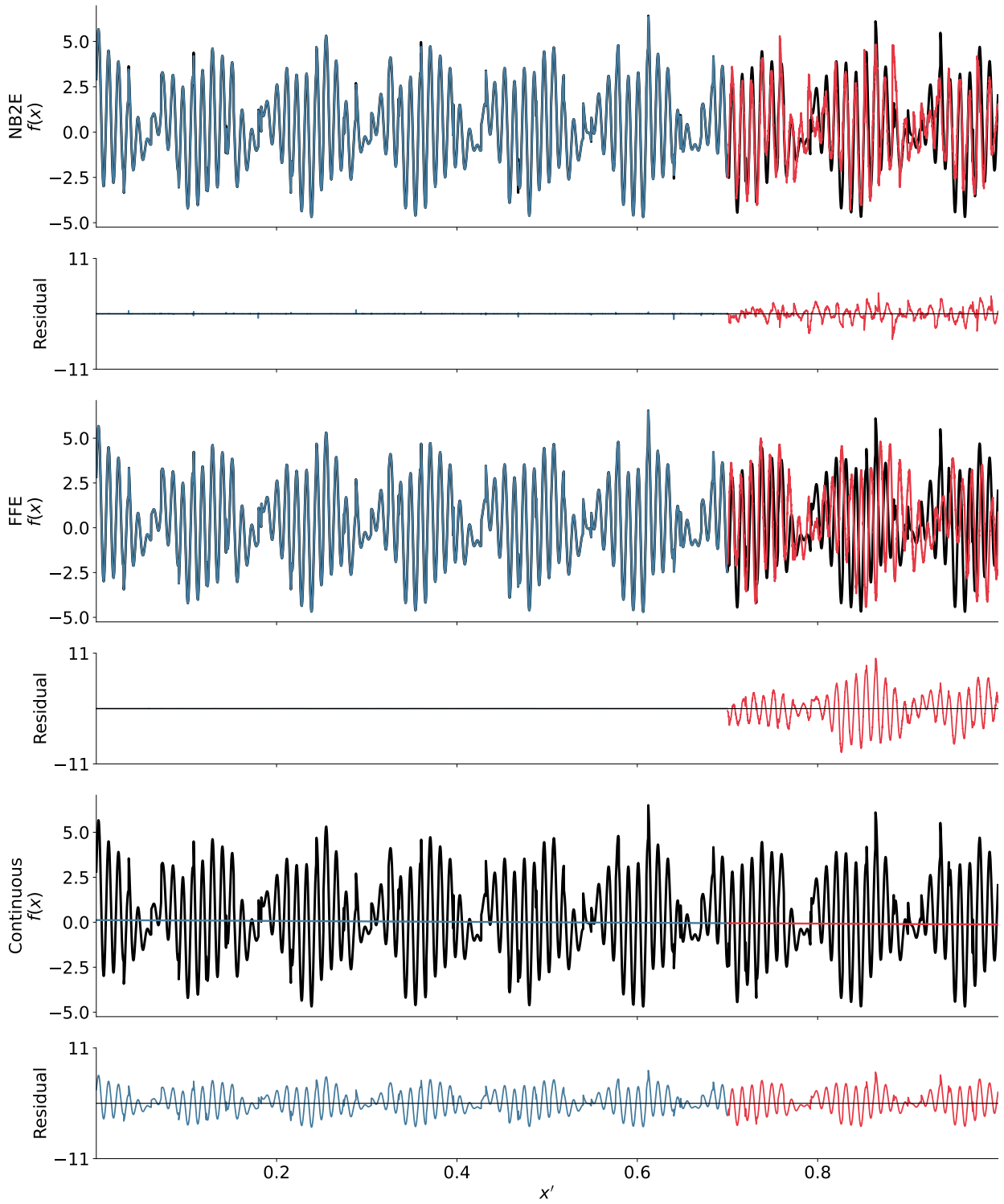

Figure 6: Same as Figures 3, 4, and 5 but for the composite function given in Appendix A

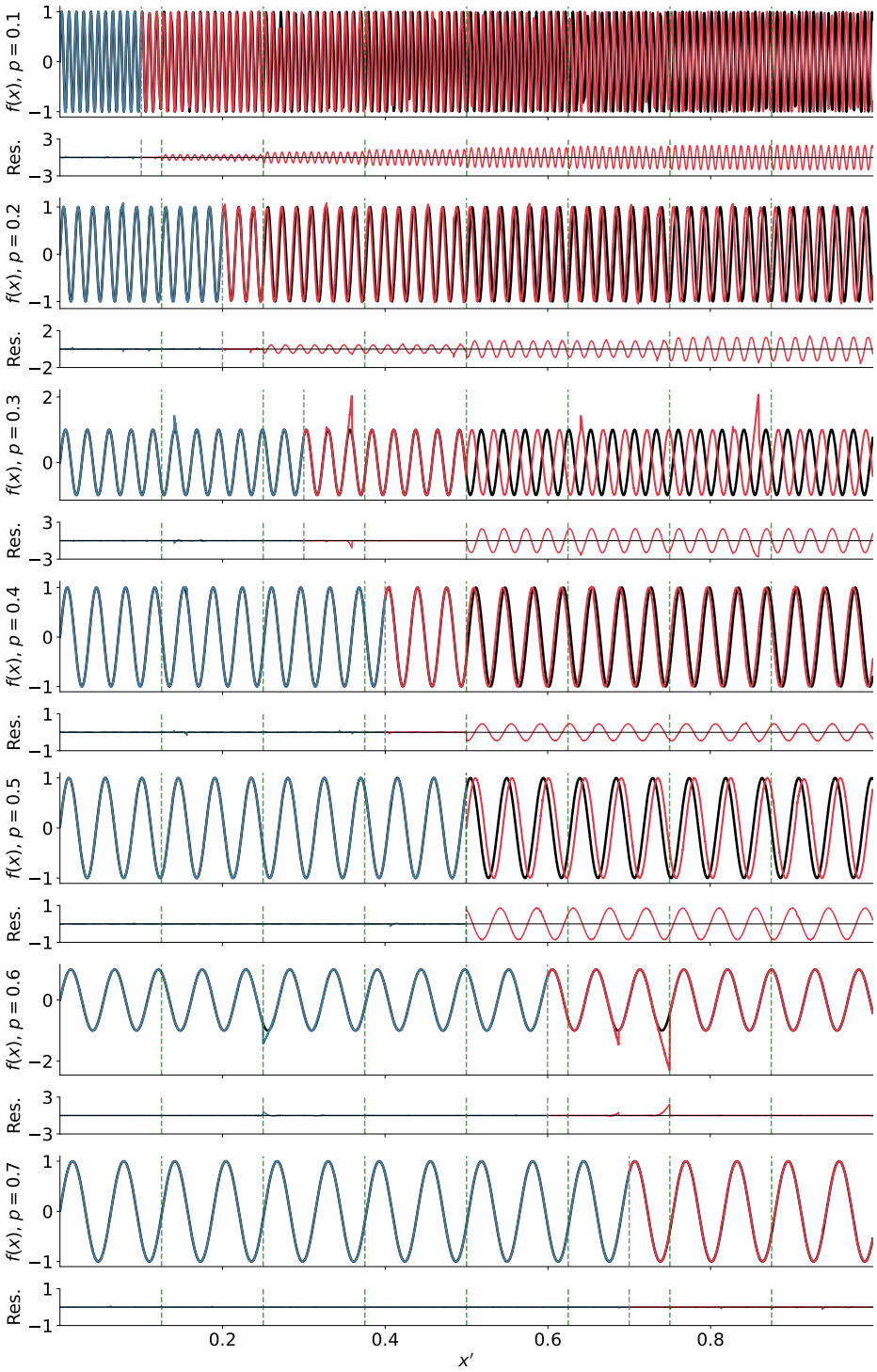

Figure 7: For $X'_{\text{train}} < p < X'_{\text{test}}$, from top to bottom, predictions and residuals for the training data (blue) and the test data (red) for $p = 0.1$, $p = 0.2$, $p = 0.3$, $p = 0.4$, $p = 0.5$, $p = 0.6$, and $p = 0.7$. Green dashed vertical lines in each plot identify the locations of, from left to right, $x' = 0.125$, $x' = 0.25$, $x' = 0.375$, $x' = 0.5$, $x' = 0.625$, $x' = 0.75$, and $x' = 0.875$, where any change occurs in the first three bits of the NB2E.

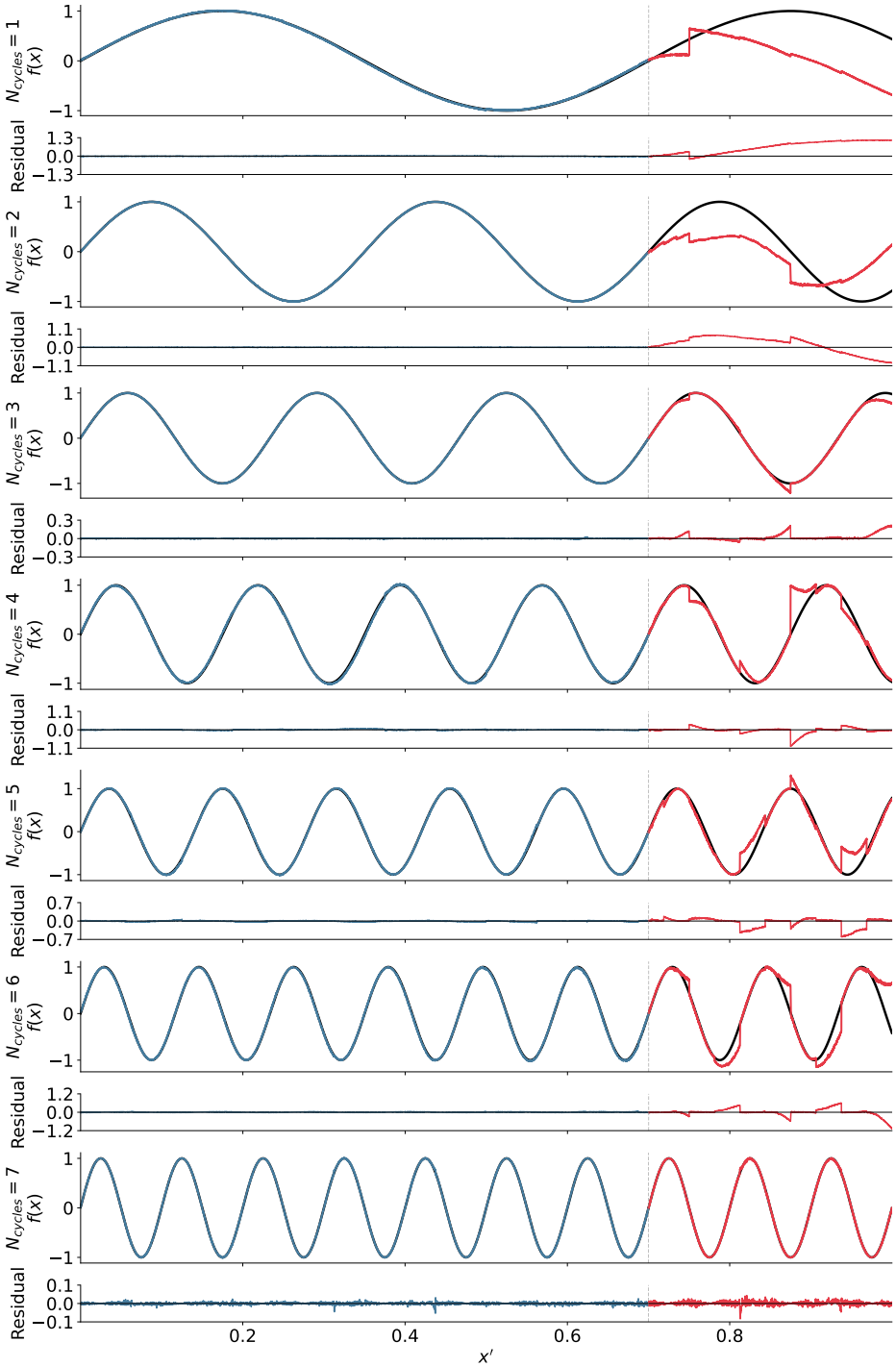

Figure 8: (*upper panels*) Predictions of the sine curve for the training data (blue) and the test data (red) shown with the true function (black). (*lower panels*) Residuals of the predictions. The number of complete cycles trained, from top to bottom, is in integer increments from 1 to 7. The dashed gray vertical line shows the train/test split at $x' = 0.7$.

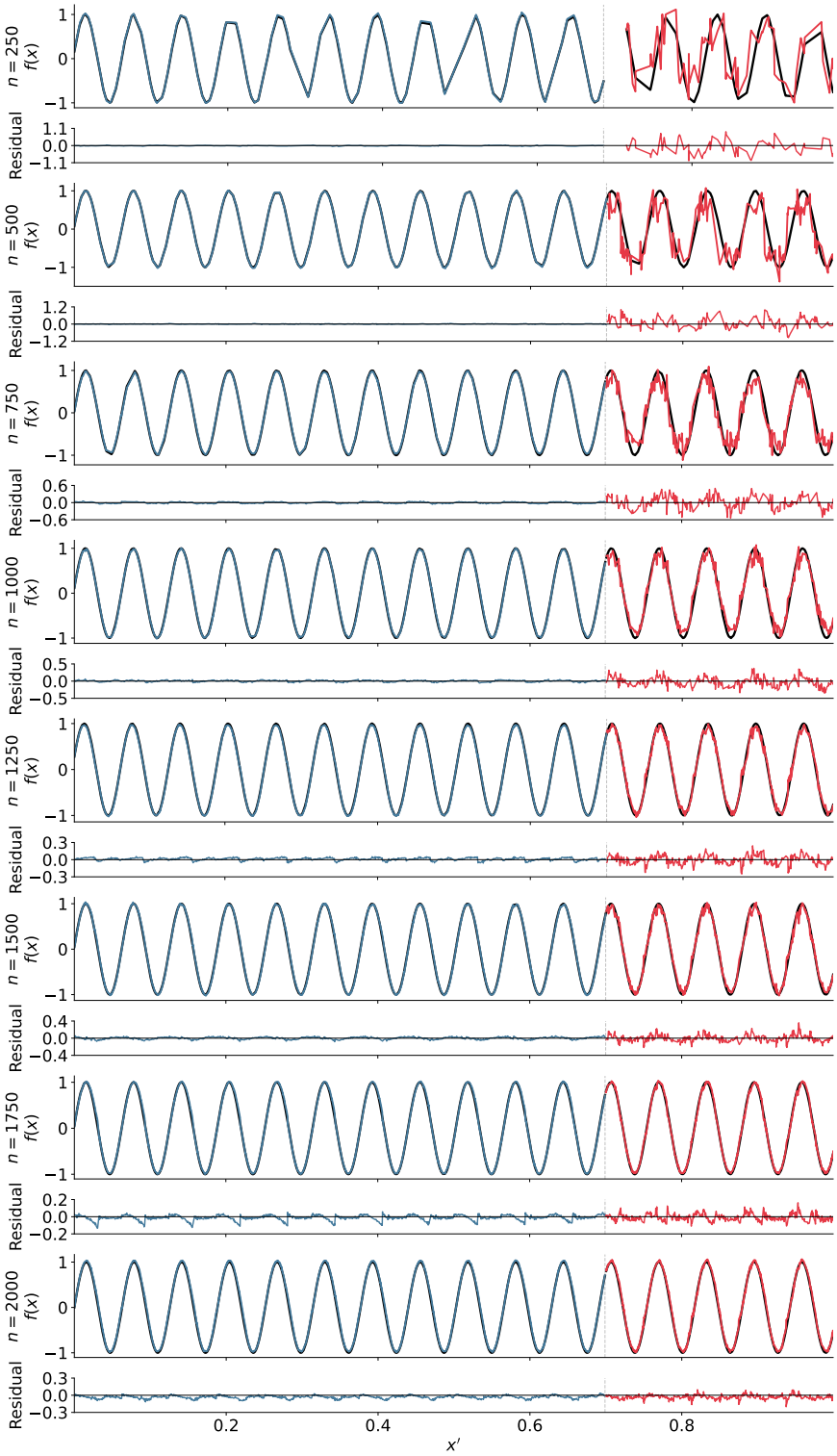

Figure 9: (*upper panels*) Predictions of the sine curve for the training data (blue) and the test data (red) shown with the true function (black). (*lower panels*) Residuals of the predictions. The number of data points in each data set, labeled accordingly, goes from 2000 in the bottom panel to 250 in the top panel in increments of 250. The dashed gray vertical line shows the train/test split at $x' = 0.7$.

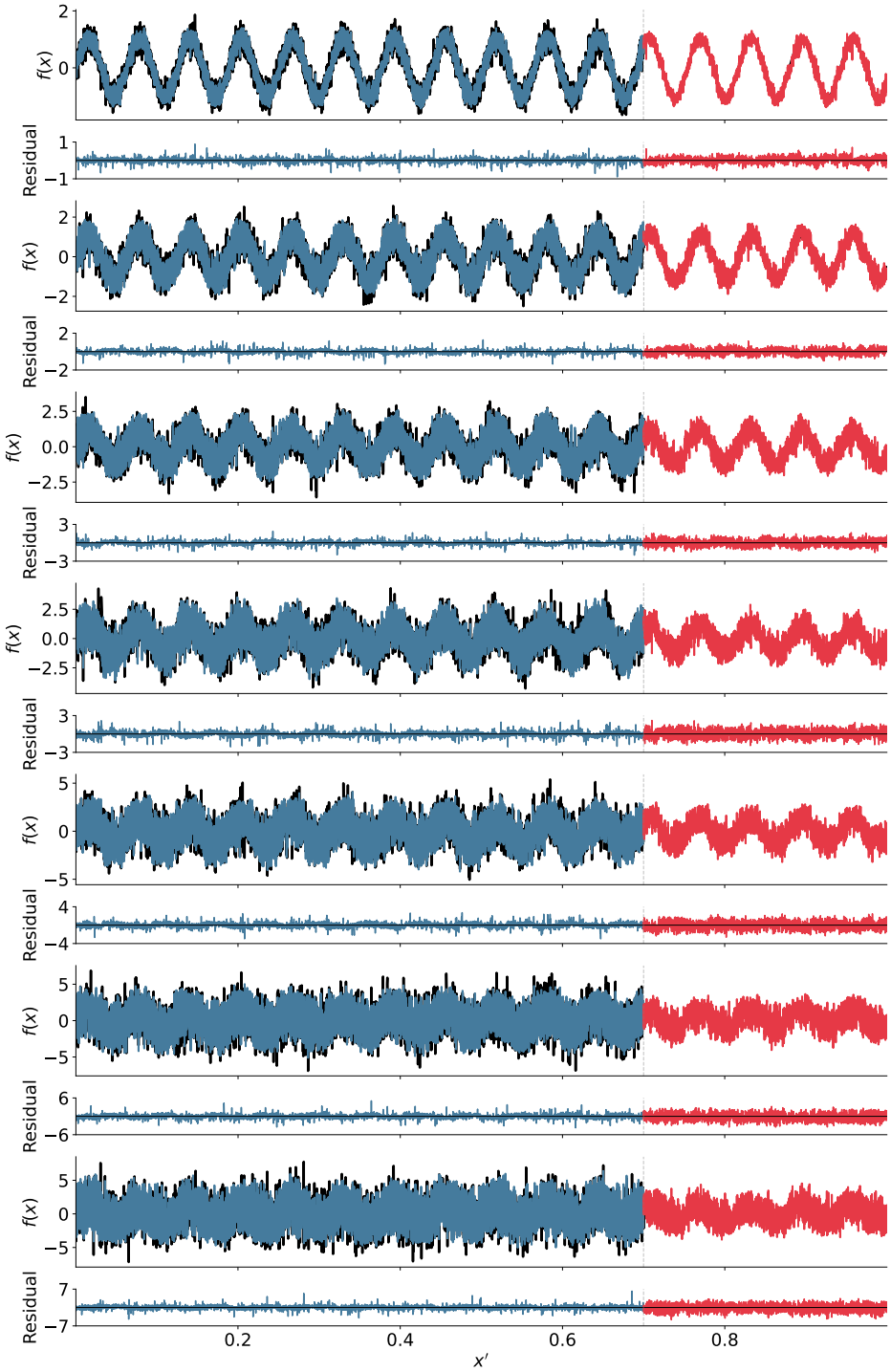

Figure 10: Predictions of the training data (blue) and test data (red) for a sine curve with noise drawn randomly from a Gaussian distribution with zero mean and standard deviations, from top to bottom, of 0.25, 0.5, 0.75, 1.0, 1.25, 1.75, and 2.0. The gray, dashed vertical line at $x' = 0.7$ shows the train/test split.

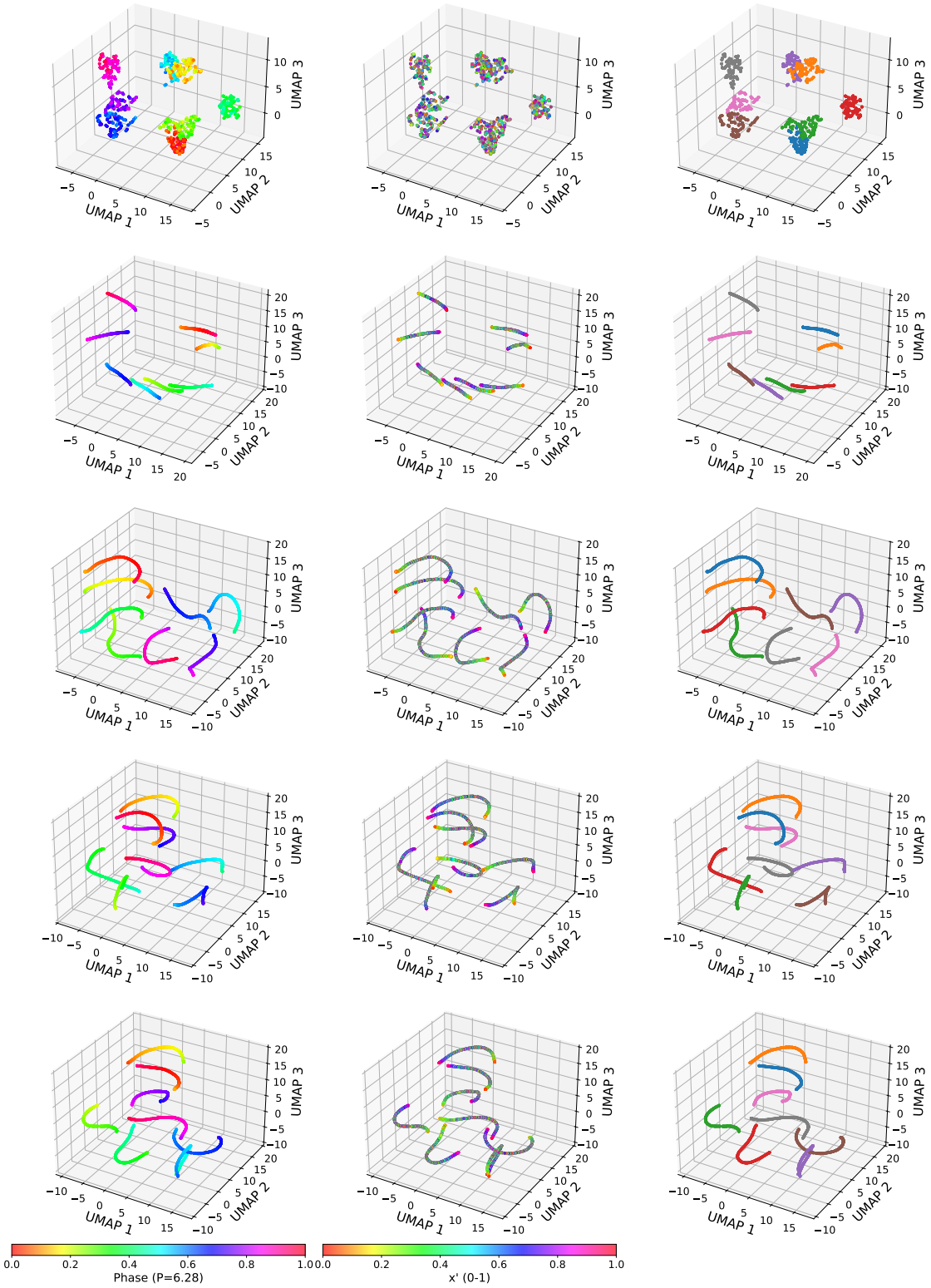

Figure 11: UMAP-reduced activation projections of the NB2E MLP for, from top to bottom, layers 1, 2, 3, 4, and 5. The activations in the left column are colored by phase of the sine curve, the middle column is colored by normalized position, and the right column is colored by clustered group (corresponding to Figure 12).

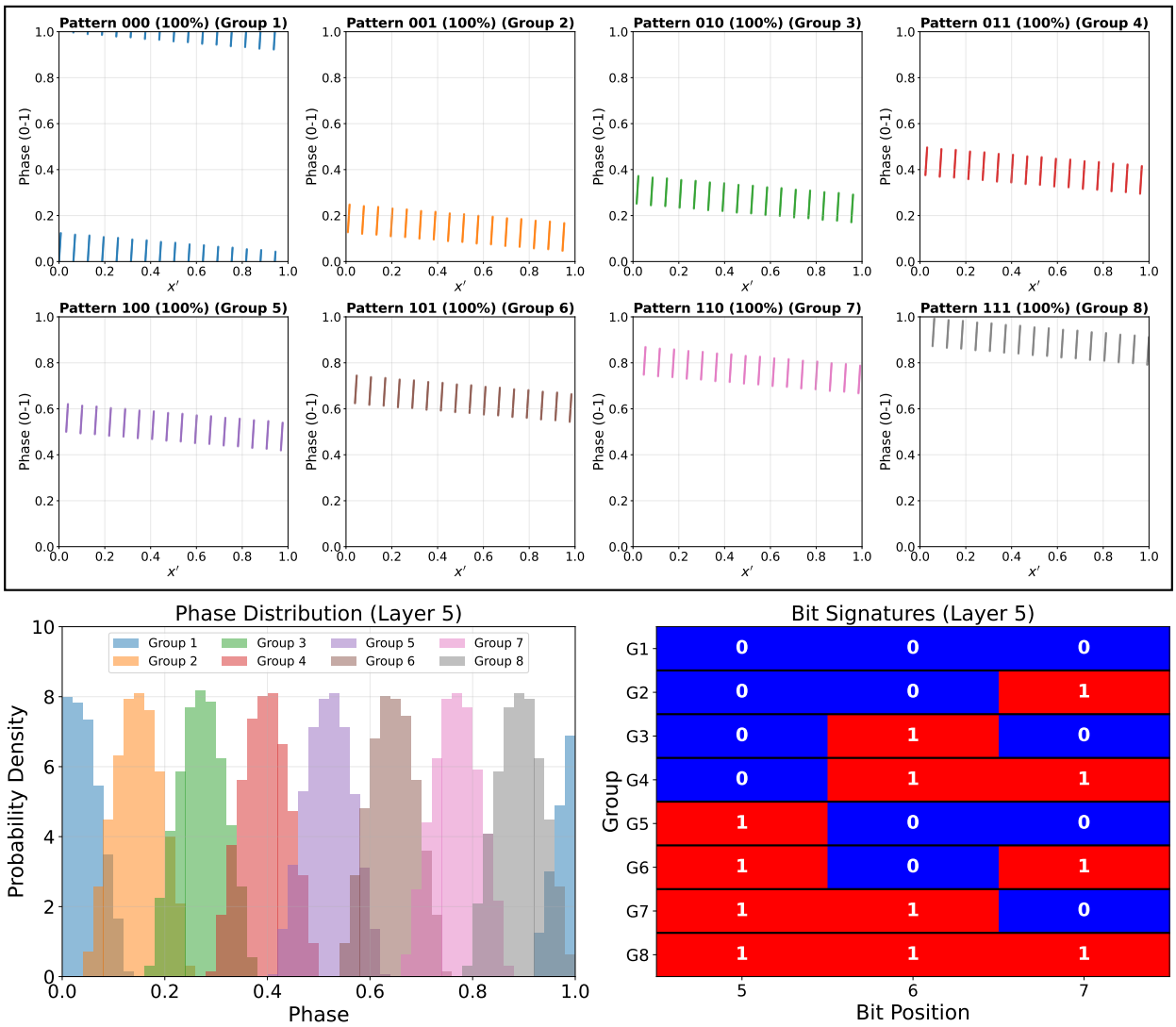

Figure 12: (*top*) Range of bits 5, 6, and 7 for each clustered group. (*bottom left*) Histogram of the density of values in each clustered activation grouping. (*bottom right*) For bit positions 5, 6, and 7, the unique bit signature of each clustered group.

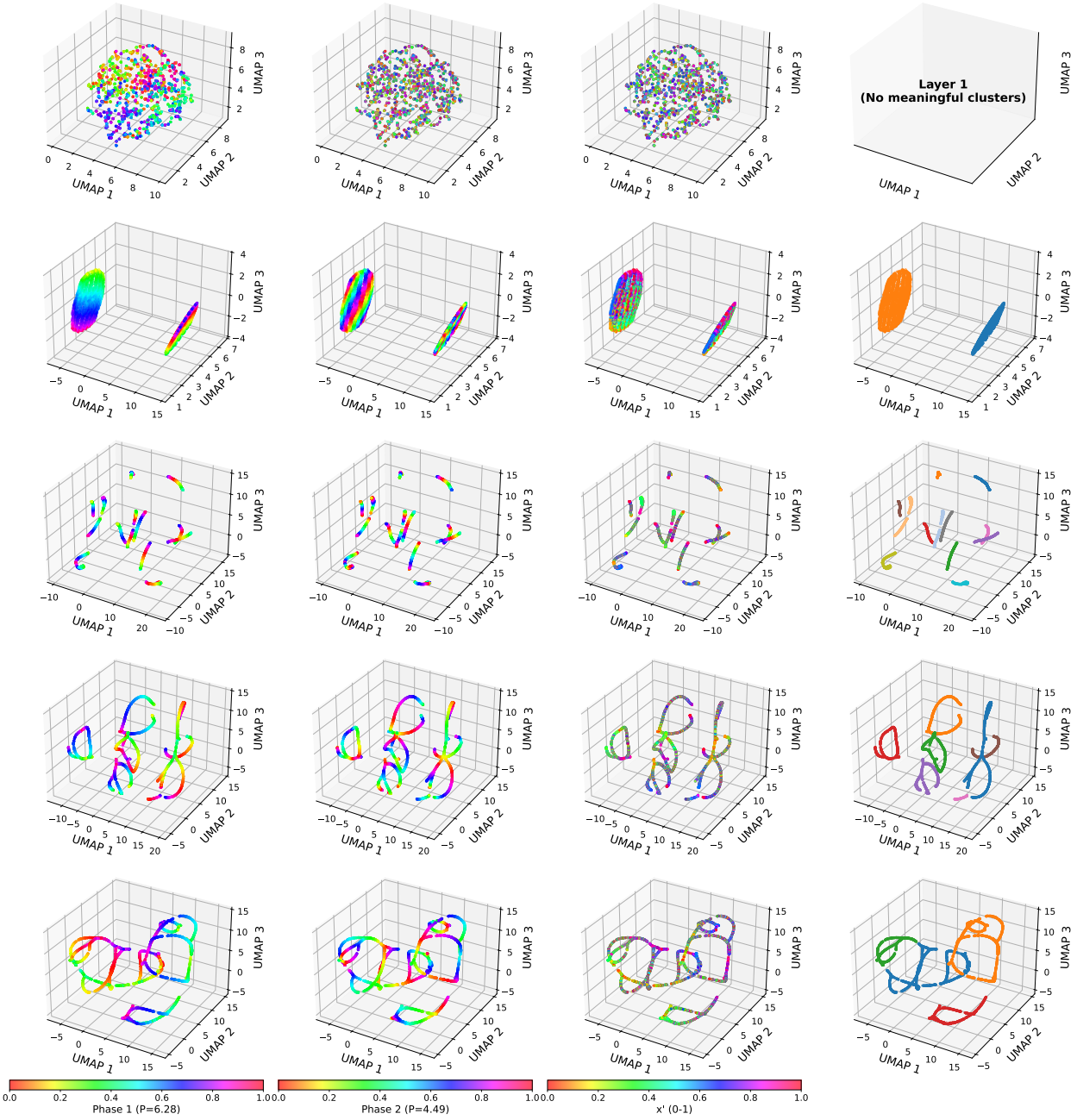

Figure 13: UMAP-reduced activation projections of the MLP with NB2E input of the composite sine function for, from top to bottom, layers 1, 2, 3, 4, and 5. The first column is colored by phase of Period 1, the second column is colored by phase of Period 2, the third column is colored by x', and the fourth column is colored by clustered group (corresponding to Figure 14 for the fifth layer).

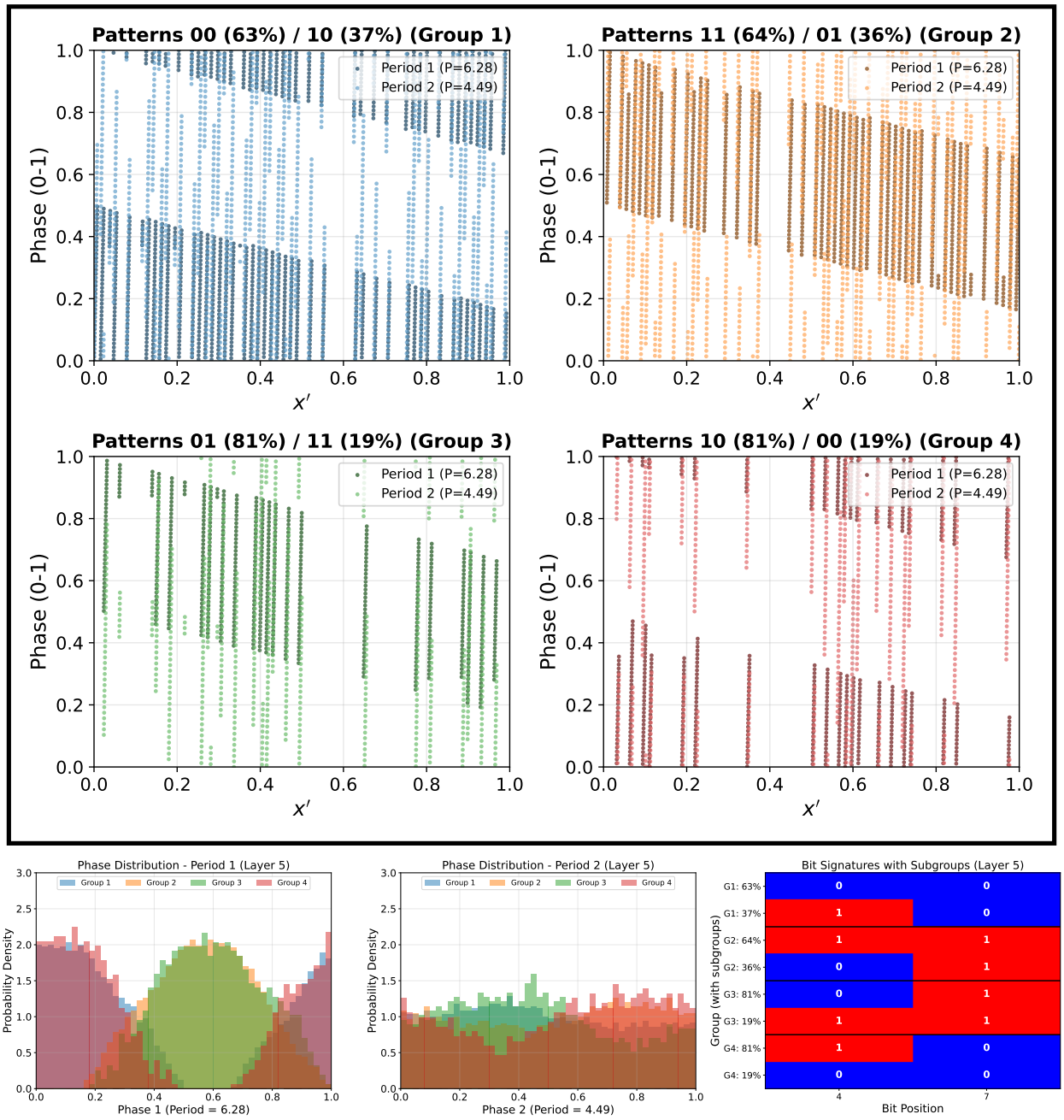

Figure 14: (*top*) x' vs. phase for each clustered group. (*bottom left*) Density histogram of the phases of Period 1 in each group. (*bottom middle*) Density histogram of the phases of Period 2 in each group. (*bottom right*) Unique values of bits 4 and 7 in each subgroup.

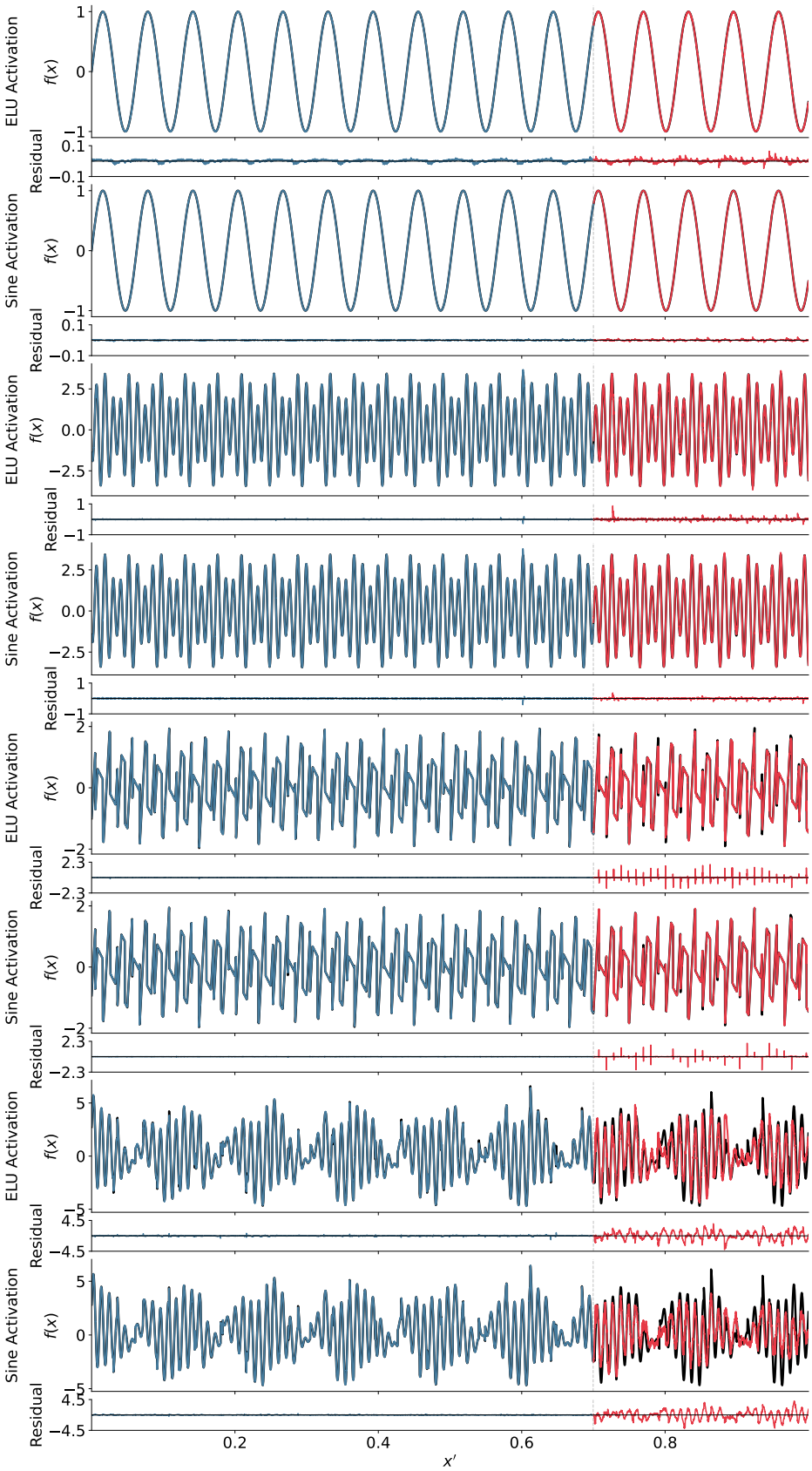

Figure 15: Predictions and residuals for the training data (blue) and test data (red) from the MLP with ELU activations and sine activations, labeled accordingly.