# ReLU Can Compute Numerical Integration Mechanistic Interpretation of a Non-linear Activation

**Anonymous Authors**[1]

## Abstract

Extending the analysis from Nanda et al. (2023) and Zhong et al. (2023), we offer an end-to-end interpretation of the 1 layer MLP-only modular addition transformer model with symmetric embeds. We present a clear and mathematically rigorous description of the computation at each layer, in preparation for the proofs-based verification approach as set out in Author et al. (2024). In doing so, we present a new interpretation of the ReLU: that it implement quadrature schemes to carry out numerical integration, and we provide anecdotal and mathematical evidence in support.

## 1. Introduction

An open problem in mechanistic interpretation of toy models is understanding non-linear activations (Elhage et al., 2021). Current approaches largely either treat non-linear activations as detecting binary ("on-off") features with scaling, or treat them as black-boxes, only describing input-output behavior.

The key challenge in interpreting non-linearities is their expressiveness. While $k$-layer linear networks are not any more expressive than 1-layer linear networks of the same width, deep non-linear networks are not easily compressed. Concretely, if matrices $A$ and $B$ are both $n \times n$, a complete description of input-output behavior of $AB$ or $A + B$ requires only $n^2$ parameters. Since there are $2n^2$ parameters in specifying $A$ and $B$, half of them are assumed to be completely arbitrary, and the effective parameter count of the computation is only $n^2$. By contrast, no such compression of $x \mapsto \mathrm{ReLU}(A\,\mathrm{ReLU}(Bx))$ nor $x \mapsto \mathrm{ReLU}(Ax) + \mathrm{ReLU}(Bx)$ is possible in general.

Nanda et al. (2023) and Zhong et al. (2023) provide exciting mechanistic interpretations of transformers with non-linear ReLU activations trained to perform modular addition. They infer an algorithm where the binary features are frequencies, by comprehensively analyzing the linear components of the models and black-boxing the ReLUs, describing only its input-output behavior.

Following the intuitions of concurrent work (Author et al., 2024) on quantitatively grading explanation compression using formal proofs, we aim to interpret the model in enough detail that we can: provide a parameterisation of a "hand-coded" model of the same architecture, where (1) the parameterisation is at most *linear* in the effective parameter count of the original architecture, and (2) the time to check that a collection of parameters is valid is *linear* in the number of (hand-coded) parameters. Then, the faithfulness of such explanations is established by show that (3) when expressing every weight and bias of the model as a sum of our "hand-coded" model and a "noise" term, the contributions of the noise terms are always small.

In the course of finding such an interpretation, we found that the ReLU implements a quadrature scheme to carry out numerical integration to double the frequency of trigonometric functions (section 4) used in the rest of the algorithm described in prior work.

## 2. Setup

We work with the 'pizza' model from Zhong et al. (2023): a one-layer ReLU transformer with constant attention $= \frac{1}{2}$ for all tokens, trained to compute $M : (x, y) \mapsto (x + y) \bmod p$. The model takes input $(x, y)$ encoded as one-hot vectors, and outputs $\mathrm{logit}(x, y, z)$ for all possible values $\pmod p$, with the largest logit representing its predicted answer. We represent the model computation of logits as

$$\mathrm{logit}(x, y, z) = \textstyle\sum_i \mathrm{ReLU}(\mathrm{OV}(x)_i + \mathrm{OV}(y)_i + \text{embed}(y)_i) \cdot (W_{\text{out}})_{zi} + \text{residual terms}$$

Here, the $W_{\text{out}}$ matrix represents the unembed layer and final linear layer combined (we ignore the bias term because it is very small), and the notation $\mathrm{OV}(x)$ represents the OVE circuit applied to the one-hot vector for $x$.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Zhong et al. (2023) state that the model computation of logits can be approximated as

$$\text{logit}(x, y, z) = |\cos(k(x - y)/2)| \cos(k(x + y - z))$$

and offer an argument of how this expression arises. As is standard in mechanistic interpretability, part of their argument is justified visually, by noting in Appendix A via plots that $|\cos(t)| - |\sin(t)| \approx \cos(2t)$, and a more complicated function in Appendix L. In order to check that a parameterisation using their stated algorithm is valid, we'd have to sum $d_{mlp}$ terms for all $p$ points separately.

## 3. Finding numerical integration

As explained in (Nanda et al., 2023), we carry out Fourier transform on the OVE vectors in the token dimension:

$$\text{OV}(x)_i \approx \sum_k a_i^k cos(kx) + b_i^k sin(kx) = \sum_k a_i^k cos(kx + \phi_i^k)$$

We find that each neuron pre-ReLU, i.e. the values $\text{OV}(x)_i$, are 'single-frequency'. That is, $\text{OV}(x)_i \approx a_i \cos(k(i)x + \phi_i)$. So, we use the angle shift representation of Fourier transforms as it helps us compare neurons of the same frequency by their angle $\phi_i$. This also allows us to look at the corresponding angles for the $W_{\text{out}}$ matrix. Note that the frequency $k$ can depend on the neuron, but only a few frequencies occur (in many models we see 4 different frequencies used). Denote the pre-ReLU input as $F_1(x, y)_i$ and split the ReLU layer into $\text{ReLU}(x) = \frac{x+|x|}{2}$. We do this because the absolute value function may be easier to deal with than the raw ReLU function, especially when we have products. We have (ignoring biases and the residual stream from the embeds)

$$\begin{aligned}
&F_1(x, y)_i \\
&\approx a_i(\cos(k(i)x + \phi_i) + \cos(k(i)y + \phi_i)) \\
&= 2a_i \cos(k(i)(x - y)/2) \cos(k(i)(x + y)/2 + \phi_i) \\
&\text{ReLU}(F_1(x, y)_i) \\
&\approx a_i |\cos(k(i)(x - y)/2)| \, |\cos(k(i)(x + y)/2 + \phi_i)| \\
&\quad + a_i \cos(k(i)(x - y)/2) \cos(k(i)(x + y)/2 + \phi_i)
\end{aligned}$$

The columns of $W_{\text{out}}$ are also 'single-frequency', with frequencies for each column matching the corresponding OVE frequency. That is, $(W_{\text{out}})_{zi} \approx b_i \cos(k(i)z + \psi_i)$. Thus,

$$\begin{aligned}
&\text{logit}(x, y, z) = \sum_i b_i \cos(k(i)z + \psi_i) \cdot \\
&\quad a_i(|\cos(k(i)(x - y)/2)| \, |\cos(k(i)(x + y)/2 + \phi_i)| \\
&\quad + \cos(k(i)(x - y)/2) \cos(k(i)(x + y)/2 + \phi_i)) \\
&= \sum_k |\cos(k(x - y)/2)| \sum_{i:k(i)=k} \\
&\quad r_i(\cos(kz) \cos(\psi_i) \, |\cos(k(x + y)/2 + \phi_i)| \\
&\qquad - \sin(kz) \sin(\psi_i) \, |\cos(k(x + y)/2 + \phi_i)|) \\
&\qquad + \text{similar terms for the identity component}
\end{aligned}$$

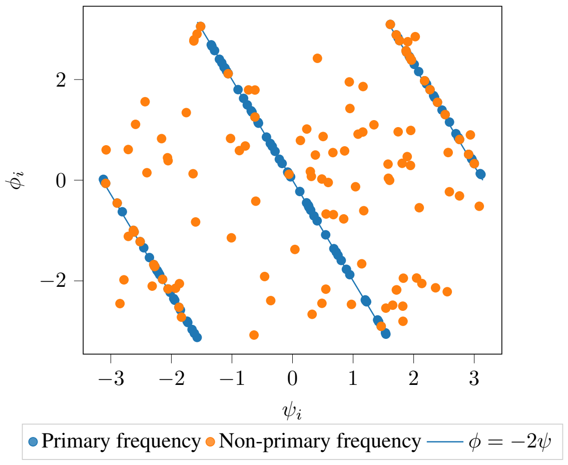

*Figure 1.* Angles for frequency $k = 12$. $\psi_i \approx -2\phi_i$ for the primary frequency of each neuron but not in general.

where $r_i = a_i b_i$. Thus, if we want to produce the logit formula, one way to do this is to match the coefficients:

$$\sum_{i:k(i)=k} r_i \cos(\psi_i) | \cos(\frac{k(x+y)}{2} + \phi_i)| \approx C_k \cos(k(x + y))$$

and similarly for sine. Then we can apply the cosine addition formula and obtain the required expression for the logit.

## 4. How numerical integration works

We find that this explanation of ReLU bears out in the model. As suggested by our expressions, we investigate the relationship between $\psi_i$ and $\phi_i$ and notice that $\psi_i \approx -2\phi_i$ with very high accuracy, see Figure 1. We also have

$$\int_{-\pi}^{\pi} \cos(-2\phi) \left|\cos(\tfrac{k}{2} + \phi)\right| \, d\phi = \tfrac{4}{3} \cos(k)$$
$$\int_{-\pi}^{\pi} \sin(-2\phi) \left|\cos(\tfrac{k}{2} + \phi)\right| \, d\phi = \tfrac{4}{3} \sin(k)$$

So it is plausible that the MLP neurons are working together to implement a quadrature scheme to carry out the two integrals listed above:[1]

$$\int_{-\pi}^{\pi} f(\phi) d\phi \approx \sum_i r_i f(\phi_i)$$

Interpreting the RHS as rectangles with width $r_i$ sampled at the point $\phi_i$ gives the visualization of Figure 2, as follows:

First normalise the widths $r_i$ such that $\sum_i r_i = 2\pi$, the range of integration. Then, sort the angles and line up the rectangles with width $r_i$ correspondingly to form end-points $a_0 = -\pi, a_1 = -\pi + r_1, a_2 = -\pi + r_1 + r_2, \ldots, a_n = -\pi + r_1 + \ldots + r_n = \pi$. Note that since the function is $2\pi$-periodic, we can shift the intervals formed above by any

[1]The toy approximation offered in Zhong et al. (2023, Appendix A), is just sampling the integral by two points (0 and $\pi/2$).

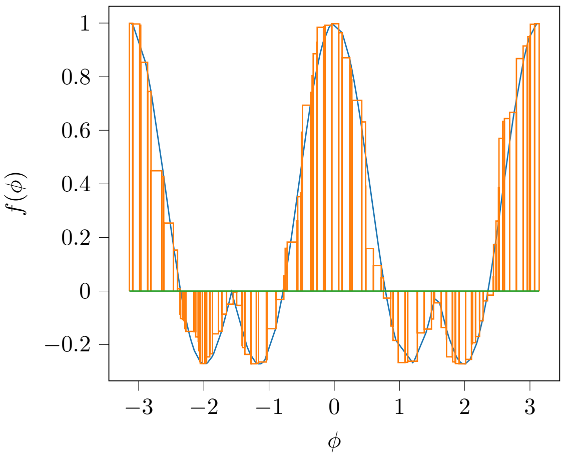

*Figure 2.* Converting the weighted sum into rectangles to estimate an integral (for frequency $k = 12$).

constant. In what follows, when bounding approximation error, we use the shift that gives the lowest bound.

Additionally, by utilising the fact that the function $f(x)$ is actually $\pi$-periodic rather than $2\pi$-periodic, we can overlap the two halves of sampled points to try and reduce the error of integration. (In this way, the rectangles in the approximation are narrower and so the error would be smaller.)

## 5. Error bound calculation

The discrepancy is: $\left|\int_{-\pi}^{\pi} f(x) - f(\phi_i)\, \mathrm{d}x\right|$ where $\phi_i$ is the angle s.t. $x \in [a_{i-1}, a_i]$. A crude bound is:

$$\begin{aligned} &\leq \textstyle\int_{-\pi}^{\pi} |f(x) - f(\phi_i)|\, \mathrm{d}x \\ &\leq \textstyle\int_{-\pi}^{\pi} |x - \phi_i| \cdot \sup_x |f'(x)|\, \mathrm{d}x \\ &\leq \sup_x |f'(x)| \textstyle\sum_i \left(\int_{a_{i-1}-\phi_i}^{a_i-\phi_i} |x|\, \mathrm{d}x\right) \end{aligned}$$

We must also include approximation error from $\psi_i \approx 2\phi_i$: we have $\sum_i r_i \left|\cos(k_i(x+y)/2 - \phi_i)\right| \cdot (\cos\psi_i - \cos 2\phi_i) \leq \sum_i r_i \left|\cos(k_i(x+y)/2 - \phi_i)\right| \cdot |\psi_i - 2\phi_i| \leq \sum_i r_i |\psi_i - 2\phi_i|$.

Using the specific frequency $k = 12$, the angle discrepancy gives us a bound of 0.15 using our data.

We can bound $\sup_x |f'(x)| \leq 2$ from differentiating the function analytically, and the integral within the sum can be evaluated easily. This gives us a way of bounding the error in integration in time linear in the number of sample points ($p$). Using this method on our data gives a bound of 0.45, so combined with above, this gives a bound of 0.60. This is compared to the naive estimate of integral $= 0$, which gives an average absolute error of 0.74. In comparison, if we simply evaluate the integral at all positions and combine it to the weighted sum, for all values of $x+y$, the maximum error here is just 0.043. This shows that numerical integration is the right idea, but we still have some way to go to produce a sublinear bound approaching the actual error.

Note that our bounding argument is $O(n_{\text{freq}} d_{mlp})$, while the procedure to check the maximum error is $O(p d_{mlp})$. Since there are $p d_{mlp}$ effective parameters in the MLP layer, our argument is of order square root in the number of parameters (assuming $p \approx d_{mlp}$ and $n_{\text{freq}} \ll p$). This showcases the tradeoff between computational complexity and bound accuracy as set out in Author et al. (2024). The fact that we get a valid bound (better than the naive estimate) in sublinear time shows that our guess is very likely on the right track. However, our large bound shows that there is more to understand about how the quadrature scheme works, such as how do we align the intervals, what do the values of the $\phi_i$ and $r_i$ play here, etc.

## 6. The rest of the model

Recall that we broke down ReLU into two parts, the absolute value component (considered above) and the identity component. We can apply the same argument to the term $G_1^i(x,y)/2$: we have the same quadrature scheme with

$$f(\phi) = \cos(-2\phi)\cos(\tfrac{k}{2}+\phi) \text{ and } f(\phi) = \sin(-2\phi)\cos(\tfrac{k}{2}+\phi)$$

In this case, the integrals evaluate to 0, which means the main contribution to the logits comes from the absolute value component rather than the identity component. Indeed, calculating the weighted sum in the expression above gives a maximum modulus of 0.06, suggesting our claim is reasonably solid. Hence, if we ignore the residual stream (from the attention layer), then we obtain the approximate expression for the logits (as above)

$$\textstyle\sum_k \frac{4}{3} C_k |\cos(k(x-y)/2)| \cos(k(x+y-z)).$$

We have the following results for the other frequencies included in this models:

| Error Bound Type \ Freq. | 12 | 18 | 21 | 22 |
|---|---|---|---|---|
| Normalised abs error | 0.04 | 0.03 | 0.04 | 0.03 |
| Normalised id error | 0.06 | 0.05 | 0.04 | 0.04 |
| Numerical $\int$ bound | 0.60 | 0.41 | 0.46 | 0.41 |
| Naive bound | 0.74 | 0.74 | 0.74 | 0.74 |

Alas, our approximations above are not completely correct, because when we regress the logits against the factors $|\cos(k(x-y)/2)|\cos(k(x+y-z))$, we obtain an $R^2$ of 0.86, while if we regress them against just $\cos(k(x+y-z))$, we obtain an $R^2$ of 0.98. (A similar value is obtained if we just use the MLP output and drop the residual streams.) However, if we only consider the contribution to the logits from the absolute value component of ReLU, the $R^2$ values become 0.99 and 0.85 respectively. Therefore, although the contribution from the identity component of ReLU is small, it does make a difference towards reducing the logit dependence on $x-y$, in particular $|\cos(k(x-y)/2)|$. This is a good thing because when $\cos(k(x-y)/2)$ is small, the logit difference between the correct logit $(x+y)$ and other logits will also be small, which will lead to a higher loss. The identity component slightly counters this effect. Then

$$W_{\text{out}}\text{OV}(x)/2 + W_{\text{out}}\text{OV}(y)/2 + W_{\text{out}}\text{embed}(y)/2$$

Thus, we can store the matrices $\text{logit\_id1}[:,x] = W_{\text{out}}\text{OV}(x)/2$ and $\text{logit\_id2}[:,x] = W_{\text{out}}\text{embed}(x)/2$, then we have

$$\begin{aligned} F_2(x,y)_z &= \text{residual stream} + \text{absolute value terms} \\ &\quad + \text{logit\_id1}[z,x] + \text{logit\_id1}[z,y] + \text{logit\_id2}[z,y] \end{aligned}$$

We carry out a 2D Fourier transform to find out the decomposition of the logit_id1 and logit_id2 matrices (because $W_{\text{out}}$ and $\text{OV}(x)$ are sparse in the (1D) Fourier basis, so their product will naturally be sparse in the 2D Fourier basis). We get $\text{logit\_id1}[z,x] \approx 2\Re(\sum_k a_k e^{i(kz-2kx)})$, where the frequencies $k$ here are the same as section 3. Hence, the output from the identity component of ReLU is (ignoring logit_id2 for now, which comes from the residual stream and is smaller): $\sum_k D_k(\cos(kz-2kx)+\cos(kz-2ky)) + E_k(\sin(kz-2kx)+\sin(kz-2ky)) = \sum_k \cos(k(y-x))(D_k\cos(k(z-x-y)) + E_k\sin(k(z-x-y)))$.

The imaginary component of the FT is very small, $c_k \approx 0$; so the contribution is $\sum_k b_k \cos(k(y-x))\cos(k(x+y-z))$.

Why does this happen, and why does it help explain the $R^2$ values we got above? We first list the approximate coefficients $a_k$:

| Frequency | 12 | 18 | 21 | 22 |
|---|---|---|---|---|
| abs coefs ($C_k$) | 13.9 | 15.1 | 12.1 | 11.2 |
| id coefs ($D_k$) | -3.7 | -3.9 | -3.2 | -3.3 |

Thus, the overall expression for the logits is

$$\begin{aligned} F_2(x,y)_z &\approx \textstyle\sum_k (C_k\,|\cos(k(y-x)/2)| \\ &\qquad + D_k\cos(k(y-x)))\cos(k(x+y-z)) \\ &= \textstyle\sum_k (2D_k^2\,|\cos(k(y-x)/2)|^2 \\ &\qquad + C_k\,|\cos(k(y-x)/2)|)\cos(k(x+y-z)) \\ &\qquad - D_k\cos(k(x+y-z)) \end{aligned}$$

using double angle formula. Since $D_k < 0$, the $\sum_k -D_k\cos(k(x+y-z))$ term gives some cushion for the base performance of the model (since as we discussed, the $\cos(k(x+y-z))$ term is why the model gives the highest logit when $z = x+y$). Moreover, the $2D_k^2|\cos(k(y-x)/2)|^2$ term also further improves the model since it is always non-negative. Hence, the contribution of the identity term evens out parts of the model and improves the logit difference when $|\cos(k(y-x)/2)|$ is small (where the absolute value part doesn't do well). Note that the model would work on its own if we only use the absolute value part, but since ReLU is composed of both the absolute value and identity part and the coefficients combine both parts in a way that improve model performance.

## 7. Discussion, interpretation, and further work

First, note the correspondence between the complexity of the argument and mechanistic understanding of the model. In the first stage, restricting each node to a single frequency (using mechanistic insight) means we only need $\mathcal{O}(p \cdot d_{mlp})$ time complexity to compute the neuron activations (the $x-y$ term can be grouped together and considered later since it is the same factor for all neurons of the same frequency). In the second stage, there is a lack of explanation of how the ReLU layer works in Zhong et al. (2023, Appendix L), where they just evaluated all the combinations in the simplified expression, to check that we indeed get $\cos(k(x+y))$ terms. Our numerical integration argument can infer about the accuracy of the scheme just by looking at the angle samples $\phi_i$ and $\psi_i$, which is $\mathcal{O}(p + d_{mlp})$ complexity. This corresponds to a better explanation of how the ReLU layer works. In particular, this argument should allow us to produce a large family of hand-coded, parameterised models which we know will perform the task well, and this family of models approximates what happens when they are trained empirically. Producing this family of models is future work.

Second, the argument presented here is not fully rigorous. We have not obtained a proof that the network as a whole indeed well-approximates numerical integration. (In fact, it is difficult even to obtain good error bounds for the 'single frequency' approximation of the neurons.) This reflects a general problem of mechanistic interpretability – a lot of the evidence is merely demonstrative or anecdotal, and something which looks 'obvious' to some human eyes may actually be impossible to prove. Thus, to have truly convincing interpretability work, we should put more effort into rigorous proofs or guarantees about the correctness of our observations and approximations.

## Impact Statement

This paper presents work whose goal is to advance the field of Mechanistic Interpretability. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

## References

Author, Suppressed for Anonymity, and . Compact proofs of model performance via mechanistic intepretability, 2024.

Attached as supplementary material.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Nanda, N., Chan, L., Lieberum, T., Smith, J., and Steinhardt, J. Progress measures for grokking via mechanistic interpretability. *arXiv preprint*, 2023. doi: 10.48550/arXiv.2301.05217.

Zhong, Z., Liu, Z., Tegmark, M., and Andreas, J. The clock and the pizza: Two stories in mechanistic explanation of neural networks, 2023.

## A. Further work

There are obvious problems with generalising this sort of approach to neural network interpretation. Not only does it depend on the specifics of this problem (how we can have a specific desired formula for the logits), but also it is highly labour intensive. Thus, in order for the approach to have any practical use, we need to develop automated tools to make these approximations and interpretations. For example, we may want to use the first few terms of the Fourier expansion (or other low-rank approximations) to approximate the action of various layers in a neural network, and then combine those to get algebraic expressions for certain neuron outputs of interest. Such algebraic expressions will natural admit phenomena like the numerical integration we described above. This sort of method may be particularly fruitful on problems which Fourier transforms play a large role, such as signal processing and solutions to partial differential equations.

## B. Future technical work

To complete the technical work laid out in the introduction, we must accomplish two tasks which we discuss in this appendix section: constructing a parameterisation of the MLP which is checkable in less than $\mathcal{O}(p \cdot d_{mlp})$ time, and more generally constructing a parameterisation of the entire 'pizza' model that is checkable in time that is linear in the number of parameters; and establishing a bound on the error in the model's logits that does not neglect any terms.

### B.1. Linear parameterisation

Constructing a parameterisation of the model which is checkable in less than $\mathcal{O}(p \cdot d_{mlp})$ time is a relatively straightforward task, given the interpretation in the body of the paper. We expect that the parameters are:

- A choice of $n_{\text{freq}}$ frequencies $k_i$.
- A splitting of the neurons into groups by frequency, and an ordering of the neurons within each group.
- An assignment of widths $r_i$ to each neuron, and an assignment of angles $\phi_i$ to each neuron.
- An assignment of orthogonal planes into which each frequency is embedded by the embedding matrix, and by the unembedding matrix.
- Rotations and scaling of the low-rank subset of the hidden model dimension for each of the O and V matrices.

### B.2. Bounding the error of the MLP

To bound the error of our interpreation of the MLP precisely, we'd need to include a bound on the primary frequency contribution of the identity component (which integrates to 0 symbolically), and include bounds on the residual components – OVE on $x$ and $y$, the MLP bias, and the embed of $y$, as inputs to ReLU; and UOVE on $x$ and $y$ and UE on $y$ as output logits.

We could decompose every matrix in our model as a sum of the corresponding matrix from our parameterized model and a noise term. Expanding out the resulting expression for the logits (and expanding $|x + \varepsilon|$ as $|x| + (|x + \varepsilon| - |x|)$), we will have an expression which is at top-level a sum of our parameterized model result and a noise term which is expressed recursively as the difference between the actual model and the parameterized model. We can then ask two questions:

1. What worst-case bounds can we prove on the error terms at various complexities?
2. What are the empirical worst-case bounds on the relevant error terms?