MODULAR ADDITION WITHOUT BLACK-BOXES: COMPRESSING EXPLANATIONS OF MLPS THAT COMPUTE NUMERICAL INTEGRATION

Chun Hei Yip **Rajashree Agrawal** **Lawrence Chan** **Jason Gross**

ABSTRACT

Interpreting the behaviour of MLP layers in transformer models remains an open problem. The goal of mechanistic interpretability is identifying the specific function learnt by the model, which is challenging to do in nonlinear domains—like MLPs—where low precision approximations of the learnt function can leave many degrees of freedom. While MLP layers that approximately implement linear functions can be well understood with low-precision approximations, we take on the challenge of approximating densely-connected MLPs. We propose a formal definition for "complete interpretation" of a circuit: a compression of the explanation to a size that is linear in the parameter count of the circuit. We investigate in the classic setting of the toy models trained on modular addition Nanda et al. (2023a); Zhong et al. (2023), where the MLP layer is treated as a black box. We extend the analysis to describe exactly *how* the MLP layer computes the required trigonometric identities. We find that the MLP layer in one-layer transformers implementing the "pizza" algorithm can be understood as evaluating a quadrature scheme, where each neuron computes the area of a rectangle under the curve of a trigonometric integral identity. We confirm this interpretation by using it to compress the MLP layer of a collection of modular addition transformers and prove non-vacuous bounds on the outputs of the MLP layer on *all* inputs in total time *linear in the number of neurons*. Our code is available at `https://tinyurl.com/mod-add-integration`.

1 INTRODUCTION

A key open problem in mechanistically explaining the behaviour of neural networks is finding interpretations of MLP layers (Elhage et al., 2021; 2022). The baseline approach is to treat MLPs as black boxes (Nanda et al., 2023b; Hanna et al., 2024). In this case, MLPs are approximated by enumerating possible inputs. As model size increases, enumerating the space of inputs to MLPs becomes infeasible, making these approximations low-precision. Such low-precision approximations leave many degrees of freedom in explanations of model behaviour, which make them ultimately unsuitable for high-stakes applications like anomaly detection or worst-case guarantees .

Fundamentally, the challenge is that nested nonlinear functions are highly expressive. Contrast this with deep linear networks, which are not any more expressive than shallow linear networks of the same width. Consider a toy example of simply adding or multiplying k matrices of shape $m \times m$. A complete description of input-output behaviour requires only m^2 parameters, independent of k. However, deep non-linear networks are not similarly compressible. Inserting nonlinearities such as ReLU around binary matrix operations in the toy example blows up the effective parameter count to km^2, and complexity of the resulting function is exponential in k (Raghu et al., 2017).

Following Gross et al. (2024), we operationalize a "complete understanding" of a circuit as a compression of the explanation to a size that is linear in the parameter count of the circuit section 2.

Even on the classic mechanistic interpretability domain of small one-layer transformers trained to perform modular addition, the MLP layer is treated as black box (Nanda et al., 2023a; Zhong et al., 2023). Existing work approximates the function by enumerating all possible inputs to the network, and checking that the MLP's outputs are congruent with the rest of their interpretation. In this work, we open up the final black box remaining inside of these modular addition transformers by applying an *infinite-width lens* to find a more compact description of the nonlinearity, wherein we treat densely

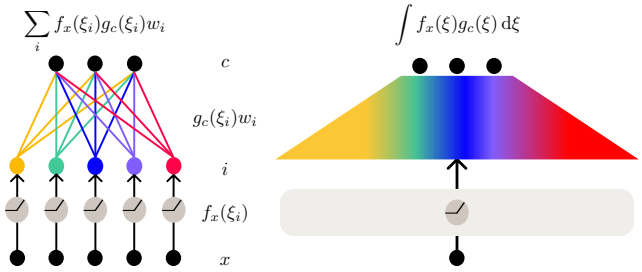


Figure 1: In the actual model (left) there are finitely many neurons (indexed by i). The function $f_x(\xi_i)$ is ReLU applied to the inputs x. The weight of the connection to each output c is $g_c(\xi_i)$ times a neuron-specific output-independent normalization factor w_i. Taking the limit as the number of neurons goes to infinity turns the sum over neurons into an integral (right).

connected MLPs as finite approximations to an infinite-MLP-width limit **??**. The infinite-width lens permits us to turn post-activation matrix multiplications into approximate integrals and study the remaining operations of the network – including the nonlinear operations – analytically.

We develop a compact description of the nonlinear function implemented by the MLPs of the "pizza" transformers (Zhong et al., 2023): doubling the frequencies of the input representations, that is, mapping from representations of the form $\cos(\frac{\omega_k}{2}(a+b)), \sin(\frac{\omega_k}{2}(a+b))$ to $\cos(\omega_k(a+b)), \sin(\omega_k(a+b))$. Building on surprising patterns in the phase-amplitude representation of the MLP pre-activations and neuron-logit map, we find that the MLP can be understood as implementing a quadrature scheme for integrals of the form

$$\int_{-\pi}^{\pi} \mathrm{ReLU}[\cos(\tfrac{\omega_k}{2}(a+b)+\phi)]\cos(\omega_k c + 2\phi)\,\mathrm{d}\phi = \frac{2}{3}\cos(\omega_k(a+b-c)),$$

for a handful of key frequencies k (subsection 3.2), where the integral can be seen as the limiting case of the MLP's summation as the number of neurons increases.

We confirm this understanding by creating non-vacuous bounds for the outputs of these MLPs in time *linear* in the number of parameters, i.e. without evaluating it on all P^2 possible inputs (section 4). We then resolve the puzzling observation that the logits of supposedly "pizza" models are closer of those of Nanda et al. (2023a)'s "clock" algorithm, by explaining how the model uses secondary frequencies equal to twice of each key frequency in order to compensate for how the pizza algorithm fails in cases when $\omega_k(a-b) \approx \pi$ (section 5).

We conclude by discussing possible takeaways for mechanistic interpretability practitioners working on non-algorithmic tasks (section 7). Notably, our approach is similar to (Elhage et al., 2022; Cunningham et al., 2023; Bricken et al., 2023) where MLP activations are treated as low dimensional projection of a high-dimensional space of sparse, linear features (Elhage et al., 2022; Cunningham et al., 2023; Bricken et al., 2023). Recent work has found evidence that sparse linear features do not fully capture the behaviour of frontier models (Marks et al., 2024; Engels et al., 2024). Our validation methodology helps to quantifying how faithfully the sparse features approximate the MLP, and notably, acts as a feedback loop in refining our interpretation which is far from complete at the point at which we name the sparse features.

2 Background & Setup

2.1 Notation for small transformers

In this work, we study the MLP layers of small transformers trained to perform modular arithmetic. Specifically, we consider small transformers trained that implement the 'pizza' model from Zhong et al. (2023): a one-layer ReLU transformer with four heads and constant attention $= \frac{1}{2}$ for all tokens, trained to compute $M : (a, b, \text{'='}) \mapsto (a+b) \bmod p$. As in Zhong et al. (2023), we take $p = 59$. The model takes input $(a, b, \text{'='})$ encoded as one-hot vectors, and we read off the logits $\mathrm{logit}(a, b, c)$ for all possible values $(\bmod p)$ above the final sequence position '=', with the largest logit representing its predicted answer.

Since the attention is constant, the logits of the model given input a, b are calculated as:

$$
\begin{aligned}
x_i^{(0)} &= W_E t_i + p_i && \text{Embedding} \\
x^{(1)} &= x_2^{(0)} + \frac{1}{2}\sum_{j=1}^{4} W_O^j W_V^j \left(x_a^{(0)} + x_b^{(0)}\right) && \text{Post attention residual stream} \\
N &= \mathrm{ReLU}(W_{\mathrm{in}} x^{(1)}) && \text{Post ReLU neuron activations} \\
x^{(2)} &= x^{(1)} + W_{\mathrm{out}} N \\
\mathrm{logits} &= W_U x^{(2)} = W_U \left(x^{(1)} + W_{\mathrm{out}}\,\mathrm{ReLU}(W_{\mathrm{in}} x^{(1)})\right)
\end{aligned}
$$

As noted in both Nanda et al. (2023a) and Zhong et al. (2023), the contribution from the skip connection around the MLP to the logits is small. Combining this with the fact that the attention is uniform across a, b, and using $W_L = W_U W_{\mathrm{out}}$ for the neuron-logit map, the logits can approximately be written as the sum of the contributions of each of the d_{mlp} neurons:

$$\mathrm{logit}(a, b, c) = \textstyle\sum_{i=1}^{d_{\mathrm{mlp}}} (W_L)_{ci} \cdot \mathrm{ReLU}(\frac{1}{2}\mathrm{OV}(a)_i + \frac{1}{2}\mathrm{OV}(b)_i)$$

2.2 The "Pizza" algorithm for modular arithmetic

Zhong et al. (2023) demonstrated that small transformers using constant attention implement modular arithmetic using the following algorithm, which they call the "pizza" algorithm:

1. $OV(a), OV(b)$ embed the one-hot encoded tokens a, b as $(\cos(\omega_k a), \sin(\omega_k a))$ and $(\cos(\omega_k b), \sin(\omega_k b))$ for a small handful of key frequencies ω_k
2. Before the MLP layer, the representation of the two tokens are averaged:

   $$(\cos(\omega_k a) + \cos(\omega_k a), \sin(\omega_k a) + \sin(\omega_k a))/2 = \cos(\omega_k \tfrac{a-b}{2})(\cos(\omega_k \tfrac{a+b}{2}), \sin(\omega_k \tfrac{a+b}{2}))$$

3. The network then uses the MLP layer to "double the frequencies":

   $$N = |\cos(\omega_k(a-b)/2)|\,(\cos(\omega_k(a+b)), \sin(\omega_k(a+b)))$$

4. Finally, W_L scores possible outputs by taking the dot product with $(\cos(\omega_k c), \sin(\omega_k c))$:

   $$\begin{aligned}\mathrm{logit}(a, b, c) &= |\cos(\omega_k(a-b)/2)|\,(\cos(\omega_k(a+b))\cos(\omega_k c) + \sin(\omega_k(a+b))\sin(\omega_k c)) \\ &= |\cos(\omega_k(a-b)/2)|\cos(\omega_k(a+b-c))\end{aligned}$$

They distinguish this from the "clock" algorithm of Nanda et al. (2023a), whose logits have no dependence on $|\cos(\omega_k(a-b)/2)|$, and where the MLP layer instead multiplies together its inputs.

Note that Zhong et al. (2023) check that the MLP layer doubles frequencies *empirically*, by treating the MLP as a black box and enumerating the MLP's outputs for all possible inputs. In this work, we seek to fully understand *how* the MLP performs its role.

2.3 "Complete understanding" MLPs

What does it mean to "completely understand" the behaviour of some component of the model? In this work, we follow Gross et al. (2024), and operationalize the degree of understanding provided by an interpretation as how much it allows us to compress a formally checkable bound of a component's behaviour on a full dataset.

In the worst case (as done in both Nanda et al. (2023a) and Zhong et al. (2023)), we can describe the MLP's behaviour by evaluating it on every possible input. For n such inputs and an MLP with width d_{mlp} and input/output dimension d_{model}, this requires checking the result of $\mathcal{O}(n\ d_{\mathrm{mlp}} d_{\mathrm{model}})$ operations. This corresponds to a null interpretation providing zero understanding of how the MLP internally functions.

Ideally, we might hope that we could formally bound the MLP's behaviour by making reference only to the model weights and without evaluating it on all inputs – in time $\mathcal{O}(d_{\text{mlp}}d_{\text{model}} + n)$, linear in the parameter count of the MLP. As this is the information-theoretic limit, we consider explanations that formally bound MLP behaviour in time linear in parameter count as providing *complete understanding* of the MLP's function.

While this may seem like a lofty goal (due to the empirical difficulty of understanding what MLPs do, even in toy cases), we demonstrate that we can indeed formally bound in linear time the behaviour of the MLP in the case of small modular arithmetic transformers using the "pizza" algorithm.

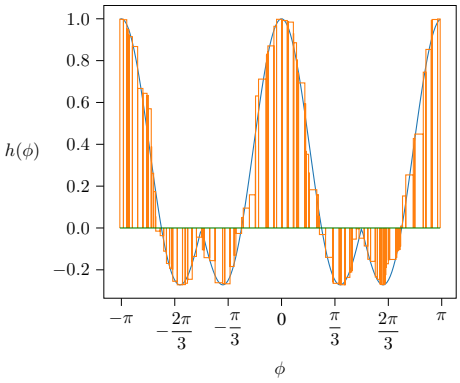


Figure 2: We argue that the MLP approximately computes integrals of this form. (Here, we display the integral being computed for frequency $k = 12$ when $a + b = c = 0$).

3 Interpreting "pizza" MLPs as performing numerical integration

Following Nanda et al. (2023a), we focus on a particular ("mainline") model in the main body of the work. We confirm that our results generalize to another 150 transformers in Appendix C.

3.1 Studying the model in the amplitude-phase Fourier form

Note that $OV(a), OV(b)$ do not actually embed all inputs to sin and cosine representations of equal magnitude; in general, for each neuron i, taking $s = \omega_k(a + b)/2$ and $s' = \omega_k(a - b)/2$, we can write the representation of the the neuron-i preactivation for an input as $\cos s'(\alpha_i \cos s + \beta_i \sin s)$ for coefficients α_i, β_i. If we rewrite this expression into the amplitude-phase form of the Fourier series, by putting (α_i, β_i) into polar coordinates, our preactivation expression becomes $r_i \cos s'(\cos\phi_i \cos s - \sin\phi_i \sin s) = r_i \cos s' \cos(s + \phi_i)$.

Similarly, W_L does not take the dot product with $(\cos t, \sin t)$ but actually multiplies by $\alpha_i' \cos t + \beta_i' \sin t$. Taking $t = \omega_k c$, we can again put this expression into the amplitude-phase Fourier form by putting (α_i', β_i') into polar coordinates, giving $r_i'(\cos\psi_i \cos t - \sin\psi_i \sin t) = r_i' \cos(t + \psi_i)$.

The contribution to the logits from neuron i is then $r_i' r_i \,\text{ReLU}[\cos s' \cos(s + \phi_i)] \cos(t + \psi_i)$.

3.1.1 Each neuron has a primary frequency for both input and output

As seen in Figure 4, the majority of preactivations for each neuron consist of terms from a single key frequency, as does the neuron-logit map W_L. Interestingly, these the largest frequency component for the ReLU preactivation almost always *equal* to the largest frequency component for the corresponding row of W_L. This allows us to divide the neurons into clusters I_k, each of which correspond to a single frequency k.

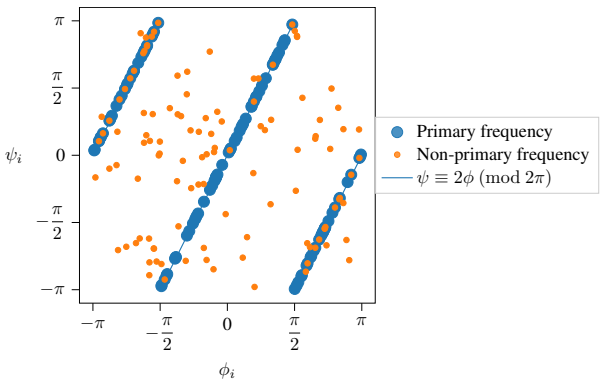
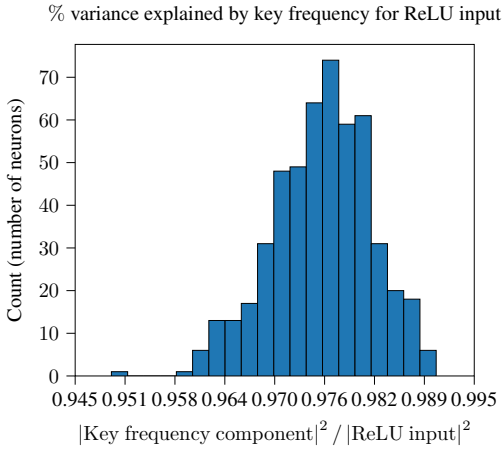
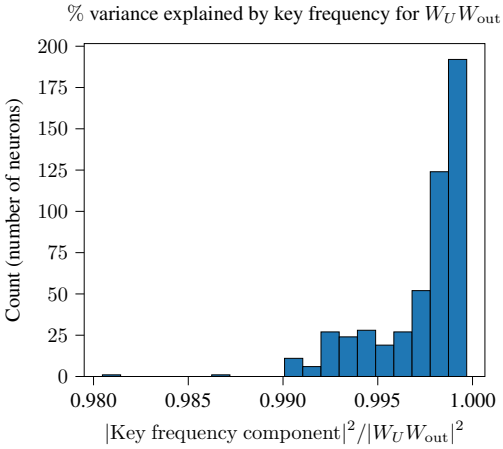


Figure 3: Angles for frequency $k = 12$. $\psi_i \approx 2\phi_i \pmod{2\pi}$ for the primary frequency of each neuron but not in general. The line has $R^2 = 0.9999$ and the intervals between angles have mean width 0.054, standard deviation 0.049. This shows that the angles are roughly uniform.



(a) For most neurons, above 96% of the variance of the ReLU input is explained by the largest Fourier frequency component. We compute the square of the largest Fourier coefficient divided by the sum of the squares of all Fourier coefficients and the square of the residual contributions.



(b) For most neurons, above 98% of the variance of the matrix W_{out} is explained by the largest Fourier frequency component. We compute the square of the largest Fourier coefficient divided by the sum of the squares of all Fourier coefficients.

Figure 4: Histograms of the variance explained by the largest Fourier frequency component for the pre-activations and neuron-logit map W_L for each of the 512 neurons in the mainline model.

3.1.2 THE OUTPUT PHASE IS DOUBLE THE INPUT PHASE

In Figure 2, we plot the input and output phase shift angles ϕ_i and ψ_i for the largest two frequencies. For each neuron's primary frequency, the neuron's output phase shift ψ_i is almost exactly twice its input phase shift ϕ_i.

3.2 FREQUENCY DOUBLING USING A TRIGONOMETRIC INTEGRAL IDENTITY

Treating the MLP layer as approximately performing numerical integration allows us to make sense of the previous observations. Recall that we can write the logits of the model as

$$\begin{aligned}\text{logit}(a,b,c) &\approx \textstyle\sum_{i=1}^{d_{\text{mlp}}} (W_L)_{ci} \cdot \text{ReLU}\left(\frac{1}{2}\text{OV}(a)_i + \frac{1}{2}\text{OV}(b)_i\right) \\ &\approx \textstyle\sum_{k\in\mathcal{K}} \sum_{i\in I_k} \cos\left(\omega_k(c+2\phi_i)\right) \text{ReLU}\left(\cos\left(\frac{\omega_k}{2}(a-b)\right)\cos\left(\frac{\omega_k}{2}(a+b+\phi_i)\right)\right)\end{aligned}$$

For each frequency k, we can write the contributions to the logits from neurons of frequency k as :

$$\text{logit}^{(k)}(a,b,c) = \left|\cos\left(\tfrac{\omega_k}{2}(a-b)\right)\right| \textstyle\sum_{i\in I_k} \cos\left(\omega_k(c+2\phi_i)\right) \text{ReLU}\left(\sigma_k \cos\left(\tfrac{\omega_k}{2}(a+b+\phi_i)\right)\right)$$

where $\sigma_k = 1$ if $\left|\cos\left(\frac{\omega_k}{2}(a-b)\right)\right| \geq 0$ and -1 otherwise.

Ignoring the $|\cos(\omega_k(a-b)/2)|$ scaling factor (which does not vary per neuron), we claim that the normalized MLP outputs

$$\sum_{i\in I_k} w_i \,\text{ReLU}[\sigma_k \cos(\tfrac{k}{2}(a+b)+\phi_i)]\cos(kc+2\phi_i) \tag{1}$$

can we well-thought of as approximating the integral (see Appendix E):

$$\int_{-\pi}^{\pi} \text{ReLU}[\sigma_k \cos(\tfrac{k}{2}(a+b)+\phi)]\cos(kc+2\phi)\,\mathrm{d}\phi = \frac{2}{3}\cos(k(a+b-c)). \tag{2}$$

Note that the above integral is valid for both $\sigma_k = -1, 1$.

This gives us the desired form for the output logits:

$$\begin{aligned}\text{logit}^{(k)}(a,b,c) &= |\cos(\omega_k(a-b)/2)|\cos(\omega_k(a+b-c)) \\ \text{logit}(a,b,c) &= \sum_{k\in\mathcal{K}} |\cos(\omega_k(a-b)/2)|\cos(\omega_k(a+b-c))\end{aligned}$$

4 VALIDATION VIA COMPACT GUARANTEES ON MLP PERFORMANCE

As evidence for the usefulness of the numerical integration interpretation, we use it to derive non-vacuous bounds on the output of the MLP *on all inputs* in time linear in the parameters.

4.1 COMPUTING NUMERICAL INTEGRATION ERROR

The validation of our interpretation as an integral then becomes a mathematical question of evaluating the efficiency of the quadrature scheme

$$\int_{-\pi}^{\pi} h_{a,b,c}(\phi)\,\mathrm{d}\phi \approx \sum_i w_i' h_{a,b,c}(\phi_i)$$

where $h_{a,b,c}(\phi_i) = \mathrm{ReLU}[\sigma_k \cos(\frac{k}{2}(a+b)+\phi_i)]\cos(kc+2\phi_i)$. The absolute error is given by

$$\varepsilon_{\int} := \left| \int_{-\pi}^{\pi} h_{a,b,c}(\phi)\,\mathrm{d}\phi - \sum_i w_i' h_{a,b,c}(\phi_i) \right| \tag{3}$$

and we can compute relative error by computing (the average value over a, b, c of)

$$\varepsilon_0 := \left| \int_{-\pi}^{\pi} h_{a,b,c}(\phi)\,\mathrm{d}\phi \right| \tag{4}$$

and then dividing Equation 3 by Equation 4 to give the relative error $\varepsilon_r := \varepsilon_{\int}/\varepsilon_0$.

Following Gross et al. (2024), if we can compute a non-vacuous error bound (i.e. a relative error that is strictly less than 1) in time linear in the number of parameters of the computation, we can be assured that our interpretation has some validity. Since there are d_{mlp} neurons and p points to evaluate our target is to compute an error bound in time $O(d_{\mathrm{mlp}} + p)$.

Recall the neuron contributions to the logits from Equation 2:

$$\mathrm{ReLU}[\sigma_k \cos(\tfrac{k}{2}(a+b)+\phi)]\cos(kc+2\phi)$$

Unfortunately, the error bound based on this equation is vacuous for some of our frequencies (see Appendix I). To improve the error bound, we can split ReLU as $\mathrm{ReLU}(x) = (x+|x|)/2$ and fold the 'id' component $x/2$ into the skip connection of the network. This will allow us to compress the contribution from the $x/2$ as a low-rank linear path, rather than having to perform an extremely crude bounding argument.

This turns $\mathrm{ReLU}[\sigma_k \cos(\frac{k}{2}(a+b)+\phi)]\cos(kc+2\phi)$ into

$$\tfrac{1}{2}\underbrace{\left|\sigma_k \cos(\tfrac{k}{2}(a+b)+\phi)\right|\cos(kc+2\phi)}_{h_{a+b,c,\sigma_k}} + \tfrac{1}{2}\sigma_k \cos(\tfrac{k}{2}(a+b)+\phi)\cos(kc+2\phi)$$

Define v_i to be the sum $v_i := \sum_{j\in I_{k_i}, \phi_j < \phi_i} w_j'$ and take ϕ_i to be the angle s.t. $x - v_i \in [0, w_i']$, so that for the $h(\phi)$ for *any* possible a, b, the discrepancy term is

$$\varepsilon_h := \left| \textstyle\int_{-\pi}^{\pi} h(x) - h(\phi_i)\,\mathrm{d}x \right|. \tag{5}$$

A crude bound is:

$$\begin{aligned}
&\leq \textstyle\int_{-\pi}^{\pi} |h(x) - h(\phi_i)|\,\mathrm{d}x \\
&\leq \textstyle\int_{-\pi}^{\pi} |x-\phi_i| \cdot \sup_x |h'(x)|\,\mathrm{d}x \\
&\leq \sup_x |h'(x)| \textstyle\sum_i \left(\int_{v_{i-1}-\phi_i}^{v_i-\phi_i} |x|\,\mathrm{d}x \right) \\
&= \sup_x |h'(x)| \sum_i \frac{1}{2} \begin{cases} \left|(v_i-\phi_i)^2 - (v_{i-1}-\phi_i)^2\right| & \text{if } \phi_i \in [v_{i-1}, v_i] \\ (v_i-\phi_i)^2 + (v_{i-1}-\phi_i)^2 & \text{otherwise} \end{cases}
\end{aligned}$$

That is, we take the sum of two error triangles when ϕ_i is within $[v_{i-1}, v_i]$ and otherwise we take the difference of triangles.

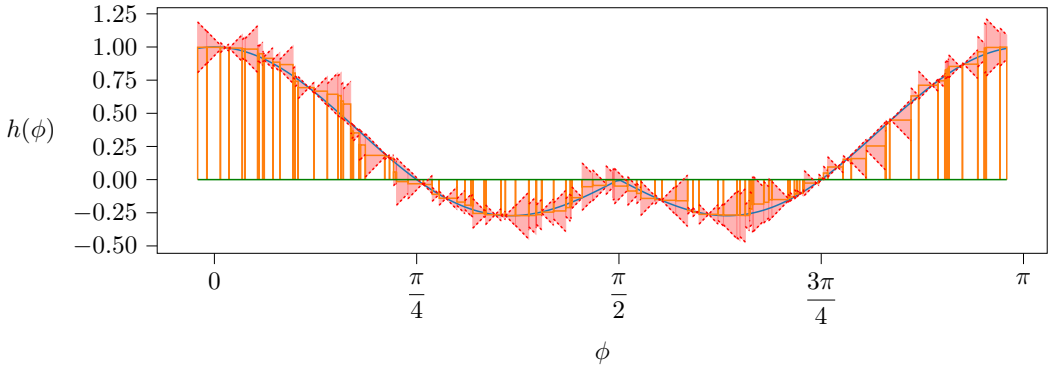


Figure 5: Error bound is the red area (for frequency $k = 12$). Note how the red area includes both the actual curve and the numerical integration approximation. Lipschitz constant is 2, so we bound things by +2 and -2,

For $h = h_{a+b,c,\sigma_k}$, we can bound

$$\sup_x |h'(x)| \leq 2$$

analytically, and the remaining sum can be evaluated easily in $\mathcal{O}(d_{\text{mlp}})$ time; call this $\varepsilon_{\approx\int}$.

Notice that this upper bound has no dependency on h, and therefore is a valid upper bound, for all possible a, b, c triplets. In other words, we can bound the error of the quadrature approximation for any a, b, c by evaluating an expression that takes time linear in the number of neurons.

This bound is represented visually in Figure 5.

We must also include approximation error from $\psi_i \approx 2\phi_i$, which we call the "angle approximation error," ε_ϕ: we have $\sum_i w'_j \left|\cos(k_i(a+b)/2 - \phi_i)\right| \cdot (\cos(kc + \psi_i) - \cos(kc + 2\phi_i)) \leq \sum_i w'_j \left|\cos(k_i(a+b)/2 - \phi_i)\right| \cdot |\psi_i - 2\phi_i| \leq \sum_i w'_j |\psi_i - 2\phi_i|$.

4.2 Empirical validation

We now provide empirical validation of our interpretation.

The network is well-approximated as doing numerical integration. We can compute the actual error $\varepsilon_\int / \varepsilon_0$ by brute force, giving numbers between 0.03 and 0.05, see Table 1.

The interpretation gives useful compression because integral explanation gives compact non-vacuous bounds. While less than 1, our bounds range from 0.48 to 0.7, see Table 1.

The bounds are so bad because we don't completely understand numerical integration. Intuitively, we'd like to center each box at its

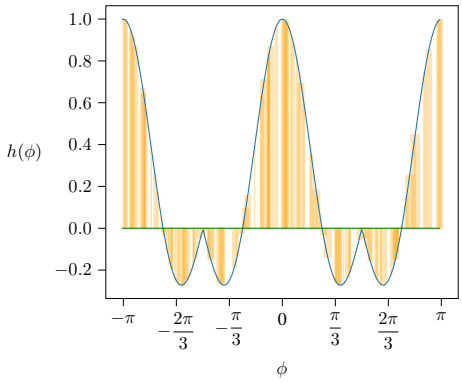


Figure 6: Numerical integration with boxes centered at ϕ_i (for frequency $k = 12$).

Error Bound Equation \ Freq.	12	18	21	22
$\varepsilon_\int / \varepsilon_0$	0.05	0.03	0.05	0.03
$(\varepsilon_{\approx\int} + \varepsilon_\phi)/\varepsilon_0$	0.70	0.49	0.54	0.48

Table 1: Error bounds by splitting ReLU into absolute value and identity components

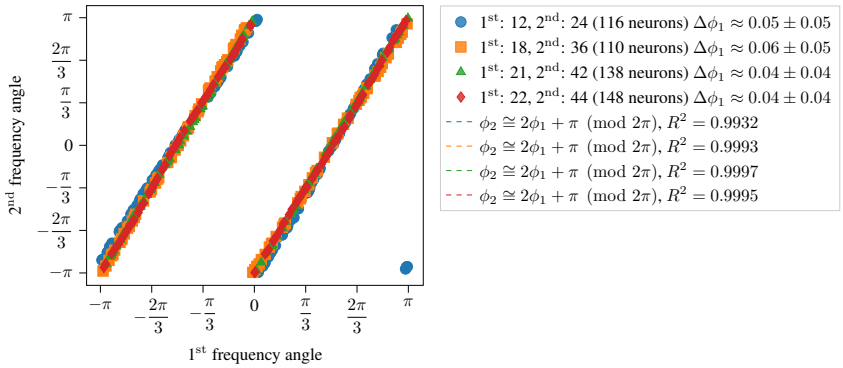


Figure 7: Not only are the secondary frequencies of neurons approximately double the primary frequencies, the input phase shift of the secondary frequencies are approximately twice the primary frequency phase shift plus π.

corresponding ϕ_i and compute the error that results from having box density above or below 1 at various points of the curve (Figure 6). We'd also like to take into account the fact that if one region has a box density above 1 and is adjacent to a region with a box density below one, these density errors *partially* cancel out. However, we don't know how to efficiently compute these effects, and the bounds we're able to give using non-uniform box densities instead of non-uniform angle locations is too crude.

5 The Role of Secondary Frequencies

5.1 Regressing Model Logits versus "Clock" and "Pizza" Logits

As noted above, Zhong et al. (2023) claim that logits are of the form ("pizza logits")

$$\text{logit}(a, b, c) \propto |\cos(\omega_k(a-b)/2)| \cos(k(a+b-c))$$

while Nanda et al. (2023a) claim that logits are of the form ("clock logits")

$$\text{logit}(a, b, c) \propto \cos(\omega_k(a+b-c))$$

Interestingly, if we regress the logits against the factors $|\cos(k(a-b)/2)| \cos(k(a+b-c))$, which gives an R^2 of 0.86, while if we regress them against just $\cos(k(a+b-c))$, we obtain an R^2 of 0.98 – substantially higher. So overall, the "clock" logits give a more accurate expression, suggesting that the analysis above is importantly incomplete. Interestingly, if we only consider the contribution to the logits from only the absolute value component of ReLU (Appendix K), the R^2 values become 0.99 and 0.85 respectively, suggesting that the network does not use nonlinearities when compensating.

In fact, the discrepancy is due to the effects of the substantially smaller non-primary frequencies. In particular, the "secondary frequency" – the second largest Fourier component – is important in the analysis here.

5.2 Using Secondary Frequencies to Better Approximate Clock Logits

For each of the neurons, the largest secondary frequency is almost always twice the primary frequency. For example, for neurons of frequency 12, the largest secondary frequency is 24, while for neurons of frequency 22, the largest secondary frequency is 15 ($= 59 - 22 \cdot 2$, note that cosine is symmetric about 0).

Note that the input shift of the secondary frequency is approximately twice the input shift of the primary frequency plus π (Figure 7).

The contribution of the doubled secondary frequency to the logits can thus be written as

$$\begin{aligned}\text{logit}^{(2k)}(a,b,c) &= \sum_{i\in I_k} \cos\left(\omega_k(c+2\phi_i)\right) \text{ReLU}\left[\cos\left(\omega_k(a-b)\right)\cos\left(\omega_k(a+b+2\phi_i+\pi)\right)\right] \\ &\approx \int_{-\pi}^{\pi} \cos(kc+2\phi)\,\text{ReLU}[-\cos\left(\omega_k(a-b)\right)\cos(k(a+b)+2\phi)]\,\mathrm{d}\phi \\ &= \int_{-\pi}^{\pi} \cos(kc+2\phi)\,\text{ReLU}[-\cos\left(\omega_k(a-b)\right)\cos(k(a+b)+2\phi)]\,\mathrm{d}\phi\end{aligned}$$

Letting $\theta = \phi + k(a+b)/2$, we get

$$= \int_{k(a+b)/2-\pi}^{k(a+b)/2+\pi} \cos(k(c-(a+b))+2\theta)\,\text{ReLU}[-\cos\left(\omega_k(a-b)\right)\cos(2\theta)]\,\mathrm{d}\theta$$

Because the integrand is 2π-periodic, the shift on the limits of integration is irrelevant:

$$= \int_{-\pi}^{\pi} \cos(k(c-(a+b))+2\theta)\,\text{ReLU}[-\cos\left(\omega_k(a-b)\right)\cos(2\theta)]\,\mathrm{d}\theta$$

By the law of cosine addition:

$$\begin{aligned}= \int_{-\pi}^{\pi} &\left[\cos(k(c-(a+b)))\cos(2\theta) - \sin(k(c-(a+b)))\sin(2\theta)\right] \\ &\cdot \text{ReLU}[-\cos\left(\omega_k(a-b)\right)\cos(2\theta)]\,\mathrm{d}\theta\end{aligned}$$

Because $\sin$ is odd and $\text{ReLU}[\cos]$ is even, the $\sin$ term integrates to 0.

$$\begin{aligned}&= \cos(k(c-(a+b))) \int_{-\pi}^{\pi} \cos(2\theta)\,\text{ReLU}[-\cos\left(\omega_k(a-b)\right)\cos(2\theta)]\,\mathrm{d}\theta \\ &= -\tfrac{\pi}{2}\cos\left(\omega_k(a-b)\right)\cos(\omega_k(a+b-c))\end{aligned}$$

Notably, when $\left|\cos\left(\frac{\omega}{2}(a-b)\right)\right|$ is close to 0, we must have that $\frac{\omega}{2}(a-b) \approx \pi/2$ or $\approx 3\pi/2$. This implies that $\omega_k(a-b) \approx \pi$, and so $\cos\left(\frac{\omega}{2}(a-b)\right) \approx -1$ and this term contributes positively to the correct logit $\cos(\omega_k(a+b-c))$, compensating for the weakness of the pizza algorithm in cases where $\left|\cos\left(\frac{\omega}{2}(a-b)\right)\right| \approx 0$.

6 Related work

Mechanistic Interpretability Our work falls squarely within the field of mechanistic interpretability (Olah et al., 2020; Elhage et al., 2021; Wang et al., 2022; Black et al., 2022), which aims to reverse engineer human-interpretable algorithms from neural networks. In particular, our work is a continuation of Nanda et al. (2023a) and Zhong et al. (2023)'s work understanding how transformers implement modular addition, which were in turn inspired by previous work studying the learning dynamics of small models on algorithmic tasks (Power et al., 2022; Liu et al., 2022).

Infinite width limits Our interpretation of MLPs as computing a finite sum numerical approximation to an integral can be thought of as understanding as taking a finite width neural network to the "infinite width" limit. The technique of the infinite width limit of an MLP layer to make the network easier to work with analytically is a common technique used in deep learning theory (Rahimi & Recht, 2008; Jacot et al., 2018; Yang & Hu, 2020; Chizat et al., 2024). However, while existing work tends to consider the properties of arbitrarily wide neural networks initialized randomly and then trained with gradient descent on arbitrary tasks, in this work we use the infinite width limit to compactly explain how the MLPs of particular trained neural networks perform modular addition.

7 Discussion

Our results provide an example of interpreting MLPs analytically in order to efficiently compress not just what an MLP does but also *how* it's doing it. Notably, the key steps in our derivation

of efficient bounds was splitting the input into orthogonal output-relevant and output-irrelevant directions, decomposing the pre-activations as a product of functions of these axes, reindexing the neurons by the output-relevant direction so that the reindexed post-activations depend *only* on the output-irrelevant directions, and compressing independently over the output-relevant direction term and the output-irrelevant direction term. We believe that some of these steps could be adapted to interpret other MLPs, even those where we cannot derive closed-form analytic representations. At the same time, the bounds we derive in subsection N.2 could be improved to understand how the network is allocating boxes under the curve, which we hope to address with future work.

Author Contributions

If you'd like to, you may include a section for author contributions as is done in many journals. This is optional and at the discretion of the authors.

Acknowledgments

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

References

Sid Black, Lee Sharkey, Leo Grinsztajn, Eric Winsor, Dan Braun, Jacob Merizian, Kip Parker, Carlos Ramón Guevara, Beren Millidge, Gabriel Alfour, and Connor Leahy. Interpreting neural networks through the polytope lens, 2022.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Lénaïc Chizat, Maria Colombo, Xavier Fernández-Real, and Alessio Figalli. Infinite-width limit of deep linear neural networks. *Communications on Pure and Applied Mathematics*, 77(10): 3958–4007, 2024.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023. URL https://arxiv.org/abs/2309.08600.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/toy_model/index.html.

Joshua Engels, Isaac Liao, Eric J. Michaud, Wes Gurnee, and Max Tegmark. Not all language model features are linear, 2024. URL https://arxiv.org/abs/2405.14860.

Jason Gross, Rajashree Agrawal, Thomas Kwa, Euan Ong, Chun Hei Yip, Alex Gibson, Soufiane Noubir, and Lawrence Chan. Compact proofs of model performance via mechanistic intepretability, June 2024. URL https://arxiv.org/abs/2406.11779.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36, 2024.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Ziming Liu, Ouail Kitouni, Niklas S Nolte, Eric Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. *Advances in Neural Information Processing Systems*, 35:34651–34663, 2022.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models, 2024. URL https://arxiv.org/abs/2403.19647.

Neel Nanda and Joseph Bloom. Transformerlens. https://github.com/TransformerLensOrg/TransformerLens, 2022.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint*, 2023a. doi: 10.48550/arXiv.2301.05217. URL https://arxiv.org/abs/2301.05217.

Neel Nanda, Senthooran Rajamanoharan, János Kramár, and Rohin Shah. Fact finding: Attempting to reverse-engineer factual recall on the neuron level, Dec 2023b. URL https://www.alignmentforum.org/posts/iGuwZTHWb6DFY3sKB/fact-finding-attempting-to-reverse-engineer-factual-recall.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. URL https://distill.pub/2020/circuits/zoom-in.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks, 2017. URL https://arxiv.org/abs/1606.05336.

Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. *Advances in neural information processing systems*, 21, 2008.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint*, 2022. doi: 10.48550/arXiv.2211.00593.

Greg Yang and Edward J Hu. Feature learning in infinite-width neural networks. *arXiv preprint arXiv:2011.14522*, 2020.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks, 2023. URL https://arxiv.org/abs/2306.17844.

A MODEL TRAINING DETAILS

We train 1-layer transformers with constant attention $= \frac{1}{2}$ (equivalently, with QK clamped to 0), as implemented by TransformerLens (Nanda & Bloom, 2022) with the following parameters

$$\begin{aligned} \text{d\_vocab} &= p + 1 = 59 + 1 \\ \text{n\_ctx} &= 2 + 1 \\ \text{d\_model} &= 128 \\ \text{d\_mlp} &= 512 \\ \text{d\_head} &= 32 \\ \text{n\_heads} &= 4 \\ \text{n\_layers} &= 1 \\ \text{act\_fn} &= \texttt{'relu'} \end{aligned}$$

We take 80% of all (a, b) pairs $\mod p$ as the training set, and the rest as the validation set. We set the loss function to be the mean log probabilities of the correct logit positions, and train for 10000 epochs using the AdamW optimizer with the default weight_decay $= 0.01$. The large number of epochs is such that the resulting model achieves 100% accuracy on all possible input pairs. We chose 151 random seeds which are pseudorandomly deterministically derived 0, along with 150 values read from `/dev/urandom`.

Each training run takes approximately 11–12 GPU-minutes to complete on an NVIDIA GeForce RTX 3090. In total, the experiments in this paper took less than 1000 GPU-hours.

B MORE FIGURES AND RESULTS FOR MAINLINE MODEL

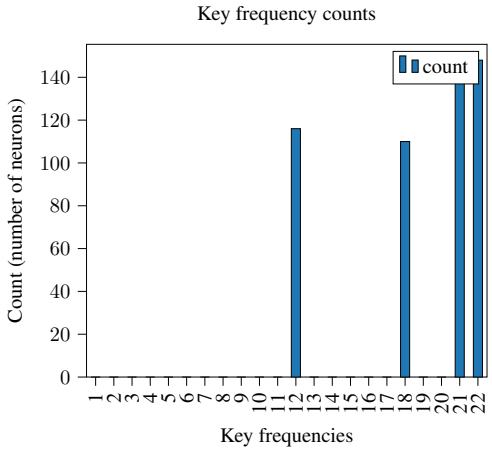


Figure 8: The key frequencies for this model are 12, 18, 21, and 22. (These are multiples of $2\pi/p$)

C RESULTS FOR OTHER RANDOM SEEDS

To ensure that this phenomenon is not specific to the model we looked at, we trained 151 models with the same setup and different random seeds (leading to different weight initialization and train-test split). We replicate some of the analysis above to show that a significant proportion of these models follow the above explanation.

For our **second observation**, we notice that most neurons are single frequency, with the largest frequency explaining more than 90% of the variance of the ReLU input, see Figure 13. The same is true for the W_{out} matrix, see Figure 14. For our **third observation**, we check that (disregarding neurons where the largest frequency is the constant term) 100 out of 151 models ('good models')

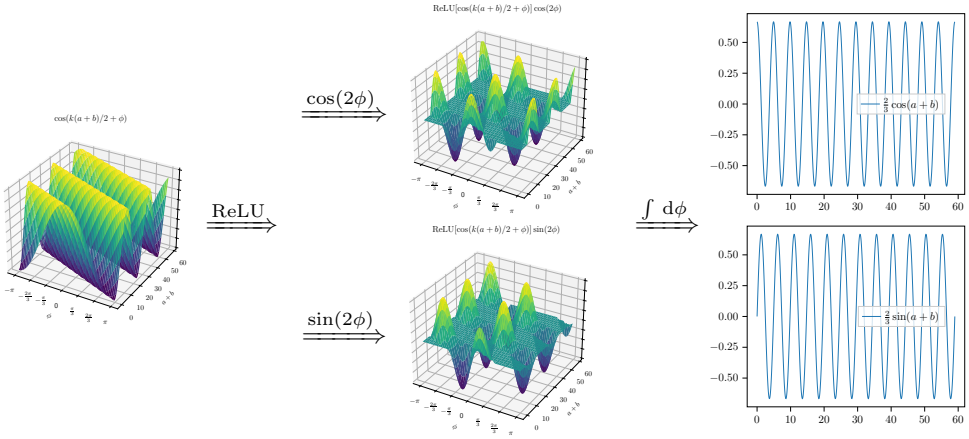


Figure 9: The network computes logit

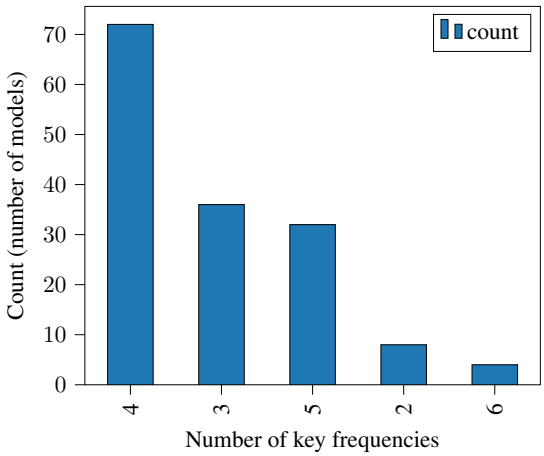


Figure 10: Most models have 3 to 5 key frequencies, which is in line with what we would expect to make the above argument work.

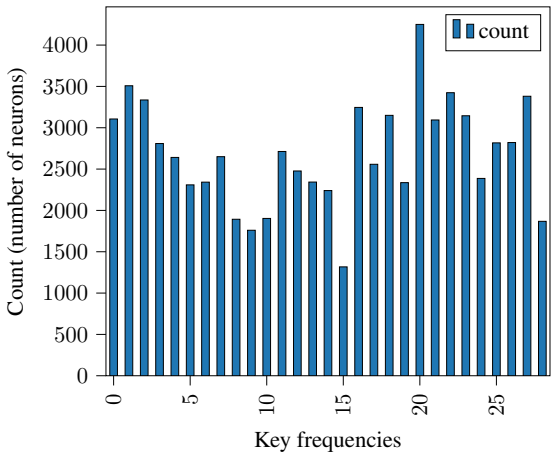


Figure 11: Key frequencies for the models are roughly uniformly distributed across all possible values.

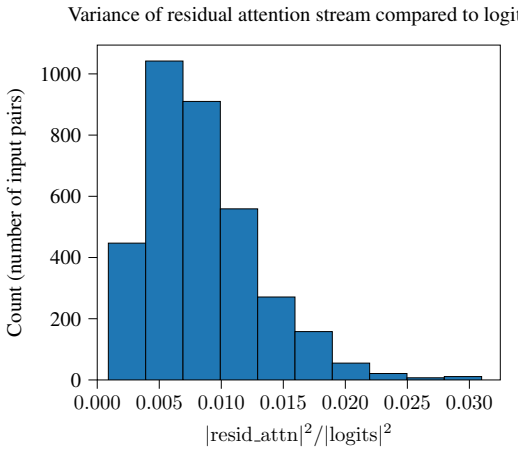


Figure 12: Residual connection from the attention stream causes mostly less than 3% of the variance of the logits.

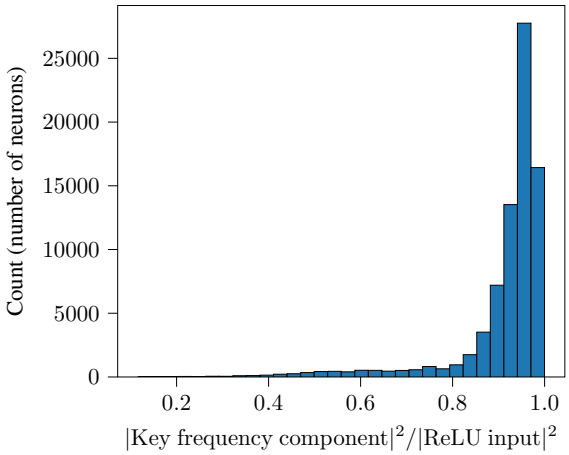


Figure 13: For most neurons, above 90% of the variance of the ReLU input is explained by the largest Fourier frequency component.

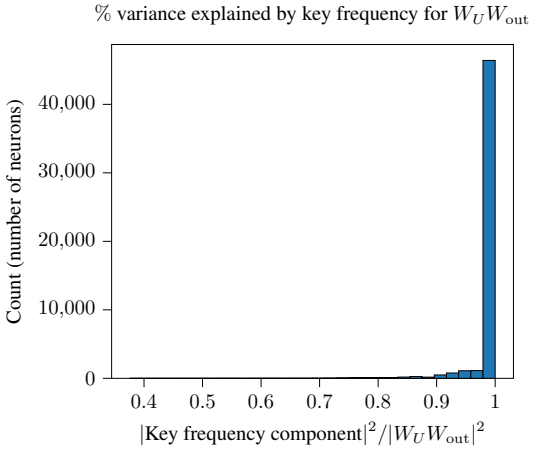


Figure 14: For most neurons in 'good' models, above 98% of the variance of the W_{out} matrix is explained by the largest Fourier frequency component.

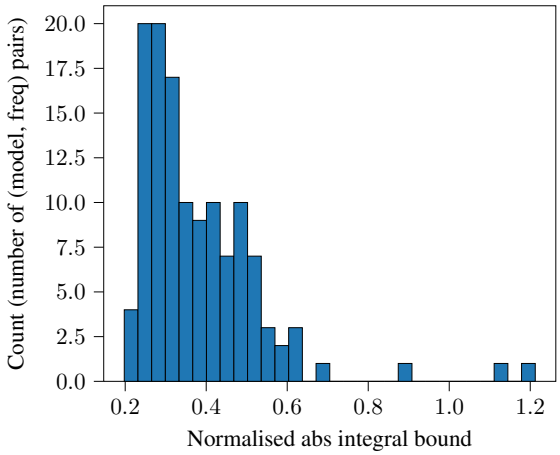


Figure 15: Most of the pairs have normalised abs integral bounds less than 0.6, compared to the naive abs error bound of 0.85.

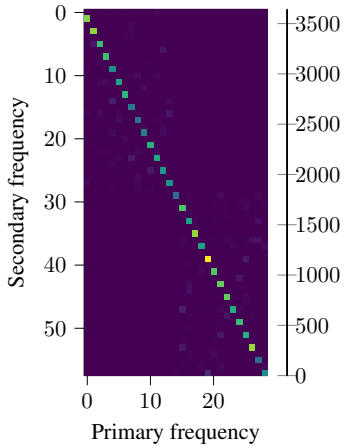


Figure 16: The second largest non-zero Fourier component is 2 times the primary frequency for 83% of the neurons.

have the frequencies matching for all neurons, with a further 27 models where the frequencies don't match for less than 10 (out of 512 neurons). For our **fourth observation**, we look at the 100 good models and see that for 78 of them, we have $R^2 > 0.9$ between the angles for all key frequencies. For our **fifth observation**, we look at the 100 good models and see that for 54 of them, we have *mean width of intervals* $>$ *standard deviation of width*, showing that angles are roughly uniformly distributed.

Finally, we carry out the error bound calculations for the 100 good models. For each (model, freq) pair, we can calculate the error bounds as in Table 1. We see that for 295 out of 358 such pairs (82%), the empirical errors (normalised abs cos error, normalised abs sin error, normalised id cos error, normalised id sin error) are all less than 0.1. For these pairs, the normalised abs integral bound has a median of 0.40, which is 47% of the naive abs bound of 0.85. Also, 99% of the normalised abs integral bounds are less than the naive abs bound (see Figure 15. Hence, the numerical integration phenomenon explained above indeed appears in most trained models, and our method of proof is able to produce a non-vacuous error bound whenever this phenomenon occurs.

For the relationship observed between primary and secondary frequencies, we see that it still holds when we use different random seeds. In particular, around 84% of the neurons have the second largest non-zero Fourier component being 2 times the primary frequency (see Figure 16). Also, amongst

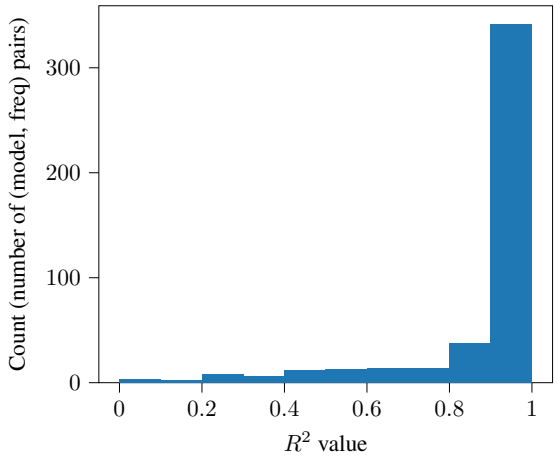


Figure 17: Most of the (model, freq) pairs have the angle for the secondary frequency closely resembling 2 times the primary frequency plus π.

these neurons, the angle for the secondary frequency is approximate 2 times the angle for the primary frequency $+\pi$, with $R^2 > 0.9$ for most (model, freq) pairs.

D TRIGONOMETRIC IDENTITIES

We use the following trigonometric identities throughout the paper:

$$\sin\alpha + \sin\beta = 2\sin\frac{\alpha+\beta}{2}\cos\frac{\alpha-\beta}{2} \tag{6}$$

$$\cos\alpha + \cos\beta = 2\cos\frac{\alpha+\beta}{2}\cos\frac{\alpha-\beta}{2} \tag{7}$$

$$\cos\alpha\cos\beta = \frac{\cos(\alpha+\beta)+\cos(\alpha-\beta)}{2} \tag{8}$$

$$\cos(\alpha+\beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta \tag{9}$$

$$\cos(\alpha-\beta) = \cos\alpha\cos\beta + \sin\alpha\sin\beta \tag{10}$$

$$\cos(\alpha+\pi) = -\cos\alpha \tag{11}$$

E DERIVATION OF TRIG INTEGRAL

Instead of splitting the summation into two chunks, converting each into an integral, and evaluating each integral, we can replace the summand with an integral directly and evaluate that. Also, we use the corrected phase shift $\psi_i = 2\phi_i$:

$$\begin{aligned}
&\text{logit}(a,b,c) \\
&\approx \sum_{k\in\mathcal{K}}\sum_{i\in I_k} w_i \,\text{ReLU}\left(\sigma_k\cos(k\tfrac{a+b}{2}+\phi_i)\right) \\
&\qquad\qquad \cos(kc+2\phi_i) \\
&\approx \sum_{k\in\mathcal{K}} Z_k \int_{-\pi}^{\pi} \text{ReLU}\left(\sigma_k\cos(k\tfrac{a+b}{2}+\phi)\right) \\
&\qquad\qquad \cos(kc+2\phi)\,\mathrm{d}\phi
\end{aligned}$$

Using $\mathcal{F}_\pm$ to denote the integral

$$\mathcal{F}_\pm = \int_{-\pi}^{\pi} \text{ReLU}\left(\pm\cos(k\frac{a+b}{2}+\phi)\right)\cos(kc+2\phi)\,\mathrm{d}\phi\,,$$

let $s = k\frac{a+b}{2}$ and $t = kc$. Using the periodicity of cosine (Equation 11), we have

$$\begin{aligned}
\mathcal{F}_- &= \int_{-\pi}^{\pi} \mathrm{ReLU}\left(-\cos(k\frac{a+b}{2} + \phi)\right)\cos(kc + 2\phi)\,\mathrm{d}\phi \\
&= \int_{-\pi}^{\pi} \mathrm{ReLU}\left(\cos(k\frac{a+b}{2} + \phi + \pi)\right)\cos(kc + 2\phi + 2\pi)\,\mathrm{d}\phi \\
&= \int_{0}^{2\pi} \mathrm{ReLU}\left(\cos(k\frac{a+b}{2} + \phi')\right)\cos(kc + 2\phi')\,\mathrm{d}\phi' \\
&= \int_{-\pi}^{\pi} \mathrm{ReLU}\left(\cos(k\frac{a+b}{2} + \phi)\right)\cos(kc + 2\phi)\,\mathrm{d}\phi \\
&= \mathcal{F}_+ \,.
\end{aligned}$$

So we may write $\mathcal{F} := \mathcal{F}_+ = \mathcal{F}_-$. Note that the integrand is non-zero only when $\phi \in [-\pi/2 - s, \pi/2 - s]$. Applying the cosine product-sum identity (Equation 8) and doing some algebra:

$$\begin{aligned}
\mathcal{F} &= \int_{-\pi/2-s}^{\pi/2-s} \cos(s + \phi)\cos(t + 2\phi)\,\mathrm{d}\phi \\
&= \frac{1}{2}\int_{-\pi/2-s}^{\pi/2-s} \cos(s - t - \phi) + \cos(s + t + 3\phi)\,\mathrm{d}\phi \\
&= \frac{1}{2}\left[\sin(\phi - s + t) + \frac{1}{3}\sin(s + t + 3\phi)\right]_{-\pi/2-s}^{\pi/2-s} \\
&= \frac{1}{2}\left[\sin(\pi/2 - 2s + t) + \frac{1}{3}\sin(3\pi/2 - 2s + t)\right] \\
&\quad - \frac{1}{2}\left[\sin(-\pi/2 - 2s + t) + \frac{1}{3}\sin(-3\pi/2 - 2s + t)\right]
\end{aligned}$$

Using the periodicity of sine and cosine, we have

$$\begin{aligned}
\mathcal{F} &= \frac{1}{2}\left[\sin(\pi/2 - 2s + t) + \frac{1}{3}\sin(3\pi/2 - 2s + t)\right] \\
&\quad - \frac{1}{2}\left[\sin(-\pi/2 - 2s + t) + \frac{1}{3}\sin(-3\pi/2 - 2s + t)\right] \\
&= \frac{1}{3}\sin(\pi/2 - 2s + t) + \frac{1}{3}\sin(\pi/2 + 2s - t) \\
&= \frac{1}{3}\cos(2s - t) + \frac{1}{3}\cos(-2s + t) \\
&= \frac{2}{3}\cos(2s - t) \\
&= \frac{2}{3}\cos(k(a + b - c))\,,
\end{aligned}$$

as desired.

That is, the pizza model computes its logits using the trigonometric integral identity:

$$\begin{aligned}
&\int_{-\pi}^{\pi} \mathrm{ReLU}\left(\cos(k(a+b)/2 + \phi)\right)\cos(kc + 2\phi)\,\mathrm{d}\phi \\
&= \frac{2}{3}\cos(k(a + b - c))
\end{aligned}$$

Or equivalently:

$$\begin{aligned}
&\cos(k(a + b - c)) \\
&= \frac{3}{2}\int_{-\pi}^{\pi} \mathrm{ReLU}\left(\cos(k(a+b)/2 + \phi)\right)\cos(kc + 2\phi)\,\mathrm{d}\phi
\end{aligned}$$

F Derivation of trig integral including secondary frequencies

Let

$$s_k = k\tfrac{a+b}{2} \qquad u_k = k\tfrac{a-b}{2} \qquad v_k = \cos(u_k) \qquad t_k = kc$$

and let β_i be the coefficient of the secondary frequency term and γ_i be the constant term.

$$\begin{aligned}
&\operatorname{logit}(a,b,c)\\
&\approx \sum_{k\in\mathcal{K}}\sum_{i\in I_k} w_i \operatorname{ReLU}\left[\cos(k\tfrac{a-b}{2})\cos(k\tfrac{a+b}{2}+\phi_i) + \beta_i \cos(k(a-b))\cos(k(a+b)+2\phi_i) + \gamma_i\right]\\
&\qquad \cos(kc+2\phi_i)\\
&\approx \sum_{k\in\mathcal{K}} Z_k \int_{-\pi}^{\pi} \operatorname{ReLU}\left[\cos(k\tfrac{a-b}{2})\cos(k\tfrac{a+b}{2}+\phi) + \bar\beta_k \cos(k(a-b))\cos(k(a+b)+2\phi) + \bar\gamma_k\right]\\
&\qquad \cos(kc+2\phi)\,\mathrm{d}\phi\\
&= \sum_{k\in\mathcal{K}} Z_k \int_{-\pi}^{\pi} \operatorname{ReLU}\left[\cos(u_k)\cos(s_k+\phi) + \bar\beta_k \cos(2u_k)\cos(2s_k+2\phi) + \bar\gamma_k\right]\\
&\qquad \cos(t_k+2\phi)\,\mathrm{d}\phi\\
&= \sum_{k\in\mathcal{K}} Z_k \int_{-\pi}^{\pi} \operatorname{ReLU}\left[v_k\cos(s_k+\phi) + \bar\beta_k(2v_k^2-1)\cos(2s_k+2\phi) + \bar\gamma_k\right]\\
&\qquad \cos(t_k+2\phi)\,\mathrm{d}\phi
\end{aligned}$$

Reindexing with $\theta = \phi + s_k$:

$$= \sum_{k\in\mathcal{K}} Z_k \int_{s_k-\pi}^{s_k+\pi} \operatorname{ReLU}\left[v_k\cos(\theta) + \bar\beta_k(2v_k^2-1)\cos(2\theta) + \bar\gamma_k\right]\cos(t_k-2s_k+2\theta)\,\mathrm{d}\theta$$

Using the fact that the integrand is 2π-periodic:

$$= \sum_{k\in\mathcal{K}} Z_k \int_{-\pi}^{\pi} \operatorname{ReLU}\left[v_k\cos(\theta) + \bar\beta_k(2v_k^2-1)\cos(2\theta) + \bar\gamma_k\right]\cos(t_k-2s_k+2\theta)\,\mathrm{d}\theta$$

Define

$$f_k(\theta) = \operatorname{ReLU}\left[v_k\cos(\theta) + \bar\beta_k(2v_k^2-1)\cos(2\theta) + \bar\gamma_k\right]$$

and use the cosine addition formula to get:

$$\begin{aligned}
&= \sum_{k\in\mathcal{K}} Z_k \int_{-\pi}^{\pi} f_k(\theta)(\cos(t_k-2s_k)\cos(2\theta) - \sin(t_k-2s_k)\sin(2\theta))\,\mathrm{d}\theta\\
&= \sum_{k\in\mathcal{K}} Z_k\cos(t_k-2s_k)\int_{-\pi}^{\pi} f_k(\theta)\cos(2\theta)\,\mathrm{d}\theta - Z_k\sin(t_k-2s_k)\int_{-\pi}^{\pi} f_k(\theta)\sin(2\theta)\,\mathrm{d}\theta
\end{aligned}$$

Since f_k is even and $\sin$ is odd, the second integral evaluates to zero, giving

$$= \sum_{k\in\mathcal{K}} Z_k\cos(t_k-2s_k)\int_{-\pi}^{\pi} f_k(\theta)\cos(2\theta)\,\mathrm{d}\theta$$

G Derivation of another integral

Replicating Equation 13, we have

$$\mathbb{E}_{0<a+b\le p}\left|\tfrac{4}{3}\cos(2\pi k(a+b)/p)\right| \approx \mathbb{E}_{0<a+b\le p}\left|\tfrac{4}{3}\sin(2\pi k(a+b)/p)\right|$$

For sufficiently large p, we can approximate the expectation as an integral:

$$\begin{aligned}
&\mathbb{E}_{0<a+b\leq p}\left|\frac{4}{3}\cos(2\pi k(a+b)/p)\right| \\
&\approx \frac{1}{p}\int_0^p \left|\frac{4}{3}\cos(2\pi kx/p)\right| \,\mathrm{d}x \\
&= \frac{4}{3}\int_0^1 |\cos(2\pi kx')| \,\mathrm{d}x'
\end{aligned}$$

H Derivation of the logit expression

The model we use is a 1L transformer with W_QW_K clamped to 0, taking as input $(a, b, \text{'='})$ and reading predictions off the final token.

A more precise equation of the model is

$$\begin{aligned}
\text{logit}(a,b,c) = \textstyle\sum_i \text{ReLU}(\text{OV}(a)_i + \text{OV}(b)_i + \text{embed}(b)_i) \\
\cdot (W_{\text{out}})_{ci} + \text{residual terms}
\end{aligned} \tag{12}$$

A more precise version of the breakdown from **??** is

$$\begin{aligned}
F_1(a,b) &= \text{OV}(a) + \text{OV}(b) + \text{Embed}(b) \\
F(a,b) &= W_{\text{in}}F_1(a,b) + b_{\text{in}} \\
\text{logits}(a,b) &= W_{\text{out}}\,\text{ReLU}(F(a,b)) + b_{\text{out}} + W_UF_1(a,b)
\end{aligned}$$

where W_{in}, W_{out}, W_U, b_{in}, b_{out}, are parameters of the model.

We start with

$$\text{OV}(a)_i \approx \alpha_i'' \cos(k_ia) + \beta_i'' \sin(k_ia) = \alpha_i' \cos(k_ix + \phi_i)$$

Then

$$\begin{aligned}
F_1(a,b) &= \text{OV}(a) + \text{OV}(b) + \text{Embed}(b) \\
&\approx a_i(\cos(k_ia) + \cos(k_ib)) + b_i(\sin(k_ia) + \sin(k_ib)) \\
&+ \text{Embed}(b) \\
&= c_i(\cos(k_ia + \phi_i) + \cos(k_ib + \phi_i)) + \text{Embed}(b)
\end{aligned}$$

where we use the reverse direction of the cosine addition formula

$$\cos(k_ia + \phi_i) = \cos(k_ia)\cos(\phi_i) - \sin(k_ix)\sin(\phi_i)$$

We ignore the Embed term (which is the residual stream), and use the trigonometric identity

$$\cos(a) + \cos(b) = 2\cos((a+b)/2)\cos((a-b)/2)$$

to get

$$\begin{aligned}
F(a,b) &= W_{\text{in}}F_1(a,b) + b_{\text{in}} \\
&\approx W_{\text{in}}c_i(\cos(k_ia + \phi_i) + \cos(k_ib + \phi_i)) + b_{\text{in}} \\
&\approx \alpha_i \cos(k_i(a-b)/2)\cos(k_i(a+b)/2 + \phi_i)
\end{aligned}$$

where $d_i = 2(W_{\text{in}}c)_i$.

Finally, for the logit expression, using

$$(W_{\text{out}})_{ci} \approx \beta_i \cos(k_ic + \psi_i)$$

we have

$$\begin{aligned}
\text{logits}(a,b) &= W_{\text{out}}\,\text{ReLU}(F(a,b)) + b_{\text{out}} + W_UF_1(a,b) \\
&\approx W_{\text{out}}\,\text{ReLU}(F(a,b))
\end{aligned}$$

(again ignoring the residual stream)

Then

$$\begin{aligned}
&\text{logits}(a,b,c)\\
&= \sum_i \text{ReLU}((F(a,b))_i)(W_{\text{out}})_{ci}\\
&\approx \sum_i \text{ReLU}(\alpha_i \cos(k_i(a-b)/2)\\
&\cos(k_i(a+b)/2+\phi_i))\beta_i \cos(k_i c+\psi_i))\\
&= \sum_k |\cos(k(a-b)/2)| \sum_i |\alpha_i|\beta_i \,\text{ReLU}[(-1)^{s_{i,k(a-b)}}\\
&\cos(k(a+b)/2+\phi_i))]\cos(kc+\psi_i)\\
&\approx \sum_k |\cos(k(a-b)/2)|\{\sum_i |\alpha_i|\beta_i \,\text{ReLU}[(-1)^{s_{i,k(a-b)}}\\
&\cos(k(a+b)/2+\phi_i)]\cos(kc+2\phi_i)\}
\end{aligned}$$

where we sum over the 'key frequencies' k (in our example $k = 12, 18, 21, 22$), and take out the sign of α_i, and we use again the cosine addition formula.

I Numerical integration error bound for ReLU function

Recall that in sec:empirical-validation, we computed the error bound for integrating the absolute value function rather than the ReLU function in the network. This is because we can break down

$$\text{ReLU}(x) = \frac{x}{2} + \frac{|x|}{2}$$

and the identity part of ReLU integrates to zero. Hence, the baseline result (of approximating the integral to zero) makes more sense when we only consider the absolute value part of ReLU.

However, using the general form of our error bound (Equation 5), it is simple to replicate the analysis for the whole ReLU function. We have the same bound

$$\sup_x |h'(x)| \leq 2$$

Moreover, utilising the fact that ReLU evaluates to 0 on half the range of integration, we can reduce our error bound by only considering the largest half of the boxes (with some boundary effects). This gives us the following error bounds:

Error Bound Type \ Freq.	12	18	21	22	Equation
Normalised ReLU cos error	0.05	0.04	0.04	0.03	(3)/(5)
Normalised ReLU sin error	0.03	0.05	0.04	0.03	(3)/(5)
Angle approximation error	0.13	0.07	0.06	0.05	(14)
Numerical ReLU $\int_{-\pi}^{\pi}$ bound	0.55	0.44	0.42	0.37	(15)
Numerical ReLU $\int_{0}^{\pi}$ bound	0.21	0.15	0.17	0.15	(16)
Total numerical ReLU $\int_{-\pi}^{\pi}$ bound	0.68	0.50	0.48	0.42	$(15)+(14)$
Total numerical ReLU $\int_{0}^{\pi}$ bound	0.55	0.37	0.40	0.34	$2\cdot(16)+(14)$
Naive ReLU cos baseline	0.42	0.42	0.42	0.42	(4), (13)
Naive ReLU sin baseline	0.42	0.42	0.42	0.42	(4), (13)

Table 2: Error bounds for the ReLU function

Note the baseline is halved because the identity component of ReLU yields a zero integral.

J Detailed computation of

The maximum empirical error (obtained by evaluating the expression for each value of $a+b \mod p$) is shown below:

Error Bound Type \ Freq.	12	18	21	22	Equation
Normalised abs $\cos$ error	0.04	0.03	0.04	0.03	(3)/(5)
Normalised abs $\sin$ error	0.05	0.05	0.03	0.02	(3)/(5)
Normalised id $\cos$ error	0.06	0.05	0.04	0.04	(3)/(5)
Normalised id $\sin$ error	0.02	0.05	0.04	0.04	(3)/(5)
Angle approximation error	0.14	0.07	0.06	0.06	(14)
Numerical abs $\int_{-\pi}^{\pi}$ bound	0.59	0.52	0.50	0.44	(15)
Numerical abs $\int_{0}^{\pi}$ bound	0.23	0.17	0.20	0.17	(16)
Total numerical abs $\int_{-\pi}^{\pi}$ bound	0.73	0.59	0.56	0.50	$(15) + (14)$
Total numerical abs $\int_{0}^{\pi}$ bound	0.59	0.41	0.46	0.40	$2 \cdot (16) + (14)$
Naive abs $\cos$ baseline	0.85	0.85	0.85	0.85	(4), (13)
Naive abs $\sin$ baseline	0.85	0.85	0.85	0.85	(4), (13)

Table 3: Error bounds by splitting ReLU into absolute value and identity components

The first four rows (normalised abs & id $\cos$ & $\sin$ error) compute the error by brute force exactly: $\left|\int_{-\pi}^{\pi} h(x) - h(\phi_i)\,\mathrm{d}x\right|$ for $h \in \{h_{a+b,|C|}, h_{a+b,|S|}, h_{a+b,C}, h_{a+b,S}\}$. Rows ten and eleven compute the baseline for the error $\left|\int_{-\pi}^{\pi} h(x)\,\mathrm{d}x\right|$ for $h \in \{h_{a+b,|C|}, h_{a+b,|S|}\}$, which is given by

$$\mathbb{E}_{0<a+b\leq p}\left|\tfrac{4}{3}\cos(2\pi k(a+b)/p)\right| \approx \mathbb{E}_{0<a+b\leq p}\left|\tfrac{4}{3}\sin(2\pi k(a+b)/p)\right| \tag{13}$$

Note that the baseline for $h \in \{h_{a+b,C}, h_{a+b,S}\}$ is 0. Line five (angle discrepancy) computes the error from $\psi \approx 2\phi$:

$$\textstyle\sum_i w'_j|\psi_i - 2\phi_i|. \tag{14}$$

Line six (numerical abs $\int_{-\pi}^{\pi}$ bound) computes the error bound from the integral:

$$\min_\theta \textstyle\sum_i \begin{cases} \left|(v_i - (\phi_i - \theta))^2 - (v_{i-1} - (\phi_i - \theta))^2\right| & \text{if } \phi_i \in [v_{i-1}, v_i] \\ (v_i - (\phi_i - \theta))^2 + (v_{i-1} - (\phi_i - \theta))^2 & \text{otherwise} \end{cases} \tag{15}$$

Shifting by θ is permitted because our functions are periodic with period π; it does not matter where we start the integral. Line seven (numerical abs $\int_{0}^{\pi}$ bound) takes advantage of the fact that h is π-periodic to integrate from 0 to π: taking $\hat{w}_i := w'_i/2$, $\hat{v}_i := v_i/2$, and $\hat{\phi}_i := \phi_i$ for $\phi_i \geq 0$ and $\hat{\phi}_i := \phi_i + \pi$ for $\phi_i < 0$, and we compute

$$\min_\theta \textstyle\sum_i \begin{cases} \left|(\hat{v}_i - (\hat{\phi}_i - \theta))^2 - (\hat{v}_{i-1} - (\hat{\phi}_i - \theta))^2\right| & \text{if } \hat{\phi}_i \in [\hat{v}_{i-1}, \hat{v}_i] \\ (\hat{v}_i - (\hat{\phi}_i - \theta))^2 + (\hat{v}_{i-1} - (\hat{\phi}_i - \theta))^2 & \text{otherwise} \end{cases} \tag{16}$$

Line eight (total numerical abs $\int_{-\pi}^{\pi}$ bound) computes the combined error bound from the integral and angle discrepancy. Line nine (total numerical abs $\int_{0}^{\pi}$ bound) takes advantage of the fact that h is π-periodic to integrate from 0 to π. This allows us to 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.)

Lines six and seven (numerical abs $\int$ bound) also both take advantage of the fact that the function is 2π-periodic, allowing us to shift the intervals formed above by any constant. When bounding approximation error, we use the shift that gives the lowest bound.

The exact error is much smaller than the size of the integral, and the mathematical error bound is also smaller than the size of the integral. This gives convincing evidence that the model is indeed performing numerical integration.

K Analysis of the 'identity' component of ReLU

We can break down ReLU into two parts,

$$\mathrm{ReLU}(x) = \frac{x}{2} + \frac{|x|}{2}$$

The integrals then split into

$$\begin{aligned}\textstyle\int_{-\pi}^{\pi} \cos(-2\phi)\tfrac{1}{2}\cos(\tfrac{k}{2}+\phi)\,\mathrm{d}\phi &= \tfrac{2}{3}\cos(k)\\ \textstyle\int_{-\pi}^{\pi} \sin(-2\phi)\tfrac{1}{2}\cos(\tfrac{k}{2}+\phi)\,\mathrm{d}\phi &= \tfrac{2}{3}\sin(k)\\ \textstyle\int_{-\pi}^{\pi} \cos(-2\phi)\tfrac{1}{2}|\cos(\tfrac{k}{2}+\phi)|\,\mathrm{d}\phi &= 0\\ \textstyle\int_{-\pi}^{\pi} \sin(-2\phi)\tfrac{1}{2}|\cos(\tfrac{k}{2}+\phi)|\,\mathrm{d}\phi &= 0\end{aligned}$$

We see that the 'identity' part of the ReLU yields a zero integral. So does this part of the model contribute to the logits? It turns out that the answer is yes. To resolve this issue, we look at the discrepancy between the results suggested by previous work: Zhong et al. (2023) claim that logits are of the form

$$\mathrm{logit}(a,b,c) \propto |\cos(k(a-b)/2)|\cos(k(a+b-c))$$

while Nanda et al. (2023a) claim that logits are of the form

$$\mathrm{logit}(a,b,c) \propto \cos(k(a+b-c))$$

To check which is correct, we regress the logits against the factors $|\cos(k(a-b)/2)|\cos(k(a+b-c))$, which gives an R^2 of 0.86, while if we regress them against just $\cos(k(a+b-c))$, we obtain an R^2 of 0.98. So overall, Nanda et al. (2023a) give a more accurate expression, but this seems to go against the analysis we did above, which led to the expression in Zhong et al. (2023). (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 $a-b$, in particular $|\cos(k(a-b)/2)|$. This is a good thing because when $\cos(k(a-b)/2)$ is small, the logit difference between the correct logit $(a+b)$ and other logits will also be small, which will lead to a higher loss. The identity component slightly counters this effect. We can rewrite the identity component as:

$$W_{\text{out}}\mathrm{OV}(a)/2 + W_{\text{out}}\mathrm{OV}(b)/2 + W_{\text{out}}\text{embed}(b)/2$$

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

$$\begin{aligned}F_2(a,b)_c = \text{residual stream} + \text{absolute value terms}\\ + \mathrm{logit\_id1}[c,a] + \mathrm{logit\_id1}[c,b] + \mathrm{logit\_id2}[c,b]\end{aligned}$$

We carry out a 2D Fourier transform to find out the decomposition of the $\mathrm{logit\_id1}$ and $\mathrm{logit\_id2}$ matrices (because W_{out} and $\mathrm{OV}(a)$ are sparse in the (1D) Fourier basis, so their product will naturally be sparse in the 2D Fourier basis). We get $\mathrm{logit\_id1}[c,a] \approx 2\Re(\sum_k a_k e^{i(kc-2ka)})$, where the frequencies k here are the same as **??**. Hence, the output from the identity component of ReLU is (ignoring $\mathrm{logit\_id2}$ for now, which comes from the residual stream and is smaller): $\sum_k D_k(\cos(kc-2ka)+\cos(kc-2kb)) + E_k(\sin(kc-2ka)+\sin(kc-2kb)) = \sum_k \cos(k(b-a))(D_k\cos(k(c-a-b)) + E_k\sin(k(c-a-b)))$.

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

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

$$\begin{aligned}F_2(a,b)_c &\approx \textstyle\sum_k (C_k\,|\cos(k(b-a)/2)|\\ &\qquad + D_k\cos(k(b-a)))\cos(k(a+b-c))\\ &= \textstyle\sum_k (2D_k^2\,|\cos(k(b-a)/2)|^2\\ &\qquad + C_k\,|\cos(k(b-a)/2)|)\cos(k(a+b-c))\\ &\qquad - D_k\cos(k(a+b-c))\end{aligned}$$

using double angle formula. Since $D_k < 0$, the $\sum_k -D_k \cos(k(a+b-c))$ term gives some cushion for the base performance of the model (since as we discussed, the $\cos(k(a+b-c))$ term is why the model gives the highest logit when $c = a + b$). Moreover, the $2D_k^2 |\cos(k(b-a)/2)|^2$ 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 $|\cos(k(b-a)/2)|$ 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.

L OTHER PLOTS

M 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.

N FUTURE TECHNICAL WORK

To complete the technical work laid out in section 7, 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.

N.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_{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 w_i to each neuron, and an assignment of angles ϕ_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.

N.2 BOUNDING THE ERROR OF THE MLP

To bound the error of our interpretation 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?