The Inside of Reasoning: Uncovering Algebraic Structure in Bilinear MLPs

Kaustubh Sharma and Aditya Chauhan
Indian Institute of Technology (IIT) Roorkee, Uttarakhand, India
kaustubh_s@ee.iitr.ac.in; aditya_c@mfs.iitr.ac.in

Abstract

Standard ReLU MLPs can solve simple algebraic and relational tasks, but the internal mechanisms they learn are often hard to express in simple symbolic terms. Recent work has proposed bilinear MLP layers as an alternative with more structured, tensor-based interactions that may be analyzed directly from weights. In this paper, we compare bilinear and ReLU MLPs on two controlled settings where the ground-truth computation is exactly known. First, on modular addition and multiplication, we extract class-wise interaction matrices from each model and study them in Fourier and singular-value bases. For addition, the bilinear model recovers operators whose Fourier spectra closely match the ideal circulant structure, whereas the ReLU model concentrates energy on a few frequencies. For multiplication, the bilinear operators are consistently lower-rank and show steeper singular-value decay. Second, on a relational composition task over a cycle of $N = 400$ entities, the bilinear model learns an almost permutation-like successor operator whose powers T^k remain accurate for many steps, while the ReLU operator is diffuse and collapses to near-uniform under composition. These results suggest that bilinear MLPs induce internal operators that are closer to explicit symbolic rules, and are promising building blocks for more interpretable neuro-symbolic systems.

Introduction

Multi-layer perceptrons (MLPs) with ReLU activations are a core component in modern deep learning. Their simplicity and expressivity are well understood on the function level, but much less is known about the algorithms they learn internally. In practice, MLP layers often implement *superposition* of many features in a shared basis (Olah et al. 2020; Elhage et al. 2022), leading to *polysemantic* neurons that activate for multiple unrelated patterns. This makes it difficult to read off a clean computation from the weights.

Mechanistic interpretability aims to reverse-engineer such computations in small models (Olah et al. 2020). A common strategy is to train models on toy algorithmic tasks where the ground truth is known, and then analyze weights and activations. Prior work has studied modular addition and related group-structured problems (Nanda et al. 2023; Chughtai, Chan, and Nanda 2023), and transitive inference on graphs (Lippl et al. 2024). These studies show that ReLU networks can represent the correct function, but often via messy, high-rank or clustered internal codes that are hard to describe in simple algebraic terms.

Bilinear MLP layers have been proposed as a more structured alternative (Shazeer 2020; Sharkey 2023; Pearce et al. 2024). Instead of applying a non-linearity to a single linear projection, they take an elementwise product of two linear projections. The resulting layer can be written as a linear map over a third-order interaction tensor, while still implementing a nonlinear function of the inputs. This makes it possible to analyze the learned interaction tensor directly, using familiar linear algebra tools such as eigendecomposition and singular value decomposition.

In this paper, we ask a simple question: *when trained on small reasoning tasks, do bilinear MLPs learn internal operators that are closer to the underlying symbolic rules than those learned by ReLU MLPs?* We focus on settings where the ground-truth operator is exactly known, and we study not only accuracy but also the structure of the learned interaction matrices.

We consider two families of tasks:

- **Modular arithmetic.** Following prior work, we train both architectures to compute addition and multiplication in $\mathbb{Z}_{97}$. For each output class, we extract a matrix $M_k[a, b]$ from the logits and analyze it in the Fourier basis (for addition) and via singular-value spectra (for multiplication), comparing to the exact algebraic operators.
- **Relational composition on a cycle.** We train both models on a single successor relation $R_1(h) = (h + 1) \bmod N$ over a cycle with $N = 400$ entities. From the trained models we extract the one-step transition operator T, measure how sharp its columns are, and study the behavior of its powers T^k compared to the true k-step successor.

Across both settings, we find that the bilinear architecture induces internal operators that are more structured and closer to the ground-truth symbolic ones. In modular addition, its Fourier spectra nearly match the ideal circulant structure; in multiplication, its interaction matrices are lower-rank. In the relational task, the bilinear model learns an almost permutation-like transition matrix whose compositions remain accurate for many steps, whereas the ReLU

*Members of Data Science Group, IIT Roorkee

model learns a diffuse kernel whose compositions become unusable.

In summary, our contributions are:

1. We formalize a simple bilinear MLP architecture for binary operations and relational reasoning, and show how to extract interaction matrices and transition operators from its weights and logits.
2. We provide a mechanistic comparison of bilinear and ReLU MLPs on modular addition and multiplication, using Fourier entropy and singular-value spectra to quantify how closely their learned operators match the true algebraic structure.
3. We show that on a synthetic transitivity task, bilinear MLPs learn sharp, permutation-like transition operators that support multi-step composition, while ReLU MLPs learn diffuse kernels whose compositional behavior quickly degenerates.

Background

Mechanistic interpretability on algorithmic tasks

Mechanistic interpretability aims to understand *how* a trained network implements a function, not only whether it achieves high accuracy (Olah et al. 2020). A common approach is to train small models on tasks where the ground-truth computation is exactly known, and then analyze weights, activations and intermediate representations.

Several works have studied algorithmic tasks with group structure. Nanda et al. (2023) analyze modular addition in transformers, showing that models often implement the operation via Fourier-mode circuits rather than a direct table lookup. Chughtai, Chan, and Nanda (2023) study group composition tasks and find that standard architectures can learn distributed, high-rank representations of group actions. Lippl et al. (2024) investigate transitive inference on synthetic graphs, identifying failure modes where networks memorize local comparisons instead of learning a global relation.

These results suggest that, even on simple problems, standard ReLU-based MLPs may solve the task using mechanisms that are algebraically correct but difficult to express in a clean symbolic form. This motivates studying architectures whose internal computation is more structured and easier to analyze.

Bilinear MLP layers

A standard MLP layer takes an input vector $x \in \mathbb{R}^d$, applies an affine map, and passes it through an elementwise nonlinearity:

$$h = \sigma(Wx + b), \tag{1}$$

where $W \in \mathbb{R}^{m \times d}$ and σ is typically ReLU (Nair and Hinton 2010). This architecture is highly expressive but entangles features through both the weight matrix and the nonlinearity, which complicates weight-based analysis.

Bilinear MLP layers replace the nonlinearity with an elementwise product of two linear projections (Shazeer 2020; Sharkey 2023; Pearce et al. 2024). In the setting we study, the layer receives two inputs $x, y \in \mathbb{R}^d$ (e.g., embeddings of two tokens) and computes

$$h = (xW_1) \odot (yW_2) \in \mathbb{R}^m \tag{2}$$

optionally followed by a linear output projection. Here $W_1, W_2 \in \mathbb{R}^{d \times m}$ and $\odot$ denotes elementwise multiplication, also known as the Hadamard product.

This layer can be written as a purely linear operation over a third-order interaction tensor $T \in \mathbb{R}^{d \times d \times m}$. For each hidden unit k,

$$h_k = \sum_{i,j} T_{ijk}\, x_i\, y_j, \qquad T_{ijk} = (W_1)_{ik}\,(W_2)_{jk}. \tag{3}$$

Thus the nonlinearity in input space arises from quadratic interactions between features, but the parameters enter linearly in T. This makes it possible to analyze the learned interactions by inspecting T or induced matrices derived from it. Figure 1 schematically compares a standard ReLU MLP block with the bilinear block we use.

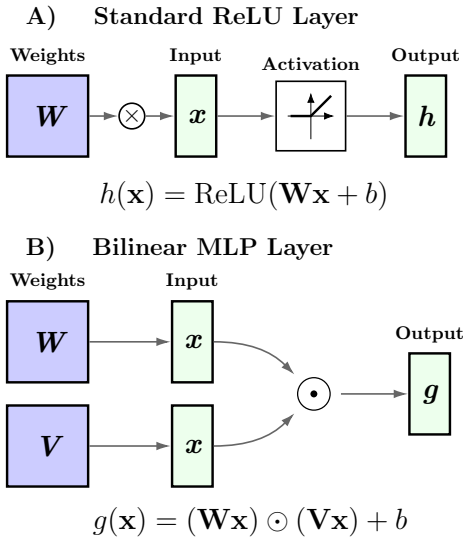


Figure 1: Comparison between a standard ReLU MLP layer (top) and the bilinear MLP layer used in this work (bottom). The bilinear layer replaces the elementwise nonlinearity with a Hadamard product of two linear projections.

Notation

We use the following notation throughout.

- For modular arithmetic, we work over $\mathbb{Z}_p$ with $p = 97$ prime. Inputs are pairs $(a, b) \in \{0, \ldots, p-1\}^2$. The target for addition is $k = (a + b) \bmod p$, and for multiplication $k = (a \cdot b) \bmod p$.
- For relational composition, we consider N entities indexed by $h \in \{0, \ldots, N-1\}$ arranged on a directed cycle, and a single successor relation $R_1(h) = (h + 1) \bmod N$.

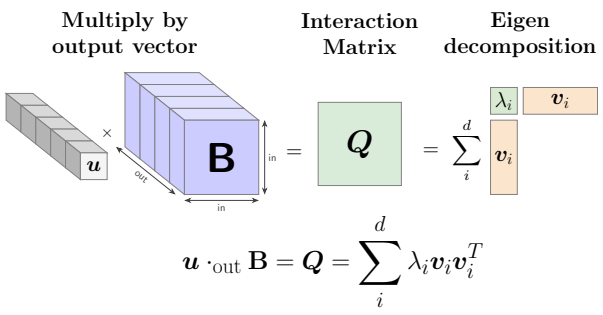


Figure 2: From bilinear weights to interaction matrices. For a fixed output class, the bilinear parameters define an interaction matrix M_k over input pairs. This matrix can be analyzed with standard linear algebra tools such as eigendecomposition or SVD to study the induced operator.

Methodology

In this section, we define the models we study, how we extract interaction operators from them, and the analysis metrics we apply.

Models for modular arithmetic

Inputs and outputs. For both modular addition and multiplication, we treat the problem as a p-way classification task. The model receives a pair of integers $(a, b) \in \{0, \dots, p-1\}^2$ and must predict the result $k \in \{0, \dots, p-1\}$.

We embed each integer via a learned embedding matrix $E \in \mathbb{R}^{p \times d}$. Let $e_a, e_b \in \mathbb{R}^d$ denote the embeddings of a and b.

Bilinear MLP. The bilinear model applies a single bilinear layer followed by a linear readout:

$$u = e_a W_1, \tag{4}$$
$$v = e_b W_2, \tag{5}$$
$$h = u \odot v, \tag{6}$$
$$\ell = h W_{\text{out}}, \tag{7}$$

where $W_1, W_2 \in \mathbb{R}^{d \times m}$, $h \in \mathbb{R}^m$ and $W_{\text{out}} \in \mathbb{R}^{m \times p}$. The predicted distribution over outputs is $\text{softmax}(\ell)$.

Given one-hot inputs and fixed embeddings, this architecture induces, for each class k, a bilinear form

$$M_k[a, b] = \ell_k(a, b), \tag{8}$$

which we can express analytically as

$$M_k[i, j] = \sum_{r=1}^{m} (W_1)_{ir} \, (W_2)_{jr} \, (W_{\text{out}})_{rk}, \tag{9}$$

where i, j index embedding rows. In practice, we also verify that this weight-derived M_k matches the matrix obtained from logits (up to numerical precision). Figure 2 illustrates the extraction and analysis of such interaction matrices.

ReLU MLP. The ReLU baseline uses a standard one-hidden-layer MLP on the concatenated embeddings:

$$x = [e_a; e_b] \in \mathbb{R}^{2d}, \tag{10}$$
$$h = \text{ReLU}(Wx + b) \in \mathbb{R}^m, \tag{11}$$
$$\ell = h W_{\text{out}}, \tag{12}$$

with $W \in \mathbb{R}^{m \times 2d}$ and $W_{\text{out}} \in \mathbb{R}^{m \times p}$. This model does not have an inherent bilinear factorization, but since the input space is finite we can still define

$$M_k[a, b] = \ell_k(a, b) \tag{13}$$

by evaluating the logits on all (a, b) pairs and stacking them into a $p \times p$ matrix. This gives a common object M_k for both models.

Models for relational composition

For the relational task, the input is a head entity $h \in \{0, \dots, N-1\}$ and a relation identifier r. In our experiments we use a single relation R_1, so $r \in \{0\}$, and the target is the successor tail $t = (h + 1) \bmod N$.

Bilinear KG model. We use a bilinear block analogous to the modular case, but now the two inputs are an entity and a relation. Let $E_{\text{ent}} \in \mathbb{R}^{N \times d}$ and $E_{\text{rel}} \in \mathbb{R}^{R \times d}$ be embedding matrices for entities and relations. For head h and relation r,

$$e_h = E_{\text{ent}}[h], \tag{14}$$
$$e_r = E_{\text{rel}}[r], \tag{15}$$

and the model computes

$$\ell = \big((e_h W_h) \odot (e_r W_r)\big) \, W_{\text{out}}, \tag{16}$$

with $W_h, W_r \in \mathbb{R}^{d \times m}$ and $W_{\text{out}} \in \mathbb{R}^{m \times N}$. The logits $\ell \in \mathbb{R}^N$ are over possible tails.

ReLU KG model. The ReLU baseline concatenates the head and relation embeddings and applies a standard MLP:

$$x = [e_h; e_r] \in \mathbb{R}^{2d}, \tag{17}$$
$$z = \text{ReLU}(Wx + b) \in \mathbb{R}^m, \tag{18}$$
$$\ell = z W_{\text{out}}, \tag{19}$$

with logits over tails.

Transition operator. For a fixed relation R (here R_1), both models define a conditional distribution over tails given head:

$$p(t \mid h, R) = \text{softmax}(\ell(h, R))_t. \tag{20}$$

We collect these into a transition operator $T \in \mathbb{R}^{N \times N}$ by setting

$$T[t, h] = p(t \mid h, R_1), \tag{21}$$

so each column h is the distribution over tails from head h in one step.

Interaction matrices and analysis metrics

We now summarize the operators we analyze and the metrics we use.

Class-wise interaction matrices M_k. For modular tasks, we define, for each class k, a matrix

$$M_k[a, b] = \ell_k(a, b), \tag{22}$$

where (a, b) ranges over all input pairs. For the bilinear model we can compute M_k either from weights or from logits; for the ReLU model we use logits only.

Fourier analysis for addition. Modular addition in $\mathbb{Z}_p$ corresponds to a circulant operator that is diagonal in the Fourier basis. To quantify how close the learned operators are to this structure, we compute the two-dimensional discrete Fourier transform (DFT) of each M_k:

$$\widehat{M}_k = F M_k F^*, \tag{23}$$

where F is the $p \times p$ DFT matrix and $*$ denotes conjugate transpose. We then form a normalized power spectrum

$$P_k(u, v) = \frac{\left|\widehat{M}_k[u, v]\right|^2}{\sum_{u',v'} \left|\widehat{M}_k[u', v']\right|^2}, \tag{24}$$

and define the Fourier entropy

$$H_k = -\sum_{u,v} P_k(u, v) \log P_k(u, v). \tag{25}$$

For the true addition operator, P_k is concentrated along a diagonal in frequency space, and H_k is close to $\log p$. We report the mean entropy over classes and compare bilinear and ReLU models to this ground truth.

Singular-value spectra for multiplication. For modular multiplication, the ideal operator is not diagonal in the Fourier basis, but is still structured and relatively low-rank. For each class k, we first *center* M_k by subtracting row and column means, then compute its singular values $\sigma_1 \geq \sigma_2 \geq \ldots$. We define the effective rank at energy level $\alpha \in (0, 1)$ as the smallest r such that

$$\sum_{i=1}^{r} \sigma_i^2 \;\geq\; \alpha \sum_{i=1}^{p} \sigma_i^2. \tag{26}$$

We report the distribution of effective ranks across classes for each model, and visualize the normalized singular-value decay σ_i/σ_1 to compare how quickly the spectrum drops.

Column entropy and multi-step accuracy for relational composition. For the relational task, we treat each column $T[:, h]$ as a probability distribution over tails and compute its Shannon entropy

$$H(h) = -\sum_{t} T[t, h] \log T[t, h]. \tag{27}$$

A true permutation has one-hot columns with entropy close to zero; a diffuse kernel has higher entropy.

To study composition, we consider powers of the learned operator. Let T^k denote the k-step transition matrix obtained by repeated matrix multiplication. For each k and each head h, we compare the predicted tail

$$\hat{t}_k(h) = \arg\max_t T^k[t, h] \tag{28}$$

to the ground-truth k-step successor $t_k(h) = (h + k) \bmod N$. We define the k-step accuracy as

$$\mathrm{Acc}(k) = \frac{1}{N} \sum_{h=0}^{N-1} 1\{\hat{t}_k(h) = t_k(h)\}. \tag{29}$$

We plot $\mathrm{Acc}(k)$ as a function of k for bilinear and ReLU models.

We train all models with Adam (lr = 1e-3, batch size = 16) for up to 500 epochs with early stopping on validation accuracy, using the same hyperparameters for bilinear and ReLU models. Unless otherwise stated, results are from a single random seed (0); in preliminary checks with over 10 seeds we observed qualitatively similar behaviour.

Results

Experiment 1: Modular arithmetic in $\mathbb{Z}_{97}$

Training behaviour For both addition and multiplication tasks, we train bilinear and ReLU MLPs with the same embedding and hidden dimensions. On both modular addition and modular multiplication, both models achieve very low training loss and high validation accuracy. Training loss and validation accuracy curves for both tasks are shown in Figure 3.

Addition: Fourier structure of learned operators For modular addition, we compare the learned class-wise interaction matrices M_k to the exact operator in the Fourier basis. As a sanity check, for the bilinear model we verify that the matrix computed analytically from the weights matches the matrix obtained from logits: the mean absolute difference between the two versions of M_k is 1.0×10^{-6} and the maximum difference is 1.5×10^{-5}.

We then compute the two-dimensional DFT of each M_k and its normalized power spectrum P_k, and report the Fourier entropy H_k as defined in the methodology. For the true modular addition operator, the mean entropy across classes is

$$\mathbb{E}[H_k^{\text{true}}] \approx 4.57 \pm 4.3 \times 10^{-16},$$

reflecting a spectrum concentrated along a diagonal in frequency space. The bilinear model closely tracks this value:

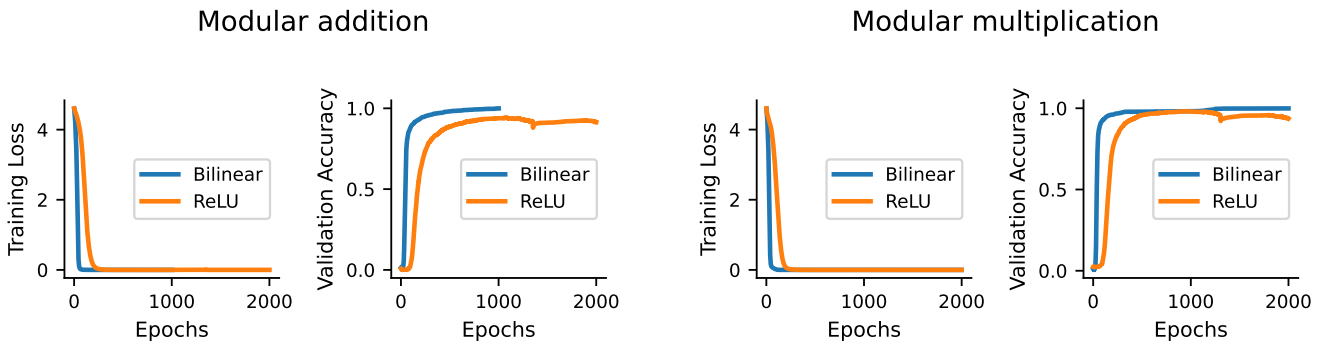
$$\mathbb{E}[H_k^{\text{bilinear}}] \approx 4.33,$$

with a small standard deviation across classes. In contrast, the ReLU model yields much lower entropies:

$$\mathbb{E}[H_k^{\text{ReLU}}] \approx 0.37,$$

indicating that most spectral energy is concentrated in a few frequencies. The distribution of H_k for all three operators is shown in Figure 4.

Fourier-domain heatmaps for representative classes (Figure 6) highlight this difference visually. For the bilinear model, $\widehat{M}_k$ exhibits a clear diagonal band of high magnitude in the (u, v) plane, closely resembling the ground-truth circulant structure. For the ReLU model, the spectrum is sparse and irregular: energy is scattered in a small number of isolated peaks with extremely sparse diagonal pattern. This



(a) (b)

Figure 3: Training loss and validation accuracy for modular addition (left) and modular multiplication (right), comparing bilinear and ReLU MLPs. Both models fit the training data well.

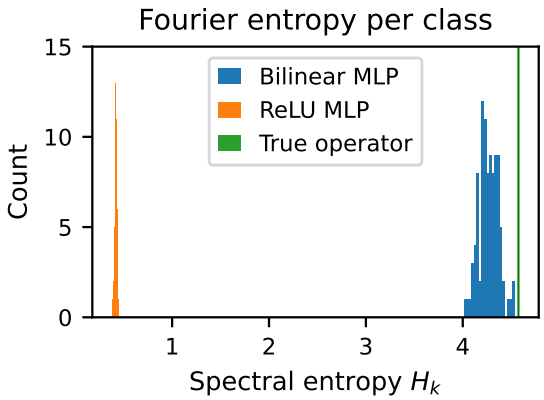


Figure 4: Fourier entropy H_k of the interaction matrices for modular addition, comparing the true operator, the bilinear model, and the ReLU model. The bilinear entropies cluster near the true value, while the ReLU entropies are much lower.

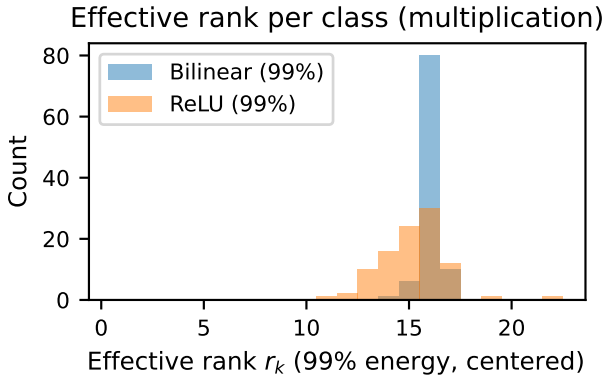


Figure 5: Effective rank r_k at 99% energy for modular multiplication. Bilinear interaction matrices are less variable than those of the ReLU model.

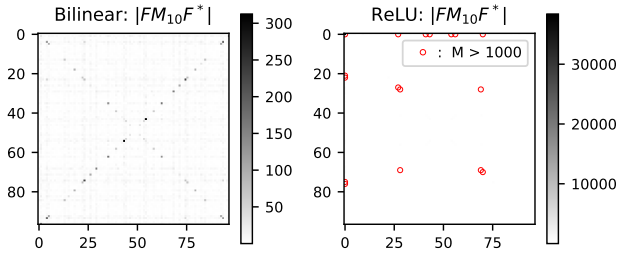


Figure 6: Fourier-domain magnitude of a representative interaction matrix M_k for modular addition. Left: bilinear model; right: ReLU model. The bilinear spectrum shows a clear diagonal band, closely matching the circulant structure of the true operator. The ReLU spectrum is extremely sparse, with energy scattered across a few isolated frequencies.

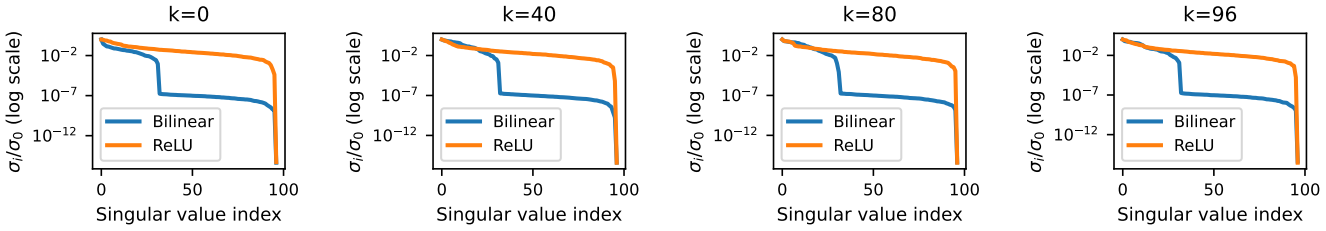
suggests that the bilinear network has learned an operator that closely matches the ideal group action in the Fourier basis, whereas the ReLU network implements addition via a more idiosyncratic spectral combination.

Multiplication: spectral decay and effective rank For modular multiplication, we analyze the centered interaction matrices M_k via singular value decomposition. After subtracting row and column means, we compute the singular values σ_i of each M_k and examine the normalized spectra σ_i/σ_1 on a log scale for several representative classes (Figure 7). In the bilinear model, the spectra decay steeply: a small number of singular values account for most of the variance, and the tail drops rapidly. In the ReLU model, the spectra are noticeably flatter: the singular values decrease more slowly, and the tail remains substantial.

We summarize this quantitatively using the effective rank at two energy levels, $\alpha = 0.90$ and $\alpha = 0.99$. At 90% energy, the bilinear model has

$$r_{\text{eff}}^{\text{bilinear}}(0.90) \approx 22.1 \pm 0.77,$$

while the ReLU model requires roughly twice as many sin-



(a) (b) (c) (d)

Figure 7: Normalized singular value decay for centered interaction matrices M_k in modular multiplication, for four representative classes $k \in \{0, 40, 80, 96\}$. Each panel plots σ_i/σ_1 on a log scale. The bilinear spectra (blue) decay much more sharply than the ReLU spectra (orange), indicating that the bilinear operators are effectively lower-dimensional.

gular directions:

$$r_{\text{eff}}^{\text{ReLU}}(0.90) \approx 41.9 \pm 5.45.$$

At 99% energy, the gap narrows but remains in the same direction:

$$r_{\text{eff}}^{\text{bilinear}}(0.99) \approx 16.4 \pm 1.00, \qquad r_{\text{eff}}^{\text{ReLU}}(0.99) \approx 18.0 \pm 3.64.$$

The distributions of effective ranks are shown in Figure 5.

Overall, these results indicate that for modular multiplication the bilinear model realizes each output class via a more concentrated, low-dimensional operator, whereas the ReLU model spreads the computation over a higher-dimensional subspace. Together with the addition results, this supports the view that bilinear MLPs tend to learn interaction matrices that more closely track the underlying algebraic structure.

Experiment 2: Relational composition on a cycle

One-step transition operators On the relational task, we train both models on a single successor relation $R_1(h) = (h+1) \bmod N$ over a cycle with $N = 400$ entities. Training is performed on the full set of one-step triples $(h, R_1, (h+1) \bmod N)$, and both models reach perfect training accuracy within $\approx$ 12–14 epochs.

From the trained models we extract the one-step transition operator $T \in \mathbb{R}^{N \times N}$ as described in the methodology. For each head h, the column $T[:, h]$ is the predicted distribution over tails after one application of R_1. We measure the sharpness of these columns via their Shannon entropy $H(h)$.

The bilinear model produces columns that are close to one-hot: the mean column entropy is

$$\mathbb{E}[H^{\text{bilinear}}(h)] \approx 0.74 \pm 0.57,$$

much closer to the ideal permutation (entropy $= 0$) than to a uniform distribution over 400 entities (entropy $\approx \log 400 \approx 5.99$). In contrast, the ReLU model yields much more diffuse columns:

$$\mathbb{E}[H^{\text{ReLU}}(h)] \approx 4.89 \pm 0.51.$$

The histogram of column entropies (Figure 9) shows a tight concentration near zero for the bilinear model, and a broad concentration around ~ 5 for the ReLU model. Thus, although both models fit the training triples, they do so with very different internal operators: one that is almost a permutation, and one that behaves more like a noisy transition kernel. Training curves are shown in Figure 8.

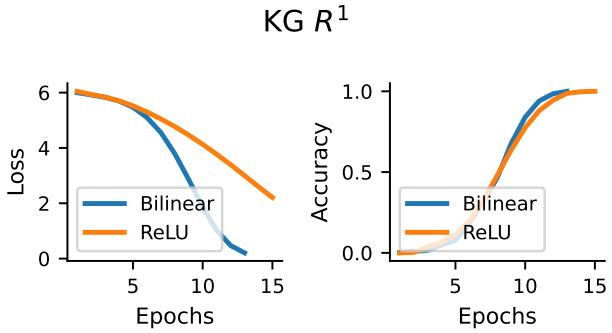


Figure 8: Training behaviour on the successor relation $R_1(h) = (h+1) \bmod N$ over a cycle with $N = 400$ entities. Both the bilinear and ReLU KG models quickly reach perfect accuracy, albeit with dissimilar loss curves.

Multi-step composition behaviour To probe compositional reasoning, we study powers of the learned operator T. For each $k \geq 1$, we form T^k by repeated matrix multiplication, and for each head h we compare the predicted tail

$$\hat{t}_k(h) = \arg\max_t T^k[t, h]$$

to the true k-step successor $t_k(h) = (h + k) \bmod N$. We then compute the k-step accuracy $\text{Acc}(k)$ across all heads.

For the bilinear model, $\text{Acc}(k)$ is exactly 1.0 for $k = 1, \ldots, 31$: the learned operator behaves as an exact successor for more than thirty compositions. Beyond this point, accuracy decays gradually: it remains above 0.95 until around $k \approx 41$, above 0.80 until around $k \approx 49$, and crosses 0.5 near $k \approx 54$. Only for larger k does accuracy approach the random-chance baseline of $1/N = 0.0025$. The full accuracy curve is plotted in Figure 10.

In contrast, the ReLU model's multi-step behaviour degrades almost immediately. While $\text{Acc}(1) = 1.0$ by construction, it drops to 0.895 at $k = 2$, 0.59 at $k = 3$, and 0.1075 at $k = 4$. By $k = 5$, accuracy is already close to

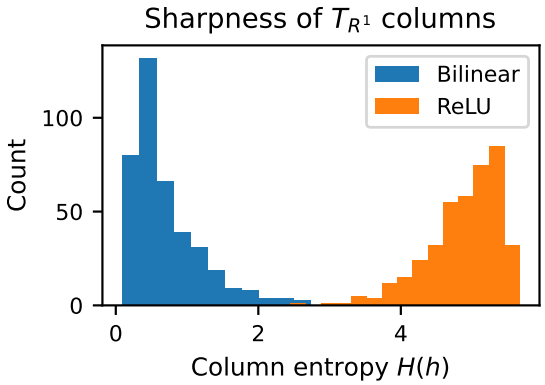


Figure 9: Distribution of column entropies $H(h)$ for the learned transition operator T on the cycle graph. The bilinear model produces columns that are nearly one-hot (low entropy), while the ReLU model yields high-entropy, diffuse columns.

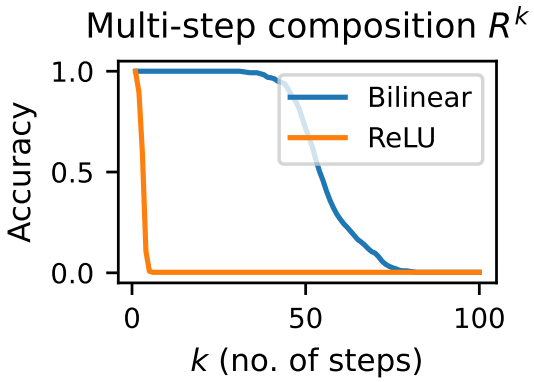


Figure 10: Multi-step composition behaviour of the learned transition operator T. The plot shows k-step accuracy $\mathrm{Acc}(k)$ of T^k compared to the true k-step successor on the cycle. The bilinear operator remains accurate for many compositions, while the ReLU operator collapses to near-random performance after only a few steps.

random (0.0075), and for $k \geq 7$ it remains essentially at the chance level ≈ 0.0025. This is consistent with the high column entropies: applying a diffuse kernel repeatedly rapidly washes out information about the starting head.

Taken together, these results show a sharp qualitative difference in how the two architectures represent the same relation. The bilinear model learns an operator that is not only correct on the training triples, but also composes cleanly for many steps in a way that closely matches the underlying group structure of the cycle. The ReLU model, by contrast, fits the one-step mapping but does so with a high-entropy operator whose compositions quickly collapse. This mirrors the modular arithmetic results and supports the view that bilinear MLPs provide a more interpretable and algebraically faithful internal representation of simple reasoning tasks.

Discussion

Our experiments asked a narrow question: when trained on simple algebraic and relational tasks, do bilinear MLPs learn internal operators that more closely match the underlying symbolic rules than those learned by ReLU MLPs? Within our synthetic settings, the answer appears to be "yes".

What the results suggest. Across modular arithmetic and relational composition, the bilinear models learn interaction matrices that align better with natural linear-algebraic structure. For modular addition, their Fourier spectra resemble the true circulant operator, both in entropy and in the diagonal pattern of the heatmaps, whereas the ReLU spectra are spiky and irregular. For multiplication, the bilinear interaction matrices have steeper singular-value decay and lower effective rank, indicating that each class is implemented by a simpler, lower-dimensional operator.

In the relational setting, the contrast is even clearer: the bilinear model learns an almost permutation-like transition matrix whose powers T^k track the k-step successor relation for many steps, while the ReLU model learns a high-entropy kernel whose compositions collapse to near-uniform after just a few iterations. Taken together, these findings support the view that bilinear MLPs favour sharp, low-dimensional operators that behave like explicit rules, in contrast to the more diffuse mechanisms learned by ReLU MLPs.

Relation to prior work. Our findings complement work on bilinear layers for weight-based interpretability (Pearce et al. 2024; Sharkey 2023). Those papers highlight that bilinear MLPs are easier to analyze because their nonlinearity factors into linear tensors. We additionally observe that, under standard training, they tend to learn operators that *themselves* are closer to the textbook algebraic objects. This fits within the broader mechanistic literature on algorithmic tasks (Nanda et al. 2023; Chughtai, Chan, and Nanda 2023; Lippl et al. 2024), where standard architectures are often found to rely on distributed or clustered codes; here we show that a simple architectural change can shift the learned mechanism towards more algebraic structure.

Limitations. Our study has several constraints. (i) All tasks are small, synthetic, and highly structured; conclusions do not automatically transfer to natural language or training LLMs. (ii) We only test a single bilinear parametrization and one-layer models; deeper or differently regularized architectures might behave differently. (iii) Our analysis focuses on a small set of summaries (Fourier entropy, singular values, column entropy, multi-step accuracy); other bases might reveal additional structure. (iv) We use standard training without sweeping over optimization hyperparameters, so our results describe typical but not guaranteed behaviour.

Future directions. Obvious extensions include richer algebraic tasks (e.g., non-abelian groups), more complex relational graphs, and bilinear or gated blocks inside transformers. One could also combine bilinear inductive biases with explicit regularizers on the learned operators, such as

penalties encouraging permutation-like or low-rank structure. More broadly, we see bilinear MLPs as a step toward architectures that are not only accurate but whose internal computations can be expressed in simple operator-level language, making mechanistic interpretability more tractable.

Related Work

Mechanistic interpretability and algorithmic tasks. Prior work has studied how neural networks solve algorithmic tasks where the ground-truth computation is exactly known. Nanda et al. (2023) investigate modular addition in small transformers, showing that the learned solution can be decomposed into structured Fourier-mode circuits. Chughtai, Chan, and Nanda (2023) analyze group composition problems and provide evidence that standard architectures can implement algebraically correct operations while relying on high-rank distributed representations. Lippl et al. (2024) study transitive inference, demonstrating that models sometimes fail to learn global relational structure and instead adopt "cooperative codes" that cluster items rather than representing relations explicitly.

Bilinear MLPs and structured representations. Recent work has proposed bilinear MLP layers as building blocks for interpretable architectures. Shazeer (2020) introduce gated and bilinear variations of MLPs in transformers, and Sharkey (2023) and Elhage et al. (2021) develop a mathematical framework in which such layers can be analyzed using tensor factorization. Most closely related to our study, Pearce et al. (2024) show that bilinear operators allow weight-based interpretability via direct eigendecomposition of learned interaction tensors. Our work complements theirs by examining how the inductive biases of bilinear layers affect the structure of solutions learned on small reasoning tasks.

Relational reasoning in neural networks. Relational generalization and compositional reasoning have been studied in the context of knowledge graph completion and neural-symbolic approaches (Nickel et al. 2016). Many architectures use multiplicative or bilinear interactions (Trouillon et al. 2016), but often with a focus on accuracy rather than mechanistic clarity. Our relational composition experiment follows the synthetic setting of Lippl et al. (2024), but emphasizes internal operator structure rather than test accuracy alone.

Overall, our work situates bilinear MLPs within the broader aim of designing architectures that not only solve tasks but do so through simpler, more algebraic internal mechanisms.

Conclusion

We compared bilinear and ReLU MLPs on modular arithmetic and relational composition tasks where the true underlying operators are precisely defined. While both architectures achieve high task accuracy, the internal structures they learn are qualitatively different. Across Fourier-domain analysis, singular-value spectra, effective rank, and multi-step relational composition behaviour, the bilinear models consistently learn interaction operators that are closer to the symbolic ground truth: they are sharper, lower-dimensional, and remain stable under composition.

These results do not imply that bilinear MLPs scale to complex domains or solve interpretability in general. However, they provide concrete evidence that architectural inductive bias can shape the mechanism by which neural networks represent reasoning-like computations. In simple settings, bilinear layers act as "white-box reasoners," enabling direct operator-level interpretation and avoiding the diffuse internal representations commonly observed in ReLU networks.

We view this work as a step toward designing architectures that are mechanistically understandable by construction. Future work could explore these ideas in richer algebraic domains, integrate bilinear inductive biases into transformer MLP blocks, or introduce regularization schemes that penalize deviations from interpretable operator structure. Ultimately, we hope these findings contribute to building models whose reasoning processes are not only correct but also transparent and expressible in symbolic terms.

References

Chughtai, B.; Chan, L.; and Nanda, N. 2023. A toy model of universality: Reverse engineering how networks learn group operations. In *International Conference on Machine Learning*, 6243–6267. PMLR.

Elhage, N.; Hume, T.; Olsson, C.; Schiefer, N.; Henighan, T.; Kravec, S.; Hatfield-Dodds, Z.; Lasenby, R.; Drain, D.; Chen, C.; Grosse, R.; McCandlish, S.; Kaplan, J.; Amodei, D.; Wattenberg, M.; and Olah, C. 2022. Toy Models of Superposition. arXiv:2209.10652.

Elhage, N.; Nanda, N.; Olsson, C.; Henighan, T.; Joseph, N.; Mann, B.; Askell, A.; Bai, Y.; Chen, A.; Conerly, T.; et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1): 12.

Lippl, S.; Kay, K.; Jensen, G.; Ferrera, V. P.; and Abbott, L. 2024. A mathematical theory of relational generalization in transitive inference. *Proceedings of the National Academy of Sciences*, 121(28): e2314511121.

Nair, V.; and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814.

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

Nickel, M.; Murphy, K.; Tresp, V.; and Gabrilovich, E. 2016. A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE*, 104(1): 11–33.

Olah, C.; Cammarata, N.; Schubert, L.; Goh, G.; Petrov, M.; and Carter, S. 2020. Zoom in: An introduction to circuits. *Distill*, 5(3): e00024–001.

Pearce, M. T.; Dooms, T.; Rigg, A.; Oramas, J. M.; and Sharkey, L. 2024. Bilinear MLPs enable weight-based mechanistic interpretability. *arXiv preprint arXiv:2410.08417*.

Sharkey, L. 2023. A technical note on bilinear layers for interpretability. *arXiv preprint arXiv:2305.03452*.

Shazeer, N. 2020. GLU Variants Improve Transformer. arXiv:2002.05202.

Trouillon, T.; Welbl, J.; Riedel, S.; Éric Gaussier; and Bouchard, G. 2016. Complex Embeddings for Simple Link Prediction. arXiv:1606.06357.