Compositionality Unlocks Deep Interpretable Models

Thomas Dooms[*1], **Ward Gauderis**[*2], **Geraint A. Wiggins**[2], **José Oramas**[1]

[1]Universiteit Antwerpen, Belgium
[2]Vrije Universiteit Brussel, Belgium
thomas.dooms@uantwerpen.be, ward.gauderis@vub.be

Abstract

We propose $\mathcal{X}$-net, an intrinsically interpretable architecture combining the compositional multilinear structure of tensor networks with the expressivity and efficiency of deep neural networks. $\mathcal{X}$-nets retain equal accuracy compared to their baseline counterparts. Our novel, efficient diagonalisation algorithm, ODT, reveals linear low-rank structure in a multilayer SVHN model. We leverage this toward formal weight-based interpretability and model compression.

1 Introduction

This paper presents preliminary results on developing interpretable neural networks while retaining competitive accuracy. We introduce the $\mathcal{X}$-net architecture and the ODT algorithm, which diagonalises and truncates such networks. We discuss the background and rationale behind these networks and provide evidence that these networks are easily interpretable based on their weights. The contributions of this work are as follows:

- We propose $\mathcal{X}$-net, an intrinsically interpretable architecture with competitive accuracy.
- We propose the ODT algorithm to diagonalise the $\mathcal{X}$-net and truncate it up to arbitrary precision.
- We extract meaningful learned patterns from the resulting highly structured model.

Mechanistic interpretability provides a microscope into a neural network's inner mechanisms, without which we cannot meaningfully verify their safety, reliability, or alignment with human values. The field aims to reverse-engineer deep neural networks into a set of understandable algorithms or components. Just as one decompiles a computer program into variables and functions, researchers aim to reduce deep networks into a collection of semantically meaningful circuits (Olah et al. 2020). Such circuits are broadly defined as small or sparse 'computation' graphs (Olah et al. 2020; Loconte et al. 2024). However, due to the inherent non-decomposability of many modules in neural networks, this task is often rooted in post-hoc approximations (Bricken et al. 2023). Detrimentally, since interpretability is an ill-defined metric (Leavitt and Morcos 2020), much research cherry-picks examples and draws incomplete conclusions, following a practice referred to as *streetlight interpretability* (Casper 2024). Interpretability should focus on finding mechanisms important to the model, not humans. Our results indicate that rooting interpretability in rigorous decompositions is a promising avenue to achieving this.

Compositional AI emphasises understanding an artificial system's behaviour by examining how its constituent processes interact and compose, rather than studying components in isolation (Coecke and Kissinger 2018; Coecke 2021). It uses the principle of *compositionality* from categorical quantum mechanics – studying systems through their components and interactions – to provide a common mathematical foundation for symbolic and subsymbolic AI, and to break the curse of dimensionality in learning and understanding complex models. The field of *Applied category theory in AI* seeks to provide an overarching framework that unifies theory and practice (Shiebler, Gavranović, and Wilson 2021; Dudzik et al. 2022) through its ability to structure, uncover and facilitate cross-disciplinary insights. By formalising the compositional structure, neuro-symbolic methods can integrate symbolic domain knowledge to improve data efficiency and interpretability of subsymbolic models. (Lorenz et al. 2023). Categorical deep learning uses compositional domain structure to derive inductive biases in deep learning (Gavranović et al. 2024; Dudzik et al. 2022; Bronstein et al. 2021). Categorical string diagrams (Coecke and Kissinger 2018) formalise *compositional models* in symmetric monoidal categories and recognise 'intrinsically interpretable' models as *compositionally-interpretable models* (Tull et al. 2024).

Rationale String diagrams interpreted in the category of real finite-dimensional Hilbert spaces create tensor network diagrams where boxes represent linear maps, and wires represent vector spaces (Coecke and Kissinger 2018). Tensor networks offer rich compositional structures that enable principled network design (Levine et al. 2018) and can serve as representations bridging symbolic and subsymbolic reasoning methods (Gauderis 2023). Tensor networks capture structural relations in high-dimensional Hilbert spaces, but are limited to linear relations. Neural networks address

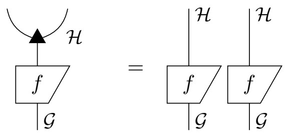
[*]These authors contributed equally.

this with non-linear activation functions, yet the composition of non-linearities remains poorly understood. This work bridges the representational power of non-linear neural networks and the compositional structure of linear tensor networks, which has been shown capable of retaining near-SoTA accuracy while providing appealing mathematical properties (Pearce et al. (2024); Cheng et al. (2024); inter alia).

2 $\mathcal{X}$-net Architecture

We propose the deep $\mathcal{X}$-net (CHI-net) architecture, short for *Compositionally and Hierarchically Interpretable network*. Through appropriate coordination of tensor composition, principled weight-tying patterns, and a non-linear cloning operator, $\mathcal{X}$-nets can be efficiently implemented, trained, and evaluated as a deep non-linear neural architecture, but analysed, recomposed and interpreted as a linear tree tensor network, combining the best of both worlds.

Using a non-linear cloning operator, multi-variable polynomials are learned as multilinear maps that can be reinterpreted as linear maps over a tensor product space. Inspired by the linear no-cloning theorem (Coecke and Kissinger 2018), we introduce the non-linear cloning operator, a purely diagrammatic manipulation:



(1)

This unfolds into a multilinear map to which multiple copies of the input are fed. Cloning has already been leveraged implicitly in network design and interpretability (Pearce et al. 2024; Cheng et al. 2024; Dubey, Radenovic, and Mahajan 2022). Through (de)composition, tensor networks avoid exponential parameter growth when learning higher-order multilinear maps (Chrysos et al. 2020; Grelier, Nouy, and Chevreuil 2018). Since any multi-linear map can be represented as a linear map over a tensor product space, the network can be viewed as a linear map with weight-tying induced by the cloning operator over a high-dimensional kernel vector space.

The architecture of a $\mathcal{X}$-net $\mathcal{X} : \mathcal{I} \to \mathcal{O}$ from input space $\mathcal{I}$ to output space $\mathcal{O}$ is shown in Figure 1. The input space $\mathcal{I}$ is embedded into a *hidden bond space* $\mathcal{H}_1$ by a linear embedding map $e : \mathcal{I} \to \mathcal{H}_1$. This feeds into L layers of the cloning operator followed by a third-order tensor *core* $f_i : \mathcal{H}_i \to \mathcal{H}_{i+1}$ for $i = 1, 2, \dots, L$. The output space $\mathcal{O}$ is obtained by unembedding $\mathcal{H}_{L+1}$ using a linear unembedding map $u : \mathcal{H}_{L+1} \to \mathcal{O}$. For notational simplicity, the maps e and u are also referred to as f_0 and f_{L+1}, respectively. When the output $\mathcal{O}$ is a probability distribution, the unembedding map u can be fed into a softmax function, omitted here for simplicity.

Intuitively, a $\mathcal{X}$-net replaces the standard hidden layer consisting of a linear map and a non-linear activation function by a cloning operator and a multilinear map. The network's forward pass is efficiently computed bottom-up, starting from the input $x \in \mathcal{I}$ and evaluating the cores f_i, in sequence, alternating with the cloning operator, as shown in

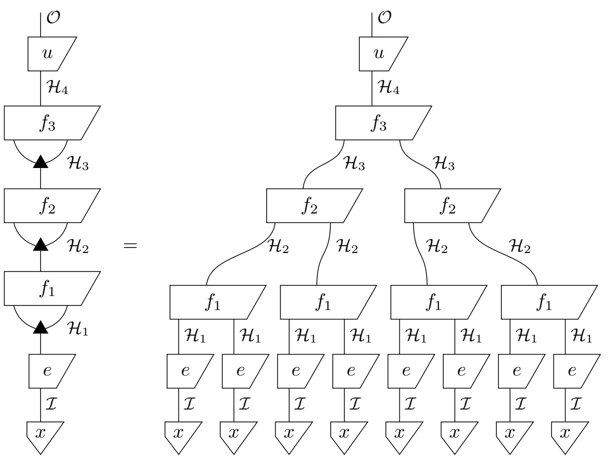
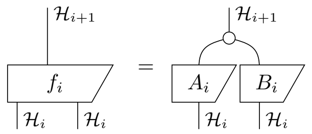


Figure 1: String diagram of a 3-layer $\mathcal{X}$-net with input $x \in \mathcal{I}$, linear embedding and unembedding maps $e : \mathcal{I} \to \mathcal{H}_1$ and $u : \mathcal{H}_4 \to \mathcal{O}$, and multilinear cores $f_i : \mathcal{H}_i \to \mathcal{H}_{i+1}$ for $i = 1, 2, 3$. On the left side, the network is interpreted bottom-up, reflecting the efficient forward pass evaluation. The right side contains the unfolded tree tensor network, where contraction is no longer restricted to unidirectional evaluation. The expansion introduces weight-tying patterns per layer.

Figure 1. By unrolling the cloning operators using Equation 1, the network can be interpreted as a tree tensor network featuring weight-tying patterns per layer.

Every core f_i is a bilinear map of its inputs $\mathcal{H}_i \otimes \mathcal{H}_i$. Combined with the non-linear immediately preceding cloning operator, f_i captures quadratic interactions between inputs. A $\mathcal{X}$-net of depth L can only express multilinear polynomials over its inputs with monomials of degree 2^L. This restriction to high-degree interactions is detrimental to generalisation, as real-world problems often depend on lower-degree terms (Lin, Tegmark, and Rolnick 2017; Li, Zhu, and Cambria 2021). Adding a constant input as a bias term overcomes this limitation: $x \leftarrow (1, x)$. With this modification, an L-layer $\mathcal{X}$-net can express multilinear polynomials of degree up to 2^L. We impose further structure on the cores to restrict the number of learnable parameters. Specifically, each core f_i is parametrised using a transposed Khatri-Rao product of two matrices A_i and B_i:



(2)

Here, the white circle represents the third-order tensor identified by a Kronecker delta δ^l_{jk} over input indices j and k and output index l and coincides with an element-wise multiplication of the input vectors (Coecke and Kissinger 2018). This parameterization, known as a *bilinear layer* (Shazeer 2020; Sharkey 2023), is increasingly popular in neural network design but is less expressive than a full bilinear map. Due to the preceding cloning operator, the bilinear layer's parameterisation can be symmetrised, with respect to its inputs, post-training without affecting its expressivity:

$$(f_i)^l_{jk} \leftarrow \frac{1}{2}\left((f_i)^l_{jk} + (f_i)^l_{kj}\right), \quad \forall i, j, k$$

Throughout, we assume this symmetry is enforced.

3 ODT Algorithm

We present the novel *Orthogonalisation, Diagonalisation, and Truncation* (ODT) algorithm, inspired by the hierarchical singular value decomposition (HSVD) for tree-structured tensor networks (Oseledets and Tyrtyshnikov 2009; Grasedyck 2010). The ODT algorithm efficiently compresses trained $\mathcal{X}$-nets and extracts their low-rank interpretable features. The overall time complexity of the algorithm is $\mathcal{O}(L \cdot h^4)$, with hidden dimension h and depth L, discussed in Appendix F. Each step is discussed individually, and a compression error bound is given. Accompanying diagrammatic proofs and further explanations are presented in Appendix G.

Orthogonalisation The objective is to make the cores f_i orthonormal for $i = 0, \ldots, L$ without changing the network behaviour. This is done bottom-up, starting from the input embedding $e = f_0$ and proceeding to the final core f_L, by applying the RQ decomposition to each matricised core f_i, where f_i is replaced by the unitary Q_i and contracting the non-orthogonal part R_i into the next core f_{i+1}. To ensure that the bond dimensions do not grow during this process, reduced RQ decomposition is used so that the adjoints of the new cores $f_i^\perp = Q_i$ are isometric with orthonormal columns. The result is a network in which all cores f_i are isometries, except for the unembedding $u = f_{L+1}$, which contains the non-orthogonal part of the network.

Diagonalisation This step computes an orthogonal projection matrix P_i for each bond H_i that projects onto the left singular vectors of the $\mathcal{X}$-net, matricised with respect to a single H_i bond. These projectors are computed by diagonalising the Gram matrix G_i of the $\mathcal{X}$-net for that bond. This calculation, illustrated in Figure 11, consists of three steps. First, the $\mathcal{X}$-net is composed with its adjoint and the tensor contraction at a bond H_i is removed. This results in a self-adjoint Gram matrix $G_i : H_i \to H_i$. Second, its spectral decomposition $G_i = V_i \Lambda_i V_i^*$ is computed. V_i is the unitary that diagonalises G_i and contains the left-singular vectors of the matricised $\mathcal{X}$-net, and Λ_i is a diagonal matrix of the corresponding real and positive singular values, sorted in decreasing order of magnitude. Finally, the self-adjoint, idempotent and therefore orthogonal projector P_i on the left singular vectors are constructed as:

$$P_i = V_i V_i^* \tag{3}$$

Inserting P_i into bond H_i of the $\mathcal{X}$-net does not affect its behaviour (Coecke and Kissinger 2018, Prop. 5.77).

Truncation Given the singular vectors and values of each bond H_i, the last step is to reduce the rank r_i of each projection P_i to project only onto the r_i most important singular vectors. This is achieved by truncating the unitary V_i to

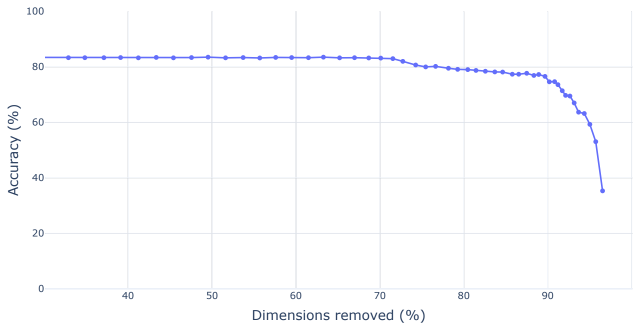


Figure 2: The accuracy curve shows that it is possible to truncate about 70% of the model's dimensions without compromising accuracy. About 90% of dimensions can be truncated with a small drop in accuracy.

the first r_i columns (preserving isometric properties) so that $\dim(\mathcal{H}_i') = r_i$ in Equation 3, while the truncated projector is the low-rank approximation that minimises the Frobenius distance to the original projector. Finally, the truncated projectors P_i are partially contracted in both $f_{i-1}^\perp$ and $f_i^\perp$. In this way, the new bond dimensions of $\mathcal{H}_i'$ are reduced to r_i in the truncated $\mathcal{X}'$-net. Note that the truncated $\mathcal{X}'$-net is a low-rank approximation of the original $\mathcal{X}$-net that retains the symmetric isometric properties imposed by the symmetrisation and orthogonalisation steps.

We establish the compression error bounds by following the proof for HSVD (Grasedyck 2010; Grelier, Nouy, and Chevreuil 2018). Let $\epsilon > 0$ be any threshold. If each G_i' is the truncated G_i with rank r_i such that:

$$\|G_i\|_F^2 - \|G_i'\|_F^2 \leq \frac{\epsilon^2}{2^{(L+1)} - 1}\|G_i\|_F^2$$

where the denominator is the number of projections in the unfolded tensor network, then the truncated $\mathcal{X}'$-net is a low-rank approximation of the original $\mathcal{X}$-net with relative precision ϵ:

$$\|\mathcal{X} - \mathcal{X}'\|_F \leq \epsilon \|\mathcal{X}\|_F$$

The prospect of bounding the loss is left for future work.

4 Experiments

We consider a 3-layer model trained on the SVHN dataset (a real-world variant of MNIST). This model achieves about 85% test accuracy. Details about the setup can be found in Appendix A. The diagonalised cores are sparse, and the truncation discards about 70% of bond dimensions with no decrease in accuracy (Figure 2). This section discusses leveraging this structure to extract learned patterns from the model.

Weight interpretation The $\mathcal{X}$-net consists of an embedding matrix ($e = f_0$), whose rows are called *atoms*, and core tensors (f_1-f_3), whose symmetric input slices are called *interaction matrices*. We call a composition of atoms a *feature* and define its *activation* as its composition with an input. An atom's interpretation depends on its subsequent interaction matrix; we distinguish between three kinds. Diagonal entries show an input's self-multiplication, with the weight

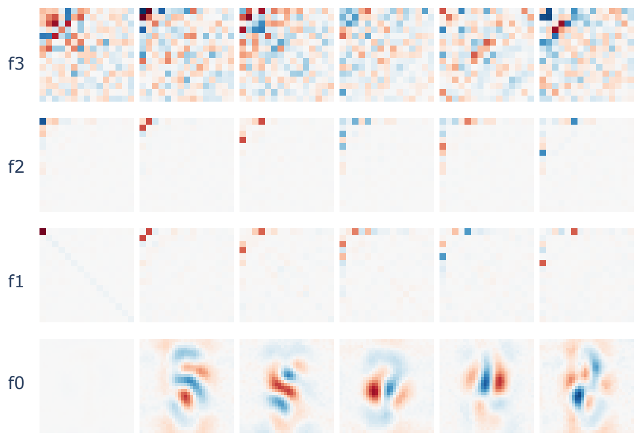


Figure 3: The six most important atoms (e=f_0) and interaction matrices (f_1-f_3) for each layer of the decomposed model. The unembedding (u=f_4) is contracted into f_3 for brevity. The atoms are reshaped into the image's dimensions for visual clarity; they contain patterns such as edge detectors and proto-digits. The middle interaction matrices are highly sparse and are dominated by constant interactions. The root (f_3) is denser, combining many learned features.

determining positive or negative value. Off-diagonal entries act as gates by multiplying different inputs together. Entries on the first row/column are scalar values since they multiply with a constant (the bias). All figures show positive weights in blue and negative weights in red.

Tensor anatomy Our diagonalised model's atoms are interpretable, and its interaction matrices (except the root) are sparse, as shown in Figure 3. Interestingly, the constant dimension stays separated despite not being hard-coded into the diagonalisation; it does not match with any input except the appended bias (not shown in the figure). The other atoms have coherent spatial structures; some resemble proto-digits, while others are localised edge detectors. Constant interactions dominate the interaction matrices of f_1 and f_2, comprising about 90% of their norm. This effectively corresponds to sparsely summing certain atoms; note that this is a fully linear operation. Local decompositions miss this structure, as discussed in Appendix E. The root (f_3) is denser, capturing more intricate interactions. Its effective input dimension is about 13, indicating that it is still structured.

Digit extraction We extract interpretable features from this model corresponding to the output classes. Our approach is inspired by Pearce et al. (2024), who consider the highest eigenvector for a given output class and show them to be interpretable in single-layer models. Our models span multiple layers, so we cannot simply reuse this approach. However, given that linear interactions dominate the first few layers, it is possible to directly use this to project the vectors onto the input space. This approach yields highly-interpretable digits (Figure 4). The other eigenvectors of digits are covered in Appendix D. In short, using the near-linearity found by the decomposition provides an excellent lens through which to understand latent features.

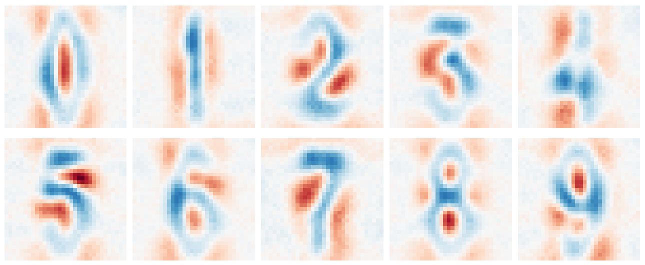


Figure 4: The most important eigenvector of the root core ($u \circ f_3$) per digit. These are linearly traced through the previous cores onto the input space. These represent the prototypical digits from the training data.

5 Conclusion

Summary This paper introduces χ-nets, an interpretable neural network architecture that maintains competitive accuracy. We present the ODT algorithm, which efficiently diagonalises χ-nets, revealing interpretable low-rank linear structures in multilayer models and allowing model compression by ranking hidden dimensions. Through tensor composition, weight-tying, and a non-linear cloning operator, χ-nets can be implemented as deep neural networks but analysed as tree tensor networks, combining the advantages of both.

Ongoing work Despite the scalability of the proposed architecture and algorithm, this paper focuses on small models. While the ODT algorithm identifies shallow neural circuits, deeper models may involve complex multilayer interactions, which will be harder to interpret comprehensively. Ongoing work includes extending this approach to modern architectures, exploring probabilistic generative modelling (Loconte et al. 2024), and involving the dataset in the decomposition to capture input statistics or integrate metadata.

Appendix

A Experimental setup

Dataset To train our model, we use the SVHN dataset, which is essentially a harder version of MNIST. The motivation is that this provides a reasonable middle ground between MNIST, which is solvable using linear models, and ImageNet, which requires huge models. Furthermore, SVHN has easily discernible classes, making it a perfect testbed for our purposes. We concatenate the ”train” and ”extra” splits to train and use the ”test” split to measure accuracy. Our input processing step grayscales the images and adds Gaussian noise; we use no other regularization.

Architecture The discussed model consists of 3 hidden (bilinear) layers surrounded by an embedding and unembedding (head). These hidden layers all use biases of the form. The embedding projects the input onto a 256-dimensional latent space, and the unembedding projects this onto 10 the output classes. This is followed by a softmax operation to acquire the prediction probabilities. We use a variant of BatchNorm as normalisation that only divides by the L_2 norm

	1 layer	**2 layers**	**3 layers**	**4 layers**
χ-net	83.8%	84.2%	85.4%	86.5%
Baseline	85.7%	86.4%	87.3%	88.0%

Table 2: Accuracy of χ-nets and a ReLU baseline across depths. A linear model only attains 23%.

(akin to RMSNorm). After training, this single average is contracted into its neighbouring matrix.

Hyperparameters	
input noise norm	0.3
weight decay	1.0
learning rate	0.001
normalisation	RmsBatchNorm
batch size	2048
optimiser	AdamW
schedule	cosine
epochs	20

Table 1: Training setup for the studied model(s).

B χ-net ablations

This work aims to establish deep χ-nets as a viable and interpretable alternative to ordinary multi-layer perceptrons. Previous works (Pearce et al. 2024; Shazeer 2020; Cheng et al. 2024) provide evidence that multilinear networks can retain competitive accuracy when scaling both dataset complexity and model size. Specifically, Pearce et al. (2024) finds transformers that use bilinear layers to be 6% less sample efficient. In a data-abundant regime, training for slightly longer can offset this. However, in a data-constrained regime, training for more epochs may not close that gap. Empirically, we find that ReLU-based networks benefit from training for more epochs while χ-nets sometimes do not.

We perform a set of ablations on our small models to show that χ-nets achieve near-accuracy parity with their baseline counterparts (Table 2). These experiments follow the setup from Appendix A, only varying the depth.

C Explaining predictions

We employ our extracted features to evaluate their usefulness in explaining a given classification output. As building blocks in our explanation, we use the eigenvectors for all digits, of which the highest ones are shown in Figure 4 and Figure 6. We then measure the activation of each feature, which is computed as their inner product with the input. These activations are then multiplied by the eigenvalues for that feature.

In Figure 5, we consider a particular 3 almost resembling a 9 (possibly due to the grayscaling). This shows a feature for the digit 9 has the highest positive impact. However, a long tail of negative features inhibits the digit 9, resulting in a slightly higher logit for the digit 3.

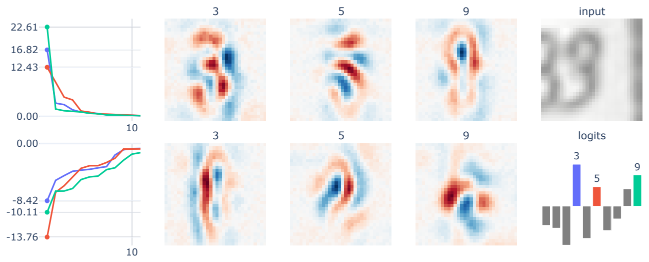


Figure 5: Using extracted features from the model to explain the classification logits. The left shows the importance scores for all features (split by positive and negative contribution), the most important ones of which are shown in the middle section. The right shows the logits along with the evaluated input.

	Bond 0	**Bond 1**	**Bond 2**	**Bond 3**
SVD	83.4	165.9	100.5	229.2
ODT	2.1	3.3	13.3	5.7

Table 3: Comparison of the effective (L_2) bond dimensions for SVD and ODT.

D Additional features

Each digit has multiple eigenvectors; this section discusses the top eigenvectors and the eigenvalues for the digits 2, 5 and 8 (shown in Figure 6). The first positive eigenvector often indicates a prototypical instance of the given digit, the second the most salient stroke, and the third depends on the digit. Strokes are often detected by Gabor-like filters, matching across a range of possible locations.

The negative eigenvectors are slightly harder to interpret but correspond to a stroke that would strongly change the given digit to another. For instance, the negative eigenvectors of a 5 correspond to a stroke closing the top part; if that were present, the feature would likely become a 9.

E ODT in numbers

While the ODT algorithm has a solid theoretical foundation, this section discusses how that translates to practice. We compare ODT to SVD and perform numerous ablations concerning the truncation procedure.

Rank We compare the ODT algorithm versus the commonly used SVD to measure the bond dimensions of our model (Figure 7). SVD has a very long tail, indicating none of the core tensors are low-rank by themselves. In contrast, the singular values from ODT immediately drop to zero; it is only by diagonalising that their structured nature becomes apparent. The effective dimensions are shown in Table 3.

An effective bond dimension smaller than 2 indicates the model is fully linear (only the constant dimension is used). This is almost the case in the first two layers. Computing the effective dimension without the constant results in about 14 for all layers. We believe this to reflect the dataset's intrinsic dimensionality.

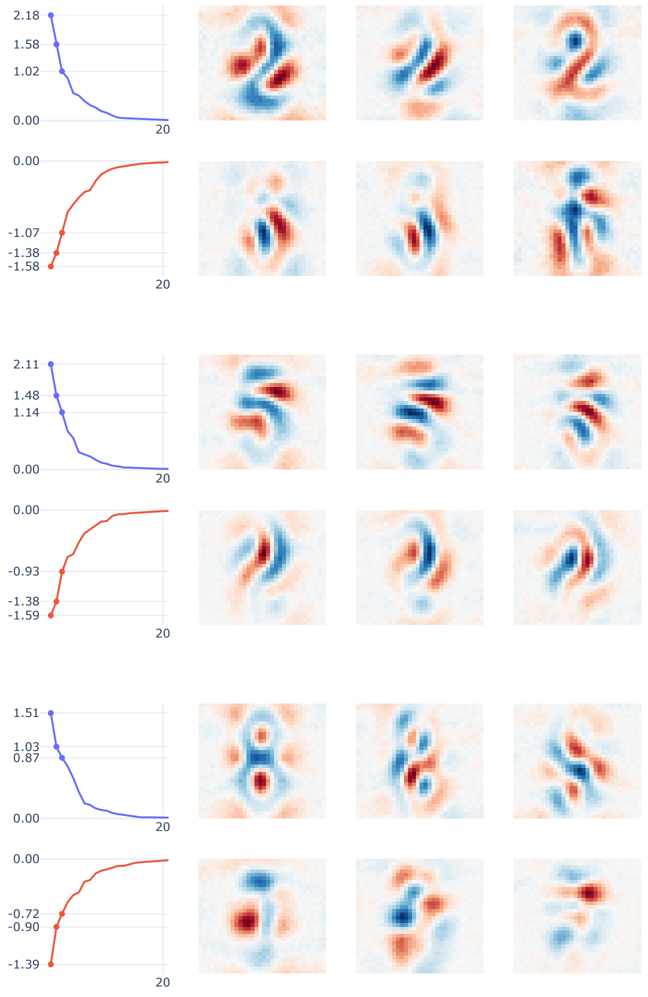


Figure 6: Most important projected eigenvectors for the digits 2 (top), 5 (middle) and 8 (bottom). The left graphs indicate the magnitude of the eigenvalues, split by positive and negative. The right side depicts the eigenvectors corresponding to the indicated eigenvalues.

Loss In addition to Figure 2, we provide the resulting loss after any truncation (Figure 8). This paints a similar picture as the accuracy comparison, but there is a dip in the loss after truncating about 90% of dimensions. We are currently uncertain about its cause. One hypothesis is that the network has a generalizing backbone and a long tail of spurious correlations. Removing a part of these correlations may destabilise the model, but removing all of them is beneficial. Once parts of the backbone are removed, the loss spikes.

Norm We measure which dimensions contribute to the Frobenius norm of the whole tensor (Figure 9). This reveals only a handful of dimensions contribute.

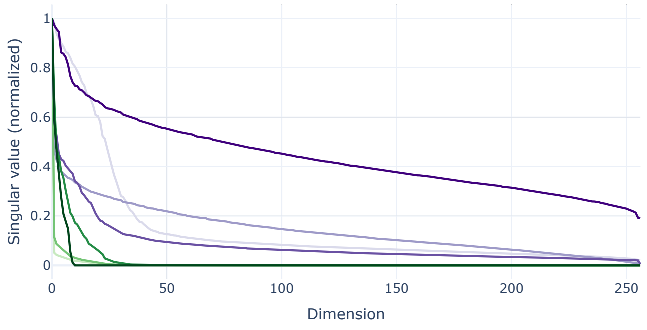


Figure 7: The normalized singular values of the bond dimension (increasing opacity with depth) for SVD (purple) and ODT (green). SVD does not find a clean low-rank structure, while ODT does.

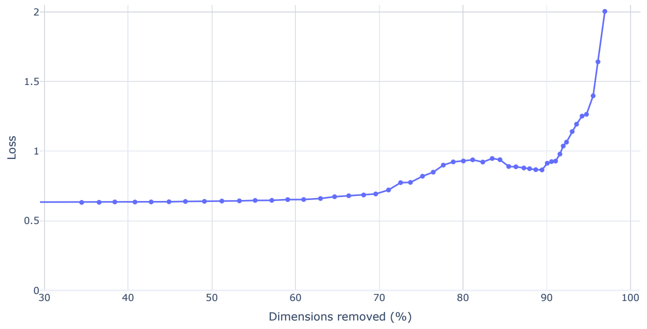


Figure 8: The loss in function of removed dimensions; large parts can be removed without affecting loss.

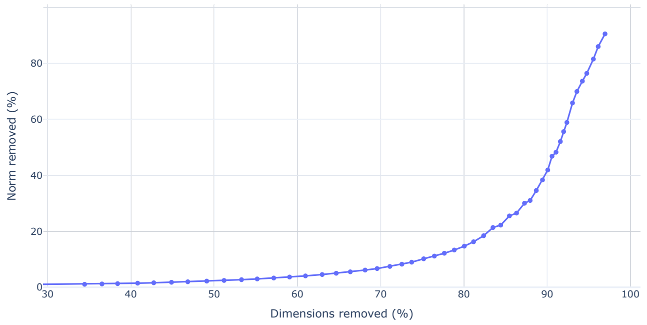


Figure 9: The tensor (Frobenius) norm as a function of removed dimensions. The norm is concentrated in only a fraction of the network.

F ODT in complexity

The overall time complexity of the ODT algorithm is $\mathcal{O}(L \cdot h^4)$, which is linear in the depth L of the network and quartic in the hidden dimension h. This section explains the complexity of the algorithm's steps per layer.

Orthogonalisation The complexity of a reduced RQ decomposition (for a matrix with dimensions n and m) is $\mathcal{O}(\max(n,m) \cdot \min(n,m)^2)$. Each matricised core has dimensions h and h^2, resulting in a complexity of $\mathcal{O}(h^4)$. Contracting the R_i matrix into the next core f_{i+1} (a third-order tensor) likewise has a complexity of $\mathcal{O}(h^4)$.

Diagonalisation Due to the imposed symmetry of the cores $f_i^{\perp}$, the projector P_i only needs to be calculated once per layer. Furthermore, one can avoid computing repeated tensor contractions by computing the Gram matrices G_i iteratively from top to bottom using the isometric properties of the cores $f_i^{\perp}$. Consequently, each Gram matrix can be computed in $\mathcal{O}(h^4)$ time. The spectral decomposition of a Gram matrix can be computed in $\mathcal{O}(h^3)$ time, and contracting V_i into their neighbouring cores takes $\mathcal{O}(h^4)$.

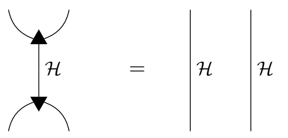
Truncation If the eigenvalues Λ_i from the previous step are stored, computing the truncation can be done in $\mathcal{O}(h)$ since it only involves removing dimensions. Contracting each projection into the adjacent cores can be done $\mathcal{O}(h^4)$.

In summary, the ODT algorithm has a bottom-up step (orthogonalisation) with complexity $\mathcal{O}(L \cdot h^4)$, a top-down step (diagonalisation) and a truncation step with the same complexity and a truncation step with negligible similar complexity. Despite the algorithm's quartic complexity, runtime rarely constrains performance. The $\mathcal{O}(h^3)$ memory requirement becomes the primary bottleneck for wide models.

G ODT in pictures and code

Each of the three steps in the ODT algorithm discussed in Section 3 is illustrated with string diagrammatic equations and pseudocode.

Orthogonalisation In Figure 10 and Algorithm 1, cores f_i (for $i = 0, \ldots, L$) are converted into isometries to achieve the isometric property shown in Figure 12, using the elimination rule for the adjoint of the cloning operator:



(4)

Diagonalisation The diagonalising projection matrices $P_i = V_i V_i^*$ for each bond $\mathcal{H}_i$ are constructed based on the eigenvectors of the gram matrix G_i, efficiently calculated as shown in Figure 11 and Algorithm 2.

Truncation The truncated projections P_i are contracted into the adjacent cores $f_{i-1}^{\perp}$ and $f_i^{\perp}$ in Figure 13. When all P_i are contracted in parallel, a new $\mathcal{X}'$- net is formed as shown in Figure 14 and Algorithm 3.

Algorithm 1: Orthogonalisation

```
Require: Cores $f_0, \ldots, f_{L+1}$
 1: $f_0^{\not\perp} \leftarrow f_0$
 2: for $i = 0$ to $L$ do
 3:     $R_i, Q_i \leftarrow \mathrm{RQ}(f_i^{\not\perp})$
 4:     if $i = L$ then
 5:         $f_{L+1}^{\not\perp} \leftarrow f_{L+1} \circ R_i$ {Unembedding}
 6:     else
 7:         $f_{i+1}^{\not\perp} \leftarrow f_{i+1} \circ (R_i \otimes R_i)$
 8:     end if
 9:     $f_i^{\perp} \leftarrow Q_i$
10: end for
11: return $f_0^{\perp}, \ldots, f_L^{\perp}, f_{L+1}^{\not\perp}$
```

Algorithm 2: Diagonalisation

```
Require: Isometric cores $f_0^{\perp}, \ldots, f_L^{\perp}$
Require: Unembedding $f_{L+1}^{\not\perp}$
 1: for $i = L+1$ to $1$ do
 2:     if $i = L+1$ then
 3:         $G_{L+1} \leftarrow f_{L+1}^{\not\perp *} \circ f_{L+1}^{\not\perp}$ {Unembedding}
 4:     else
 5:         $G_i \leftarrow f_i^{\perp *} \circ G_{i+1} \circ f_i^{\perp}$
 6:     end if
 7:     $V_i, \Lambda_i \leftarrow \mathrm{EVD}(G_i)$
 8: end for
 9: return $V_1, \ldots, V_{L+1}, \Lambda_1, \ldots, \Lambda_{L+1}$
```

Algorithm 3: Truncation

```
Require: Isometric cores $f_0^{\perp}, \ldots, f_L^{\perp}$
Require: Unembedding $f_{L+1}^{\not\perp}$
Require: Isometries $V_1, \ldots, V_{L+1}$ of $r_1, \ldots, r_{L+1}$ most
    important singular vectors
 1: $f_0' \leftarrow f_0^{\perp}$
 2: for $i = 1$ to $L+1$ do
 3:     $f_{i-1}' \leftarrow V_i^* \circ f_{i-1}'$
 4:     if $i = L+1$ then
 5:         $f_{L+1}' \leftarrow f_{L+1}^{\not\perp} \circ V_{L+1}$ {Unembedding}
 6:     else
 7:         $f_i' \leftarrow f_i^{\perp} \circ (V_i \otimes V_i)$
 8:     end if
 9: end for
10: return $f_0', \ldots, f_{L+1}'$
```

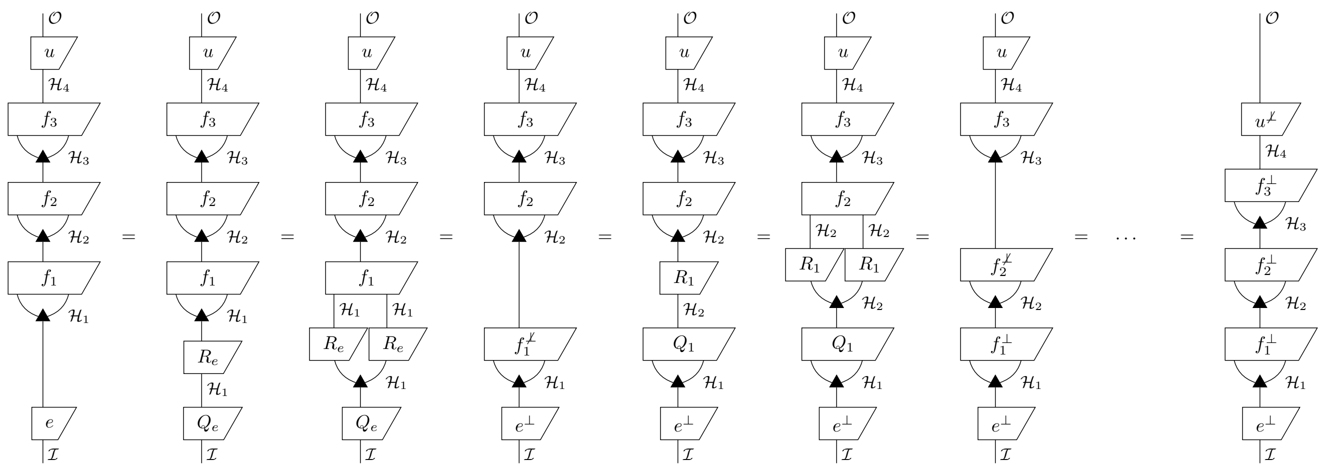


Figure 10: Orthogonalisation of a 3-layer χ-net by bottom-up application of the reduced RQ decomposition.

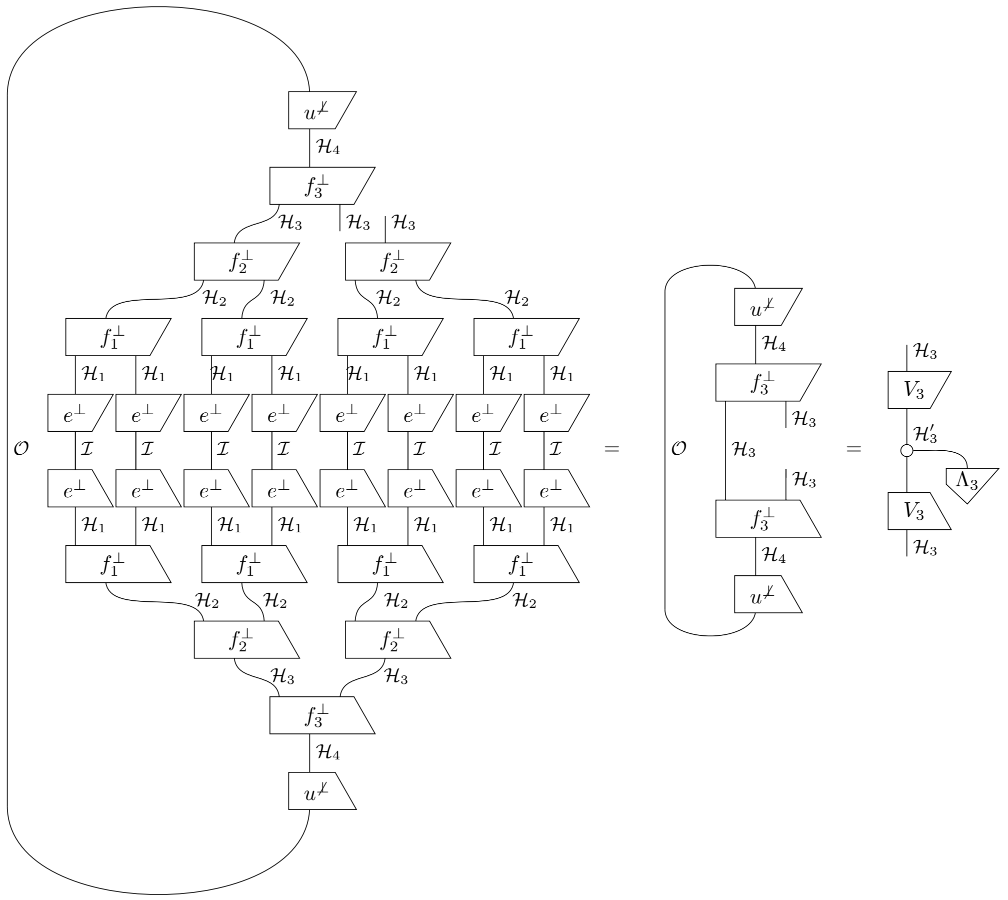


Figure 11: Example diagonalisation of the Gram matrix G_3. Step 1 is a consequence of the isometric property of the cores, as shown in Figure 12. Step 2 is the spectral decomposition of the self-adjoint Gram matrix.

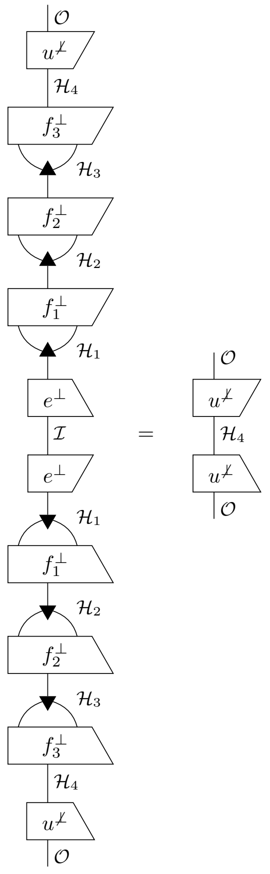


Figure 12: Isometric property of the orthogonalised χ-net. In the diagrammatic language, vertical reflection corresponds to taking the adjoint. The equality can be proven by contracting the isometries and the cloning operator using Equation 4.

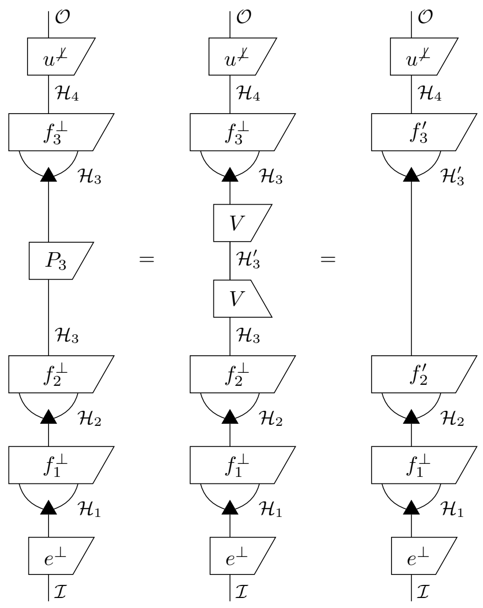


Figure 13: Example contraction of the truncated projector P_3 into the cores $f_2^{\perp}$ and $f_3^{\perp}$, resulting in a reduced bond dimension of $\mathcal{H}_3'$.

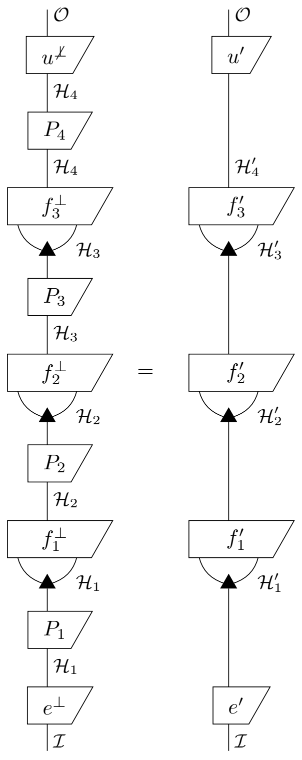


Figure 14: Example of a 3-layer χ-net after diagonalisation by the projectors P_i, shown on the left. If these P_i are truncated to a lower rank, the resulting χ'-net is a low-rank approximation of the original χ-net, as shown on the right.

Acknowledgements

This research was funded by the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme (TD, GAW, JO) and by the Department of Computer Science of the Vrije Universiteit Brussel (WG).

Contributions

TD and WG contributed equally to the research and coauthored the manuscript. GW and JO guided both the research and writing process.

References

Bricken, T.; Templeton, A.; Batson, J.; Chen, B.; Jermyn, A.; Conerly, T.; Turner, N.; Anil, C.; Denison, C.; Askell, A.; Lasenby, R.; Wu, Y.; Kravec, S.; Schiefer, N.; Maxwell, T.; Joseph, N.; Hatfield-Dodds, Z.; Tamkin, A.; Nguyen, K.; McLean, B.; Burke, J. E.; Hume, T.; Carter, S.; Henighan, T.; and Olah, C. 2023. Towards Monosemanticity: Decomposing Language Models With Dictionary Learning. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Bronstein, M. M.; Bruna, J.; Cohen, T.; and Veličković, P. 2021. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. arXiv:2104.13478.

Casper, S. 2024. EIS XIV: Is mechanistic interpretability about to be practically useful?

Cheng, Y.; Chrysos, G.; Georgopoulos, M.; and Cevher, V. 2024. Multilinear Operator Networks. In *The Twelfth International Conference on Learning Representations*.

Chrysos, G.; Moschoglou, S.; Bouritsas, G.; Deng, J.; Panagakis, Y.; and Zafeiriou, S. 2020. Deep Polynomial Neural Networks. https://arxiv.org/abs/2006.13026v2.

Coecke, B. 2021. Compositionality as We See It, Everywhere around Us. arXiv:2110.05327.

Coecke, B.; and Kissinger, A. 2018. Picturing Quantum Processes. In Chapman, P.; Stapleton, G.; Moktefi, A.; Perez-Kriz, S.; and Bellucci, F., eds., *Diagrammatic Representation and Inference*, Lecture Notes in Computer Science, 28–31. Cham: Springer International Publishing. ISBN 978-3-319-91376-6.

Dubey, A.; Radenovic, F.; and Mahajan, D. 2022. Scalable Interpretability via Polynomials. https://arxiv.org/abs/2205.14108v4.

Dudzik, A.; Gavranović, B.; João Guilherme Araújo; Petar Veličković; and Pim de Haan. 2022. Categories for AI.

Gauderis, W. 2023. *Quantum Theory in Knowledge Representation: A Novel Approach to Reasoning with a Quantum Model of Concepts*. Master's thesis, Vrije Universiteit Brussel, Brussel.

Gavranović, B.; Lessard, P.; Dudzik, A.; von Glehn, T.; Araújo, J. G. M.; and Veličković, P. 2024. Categorical Deep Learning: An Algebraic Theory of Architectures. arXiv:2402.15332.

Grasedyck, L. 2010. Hierarchical Singular Value Decomposition of Tensors. *SIAM Journal on Matrix Analysis and Applications*, 31(4): 2029–2054.

Grelier, E.; Nouy, A.; and Chevreuil, M. 2018. Learning with Tree-Based Tensor Formats. https://arxiv.org/abs/1811.04455v2.

Leavitt, M. L.; and Morcos, A. 2020. Towards falsifiable interpretability research. arXiv:2010.12016.

Levine, Y.; Sharir, O.; Cohen, N.; and Shashua, A. 2018. Quantum Entanglement in Deep Learning Architectures. https://arxiv.org/abs/1803.09780v3.

Li, W.; Zhu, L.; and Cambria, E. 2021. Taylor's Theorem: A New Perspective for Neural Tensor Networks. *Knowledge-Based Systems*, 228: 107258.

Lin, H. W.; Tegmark, M.; and Rolnick, D. 2017. Why Does Deep and Cheap Learning Work so Well? *Journal of Statistical Physics*, 168(6): 1223–1247.

Loconte, L.; Mari, A.; Gala, G.; Peharz, R.; de Campos, C.; Quaeghebeur, E.; Vessio, G.; and Vergari, A. 2024. What Is the Relationship between Tensor Factorizations and Circuits (and How Can We Exploit It)? arXiv:2409.07953.

Lorenz, R.; Pearson, A.; Meichanetzidis, K.; Kartsaklis, D.; and Coecke, B. 2023. QNLP in Practice: Running Compositional Models of Meaning on a Quantum Computer. *Journal of Artificial Intelligence Research*, 76: 1305–1342.

Olah, C.; Cammarata, N.; Schubert, L.; Goh, G.; Petrov, M.; and Carter, S. 2020. Zoom In: An Introduction to Circuits. *Distill*. Https://distill.pub/2020/circuits/zoom-in.

Oseledets, I. V.; and Tyrtyshnikov, E. E. 2009. Breaking the Curse of Dimensionality, Or How to Use SVD in Many Dimensions. *SIAM Journal on Scientific Computing*, 31(5): 3744–3759.

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

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

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

Shiebler, D.; Gavranović, B.; and Wilson, P. 2021. Category Theory in Machine Learning. arXiv:2106.07032.

Tull, S.; Lorenz, R.; Clark, S.; Khan, I.; and Coecke, B. 2024. Towards Compositional Interpretability for XAI. arXiv:2406.17583.