Deep Tree Tensor Networks

Chang Nie
Nanjing University of Science and Technology
Nanjing, China
changnie@njust.edu.cn

Abstract

Originating in quantum physics, tensor networks (TNs) have been widely adopted as exponential machines and parametric decomposers for recognition tasks. Typical TN models, such as Matrix Product States (MPS), have not yet achieved successful application in natural image recognition. When employed, they primarily serve to compress parameters within pre-existing networks, thereby losing their distinctive capability to capture exponential-order feature interactions. This paper introduces a novel architecture named ***D**eep **T**ree **T**ensor **N**etwork* (DTTN), which captures 2^L-order multiplicative interactions across features through multilinear operations, while essentially unfolding into a *tree*-like TN topology with the parameter-sharing property. DTTN is stacked with multiple antisymmetric interaction modules (AIMs), and this design facilitates efficient implementation. Furthermore, our theoretical analysis demonstrates the equivalence between quantum-inspired TN models and polynomial/multilinear networks under specific conditions. We posit that the DTTN could catalyze more interpretable research within this field. The proposed model is evaluated across multiple benchmarks and domains, demonstrating superior performance compared to both peer methods and state-of-the-art architectures. Our code is publicly available at https://github.com/NieCha/deep_tree_tensor_network.

1 Introduction

"Simplicity is the ultimate sophistication." — Leonardo da Vinci

The wavefunction of a quantum many-body system typically resides in an extremely high-dimensional Hilbert space, with its complexity increasing exponentially as the particle count grows [27, 60]. For example, consider a system consisting of N spin-$\frac{1}{2}$ particles; the dimensionality of the corresponding Hilbert space would be 2^N. Tensor networks (TNs) offer powerful numerical techniques for tackling the "*Curse of Dimensionality*" [12]. By leveraging the local entanglement properties of quantum states, TNs represent complex wavefunctions into multilinear representations of multiple low-dimensional cores, thereby significantly reducing computational and storage requirements [11, 44][1]. This class of methods allows for an accurate representation of quantum states while mitigating the exponential growth in complexity, making it feasible to simulate large-scale quantum systems [26, 44].

Recently, TN-based interpretable and quantum-inspired white-box machine learning has attracted the attention of researchers. It holds the potential to generate novel schemes that can run on quantum hardware [25, 43]. Typical TN models, including Matrix Product States [13] (MPS[2]), Tree Tensor Network (TTN) [5], and Multi-Scale Entanglement Renormalization Ansatz (MERA) [45] are

[1] In tensor network theory, states that satisfy the area law for entanglement entropy can be efficiently approximated using TNs with finite bond dimensions.

[2] The MPS is also referred to as Tensor Train [41], or Tensor Ring [61] with the periodic condition in classical machine learning.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

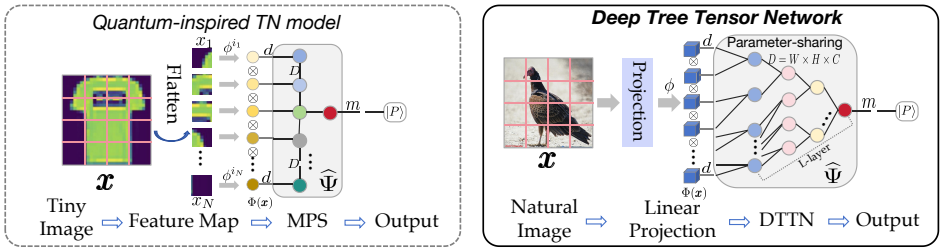


Figure 1: Schematic diagram of the quantum-inspired MPS model and DTTN towards image recognition task. The former is applied for tiny inputs and setting a small local mapping dimension $d = 2$ and bond dimension $D \leq 64$ in general[43]. DTTN handles complex inputs while retaining spatial locality in linear projection, and its parameter-sharing nature allows for maintaining a high bond dimension.

skillfully applied in image classification. Routine practice is to map each pixel or local patch of the image to a d-dimensional vector by a local map function $\phi(\cdot)$, and then use the *tensor product* to obtain a joint feature map $\Phi(\boldsymbol{x})$ of d^N dimensionality (see Fig. 1 left). This process can be expressed as follows:

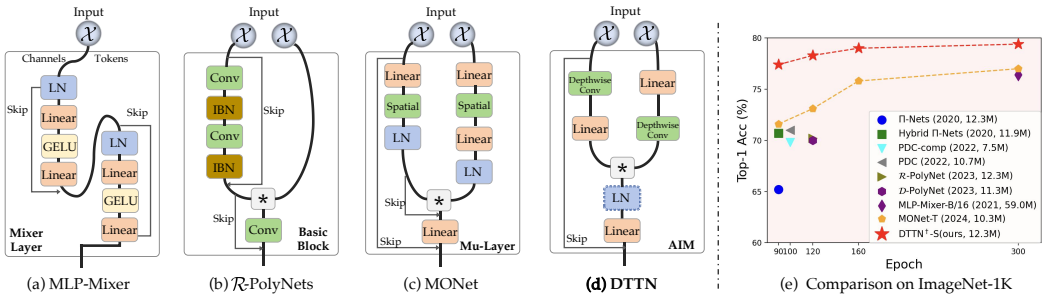
$$f(\boldsymbol{x}) = \arg\max_{m} \ \langle \boldsymbol{W}^m, \Phi(\boldsymbol{x}) \rangle \,. \tag{1}$$

Here, $\boldsymbol{W}^m$ represents a $(N+1)$-th order tensor with an output index m; $f(\cdot) : \mathbb{R}^{w \times h \times c} \to \mathbb{R}$ denotes a multilinear function. In principle, mapping samples into exponential dimensional space to achieve linear separability instead of adopting activation functions is the essence of TNs [47, 43, 42]. However, existing methods are limited to simple tasks, e.g., MNIST and Fashion MNIST [5, 43], and we reveal that this is mainly due to 1) *low computational efficiency* and 2) *lack of feature self-interaction capability* (Section 3.3 for more details). Consequently, our goal is to address these challenges by applying TNs to complex benchmarks such as ImageNet-1K, thereby bridging the gap between TNs and modern architectures.

Alternatively, TNs are popularly employed to parameterize existing network models for acceleration. For example, the variational parameters of a network can be directly decomposed into TN formats, including convolutional kernels [40], fully-connected weights [32, 39], and attention blocks [34], to name a few. During inference, one can choose to retain or merge the TN structure as required. However, such techniques are devoid of probabilistic interpretation and feature multiplicative interaction. Notably, there exist intriguing and previously unexplored similarities between advanced deep learning architectures and TNs, such as MLP-Mixer [51], polynomial networks, and multilinear networks [9, 10, 7], which have demonstrated strong performance on complex visual tasks. As shown in Fig. 2, we visualize different architectures from the TN perspective for comparison. We note that contemporary architectures exhibit several key distinctions from existing TNs, including the incorporation of *nonlinear activations* and *instance normalization*. Consequently, we can improve TNs by drawing insights from advanced architectures through rigorous equivalence analysis, thereby overcoming their limitations on complex benchmarks.

Concretely, we introduce a novel class of TN architectures, named *Deep Tree Tensor Network* (DTTN), which uses multilinear operations to capture exponential-order interactions among features and predict targets without relying on activation functions or attention mechanisms. DTTN is constructed by sequentially stacking multiple antisymmetric interaction modules (AIMs) and inherently unfolds into a *tree*-like structure to reach on-par performance compared to advanced architectures, with better interpretability and understanding. Overall, the contributions of this paper are summarized as follows:

- We introduce DTTN, a simple yet effective architecture constructed by sequentially stacking AIMs. DTTN captures feature 2^L multiplicative interactions without activation functions and achieves state-of-the-art performance compared to other polynomial and multilinear networks with faster convergence (see Fig. 2 (e)).
- We provide a theoretical analysis of the equivalence between DTTN and other architectures under specific conditions. Without instance normalization, DTTN essentially reduces to a tree-topology TN, thereby overcoming the limitations of TN-based Born machines that excel mainly in simple tasks.



(a) MLP-Mixer (b) $\mathcal{R}$-PolyNets (c) MONet **(d) DTTN** (e) Comparison on ImageNet-1K

Figure 2: (a-d) Illustration of Core Blocks for Different Architectures. The MLP-Mixer utilizes GELU activation and other networks via the Hadamard product "$*$" to enable the network to learn complex representations. We emphasize that instance batch normalization (IBN) and layer normalization (LN [57]) preceded before Hadamard product operations disrupt the polynomial unfolding nature of $\mathcal{R}$-PolyNets [10] and MONet [7]. In contrast, the succinctly designed AIM circumvents this issue. The optional LN inside AIM not only enhances performance but also facilitates faster convergence. (e) Comparison of Different Networks on ImageNet-1k. When comparing various networks trained on ImageNet over different epochs, DTTN stands out by achieving state-of-the-art performance compared to other multilinear networks, significantly outperforming them.

- We conduct a comprehensive evaluation of the proposed model's effectiveness and broader applicability across diverse benchmarks and domains, including visual recognition, recommender systems, and physics-informed neural networks. Our results show that DTTN achieves performance comparable to, and in certain cases even outperforms, carefully designed state-of-the-art architectures.

Notations. Throughout this paper, we use $\boldsymbol{x} \in \mathbb{R}^{I_1}$, $\boldsymbol{X} \in \mathbb{R}^{I_1 \times I_2}$, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ to denote first-order vectors, second-order matrices, and N-th order tensors, respectively. The blackboard letters are employed to represent a set of objects, e.g., $\mathbb{R}$ and $\mathbb{Z}$ denote real numbers and integers. In addition, $*$, $\odot$, and $\otimes$ denote the *Hadamard product*, *Khatri-Rao product*, and *tensor product*, respectively. For brevity, $|\mathbb{K}|$ denotes the cardinality of a set $\mathbb{K}$, and $\mathbb{K}_N$ denotes the positive integers set $\{1, 2, \ldots, N\}$. Moreover, for given tensors $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{I_4 \times I_5 \times I_6 \times I_7}$, with $I_2 = I_4$ and $I_3 = I_5$. The tensor contraction is executed by summing along the shared modes of $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$ to yield a new tensor $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \times_{2,3}^{1,2} \boldsymbol{\mathcal{B}} \in \mathbb{R}^{I_1 \times I_6 \times I_7}$. The entry-wise calculation can be expressed as $\boldsymbol{\mathcal{C}}_{(i_1,i_6,i_7)} = \sum_{i_2=1}^{I_2} \sum_{i_3=1}^{I_3} \boldsymbol{\mathcal{A}}_{(i_1,i_2,i_3)} \boldsymbol{\mathcal{B}}_{(i_2,i_3,i_6,i_7)}$. Refer to [29] for additional definitions.

2 Related Work

2.1 Quantum-Inspired TNs

Quantum-inspired tensor networks (TNs) enhance the performance of classical algorithms by mimicking quantum computational characteristics [25]. These methods map inputs into a Hilbert space with exponential dimensionality, achieving linear separability through local mappings and tensor products, while employing multiple low-order cores to parameterize coefficients, significantly reducing computational and storage complexity [47, 49]. The avoidance of activation functions, alongside the theoretical underpinnings rooted in many-body physics, contributes to the interpretability of TNs [43]. Recently, numerous studies have successfully applied TNs to tasks such as image classification [49], generation [5], and segmentation [48]. These studies effectively integrate established neural network techniques like residual connections [38], multiscale structures [35], and normalization [47] into TN frameworks. However, current TNs are predominantly suited for simpler tasks and face limitations in terms of computational efficiency and expressive power.

2.2 Advanced Modern Networks

In the contemporary landscape of deep learning, the design of network architectures has grown increasingly sophisticated and varied, each architecture presenting distinct advantages. Advanced models such as Convolutional Neural Networks (CNNs) renowned for efficient feature extraction [20,

22], Transformers distinguished by their powerful contextual understanding capabilities [54], MLP-based architectures celebrated for their simple yet effective designs [51], and Mamba noted for its linear complexity [18] have become pivotal across a wide array of applications. These networks leverage nonlinearities, while beneficial for model expressiveness, but limit applicability in domains such as security and encryption. Notably, Leveled Fully Homomorphic Encryption schemes can only support addition and multiplication operations [2, 7].

2.3 Polynomial and Multilinear Networks

Polynomial and multilinear networks employ addition and multiplication operations to construct intricate network representations [9, 10, 7]. Specifically, the pioneering Polynomial Network (PN) [9] constructs higher-order polynomial expansions of the input features in a modular fashion while supporting end-to-end training, achieving notable success in both image recognition and generation tasks. In their follow-up work, Chrysos et al. [10] introduce regularization strategies—such as data augmentation, instance normalization, and higher-order feature interactions—to further enhance model performance. Cheng et al. [7] drew inspiration from modern architectural designs to propose MONet, aiming to narrow the gap between multilinear networks and modern architectures. It is worth noting that both polynomial and multilinear networks can capture exponential-order feature interactions. However, a key distinction lies in their structural unfoldability: polynomial networks maintain an unfoldable structure, whereas multilinear networks may lose this property when LN is applied, as

$$\begin{cases} (\boldsymbol{A}\boldsymbol{z}) * (\boldsymbol{B}\boldsymbol{z}) = vec(\boldsymbol{z} \otimes \boldsymbol{z})(\boldsymbol{A}^T \odot \boldsymbol{B}^T), \\ \texttt{LN}(\boldsymbol{A}\boldsymbol{z}) * (\boldsymbol{B}\boldsymbol{z}) \neq vec(\boldsymbol{z} \otimes \boldsymbol{z})\ \texttt{LN}(\boldsymbol{A}^T \odot \boldsymbol{B}^T). \end{cases} \tag{2}$$

Here $\texttt{LN}(\cdot)$ represents layer normalization, while A and B are learnable matrices.

In this work, we aim to achieve two objectives: (1) re-establish the polynomial expansion form and analyze the differences between multilinear and quantum-inspired TNs; and (2) develop a layer-normalized multilinear DTTN† that outperforms existing high-performance multilinear networks [7].

3 Method

In this section, we provide a detailed description of the Deep Tree Tensor Network (DTTN). We aim to construct a tree topology network by sequentially stacking AIM blocks, which consist solely of multilinear operations. For an input image $\boldsymbol{x}$, we apply a vanilla linear projection, also known as patch embedding [51], to obtain the feature map $\phi(\boldsymbol{x}, \boldsymbol{\Lambda}_\phi) \in \mathbb{R}^{W \times H \times C}$. Similar approaches are used in other methodologies. Here $\boldsymbol{\Lambda}_\phi \in \mathbb{R}^{S^2 \times C}$ represents a learnable matrix, where $S, C \in \mathbb{N}$ denote the local patch size and the number of output channels, respectively. This procedure corresponds to the local mapping illustrated in Fig. 1, with its output serving as the input for the DTTN. It should be noted that batch normalization (BN) operations following the linear layer have been omitted for brevity, as these operations can be integrated with the nearest linear layer during inference through structural re-parameterization technique [15].

3.1 Antisymmetric Interaction Module

The antisymmetric interaction module (AIM) is the core of DTTN. As illustrated in Fig. 2(d), for the l-th block input feature map $\boldsymbol{\mathcal{X}}^l \in \mathbb{R}^{W_l \times H_l \times C_l}$, we utilize an antisymmetric two-branch structure to capture the linear interactions of the input features separately. Both parts, denoted as f_1^l, f_2^l, incorporate a depthwise convolution layer and a linear layer, but apply them in reversed order. These layers are designed to capture spatial locality and channel interactions, respectively, effectively combining the advantages of CNN and MLP architectures [20, 51]. The antisymmetric design specifically targets reducing the complexity of AIM. The ratio of parameters and FLOPs between the two branches can be expressed as follows:

$$\begin{aligned} R_{Para} &= \frac{r_{exp}k^2C_l + r_{exp}C_l^2}{r_{exp}k^2C_l + r_{exp}^2C_l^2} \sim \frac{1}{r_{exp}}, \\ R_{Flops} &= \frac{r_{exp}k^2W_lH_lC_l + r_{exp}W_lH_lC_l^2}{r_{exp}k^2W_lH_lC_l + r_{exp}^2W_lH_lC_l^2} \sim \frac{1}{r_{exp}}, \end{aligned} \tag{3}$$

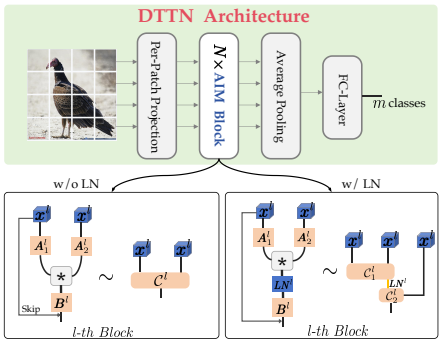


Figure 3: Schematic overview of the DTTN architecture.

Table 1: Specifications of different DTTN variants configuration. "Tiny", "Small", and "Large" refer to DTTN-T, DTTN-S, and DTTN-L configurations, respectively, each differing in parameter sizes. The primary distinction among these variants lies in the number of blocks and the hidden-sizes within their multi-stage structures. Additionally, models with layer normalization (LN) layers are denoted with a '†', such as DTTN†-S.

Specification	Tiny	Small	Large
Numbers of Blocks	34	44	56
Hidden-size	64,128,160,192	96,128,192,192	128,192,256,384
Stages	6,6,16,6	6,6,24,8	8,8,32,8
Expansion ratio	3	3	3
Parameters (M)	7.1	12.3	35.9
FLOPs (B)	2.3	4.1	12.3

where $r_{exp} \in \mathbb{N}_+$ represents the expansion ratio in the inverted bottleneck design, typically set to 3, $k \in \mathbb{N}_+$ denotes the kernel size, and C_l, W_l, H_l are the number of channels, width and height of the l-th block input feature map, respectively. We use the hardware-friendly Hadamard product ($*$) to capture the second-order interactions of the branch outputs. Following this, an optional LN and a linear projection layer are sequentially applied to the computation results. Finally, a shortcut connection is used to preserve the input signal and accelerate training (see Fig. 2). Overall, the AIM forward calculation can be expressed as

$$\boldsymbol{\mathcal{X}}^{l+1} = \boldsymbol{\mathcal{X}}^{l} + Pro\left(f_1^l(\boldsymbol{\mathcal{X}}^l) * f_2^l(\boldsymbol{\mathcal{X}}^l)\right), \forall l \in \mathbb{K}_L, \tag{4}$$

where Pro$(\cdot)$ denotes the projection transformation operation. In summary, AIM captures second-order multiplicative interactions among input elements through multilinear operations without employing *nonlinear activations*. In contrast to core blocks inside other architectures, such as the Basic Block in $\mathcal{R}$-PolyNets [10] and Mu-layer[3] in MONet [7], AIM employs an antisymmetric design with only one shortcut connection.

3.2 Network Architecture

Our proposed architecture is constructed by stacking L AIM blocks sequentially. As shown in Fig. 3, the final output is derived through average pooling and a fully-connected layer. The DTTN architecture is multi-stage, in line with its peers [9, 21, 7]. By varying the hidden-sizes and the number of blocks in each stage, we have designed three variants: *DTTN-T*, *DTTN-S*, and *DTTN-L* (see Table 1), each with distinct parameters to facilitate a comparative analysis. At a high level, we assert that the DTTN exhibits the following theoretical property:

Proposition 1. *The DTTN has the capability to capture 2^L multiplicative interactions among input elements, which can be represented in the format of Equation (1) as $\Phi(\boldsymbol{x}) = \otimes^{2^L} \phi(\boldsymbol{x}, \boldsymbol{\Lambda}_\phi)$. Consequently, the elements of $f(\boldsymbol{x})$ are homogeneous polynomials of degree 2^L over the feature map $\phi(\boldsymbol{x}, \boldsymbol{\Lambda}_\phi)$.*

It is important to note, however, that this characteristic no longer holds when LN is incorporated into the network, as LN introduces second-order statistical information. This phenomenon is clearly exemplified in models like MONet and DTTN†, where the network essentially behaves as a standard multilinear model.

Unfolding Topology. As illustrated in Fig. 3, the AIM is equivalent to a binary tree node in the absence of LN. In this case, AIM forward computation can be regarded as a TN contraction, which

[3]The MONet architecture comprises a stack of two variants of Mu-Layer. The second variant differs from the one shown in Fig. 2(c) in that it does not incorporate a spatial shift operation.

can be formulated as

$$\begin{aligned}\boldsymbol{x}^{l+1} =& \boldsymbol{x}^{l} + \boldsymbol{B}^{l}\left((\boldsymbol{A}_1^{l}\boldsymbol{x}^{l}) * (\boldsymbol{A}_2^{l}\boldsymbol{x}^{l})\right) \\ =& \boldsymbol{x}^{l} + \texttt{Reshape}\left(\boldsymbol{B}^{l}({\boldsymbol{A}_1^{l}}^{T} \odot {\boldsymbol{A}_2^{l}}^{T})^{T}\right) \times_{2,3}^{1,2} (\boldsymbol{x}^{l} \otimes \boldsymbol{x}^{l}) \\ =& \boldsymbol{\mathcal{C}}^{l} \times_{2,3}^{1,2} (\boldsymbol{x}^{l} \otimes \boldsymbol{x}^{l})\end{aligned} \tag{5}$$

Here $\boldsymbol{x}^{l} = vec(\boldsymbol{\mathcal{X}}^{l}) \in \mathbb{R}^{W_l H_l C_l}$, $\boldsymbol{\mathcal{C}}^{l}$ signifies a third-order tensor representing structural re-parameterization with the learnable matrices $\boldsymbol{A}_1^{l}$, $\boldsymbol{A}_2^{l}$ and $\boldsymbol{B}^{l}$. The AIM captures the second-order multiplicative interactions among input elements via *tensor product* and structures the three-order tensor utilizing *PyTorch* operators. Thus, one DTTN composed of L AIMs can essentially be unfolded into a tree network with 2^L leaf nodes, as pictured in Fig. 1.

3.3 DTTN vs. Other Architectures

DTTN & Polynomial Networks. DTTN can be expressed in the same polynomial expansion form as Π-Net [9], which is given by

$$f(\boldsymbol{x}) = \sum_{l=1}^{2^L} \left(\mathcal{W}^{[l]} \times_{2,\ldots,l+1}^{1,\ldots,l} (\otimes^{l}\phi(x))\right) + \beta, \tag{6}$$

where $\beta \in \mathbb{R}^m$ represents the constant term, and $\mathcal{W}^{[l]}$ is a $(l+1)$-th order learnable parameter tensor. Notably, equations (1) and (6) become equivalent upon introducing a bias term for the elements of the input vector $\boldsymbol{x}$. The networks exhibit different structured representations of the coefficients $\mathcal{W}^{[l]}$, for all $l \in \mathbb{K}_{2^L}$, due to their distinct network blocks and computational graphs.

DTTN & Multilinear Networks. We note that networks which exclusively involve multilinear operations—such as MONet, DTTN†, and $\mathcal{R}$-PolyNets—yet lack the polynomial expansion structure can be classified as multilinear networks.

DTTN & Quantum-inspired TNs. The main advantages of DTTN over Quantum-inspired TNs are twofold. (1) *The alternative of the local map function.* Existing TNs employ trigonometric functions for local mapping, which leads to the absence of higher-order (≥ 2) terms involving input elements in the network's unfolded form. This limitation results in a loss of feature self-interaction capabilities. (2) *Higher bond dimension induced by parameter-sharing properties.* Quantum-inspired TNs facilitate the parallel contraction of shared indices among N cores within the same layer. However, due to limited memory capacity, the bond dimension must often be restricted to smaller values. The following theorem establishes the equivalence between DTTNs and TNs:

Theorem 1. *Given the local mapping function $\phi^{i_j}(x_j) = [x_j^0, \ldots, x_j^{2^L}]^T$, a polynomial network with the expansion form of Equation (6) can be transformed into a quantum-inspired TNs model with finite bond dimension.*

Since the internal cores of a TN can be decomposed into a "**core-diagonal factor-core**" structure via Higher-order Singular Value Decomposition (HOSVD) [29] and subsequently merged with connectivity cores, we regard the structural differences between DTTN and quantum-inspired TNs as negligible. We believe the above theorem not only establishes an equivalence between quantum-inspired TNs and modern architectures, but also offers insights for the future development of more interpretable and high-performance TN models.

4 Experiments

In this section, we provide a comprehensive assessment of the DTTN's effectiveness. Specifically, in Section 4.1, we perform experiments on a series of image classification benchmarks to validate the model's superiority over other multilinear and TN architectures. In Section 4.2, we show the broader impact of the DTTN across other domains, including recommendation system and partial differential equation (PDE) solving. Section 4.3 presents ablation studies aimed at examining the impact of various design choices. The paper concludes with a discussion of the model's limitations. Further analysis and detailed results are provided in the appendix.

Table 2: ImageNet-1K classification accuracy for various network architectures. Models that can be polynomially expanded are marked in red, whereas other multilinear models that do not include activation functions but incorporate layer normalization are marked in green. Our DTTN†-T outperforms the previous SOTA model MONet-T by 0.9% with fewer parameters and FLOPs.

Model	Top-1(%)	Params (M)	FLOPs(B)	Epoch	Activation	Attention	Reso.
CNN-based							
ResNet-50 [20]	77.2	25.0	4.1	-	ReLU	×	224^2
A^2Net [4]	77.0	33.4	31.3	-	ReLU	✓	224^2
AA-ResNet-152 [1]	79.1	61.6	23.8	100	ReLU	✓	224^2
RepVGG-B2g4 [15]	79.4	55.7	11.3	200	ReLU	×	224^2
Transformer- and Mamba-based							
ViT-B/16 [16]	77.9	86.0	55.0	300	GeLU	✓	224^2
DeiT-S/16 [53]	81.2	24.0	5.0	300	GeLU	✓	224^2
Swin-T/16 [36]	81.3	29.0	4.5	300	GeLU	✓	224^2
Vim-S [62]	80.5	26.0	-	300	SiLU	✓	224^2
MLP-based							
MLP-Mixer-B/16 [51]	76.4	59.0	11.6	300	GeLU	×	224^2
MLP-Mixer-L/16 [51]	71.8	507.0	44.6	300	GeLU	×	224^2
CycleMLP-T [3]	81.3	28.8	4.4	300	GeLU	×	224^2
Hire-MLP-Tiny [19]	79.8	18.0	2.1	300	GeLU	×	224^2
ResMLP-24 [52]	79.4	6.0	30.0	300	GeLU	×	224^2
S^2MLP-Wide [59]	80.0	71.0	14.0	300	GeLU	×	224^2
S^2MLP-Deep [59]	80.7	10.5	51.0	300	GeLU	×	224^2
ViP-Small/14 [21]	80.5	30.0	6.5	300	GeLU	✓	224^2
AFFNet [24]	79.8	6.0	1.5	300	ReLU	✓	256^2
Polynomial- and Multilinear-based							
Π-Nets [9]	65.2	12.3	1.9	90	-	×	224^2
DTTN-S(ours)	**71.8 / 77.2**	12.3	4.1	90 / 300	-	×	224^2
Hybrid Π-Nets [9]	70.7	11.9	1.9	90	ReLU+Tanh	×	224^2
PDC [8]	71.0	10.7	1.6	100	ReLU+Tanh	×	224^2
PDC-comp [8]	70.2	7.5	1.3	100	ReLU+Tanh	×	224^2
$\mathcal{R}$-PolyNets [10]	70.2	12.3	1.9	120	-	×	224^2
$\mathcal{D}$-PolyNets [10]	70.0	11.3	1.9	120	-	×	224^2
MONet-T [7]	77.0	10.3	2.8	300	-	×	224^2
DTTN†-T(ours)	**77.9**	7.1	2.3	300	-	×	224^2
DTTN†-S(ours)	**79.4**	12.3	4.1	300	-	×	224^2
MONet-S [7]	81.3	32.9	6.8	300	-	×	224^2
DTTN†-L(ours)	**82.4**	35.9	12.3	300	-	×	224^2

4.1 Visual Recognition

Setup and Training Details. A series of benchmarks with different types, scales, and resolutions are employed for the experiments, including CIFAR-10 [30], Tiny ImageNet [31], ImageNet-100 [58], ImageNet-1K [46], MNIST, and Fashion-MNIST [56]. A detailed description of the benchmarks and training configurations is included in the supplement.

Table 3: Experimental validation of various network architectures was conducted on smaller benchmarks with differing resolutions. The top-performing results are highlighted in bold. Among these, our DTTN†-S model achieves the best performance across all benchmarks, outperforming other polynomial and multilinear networks.

	CIFAR-10	Tiny ImageNet	ImageNet-100
Resolution	32^2	64^2	224^2
Resnet18	94.4	61.5	85.6
MLP-Mixer	90.6	45.6	84.3
Res-MLP	92.3	58.9	84.8
Π-Nets	90.7	50.2	81.4
Hybrid Π-Nets	94.4	61.1	85.9
PDC	90.9	45.2	82.8
$\mathcal{D}$-PolyNets	94.7	61.8	86.2
MONet-T	94.8	61.5	87.2
DTTN†-S	**95.0**	**63.8**	**87.7**

Table 4: Experimental validation was conducted comparing the DTTN-S with quantum-inspired TNs on the MNIST and Fashion-MNIST datasets. The highest performances, marked in bold, are achieved by our DTTN-S, which outperforms previous tensor networks, including aResMPS that utilizes residual connections and ReLU activation.

Model	MNIST	Fashion-MNIST
MPS Machine	0.9880	0.8970
Bayesian TN	-	0.8692
PEPS	-	0.883
LoTeNet	0.9822	0.8949
aResMPS	0.9907	0.9142
DTTN-S	**0.9930**	**0.9236**

DTTN vs. Polynomial Networks. Table 2 reports the results of the unfoldable networks DTTN-S and Π-Nets [9] on ImageNet-1K (light red areas), using neither activation functions nor instance

Table 5: Validating AIM as a pluggable module for enhancing feature interaction in recommendation models with consistent performance gains.

Model	Criteo	Avazu
DeepFM	80.12	75.46
DeepFM+AIM	80.44$_{+0.32}$	75.73$_{+0.27}$
FiBiNet	80.42	76.01
FiBiNet+AIM	80.97$_{+0.55}$	76.08$_{+0.07}$
DCN-V2	80.93	76.14
DCN-V2+AIM	81.15$_{+0.22}$	76.52$_{+0.38}$

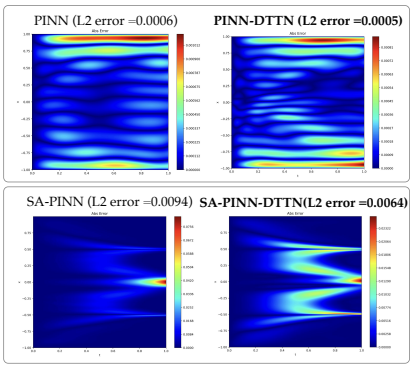


Figure 4: Performance of PINNs on linear and nonlinear Allen-Cahn PDEs: L2 error and absolute error across the Spatial-Temporal domain.

Table 6: The influence of network depth and width on model performance.

	Top-1 (%)	Params(M)
L=8, d=256	79.2	5.6
L=16,d=256	85.5	10.2
L=24,d=256	86.8	14.8
L=32,d=256	**87.2**	19.4
L=32,d=64	63.4	1.3
L=32,d=128	82.5	4.9
L=32,d=512	**87.9**	76.8

Table 7: The influence of different design choices for AIM on the performance of the DTTN variants.

	Top-1 (%)	Params(M)
SIM-Conv	86.2	9.1
SIM-Linear	84.9	4.8
DTTN†-T	**86.4**	6.9
Sim-Conv	87.8	15.9
Sim-Linear	85.2	8.3
DTTN†-S	87.7	12.1

Table 8: The influence of layer normalization inside AIM on the performance of the DTTN variants.

Model	Top-1 (%)	Params(M)
DTTN-T	85.6	6.9
DTTN†-T	**86.4**$_{+1.8}$	6.9
DTTN-S	87.3	12.1
DTTN†-S	**87.7**$_{+0.4}$	12.1
DTTN-L	87.6	35.6
DTTN†-L	**88.1**$_{+0.5}$	35.6

normalization. DTTN-S achieves Top-1 accuracy of 71.8% and 77.2% after 90 and 300 training epochs, respectively, significantly outperforming Π-Nets by 5.6% and 12%, respectively. Additionally, DTTN-S maintains a significant advantage—nearly a 10% improvement in accuracy—over Hybrid Π-Nets that incorporate activation functions (which lose their unfolding properties).

DTTN vs. Multilinear Networks. Fig. 2(e) illustrates the performance of various multilinear networks trained on ImageNet-1K over different epochs, with the curve for DTTN†-S showing superior results compared to others. Tables 2 and 3 detail the Top-1 accuracies of multilinear architectures across a range of benchmarks of varying scales, including CIFAR-10, Tiny ImageNet, ImageNet-100, and ImageNet-1K. Specifically, DTTN-S achieves improvements of 0.2%, 2.3%, 0.5%, and 2.4% over the previous best models at similar scales. Moreover, the unfoldable DTTN-S attains an impressive 77.2% accuracy on ImageNet-1K without instance normalization. Additionally, DTTN†-T surpasses the prior state-of-the-art model MONet [7] by 0.9%, while utilizing approximately 30% fewer parameters and reducing FLOPs by 20%. These comparisons demonstrate that DTTN achieves significantly faster convergence and superior performance.

DTTN vs. Quantum-inspired TNs. DTTN outperforms other quantum-inspired TNs on smaller benchmarks like MNIST and Fashion-MNIST. Notably, these models have not yet been successfully scaled to large benchmarks. The baselines included for comparison are MPS [49], Bayesian TNS, PEPS [6], LoTeNet [47], and aResMPS [38]. The test results are reported in Tab. 4, where DTTN-S consistently achieves the best performance. Specifically, on the Fashion-MNIST dataset, DTTN-S outperformed the second-best model, aResMPS, by 0.96%. This superior performance can be attributed to the higher bond dimension facilitated by parameter sharing and self-interaction capabilities, as discussed in Section 3.3.

DTTN vs. Advanced Architectures. We further demonstrate the competitive performance of DTTN relative to modern architectures, marking it as the first TN model to be applied to large-scale benchmarks. Table 2 reports the Top-1 accuracy of the DTTN family compared to other models, including CNN-based, Transformer-based, Mamba-based, and MLP-based architectures

trained on ImageNet-1K. The DTTN†-L achieves an accuracy of 82.4% with 35.9M parameters, which is competitive and outperforms models such as DeiT-S/16 [53], ViP-Small/14 [21], and Vim-S [62]. Additionally, results in Tab. 3 further illustrate that DTTN achieves the highest accuracy on small benchmarks. The ViP-Small/14 [21] with an MLP architecture and MONet exhibit slower convergence on ImageNet-100 compared to ResNet-50 (see Appendix B), which is due to their inductive bias. In contrast, DTTN demonstrates remarkable training efficiency and superior accuracy relative to its counterparts.

4.2 Broader Impact

Recommendation System. Given the critical role of feature interaction in recommendation systems [33], we select two widely-used datasets, Criteo and Avazu, to evaluate the effectiveness of AIM in enhancing feature interaction modeling. In our implementation, AIM is designed as a pluggable module and seamlessly integrated into existing Click-Through Rate (CTR) prediction models, including DeepFM, FiBiNet [23], and DCN-V2 [55][4]. In this setup, AIM replaced all linear layers in the target models, with internal convolution operations removed, leaving only linear and normalization layers. Experimental validation, reported in Tab. 5 and evaluated using AUC, shows that AIM consistently enhances performance across both datasets for all tested CTR models. Specifically, FiBiNet saw a 0.55% improvement on the Criteo dataset.

DTTN-based PDE Solver. Physics-Informed Neural Networks (PINNs) [14] incorporate physical constraints of systems into the training process, enabling the model to learn governing physical laws for describing system behaviors. The nonlinear activation functions serve as the critical component for PINNs to capture complex patterns. Here, we apply DTTN to solve specific partial differential equations (PDEs), including both the linear case and the highly nonlinear Allen-Cahn equation (defined as $u_t = \epsilon\Delta u + u - u^3$, where ϵ controls interface width and $F(u) = u^3 - u$ represents nonlinear reaction terms. The two equations to be solved are as follows:

$$\begin{cases} \frac{\partial u}{\partial t} = -u \\ u(x, t=0) = sin(\pi x) \\ u(1,t) = u(-1,t) = 0 \end{cases} \quad \begin{cases} u_t - 0.0001u_{xx} + 5u^3 - 5u = 0 \\ u(x, t=0) = x^2 cos(\pi x) \\ u(1,t) = u(-1,t) = -1. \end{cases} \tag{7}$$

We employ PINN [14] and Self-Adaptive PINN (SA-PINN [50]) with hard boundary conditions to address these problems. For comparison, we replace the linear layers with the AIM module to construct PINN-DTTN and SA-PINN-DTTN, which maintain identical configurations to their baselines but omit the Tanh nonlinearity. Numerical results, including the average L2-error and the absolute error of the prediction in the spatial-temporal domain, are presented in Fig. 4. DTTN-based PINNs achieve lower prediction errors without activation nonlinearity.

4.3 Ablation Study

Here, we conduct ablation studies to evaluate the effectiveness of various design choices, including network depth and width, the antisymmetric design of AIM, and layer normalization. We utilize ImageNet-100—a representative subset of ImageNet-1K—as our test benchmark, which is well-suited for model validation and hyperparameter tuning under limited computational resources. Throughout these experiments, all hyperparameters are kept consistent with those of the final model, except for the variable under investigation. Each model was trained from scratch. We report the experimental results in Tables 6, 7, and 8.

Depth and Width. To control the depth and width of DTTN, we vary the number of AIM blocks and the hidden-size denoted as L and d, for brevity. For ease of comparison, we kept d fixed across different stages. As illustrated in Tab. 6, we first set L to $\{8, 16, 24, 32\}$ with $d = 256$, and subsequently varied d to $\{64, 128, 256, 512\}$ while fixing L at 32. This approach enables us to explore how network depth and width affect model performance. Notably, performance declines significantly as both L and d are reduced. Specifically, Top-1 accuracy falls by 24.5% when d is reduced from 512 to 64. Furthermore, the number of model parameters grows linearly with L and quadratically with d. These observations provide indirect evidence for the effectiveness of DTTN's multi-stage design.

[4] https://github.com/shenweichen/DeepCTR-Torch

Antisymmetrical Design of AIM. We evaluate the effectiveness of the antisymmetric design of AIM on the DTTN†-T and DTTN†-S architectures. As shown in Tab. 7, we compare the impact of symmetric versus antisymmetric designs on model performance. In this context, SIM-Conv and SIM-Linear refer to the Symmetric Interaction Module (SIM), which uses prioritized convolutional and linear layers, respectively, as alternatives to AIM. We observe that DTTN†-T achieves the best results among its counterparts, whereas DTTN†-S performs only slightly worse than SIM-Conv (by 0.1%), while using approximately 20% fewer parameters. Furthermore, the reduction in the number of parameters aligns with the theoretical predictions corresponding to Equation 3.

Importance of LN. Figure 3(d) and Table 2 highlight the essential role of the Layer Normalization (LN) layer in enhancing the network's representational capacity—a key technique for improving the performance of multilinear networks [10, 7]. Table 8 reports the improvements in DTTN variants after incorporating LN, showing gains of 1.8%, 0.4%, and 0.5%, respectively. Moreover, by comparing the results in Fig. 2(e) and Tab. 2, we observe that DTTN-S achieves a Top-1 accuracy that is 5.6% and 2.2% lower than DTTN†-S after 90 and 300 training epochs, respectively.

Moreover, by combining the data from Figure 2(e) and Tab. 2, it can be seen that DTTN-S exhibits a Top-1 accuracy that is 5.6% and 2.2% lower than DTTN†-S after training for 90 and 300 epochs, respectively. This further underscores the importance of LN in boosting the performance and convergence of our proposed architecture.

Limitations. Although we have conducted an extensive evaluation of DTTN's effectiveness and advantages, computational resource limitations prevented us from performing additional experiments. These include pre-training on larger datasets like JFT-300M and evaluating model robustness. Future research will focus on a theoretical analysis of the proposed model, as well as investigate multilinear transformer architectures tailored for large language models (LLMs).

5 CONCLUSION

This paper introduces a multilinear network architecture named DTTN, which bridges the gap between quantum-inspired TNs and advanced architectures, achieving this uniquely without activation nonlinearity. Specifically, DTTN employs a modular stacking design to capture exponential interactions among input features, essentially unfolding into a tree-structured TN. We conducted extensive experiments to showcase the effectiveness of DTTNs across various benchmarks. Notably, this is the first validation of TNs' effectiveness on large-scale benchmarks, yielding competitive results compared to advanced architectures. Additionally, we explored the broader applicability of DTTN in other domains, such as recommendation systems and solving PDEs. Lastly, more theoretical discussions about DTTN will be addressed in future work.

6 Acknowledgment

We thank Yajie Chen and Junfang Chen for their valuable feedback and suggestions for improving this paper.

References

[1] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3286–3295, 2019.

[2] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[3] Shoufa Chen, Enze Xie, Chongjian Ge, Runjian Chen, Ding Liang, and Ping Luo. Cyclemlp: A mlp-like architecture for dense prediction. *arXiv preprint arXiv:2107.10224*, 2021.

[4] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Aˆ 2-nets: Double attention networks. *Advances in neural information processing systems*, 31, 2018.

[5] Song Cheng, Lei Wang, Tao Xiang, and Pan Zhang. Tree tensor networks for generative modeling. *Physical Review B*, 99(15):155131, 2019.

[6] Song Cheng, Lei Wang, and Pan Zhang. Supervised learning with projected entangled pair states. *Physical Review B*, 103(12):125117, 2021.

[7] Yixin Cheng, Grigorios G Chrysos, Markos Georgopoulos, and Volkan Cevher. Multilinear operator networks. *arXiv preprint arXiv:2401.17992*, 2024.

[8] Grigorios G Chrysos, Markos Georgopoulos, Jiankang Deng, Jean Kossaifi, Yannis Panagakis, and Anima Anandkumar. Augmenting deep classifiers with polynomial neural networks. In *European Conference on Computer Vision*, pages 692–716. Springer, 2022.

[9] Grigorios G Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Jiankang Deng, Yannis Panagakis, and Stefanos Zafeiriou. Deep polynomial neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 44(8):4021–4034, 2021.

[10] Grigorios G Chrysos, Bohan Wang, Jiankang Deng, and Volkan Cevher. Regularization of polynomial networks for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16123–16132, 2023.

[11] Andrzej Cichocki. Era of big data processing: A new approach via tensor networks and tensor decompositions. *arXiv preprint arXiv:1403.2048*, 2014.

[12] Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh-Huy Phan, Qibin Zhao, Danilo P Mandic, et al. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends® in Machine Learning*, 9(4-5):249–429, 2016.

[13] J Ignacio Cirac, David Perez-Garcia, Norbert Schuch, and Frank Verstraete. Matrix product states and projected entangled pair states: Concepts, symmetries, theorems. *Reviews of Modern Physics*, 93(4):045003, 2021.

[14] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

[15] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13733–13742, 2021.

[16] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[17] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[18] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[19] Jianyuan Guo, Yehui Tang, Kai Han, Xinghao Chen, Han Wu, Chao Xu, Chang Xu, and Yunhe Wang. Hire-mlp: Vision mlp via hierarchical rearrangement. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 826–836, 2022.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] Qibin Hou, Zihang Jiang, Li Yuan, Ming-Ming Cheng, Shuicheng Yan, and Jiashi Feng. Vision permutator: A permutable mlp-like architecture for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):1328–1334, 2022.

[22] Qibin Hou, Cheng-Ze Lu, Ming-Ming Cheng, and Jiashi Feng. Conv2former: A simple transformer-style convnet for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[23] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. Fibinet: combining feature importance and bilinear feature interaction for click-through rate prediction. In *Proceedings of the 13th ACM conference on recommender systems*, pages 169–177, 2019.

[24] Zhipeng Huang, Zhizheng Zhang, Cuiling Lan, Zheng-Jun Zha, Yan Lu, and Baining Guo. Adaptive frequency filters as efficient global token mixers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6049–6059, 2023.

[25] William Huggins, Piyush Patil, Bradley Mitchell, K Birgitta Whaley, and E Miles Stoudenmire. Towards quantum machine learning with tensor networks. *Quantum Science and technology*, 4(2):024001, 2019.

[26] Daniel Jaschke, Simone Montangero, and Lincoln D Carr. One-dimensional many-body entangled open quantum systems with tensor network methods. *Quantum science and technology*, 4(1):013001, 2018.

[27] Hong-Chen Jiang, Zheng-Yu Weng, and Tao Xiang. Accurate determination of tensor network state of quantum lattice models in two dimensions. *Physical review letters*, 101(9):090603, 2008.

[28] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6399–6408, 2019.

[29] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[30] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[31] Yann Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[32] Heng-Chao Li, Zhi-Xin Lin, Tian-Yu Ma, Xi-Le Zhao, Antonio Plaza, and William J Emery. Hybrid fully connected tensorized compression network for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–16, 2023.

[33] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1754–1763, 2018.

[34] Yingyu Liang, Zhenmei Shi, Zhao Song, and Yufa Zhou. Tensor attention training: Provably efficient learning of higher-order transformers. *arXiv preprint arXiv:2405.16411*, 2024.

[35] Ding Liu, Shi-Ju Ran, Peter Wittek, Cheng Peng, Raul Blázquez García, Gang Su, and Maciej Lewenstein. Machine learning by unitary tensor network of hierarchical tree structure. *New Journal of Physics*, 21(7):073059, 2019.

[36] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[37] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[38] Ye-Ming Meng, Jing Zhang, Peng Zhang, Chao Gao, and Shi-Ju Ran. Residual matrix product state for machine learning. *SciPost Physics*, 14(6):142, 2023.

[39] Chang Nie, Huan Wang, and Lu Zhao. Stn: Scalable tensorizing networks via structure-aware training and adaptive compression. *arXiv preprint arXiv:2205.15198*, 2022.

[40] Chang Nie, Huan Wang, and Lu Zhao. Adaptive tensor networks decomposition for high-order tensor recovery and compression. *Information Sciences*, 629:667–684, 2023.

[41] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[42] Siddhartha Patra, Saeed S Jahromi, Sukhbinder Singh, and Román Orús. Efficient tensor network simulation of ibm's largest quantum processors. *Physical Review Research*, 6(1):013326, 2024.

[43] Shi-Ju Ran and Gang Su. Tensor networks for interpretable and efficient quantum-inspired machine learning. *Intelligent Computing*, 2:0061, 2023.

[44] Shi-Ju Ran, Emanuele Tirrito, Cheng Peng, Xi Chen, Luca Tagliacozzo, Gang Su, and Maciej Lewenstein. *Tensor network contractions: methods and applications to quantum many-body systems*. Springer Nature, 2020.

[45] Justin Reyes and Miles Stoudenmire. A multi-scale tensor network architecture for classification and regression. *arXiv preprint arXiv:2001.08286*, 2020.

[46] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[47] Raghavendra Selvan and Erik B Dam. Tensor networks for medical image classification. In *Medical imaging with deep learning*, pages 721–732. PMLR, 2020.

[48] Raghavendra Selvan, Erik B Dam, Søren Alexander Flensborg, and Jens Petersen. Patch-based medical image segmentation using matrix product state tensor networks. *arXiv preprint arXiv:2109.07138*, 2021.

[49] Edwin Stoudenmire and David J Schwab. Supervised learning with tensor networks. *Advances in neural information processing systems*, 29, 2016.

[50] Shashank Subramanian, Robert M Kirby, Michael W Mahoney, and Amir Gholami. Adaptive self-supervision algorithms for physics-informed neural networks. In *ECAI 2023*, pages 2234–2241. IOS Press, 2023.

[51] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

[52] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *IEEE transactions on pattern analysis and machine intelligence*, 45(4):5314–5321, 2022.

[53] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[54] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[55] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the web conference 2021*, pages 1785–1797, 2021.

[56] H Xiao. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[57] Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. Understanding and improving layer normalization. *Advances in neural information processing systems*, 32, 2019.

[58] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3014–3023, 2021.

[59] Tan Yu, Xu Li, Yunfeng Cai, Mingming Sun, and Ping Li. S2-mlp: Spatial-shift mlp architecture for vision. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 297–306, 2022.

[60] Hui-Hai Zhao, Zhi-Yuan Xie, Qihong N Chen, Zhong-Chao Wei, Jianwei W Cai, and Tao Xiang. Renormalization of tensor-network states. *Physical Review B—Condensed Matter and Materials Physics*, 81(17):174411, 2010.

[61] Qibin Zhao, Guoxu Zhou, Shengli Xie, Liqing Zhang, and Andrzej Cichocki. Tensor ring decomposition. *arXiv preprint arXiv:1606.05535*, 2016.

[62] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024.

Appendix

A Experimental Setup

A.1 Image Classification Benchmarks and Training Settings

A detailed description of the benchmarks and training configurations is provided below.

- CIFAR-10 [30] consists of 60K color images of 32×32 resolution across 10 classes, with 50K images for training and 10K for testing. We train our model using the SGD optimizer with a batch size of 128 for 160 epochs. The MultiStepLR strategy is applied to adjust the learning rate, and data augmentation settings are in accordance with [10].
- Tiny ImageNet [31], ImageNet-100 [58], and ImageNet-1K [46] contain 100k, 100k, and 1.2M color-annotated training images with resolutions of 64×64, 224×224, 224×224 pixels, respectively. These datasets serve as standard benchmarks for evaluating image recognition models. During training, we optimized our model using a configuration consistent with [7, 51][5]. Specifically, we utilized the AdamW optimizer [37] alongside a cosine decay schedule for learning rate tuning, and applied data augmentations including label smoothing, Cut-Mix, Mix-Up, and AutoAugment. Note that we did not employ additional large-scale datasets such as JFT-300M for pre-training, nor did we use unsupervised or semi-supervised methods to optimize our model. All experiments were conducted on 8 GeForce RTX 3090 GPUs using native PyTorch.
- MNIST and Fashion-MNIST [56] both contain 60,000 grayscale images for training and 10,000 for validation, with each image sized at 28×28. We conducted comparison experiments between DTTN and other TN models on these two benchmarks, employing training configurations consistent with those used for CIFAR-10.

A.2 Hyperparameters for Training on ImageNet-1K

Table 11 shows the experimental hyperparameters used for training the DTTN family on the ImageNet-1K benchmark. We utilize the timm library[6] and ensure all settings are aligned with the comparison method MONet [7].

B More Experimental Results

Image Segmentation. We employ the Semantic FPN framework [28][7] for performing semantic segmentation on the ADE20K dataset, which includes 20,000 training images and 2,000 validation images. DTTN†-S and DTTN†-L were initialized using pre-trained ImageNet-1K weights before being integrated as the backbone of the framework. Additionally, all newly added layer weights were initialized using Xavier initialization [17]. Table 9 presents the outcomes of the DTTN model trained for 12 epochs using the AdamW optimizer, evaluated by Mean Intersection over Union (mIoU). Some experimental data are derived from [7]. Notably, DTTN†-L achieves better performance over other multilinear networks, underscoring the efficacy of our designed AIM.

Training Convergence. Figure 5 displays the Top-1 accuracy and loss curves of the proposed model alongside other architectures—such as ResNet-50 [20], ViP-Small [21] representing the MLP architecture, and MONet [7]—all trained on ImageNet-100 for 90 epochs. It can be observed that DTTN achieves significantly faster convergence and superior performance.

C Motivation and Interpretability

The principal motivation of this work is to uncover both the potential and inherent limitations of quantum-inspired tensor networks (TNs) on large-scale benchmarks, by establishing their equivalence with DTTN through Theorem 1. Although TNs have been extensively studied in the fields of quantum mechanics and white-box machine learning, state-of-the-art models are still largely restricted to small datasets and limited tasks. DTTN overcomes this barrier by extending Tensor Network Networks

[5] https://github.com/Allencheng97/Multilinear_Operator_Networks
[6] https://github.com/huggingface/pytorch-image-models
[7] https://github.com/CSAILVision/semantic-segmentation-pytorch

BackBone	mIoU(%)
Resnet18	32.9
FCN	29.3
Seg-Former	37.4
R-PDC	20.7
$\mathcal{R}$-PolyNets	19.9
MONet-S	37.5
DTTN†-S	36.9
DTTN†-L	**38.6**

Table 9: Experimental validation of semantic segmentation on ADK20K with Semantic FPN.

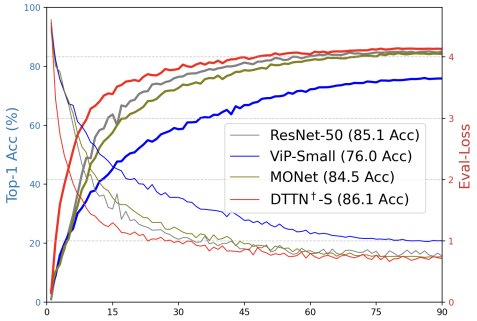


Figure 5: Top-1 accuracy and loss visualization for architectures trained from scratch on ImageNet-100.

(TNNs) to large-scale applications through techniques such as parameter sharing and exponential interaction modeling, thereby achieving competitive performance.

Furthermore, DTTNs retain a high degree of interpretability that aligns with quantum-inspired TNs, offering intuitive deterministic linear representations and probabilistic interpretations. For a deeper understanding of the theoretical underpinnings, readers are referred to related works [43, 44].

Table 10: Comparison of inference latency, throughput, and memory usage on an NVIDIA A6000 GPU.

Model	Batch Size	Latency (ms/batch)	Throughput (sample/sec)	Peak Memory (GB)
MONet_T	64	128.39	498.47	1.14
ViP-Small/14 (ours)	64	86.1	743.28	0.584
DTTN-S (ours)	64	81.94	781.08	1.099

D Latency Analysis

Table 10 presents a detailed comparison of inference latency and memory usage, benchmarked on an NVIDIA A6000 GPU. Compared to the previous state-of-the-art model, MONet-T, our DTTN-S reduces latency by nearly 35% while also achieving a 2.4% performance gain on ImageNet (see Table 1). Furthermore, DTTN demonstrates faster convergence on both the ImageNet and ImageNet-100 benchmarks, as shown by the convergence curves in Fig. 2e and Fig. 5.

Table 11: Training settings for ImageNet-1K in Section 4.1

Item	Setting
Optimizer	AdamW
Base learning rate	1e-3
Warmup-lr	1e-6
Learning rate schedule	cosine
Weight Decay	0.01 & 0.02
Batch size	320 $\times$ 4 GPU
Label smoothing	0.1
Auto augmentation	✓
Random erase	0.1
Cutmix	0.5
Mixup	0.5
Dropout	0.0

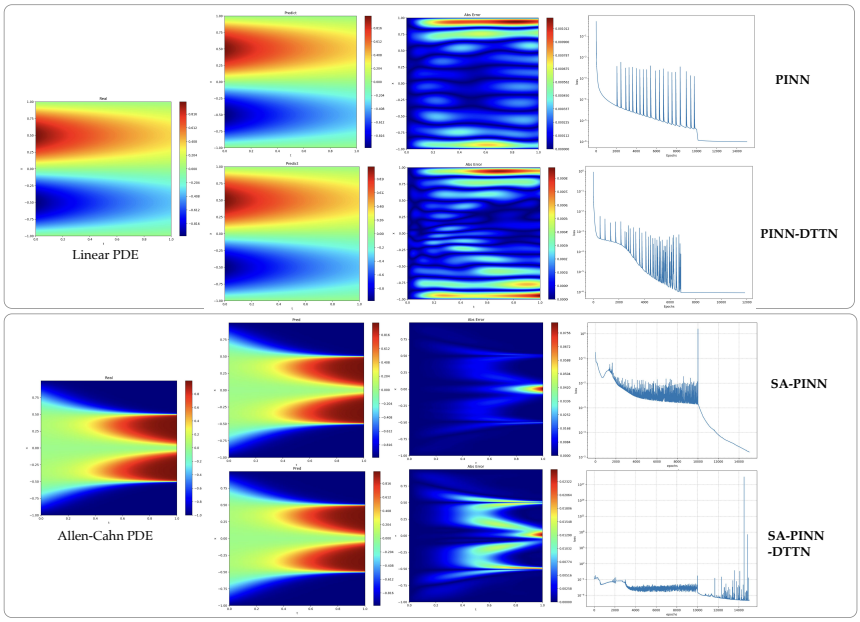


Figure 6: Visualization of training loss and prediction results of PINNs solving PDEs across the spatial-temporal domain.

E Details of DTTN Resolving PDEs

In this study, we use the Physics-Informed Neural Network (PINN) [14] and the Self-Adaptive PINN (SA-PINN) [50] as our baseline models. These are multi-layer feedforward networks with a hidden dimension of $d = 32$. Each linear layer is succeeded by a $\tanh$ activation function[8]. Subsequently, we replaced these linear layers with AIM modules while removing the activation functions, yielding the novel architectures PINN-DTTN and SA-PINN-DTTN. These enhanced networks are then applied to solve both the linear PDE and the nonlinear Allen-Cahn equation presented in Equation (8). As noted in reference [50], the Allen-Cahn PDE serves as an intriguing benchmark for testing PINNs, requiring the network to approximate solutions with sharp transitions in space and time, along with periodic boundary conditions.

Fig. 6 illustrates the prediction results and absolute errors for various PINN variants addressing the two equations across the spatial-temporal domain $\Omega \times T \to [-1, 1] \times [0, 1]$. For the first equation, which has an analytical solution of $u = e^{-t}\sin(\pi x)$, the PINN-DTTN exhibits lower errors in both training and prediction. However, for the second equation, despite the SA-PINN-DTTN achieving lower errors, there are noticeable spikes in training loss. This phenomenon may be due to instability when fitting complex functions with simple polynomials.

F Implementation and Motivation of the AIM Design

In Algorithm 1, we provide a PyTorch-style implementation of AIM. The main difference between training and inference lies in structural re-parameterization. Specifically, during training, BN layers are integrated with adjacent convolutional or linear layers. During inference, the AIM propagation process simplifies, matching the schematic shown in Fig. 2(d).

The primary motivation for designing DTTN was to develop a computationally efficient model, based solely on tensor operations, that is capable of scaling to large benchmarks. The core of our solution is the AIM, which is designed to facilitate direct multiplicative interactions. A tree structure emerges as a natural consequence of this design: since each AIM module performs a binary fusion of two feature tensors into one, stacking these modules hierarchically inherently constructs a tree. Significantly, this straightforward, hierarchical construction is provably equivalent to a powerful Tree Tensor Network

[8] https://github.com/ZzYyPp47/Attention_pinn

Algorithm 1 Code for AIM (PyTorch-like)

```
# H: height, W: width, C: channel, R: expansion ratio
# x: input tensor of shape (B, C, H, W)
############## Initialization ##############
proj_l = nn.Linear(C, C * R) # Aggregate channel information
conv_l = nn.Conv2d(C * R, C * R, kernel_size=3, groups=C * R) # Aggregate
    spatial information
proj_r = nn.Linear(C * R, C * R) # Aggregate channel information
conv_r = nn.Conv2d(C, C * R, kernel_size=3, groups=C) # Aggregate spatial
    information
proj = nn.Conv2d(C * R, C, kernel_size=1) # For information fusion
ln_norm = nn.LayerNorm([C * R]) # Layer normalization
l_norm = nn.BatchNorm2d(C * R) # Batch normalization
r_norm = nn.BatchNorm2d(C * R) # Batch normalization
res_norm = nn.BatchNorm2d(C) # Batch normalization
scale = nn.Parameter(torch.ones(1))
############## Training stage ##############
def AIM(x, use_ln):
    x_l = conv_l(proj_l(x.permute(0, 2, 3, 1).permute(0, 3, 1, 2)))
    x_r = proj_r(conv_r(x).permute(0, 2, 3, 1)).permute(0, 3, 1, 2)
    if use_ln:
        out = ln_norm(x_l * x_r)
    else:
        out = l_norm(x_l) * r_norm(x_r)
    out = res_norm(proj(out))
    return x + scale * out
############## Inference stage ##############
def AIM(x, use_ln):
    x_l = conv_l(proj_l(x.permute(0, 2, 3, 1).permute(0, 3, 1, 2)))
    x_r = proj_r(conv_r(x).permute(0, 2, 3, 1)).permute(0, 3, 1, 2)
    out = x_l * x_r
    if use_ln:
        out = ln_norm()
    return x + proj(out)
```

(TTN), bypassing the complex methods that have historically limited their application [5]. Thus, while the tree topology was not an initial design constraint, it is fundamental to our model's expressive power and theoretical grounding.

Training Stability of DTTN. The training of traditional TNs is often hindered by well-known challenges, such as gradient instability (i.e., exploding or vanishing gradients) and high sensitivity to hyperparameters. In contrast, our DTTN model trains stably across all scenarios, showing no signs of these issues, as evidenced by the smooth loss curves in Fig. 5. This stability provides a significant advantage over traditional TNs, where long chains of sequential tensor contractions can lead to optimization problems. Although DTTN is theoretically equivalent to a TTN, its hierarchical optimization process mirrors that of a standard deep network, ensuring a more robust and stable training dynamic.

G Complexity Analysis

We now analyze the complexity of DTTN in terms of its storage requirements and computational cost. Without loss of generality, we consider the multi-stage DTTN with hidden-size d for each stage, and the total depth is L. We then consider the complexity of the main components of the network, including the local mapping, the core blocks, and the classification head separately.

In this section, we analyze the complexity of a DTTN, focusing on both storage requirements and computational cost. Without loss of generality, we consider a multi-stage DTTN where each stage has a hidden size of d, and the overall depth is L. We then examine the complexity associated with the main components of the network: the local mapping function, the core blocks, and the classification head.

Local Mapping. The local mapping function $\phi(\cdot)$ consists of two consecutive convolutional layers, each with a kernel size and stride of 2. This function is applied to the input image $\boldsymbol{x}$ to produce a feature map $\phi(\boldsymbol{x}, \boldsymbol{\Lambda_\phi}) \in \mathbb{R}^{W\times H\times C}$. Here, $\boldsymbol{\Lambda}_\phi \in \mathbb{R}^{S^2\times C}$ represents a learnable matrix, where S,$C = d \in \mathbb{N}$ denote the local patch size and the number of output channels, respectively. The parameters and floating-point operations (FLOPs) involved in this process are given by:

$$\begin{aligned} Params^L &= \mathcal{O}(14 \cdot d + 4 \cdot d^2), \\ FLOPs^L &= \mathcal{O}(48 \cdot d \cdot WH + 4 \cdot d^2 \cdot WH). \end{aligned} \tag{8}$$

Core Blocks. we analyze the complexity associated with the core AIM blocks within the DTTN architecture. Assuming that each of the four stages contains an equal number of blocks, the corresponding feature map sizes are $W \times H$,$\frac{W}{2} \times \frac{H}{2}$, $\frac{W}{4} \times \frac{H}{4}$, and $\frac{W}{8} \times \frac{H}{8}$, respectively. Our analysis focuses on the parameters and FLOPs involved in the three linear layers and two convolutional layers of AIM. The total parameters and FLOPs can be expressed as follows:

$$\begin{aligned} Params^{AIM} =& \mathcal{O}\left((\sum_{s=0}^{3} 22 \cdot r_{exp} \cdot d + (2 \cdot r_{exp} + r_{exp}^2) \cdot d^2 + d) \cdot \frac{L}{4}\right) \\ =& \mathcal{O}\left((22 \cdot r_{exp} + 2 \cdot (r_{exp} + r_{exp}^2) \cdot d + 1) \cdot d \cdot L\right), \\ FLOPs^{AIM} =& \mathcal{O}\left((\sum_{s=0}^{3} 18 \cdot r_{exp} \cdot d \cdot \frac{WH}{4^s} + (2 \cdot r_{exp} + r_{exp}^2) \cdot d^2 \cdot \frac{WH}{4^s}) \cdot \frac{L}{4}\right) \\ =& \mathcal{O}\left((\frac{765}{128} \cdot r_{exp} + \frac{85}{128} \cdot (r_{exp} + r_{exp}^2) \cdot d) \cdot d \cdot WH \cdot L\right). \end{aligned} \tag{9}$$

Classification Head. For a classification head with m classes, we receive a feature map of size $\frac{W}{8} \times \frac{H}{8} \times d$, which is then processed through an average pooling layer followed by a fully-connected layer to output an m-dimensional vector. The parameters and FLOPs involved in this process are given by:

$$\begin{aligned} Params^H &= \mathcal{O}(d \cdot (m + 1)), \\ FLOPs^H &= \mathcal{O}(\frac{WH}{64} \cdot d + m \cdot d). \end{aligned} \tag{10}$$

In summary, our conclusions regarding the number of parameters and FLOPs for the DTTN architecture can be expressed as:

$$\begin{aligned} Params &= Params^L + Params^{AIM} + Params^H, \\ FLOPs &= FLOPs^L + FLOPs^{AIM} + FLOPs^H. \end{aligned} \tag{11}$$

It can be observed that the number of parameters in the DTTN architecture remains fixed, while the computational complexity exhibits a linear relationship with the input image scale. This characteristic represents a significant advantage of the DTTN over modern MLP and Transformer architectures, which typically exhibit quadratic complexity.

H Proofs

Here, we derive the proofs of Proposition 1 and Theorem 1 from the main paper and further elucidate their significance.

Proposition. *The DTTN has the capability to capture 2^L multiplicative interactions among input elements, which can be represented in the format of Equation 1 as $\Phi(\boldsymbol{x}) = \otimes^{2^L}\phi(\boldsymbol{x}, \boldsymbol{\Lambda}_\phi)$. Consequently, the elements of $f(\boldsymbol{x})$ are homogeneous polynomials of degree 2^L over the feature map $\phi(\boldsymbol{x}, \boldsymbol{\Lambda}_\phi)$.*

Proof. For each AIM block with input $\boldsymbol{x}^l = vec(\boldsymbol{\mathcal{X}}^l) \in \mathbb{R}^{W_l H_l C_l}$, let $D^l = W_l \times H_l \times C_l$ and $l \in \mathbb{K}_L$. Suppose that the left and right branches of AIM can be represented as $f_1^l(\boldsymbol{x}^l) = \boldsymbol{A}_1^l \boldsymbol{x}^l$ and $f_2^l(\boldsymbol{x}^l) = \boldsymbol{A}_2^l \boldsymbol{x}^l$, where $\boldsymbol{A}_1^l, \boldsymbol{A}_2^l \in \mathbb{R}^{D^l \times D^l}$ are obtained through structured combinations of convolutional and linear layer weights. The feedforward propagation of the AIM block can then be

expressed as:

$$\begin{aligned}\boldsymbol{x}^{l+1} &= \boldsymbol{x}^l + \boldsymbol{B}^l \left((\boldsymbol{A}_1^l \boldsymbol{x}^l) * (\boldsymbol{A}_2^l \boldsymbol{x}^l) \right) \\ &= \boldsymbol{x}^l + \texttt{Reshape}\left(\boldsymbol{B}^l ({\boldsymbol{A}_1^l}^T \odot {\boldsymbol{A}_2^l}^T)^T \right) \times_{2,3}^{1,2} (\boldsymbol{x}^l \otimes \boldsymbol{x}^l) \\ &= \boldsymbol{x}^l + \boldsymbol{\mathcal{Z}}^l \times_{2,3}^{1,2} (\boldsymbol{x}^l \otimes \boldsymbol{x}^l).\end{aligned} \tag{12}$$

where $\boldsymbol{B}^l$ denotes the fused linear layer inside AIM, and $\boldsymbol{\mathcal{Z}}^l = \texttt{Reshape}\left(\boldsymbol{B}^l ({\boldsymbol{A}_1^l}^T \odot {\boldsymbol{A}_2^l}^T)^T \right) \in \mathbb{R}^{D^l \times D^l \times D^l}$ is a structured learnable tensor. Then each element of $\boldsymbol{x}^{l+1}$ can be calculated by

$$\boldsymbol{x}_\tau^{l+1} = \boldsymbol{x}_\tau^l + \sum_w^{D^l} \sum_\rho^{D^l} \boldsymbol{\mathcal{Z}}_{(w,\rho,\tau)}^l \boldsymbol{x}_w^l \boldsymbol{x}_\rho^l. \tag{13}$$

Thus, each AIM module captures second-order multiplicative feature interactions of the input. By induction, the DTTN stacked with L AIMs captures 2^L interactions of the input $\boldsymbol{x}$. Note that the network's bias terms and shortcut connections can be eliminated by introducing an additional homogeneous dimension in the local mapping. Hence, we have $\boldsymbol{x}_\tau^{l+1} = \sum_w^{D^l+1} \sum_\rho^{D^l+1} \boldsymbol{\mathcal{Z}}_{(w,\rho,\tau)}^{*l} \boldsymbol{x}_w^l \boldsymbol{x}_\rho^l \in \mathbb{R}^{D_l+1}$. Therefore, the expression $f(\boldsymbol{x})$ is a homogeneous polynomial of degree 2^L of $\phi(\boldsymbol{x}, \boldsymbol{\Lambda}_\phi)$, which concludes our proof. □

Theorem. *Given the local mapping function $\phi^{i_1}(x_1) = [x_1^0, \cdots, x_1^{2^L}]^T$, a polynomial network with the expansion form of Equation (6) can be transformed into a quantum-inspired TN model with finite bond dimension.*

Proof. For an input image $x \in \mathbb{R}^{W \times H \times C}$, the computation in the vanilla quantum-inspired TN model can be expressed as

$$f(\boldsymbol{x}) = TN\left(\{\phi^i(\boldsymbol{x}_{(\tau,\rho,w)})\}_{i=1}^N, \{\boldsymbol{\mathcal{R}}_i\}_{i=1}^N \right), \tag{14}$$

where $N = WHC$,$\tau \in \mathbb{K}_W$,$\rho \in \mathbb{K}_H$,$w \in \mathbb{K}_C$, and $\{\mathcal{R}_\rangle\}_{i=1}^N\}$ denotes the tensor cores. $TN(\cdot) : \mathbb{R}^{\underbrace{(2^L+1)\cdots(2^L+1)}_{N}} \to \mathbb{R}^m$ represents the contraction operation that outputs an m-dimensional vector. We can further complete the contraction of 2^L physical indices and transform Equation 14 into

$$\begin{aligned}f(\boldsymbol{x}) &= TN\left(\phi^1(\boldsymbol{x}_{(\tau,\rho,w)}) \times_1^1 \boldsymbol{\mathcal{R}}_1, \cdots, \phi^N(\boldsymbol{x}_{(\tau,\rho,w)}) \times_1^1 \boldsymbol{\mathcal{R}}_N \right) \\ &= TN(\boldsymbol{\mathcal{Z}}_1, \cdots, \boldsymbol{\mathcal{Z}}_N).\end{aligned} \tag{15}$$

Since each output element of a polynomial network can be expressed as

$$\sum_{i_1}^N \cdots \sum_{i_{2^L}}^N w_\xi x_{i_1} x_{i_2} \cdots x_{i_{2^L}}, \;\; \xi \leq N^{2^L}. \tag{16}$$

The above equation is equivalent to

$$\sum_\xi w_\xi x_1^{k_1} \cdots x_N^{k_N} \quad where \;\; k_i \geq 0 \;\; and \;\; k_1 + \cdots + k_N = 2^L \tag{17}$$

Since Equation 15 encompasses each term of Equation 17, this proves our claim. □

The above proof indicates that by selecting an appropriate local mapping, the DTTN can be transformed into a standard tensor network model. We re-emphasize that this is the first work demonstrating that tensor networks can achieve competitive performance on large-scale benchmarks, such as attaining 77.2% Top-1 accuracy on ImageNet-1K. Previous TNs have been limited to much smaller tasks due to expression limitations and lower bond dimensions. We hope that this research will inspire further explorations into tensor networks.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The contributions and scope of this paper are summarized in the abstract and detailed in the introduction. Specifically, the scope of the study is outlined in the first two paragraphs of the introduction, while the contributions are highlighted in the final paragraph.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of this work in the last paragraph of Section 4.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Detailed proofs of the propositions and theorems presented in this paper are provided in Appendix H.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe the details of the experiments in the experimental section and the Appendix. We plan to make the code publicly available to ease the reproducibility.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All experimental datasets used are publicly available. The pseudocode for the AIM module is provided in Appendix F.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide detailed experimental settings, including data split, optimizer configurations, and other parameters, in Section 4 (Experimental Section) of the main paper. Additional implementation details are provided in Appendix A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Given the substantial computational resources and time required, a statistical analysis is not conducted.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We describe the software and hardware platforms for the experiments in the Experiments section, with specific resource consumption consistent with regular deep model training.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have made sure that our paper conforms with the NeurIPS Code of Ethics

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the positive impacts of our approach in the Introduction and Experimental sections.

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: There is no risk of misuse of the proposed method and the datasets used in the paper.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited the original paper or attached the link to the existing assets used in this paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.