Network compression using tensor decompositions and pruning

Yassine Zniyed
LIS UMR 7020, Université de Toulon, Aix-Marseille Université, CNRS

joint work with Van Tien Pham, and Thanh Phuong Nguyen

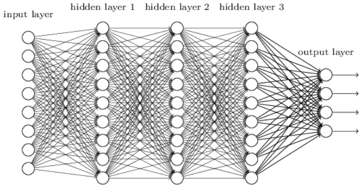
27.11.2024
Workshop on Low-rank Approximations and their Interactions with Neural Networks

LABORATOIRE D'INFORMATIQUE & DES SYSTÈMES

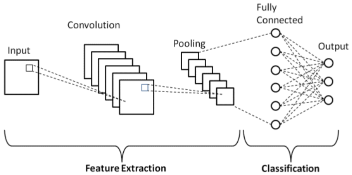
UNIVERSITÉ DE TOULON

Table of Contents

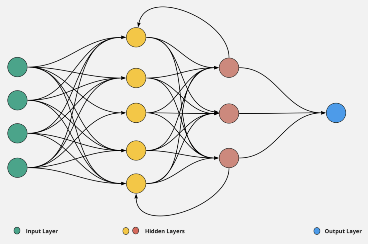
Examples of Neural Network Architectures



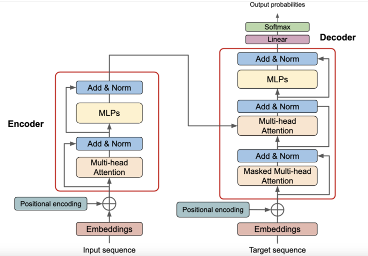
Fully-Connected Network (a.k.a MLP)



Convolutional Neural Network (CNN)



Recurrent Neural Network (RNN)



Transformer

Overparameterization in Modern DNNs

- Modern DNNs are often overparameterized to ensure sufficient capacity for learning complex patterns.
- This results in redundancy and inefficiency, making them resource-intensive.

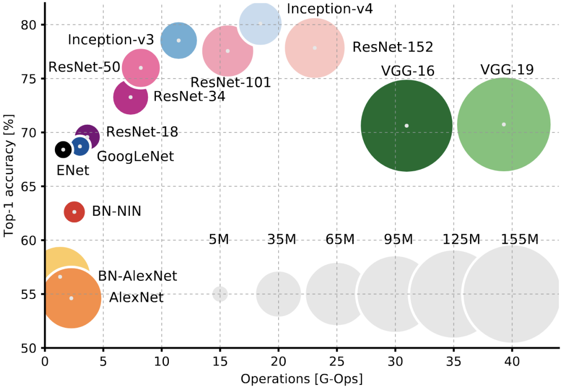
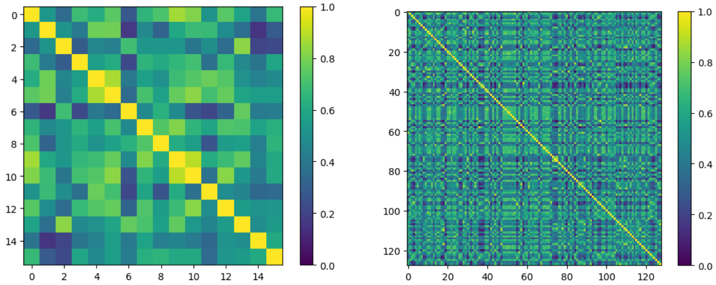


Figure : Top-1 accuracy on the ImageNet dataset vs. number of operations required for a single forward pass [*Canziani et al., 2016*].

From Overparameterization to Compression

- Modern DNNs often exhibit significant **redundancy** :
  - Many learned features across architectures (e.g., CNNs, Transformers) are overlapping or similar.
  - Weight matrices and kernels often exhibit **low-rank structures**.
- Addressing this redundancy through **compression** :
  - Reduces storage and computational requirements.
  - Facilitates deployment on resource-constrained devices.
  - Improves energy efficiency and inference speed.



Layer 1, ResNet-56 on CIFAR-10 — Layer 12, ResNet-50 on ImageNet

Figure : Similarity matrices (cosine distance) showing redundancy in filters

Compression Techniques for Neural Networks

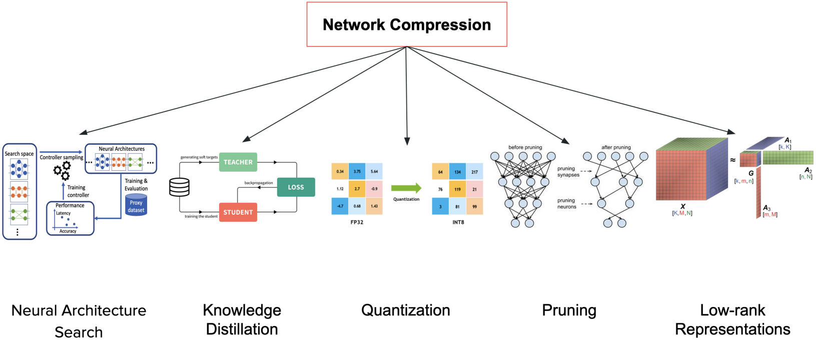


Figure : Overview of key neural network compression techniques.

Taxonomy of DNN Pruning

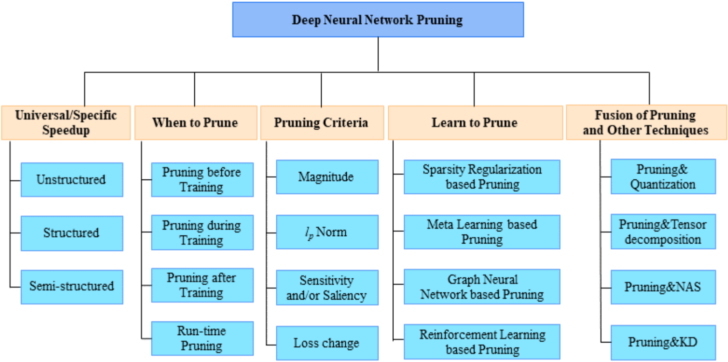
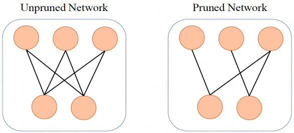
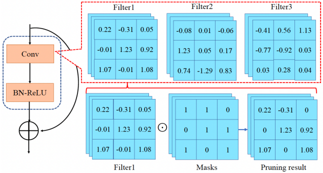


Figure : Taxonomy of pruning techniques [*Chang et al., 2024*].

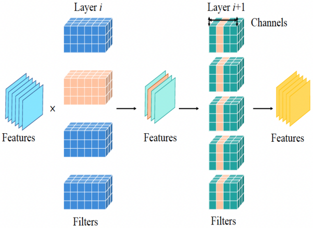
Structured vs. Unstructured Pruning



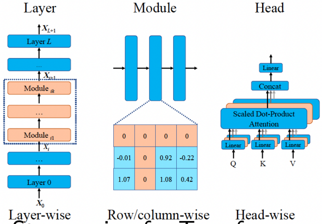
Unstr. pruning for neurons and connections



Unstr. pruning for weights and masks



Str. pruning for CNNs



Str. pruning for Transformers

Figure : Visualization of structured vs. unstructured pruning.

When to Prune ?

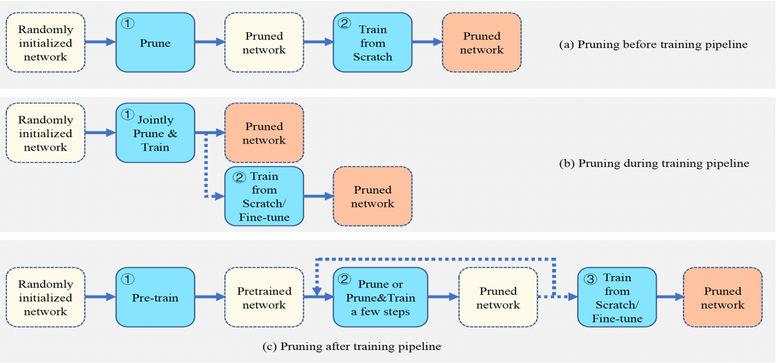


Figure : Typical pipelines of static pruning.

Weight Matrix Decomposition with SVD

- One common case of low-rank approximation involves decomposing **matrix weights** in DNNs using matrix decompositions, such as Singular Value Decomposition (SVD).
- This approach is widely used in architectures like Transformers and LLMs to reduce the dimensionality of matrix weights.

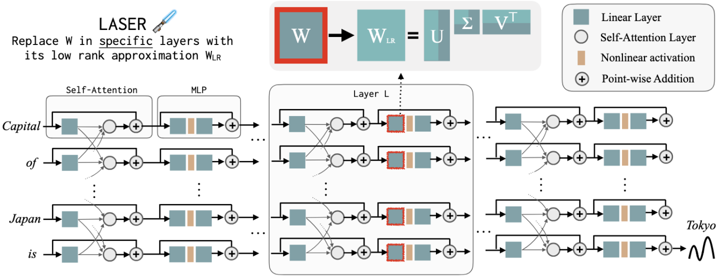
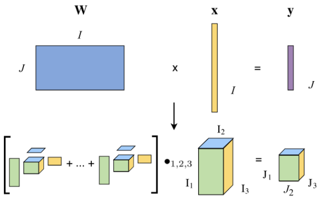


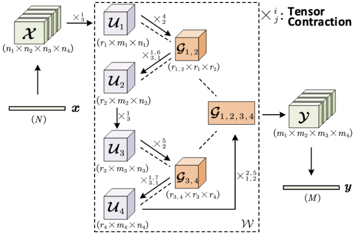
Figure : Low-rank approximation of matrix weights [*Sharma et al., 2023*].

Tensorization of Weight Matrices

- Weight matrices in neural networks can be tensorized to enable efficient computations and decompositions.
- Example : The matrix-vector product can be performed in a tensorial format using :
  - A Block Term Decomposition (BTD) format.
  - A Hierarchical Tucker (HT) network structure.



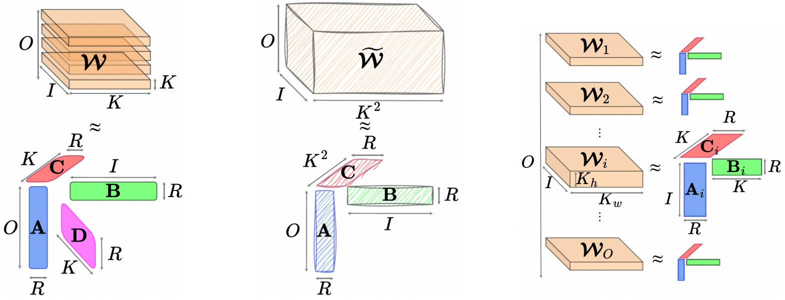
BTD format [(J. Ye et al, 2018)].



HT network [(Yin et al, 2021)].

Weight Tensor Decomposition

- Some works use **SVD** by unfolding weight tensors into matrices.
- Other works directly decompose weight tensors using tensor decomposition techniques, as illustrated below :



Layer decomp. [*Lebedev et al., 2015*] Reshaped layer decomp. [*Phan et al., 2024*] Filters decomp. [*Pham et al., 2024*]

Figure : Exemples of CPD-based approaches for convolution layer decomposition.

NORTON Approach : A Hybrid Compression Method

- NORTON : **N**etwork c**O**mp**R**ession through **T**ens**O**r decompositions and pru**N**ing.
- A hybrid method for CNN compression, combining :
  - **CP decomposition** to reduce dimensionality.
  - **Pruning techniques** to eliminate redundant filters.

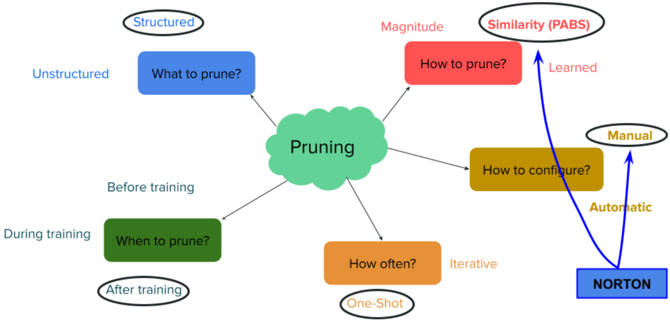


Figure : Pruning combination used in the NORTON approach.

CP Decomposition for a Single Filter

- The CPD expresses a tensor as the sum of rank-one tensors.
- For a single filter, the CP decomposition is illustrated as :

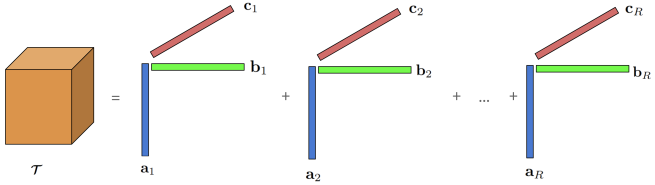


Figure : CP decomposition of a single filter into rank-one components.

- The CPD is applied to **all filters** in each **convolutional layer**.
- Compact representation of the CPD :

$$\mathcal{T} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$$

Decomposition Then Pruning Process

- NORTON starts with the CP decomposition of all filters in the convolutional layers.
- Using a CPD-based similarity, pruning is applied to remove similar filters.

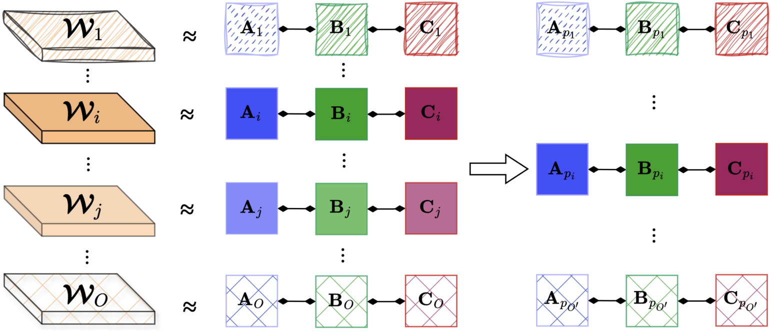


Figure : Decomposition of filters followed by pruning.

CPD-based similarity and similarity matrix

- Due to the ambiguities of the CPD, the factor matrices of two CPDs of the same tensor are not guaranteed to be identical.
- Let $\phi(.,.)$ be a function that computes the PABS (Principal Angles Between Subspaces) between two factor matrices. If the CPD is unique :

$$\begin{cases} \mathcal{W}_i = & [\![\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i]\!], \\ \mathcal{W}_j = & [\![\mathbf{A}_j, \mathbf{B}_j, \mathbf{C}_j]\!], \\ \mathcal{W}_i = & \mathcal{W}_j. \end{cases} \Rightarrow \begin{cases} \phi(\mathbf{A}_i, \mathbf{A}_j) = 0, \\ \phi(\mathbf{B}_i, \mathbf{B}_j) = 0, \\ \phi(\mathbf{C}_i, \mathbf{C}_j) = 0. \end{cases}$$

- Even in non-unique cases, PABS is effective in identifying redundancies.
- A distance matrix $\mathbf{D}$ is computed as :

$$\mathbf{D}_{ij} = \alpha \mathbf{D}_{ij}^{\mathbf{A}} + \beta \mathbf{D}_{ij}^{\mathbf{B}} + \gamma \mathbf{D}_{ij}^{\mathbf{C}},$$

where $\mathbf{D}_{ij}^{\mathbf{A}} = \phi(\mathbf{A}_i, \mathbf{A}_j)$ (similarly for $\mathbf{D}_{ij}^{\mathbf{B}}$ and $\mathbf{D}_{ij}^{\mathbf{C}}$), and α, β, and γ are weight parameters such that $\alpha + \beta + \gamma = 1$.
- A straightforward algorithm is used to iteratively eliminate the similar filters.

Convolution Under CPD Format

- **Original convolution :**

$$\mathcal{O}_k(i,j) = \sum_{m=0}^{K_h-1} \sum_{n=0}^{K_w-1} \sum_{p=0}^{I-1} \mathcal{I}(i+m, j+n, p) \cdot \mathcal{W}_k(m,n,p)$$

- **CPD of the weight tensor :**

$$\mathcal{W}_k(m,n,p) = \sum_{r=0}^{R-1} \mathbf{A}_k(m,r) \cdot \mathbf{B}_k(n,r) \cdot \mathbf{C}_k(p,r)$$

- **Convolution under CPD :**

$$\mathcal{O}_k(i,j) = \sum_{r=0}^{R-1} \underbrace{\sum_{m=0}^{K_h-1} \overbrace{\sum_{n=0}^{K_w-1} \underbrace{\sum_{p=0}^{I-1} \mathcal{I}(i+m, j+n, p) \cdot \mathbf{C}_k(p,r)}_{\mathcal{O}_k^{\mathbf{C}}(i+m,j+n,r)} \cdot \mathbf{B}_k(n,r)}^{\mathcal{O}_k^{\mathbf{B}}(i+m,j,r)} \cdot \mathbf{A}_k(m,r)}_{\mathcal{O}_k^{\mathbf{A}}(i,j,r)}$$

Implementation of CPD-based Convolution Layer

- The figure illustrates the convolution layer for an entire batch, denoted by B.
- The structure can be efficiently implemented using classical deep learning frameworks (*e.g.*, PyTorch, TensorFlow).

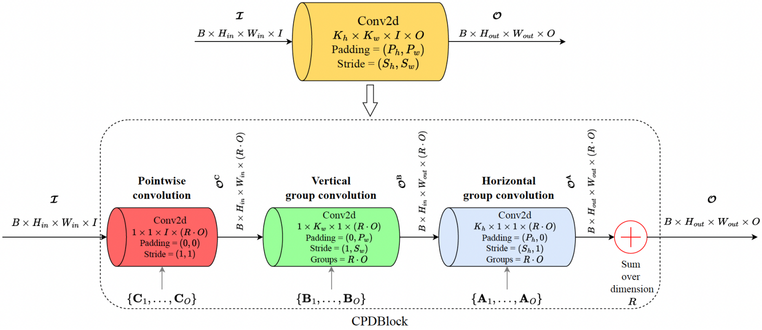


Figure : CPD-based convolution layer performing the operation for a batch of size B.

Illustration of the full NORTON Approach

- The process involves three main steps :
  - **Filter decomposition :** Filters in each convolution layer are decomposed using CPD.
  - **Filter pruning :** Similar filters are removed using a CPD-based similarity.
  - **Fine-tuning :** The pruned model is fine-tuned to restore performance.
- The result is a compact model with reduced parameters and computational cost.

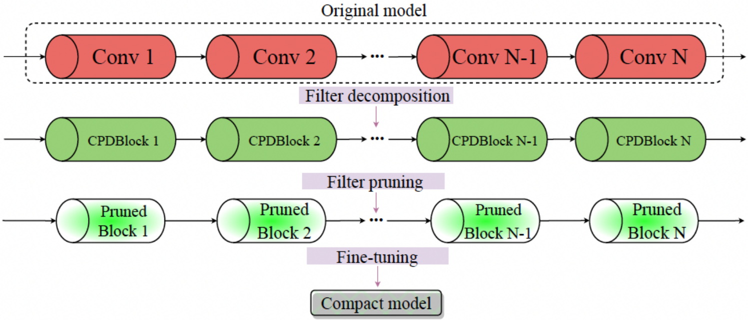


Figure : Overview of the NORTON approach applied to a CNN with N layers.

Compression Results

Method	Type	Top-1	MACs (CR)	Params (CR)
VGG-16-BN		93.96	313.73M (00)	14.98M (00)
DECORE-500 [15]	P	94.02	203.08M (35)	5.54M (63)
RGP-64_16 [7]	P	92.76	78.78M (75)	3.81M (75)
NORTON (Ours)	H	**94.11**	**74.14M (77)**	**3.60M (76)**
ALDS [10]	D	92.67	43.33M (86)	0.63M (96)
Dai *et al.* [4]	H	93.03	37.76M (87)	0.43M (97)
Lebedev *et al.* [1]	D	93.07	68.53M (78)	3.22M (78)
HALOC [8]	D	93.16	43.92M (86)	0.30M (98)
EDP [6]	H	93.52	62.40M (80)	0.66M (96)
CORING [12]	P	93.54	66.95M (79)	1.90M (87)
NORTON (Ours)	H	**93.84**	**37.68M (88)**	1.94M (87)
RGP-64_6 [7]	P	91.45	31.37M (90)	1.43M (90)
DECORE-50 [15]	P	91.68	36.85M (88)	0.26M (98)
NORTON (Ours)	H	**92.54**	**13.54M (96)**	**0.24M (98)**
NORTON (Ours)	H	**90.32**	**4.58M (99)**	**0.14M (99)**

Table 1 : VGG-16-BN on CIFAR-10

Method	Type	Top-1	Top-5	MACs (CR)	Params (CR)
ResNet-50		76.15	92.87	4.09G (00)	25.50M (00)
Kim *et al.* [9]	D	75.34	92.68	N/A	17.60M (31)
DECORE-8 [15]	P	76.31	93.02	3.54G (13)	22.69M (11)
Hinge [13]	H	74.70	N/A	2.17G (47)	N/A
NORTON (Ours)	H	**76.58**	**93.43**	**2.08G (50)**	**13.51M (47)**
CC-0.6 [5]	H	74.54	92.25	1.53G (63)	10.58M (59)
RGP-64_30 [7]	P	74.58	92.09	1.92G (53)	11.99M (53)
Phan *et al.* [2]	D	74.68	92.16	1.56G (62)	N/A
EDP [6]	H	75.34	92.43	1.92G (53)	14.28M (44)
CORING [3]	P	75.55	92.61	1.50B(64)	11.04M(57)
NORTON (Ours)	H	**75.95**	**92.91**	**1.49G (64)**	**10.52M (59)**
DECORE-5 [15]	P	72.06	90.82	1.60G (61)	8.87M (65)
RGP-64_16 [7]	P	72.68	91.06	1.02G (75)	6.38M (75)
NORTON (Ours)	H	**73.65**	**91.64**	**0.92G (78)**	**5.88M (77)**

Table 2 : ResNet-50 on ImageNet

NORTON's Efficacy in Downstream Tasks

- **FasterRCNN** : Object detection
- **MaskRCNN** : Instance segmentation
- **KeypointRCNN** : Human keypoint detection

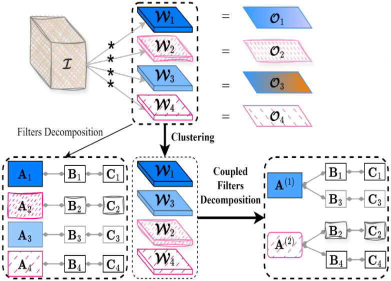
Model	$AP^{0.5:0.95}$	$AP^{0.5}$	$AP^{0.75}$	AR^{1}	AR^{10}	AR^{100}	MACs (CR)	Params (CR)	FPS	Latency(ms)
FasterRCNN [64], [79]	0.37	0.58	0.39	0.31	0.48	0.51	134.85G (00)	41.81M (00)	12	85
NORTON (Ours)	**0.38**	**0.59**	**0.42**	**0.32**	**0.50**	**0.52**	111.47G (17)	30.72M (27)	19	53
NORTON (Ours)	0.32	0.52	0.34	0.29	0.46	0.48	**93.39G (31)**	**22.01M (47)**	**25**	**41**
MaskRCNN [65], [79]	0.34	0.55	0.36	0.29	0.45	0.47	134.85G (00)	44.46M (00)	9	111
NORTON (Ours)	**0.35**	**0.57**	**0.37**	**0.30**	**0.46**	**0.48**	111.47G (17)	33.36M (25)	14	73
NORTON (Ours)	0.32	0.52	0.33	0.28	0.44	0.46	**93.39G (31)**	**24.65M (45)**	**20**	**50**
				$AR^{0.5:0.95}$	$AR^{0.5}$	$AR^{0.75}$				
KeypointRCNN [65], [79]	0.65	0.86	0.71	0.71	0.90	0.77	137.42G (00)	59.19M (00)	8	125
NORTON (Ours)	**0.65**	**0.86**	**0.71**	**0.72**	**0.91**	**0.77**	114.04G (17)	48.10M (19)	13	76
NORTON (Ours)	0.63	0.85	0.69	0.69	0.90	0.75	**95.97G (30)**	**39.39M (34)**	**17**	**59**

Table : Performance of NORTON's compressed ResNet-50/ImageNet as backbone on COCO2017.

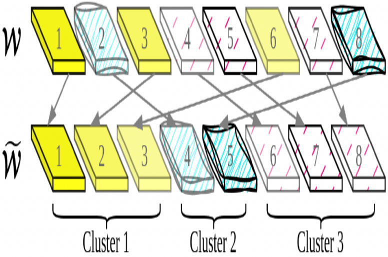
Demo

CONCATENATION Approach (in brief)

- **C**oupled tensor decompositi**ON** for **C**omp**A**ct ne**T**work repres**EN**t**ATION**.
- An ongoing work that uses CPD in a coupled manner instead of combining pruning and tensor decomposition.



Coupled decomposition approach.



Clustering of the filters.

Implementation of CONCATENATION

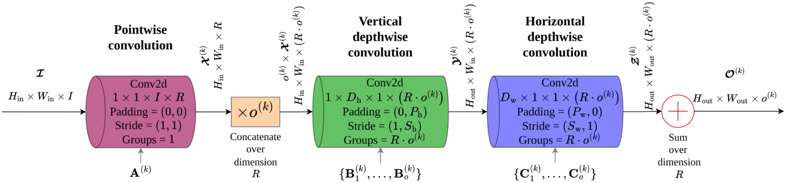


Figure : CONCATENATION implementation for convolutional layers.

Preliminary Results

Method	Type	Top-1	MACs (CR)	Params (CR)
DenseNet-40 [35]		94.81	290.14M (00)	1.06M (00)
LCT [46]	TKD	94.14	N/A	0.58M (45)
HT-2 [44]	TKD	94.51	161.19M (44)	0.50M (52)
Hinge [54]	D+P+K	94.67	161.32M (44)	0.77M (28)
NORTON [19]	CPD+P	94.67	168.23M (42)	0.58M (45)
CC [52]	SVD+P	94.67	155.19M (47)	0.51M (52)
CORING [30]	SVD+P	94.71	173.39M (40)	0.62M (41)
CEPD [16]	TTD+P	94.79	145.53M (50)	0.50M (53)
CCPD (Ours)	CPD	**94.85**	**141.22M (51)**	**0.46M (57)**
HT-2 [44]	TKD	94.21	120.89M (58)	0.41M (62)
CC [52]	SVD+P	94.40	115.95M (60)	0.38M (64)
CEPD [16]	TTD+P	94.55	110.97M (62)	0.37M (65)
CCPD (Ours)	CPD	**94.61**	**110.26M (62)**	**0.34M (68)**

Table : DenseNet-40 on CIFAR-10 using CONCATENATION.

Thank You !

References :

- V. T. Pham, Y. Zniyed, and T. P. Nguyen. "Enhanced Network Compression Through Tensor Decompositions and Pruning". *IEEE Transactions on Neural Networks and Learning Systems*, 2024, pp. 1-13.
- V. T. Pham, Y. Zniyed, and T. P. Nguyen. "Efficient tensor decomposition-based filter pruning". *Neural Networks*, 2024, 106393.

GitHub repository.