Tensorial Neural Networks: Generalization of Neural Networks and Applications to Model Compression

Furong Huang

University of Maryland

furongh@cs.umd.edu

Joint work with Jiahao Su, Jingling Li and Bobby Bhattacharjee

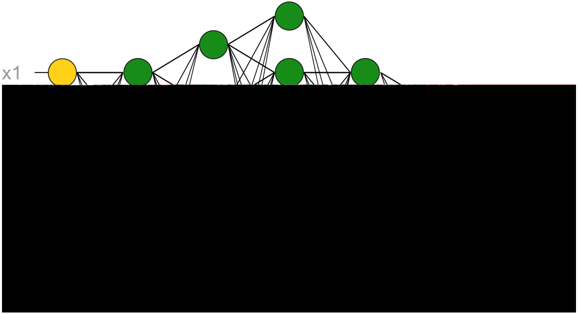
Neural Network - Nonlinear Function Approximation

Image classification

Speech recognition

Text processing

Success of Deep Neural Networks



- computation power growth
- enormous labeled data

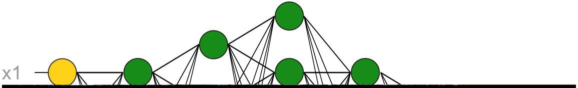
Neural Network - Nonlinear Function Approximation

Image classification

Speech recognition

Text processing

Success of Deep Neural Networks



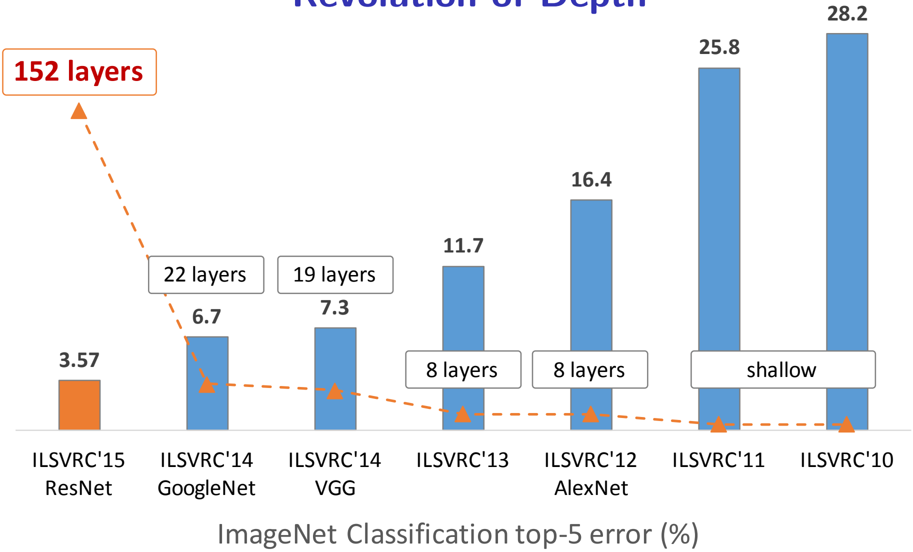
[redacted]

- computation power growth
- enormous labeled data

Express Power

- linear composition vs nonlinear composition
- shallow network vs deep structure

Revolution of Depth



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

- Deeper and wider architectures
- Large number of model parameters

Challenges For Large Deep Neural Network

Learning

- Learning takes longer, might not converge, susceptible to vanishing/exploding gradients, etc
- One-time cost.

Challenges For Large Deep Neural Network

Learning

- Learning takes longer, might not converge, susceptible to vanishing/exploding gradients, etc
- One-time cost.

Test

- Requires large amount of computation and memory storage.

  Hard to deploy on constrained devices such as smart phones or IoT device.
- Repeated high-cost in predictions.

Challenges For Large Deep Neural Network

Learning

- Learning takes longer, might not converge, susceptible to vanishing/exploding gradients, etc
- One-time cost.

Test

- Requires large amount of computation and memory storage.

  Hard to deploy on constrained devices such as smart phones or IoT device.
- Repeated high-cost in predictions.

How to design NN models with compact architectures,
but similar expressive power as large models?

How to design NN models with compact architectures, but similar expressive power as large models?

Popular Approaches

- Compress well-trained neural networks
- Find better neural network architectures

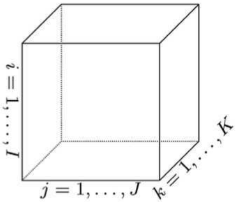
Tensorial Neural Networks (TNNs)

- Achieve both compression and better architecture
- Generalize $\left\{ \begin{array}{l} \text{matrix-vector product} \\ \text{convolution} \end{array} \right.$ to **general tensor operations**
- New tensor algebra: extend existing operations with low order operands to those with high order operands

What is a tensor? How to denote tensors effectively?

Multi-dimensional Array

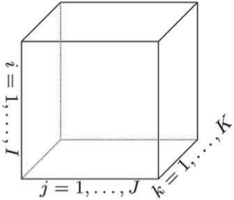
- Tensor - Higher order matrix
- The number of dimensions is called tensor order.



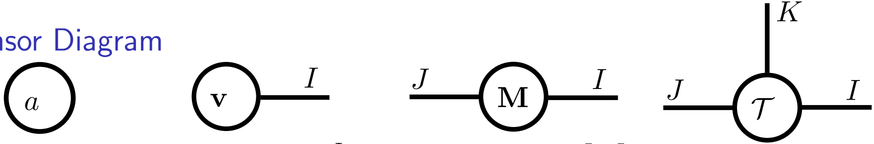
What is a tensor? How to denote tensors effectively?

Multi-dimensional Array

- Tensor - Higher order matrix
- The number of dimensions is called tensor order.

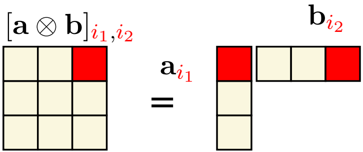


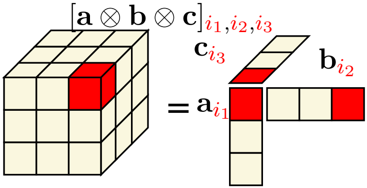
Tensor Diagram



Scalar $a \in \mathbb{R}$ | Vector $\mathbf{v} \in \mathbb{R}^I$ | Matrix $\mathbf{M} \in \mathbb{R}^{I \times J}$ | Tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$

Tensor Product





- $[\mathbf{a} \otimes \mathbf{b}]_{i_1,i_2} = \mathbf{a}_{i_1} \mathbf{b}_{i_2}$
- Rank-1 matrix

- $[\mathbf{a} \otimes \mathbf{b} \otimes \mathbf{c}]_{i_1,i_2,i_3} = \mathbf{a}_{i_1} \mathbf{b}_{i_2} \mathbf{c}_{i_3}$
- Rank-1 tensor

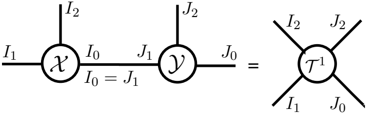
- Existing tensor operations are only defined on lower-order $\mathcal{X}$ and $\mathcal{Y}$ such as matrices and vectors.

Generalized Tensor Operations

Generalized tensor operations on *High-order tensor operands*

Mode-(0,1) tensor contraction

$\mathcal{X} \times_1^0 \mathcal{Y} \rightarrow \mathcal{T}^1$



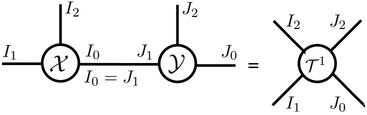
$$\mathcal{T}^1_{i_1,i_2,j_0,j_2} = \sum_r \mathcal{X}_{r,i_1,i_2}\mathcal{Y}_{j_0,r,j_2}$$

Generalized Tensor Operations

Generalized tensor operations on *High-order tensor operands*

Mode-(0,1) tensor contraction

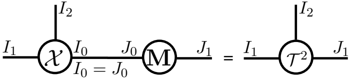
$\mathcal{X} \times_1^0 \mathcal{Y} \to \mathcal{T}^1$



$\mathcal{T}^1_{i_1,i_2,j_0,j_2} = \sum_r \mathcal{X}_{r,i_1,i_2} \mathcal{Y}_{j_0,r,j_2}$

Mode-0 tensor product

$\mathcal{X} \times_0 \mathbf{M} \to \mathcal{T}^2$



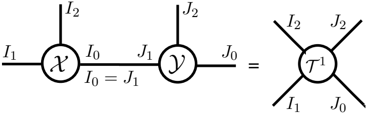
$\mathcal{T}^2_{j_0,i_1,i_2} = \sum_r \mathcal{X}_{r,i_1,i_2} \mathbf{M}_{j_0,r}$

Generalized Tensor Operations

Generalized tensor operations on *High-order tensor operands*

Mode-(0,1) tensor contraction

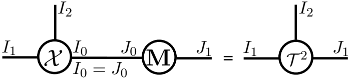
$\mathcal{X} \times_1^0 \mathcal{Y} \to \mathcal{T}^1$



$\mathcal{T}^1_{i_1,i_2,j_0,j_2} = \sum_r \mathcal{X}_{r,i_1,i_2}\mathcal{Y}_{j_0,r,j_2}$

Mode-0 tensor product

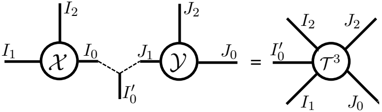
$\mathcal{X} \times_0 \mathbf{M} \to \mathcal{T}^2$



$\mathcal{T}^2_{j_0,i_1,i_2} = \sum_r \mathcal{X}_{r,i_1,i_2}\mathbf{M}_{j_0,r}$

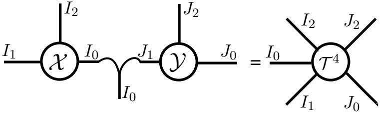
Mode-(0,1) tensor convolution

$\mathcal{X} *_1^0 \mathcal{Y} \to \mathcal{T}^3$



$\mathcal{T}^3_{:,i_1,i_2,j_0,j_2} = \mathcal{X}_{:,i_1,i_2} * \mathcal{Y}_{j_0,:,j_2}$

Mode-(0,1) tensor partial outer product $\mathcal{X} \otimes_1^0 \mathcal{Y} \to \mathcal{T}^4$

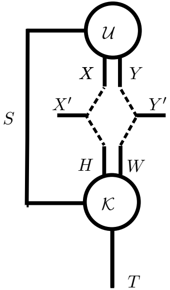


$\mathcal{T}^4_{r,i_1,i_2,j_0,j_2} = \mathcal{X}_{r,i_1,i_2}\mathcal{Y}_{j_0,r,j_2}$

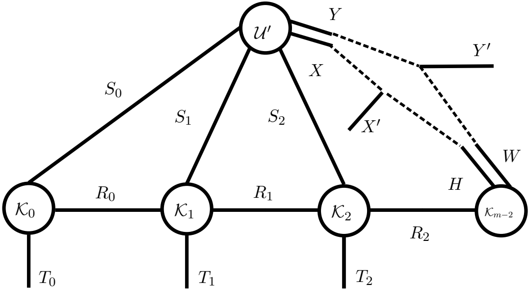
- Similar definitions apply to general mode-(i,j) tensor operations.

Outline

One-layer of CNN vs One-layer of TNN

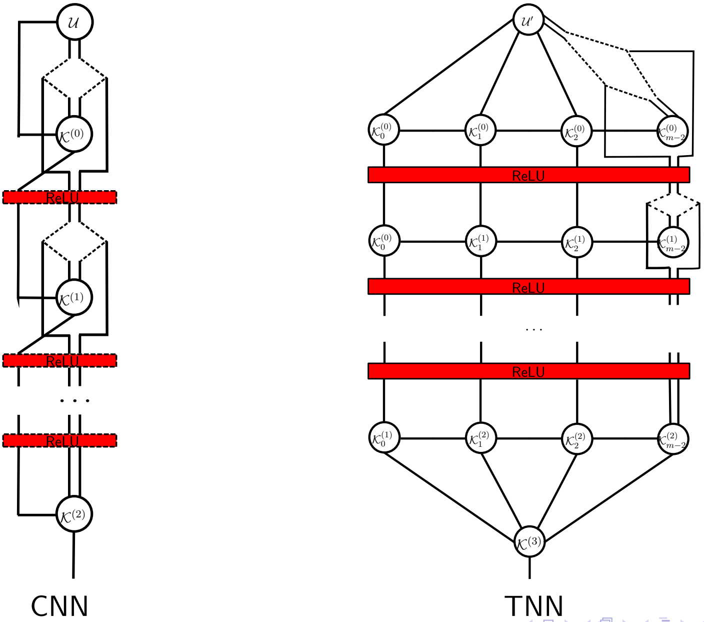


One layer of CNN



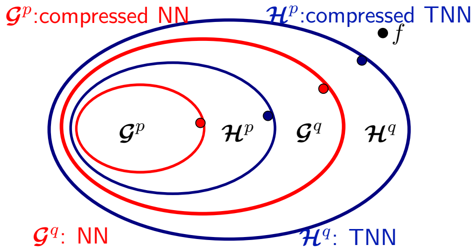
One layer of TNN

CNN vs TNN


$\mathcal{U}$
$\mathcal{K}^{(0)}$
ReLU
$\mathcal{K}^{(1)}$
ReLU
...
ReLU
$\mathcal{K}^{(2)}$
CNN
$\mathcal{U}'$
$\mathcal{K}_0^{(0)}$ $\mathcal{K}_1^{(0)}$ $\mathcal{K}_2^{(0)}$ $\mathcal{K}_{m-2}^{(0)}$
ReLU
$\mathcal{K}_0^{(0)}$ $\mathcal{K}_1^{(1)}$ $\mathcal{K}_2^{(1)}$ $\mathcal{K}_{m-2}^{(1)}$
ReLU
...
ReLU
$\mathcal{K}_0^{(1)}$ $\mathcal{K}_1^{(2)}$ $\mathcal{K}_2^{(2)}$ $\mathcal{K}_{m-2}^{(2)}$
$\mathcal{K}^{(3)}$
TNN

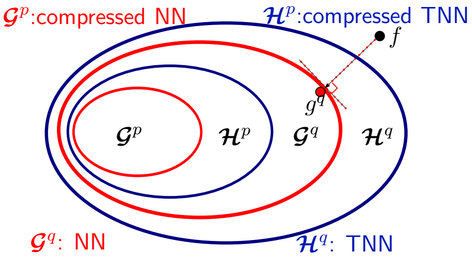
Tensorial Neural Networks

Relationship between NNs and TNNs



Tensorial Neural Networks

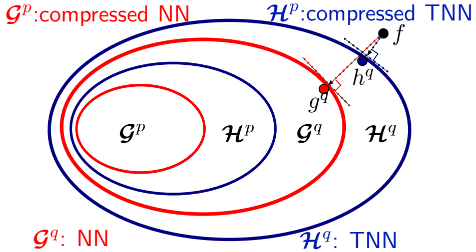
Relationship between NNs and TNNs



- **Learning** of a NN with q parameters: g^q, closest to f in $\mathcal{G}^q$

Tensorial Neural Networks

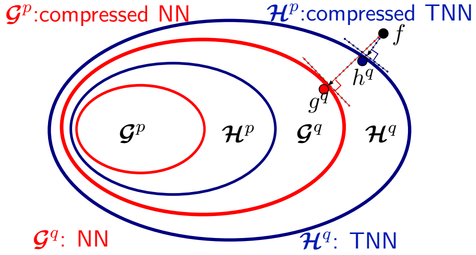
Relationship between NNs and TNNs



- **Learning** of a NN with q parameters: g^q, closest to f in $\mathcal{G}^q$
- **Learning** of a TNN with q parameters: h^q, closest to f in $\mathcal{H}^q$

Tensorial Neural Networks

Relationship between NNs and TNNs

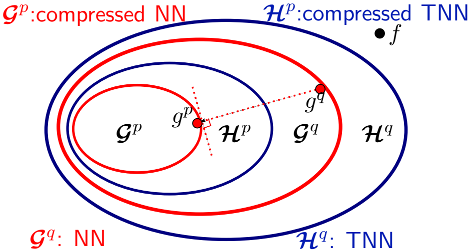


- **Learning** of a NN with q parameters: g^q, closest to f in $\mathcal{G}^q$
- **Learning** of a TNN with q parameters: h^q, closest to f in $\mathcal{H}^q$

$$\boxed{h^q \textbf{ is closer to } f \textbf{ than } g^q}$$

Tensorial Neural Networks

Relationship between NNs and TNNs



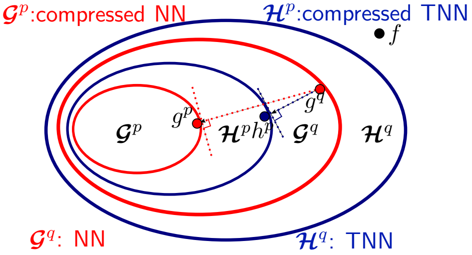
- **Learning** of a NN with q parameters: g^q, closest to f in $\mathcal{G}^q$
- **Learning** of a TNN with q parameters: h^q, closest to f in $\mathcal{H}^q$

$$\boxed{h^q \textbf{ is closer to } f \textbf{ than } g^q}$$

- **Compression** of g^q to p parameters: g^p, closest to g^q in $\mathcal{G}^p$

Tensorial Neural Networks

Relationship between NNs and TNNs



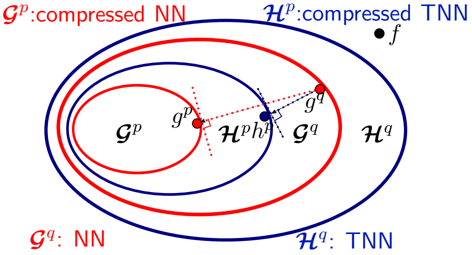
- **Learning** of a NN with q parameters: g^q, closest to f in $\mathcal{G}^q$
- **Learning** of a TNN with q parameters: h^q, closest to f in $\mathcal{H}^q$

$$\boxed{h^q \textbf{ is closer to } f \textbf{ than } g^q}$$

- **Compression** of g^q to p parameters: g^p, closest to g^q in $\mathcal{G}^p$
- **Compression** of g^q to p parameters: h^p, closest to g^q in $\mathcal{H}^p$

Tensorial Neural Networks

Relationship between NNs and TNNs



- **Learning** of a NN with q parameters: g^q, closest to f in $\mathcal{G}^q$
- **Learning** of a TNN with q parameters: h^q, closest to f in $\mathcal{H}^q$

$\boxed{h^q \textbf{ is closer to } f \textbf{ than } g^q}$

- **Compression** of g^q to p parameters: g^p, closest to g^q in $\mathcal{G}^p$
- **Compression** of g^q to p parameters: h^p, closest to g^q in $\mathcal{H}^p$

$\boxed{\textbf{Compressed TNN } h^p \textbf{ is closer to pre-trained } g^q \textbf{ than } g^p}$

Outline

Compression Using Invariant Structure In Deep Neural Networks

Common Compression Techniques

- Pruning, quantization, encoding and knowledge distillation

Compression Using Invariant Structure In Deep Neural Networks

Common Compression Techniques

- Pruning, quantization, encoding and knowledge distillation

Low Rank Approximation

- Complementary to other techniques
- Reduce the number of parameters by a factor **polynomial** in the dimension
  - Caveat: only when the weight matrices (convolutional kernels) are **low rank**

Compression Using Invariant Structure In Deep Neural Networks

Common Compression Techniques

- Pruning, quantization, encoding and knowledge distillation

Low Rank Approximation

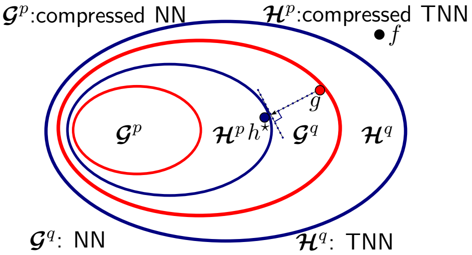
- Complementary to other techniques
- Reduce the number of parameters by a factor **polynomial** in the dimension
  - Caveat: only when the weight matrices (convolutional kernels) are **low rank**

Exploiting other invariant structure via low rank approximation?

Periodicity, modulation and low rank?

Idea for Compression using TNN

Given a pre-trained NN $g \in \mathcal{G}^q$, how do we find a TNN $h^\star \in \mathcal{H}^p$ that is as close to g as possible?



1. Tensorization
2. Generalized Tensor Decomposition
3. Mapping NN to TNN
   - End-to-End (E2E) Learning: traditional learning approach
   - Sequential (Seq) Learning: learning each layer from bottom-up sequentially

Tensorization

Reshape the object to a higher order object

- Identifying periodic and modulated structure by exploiting the low rank structure in the reshaped matrix

Toy Example

Invariant structures

$$\left\{\begin{array}{c} [1, 2, 3, 1, 2, 3, 1, 2, 3] \\ \text{Periodic structure} \\ \\ [1, 1, 1, 2, 2, 2, 3, 3, 3] \\ \text{Modulated structure} \end{array}\right. \xrightarrow{\text{reshape}} \begin{bmatrix} 1, 1, 1 \\ 2, 2, 2 \\ 3, 3, 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \begin{bmatrix} 1, 1, 1 \end{bmatrix}$$

Low-rank structure

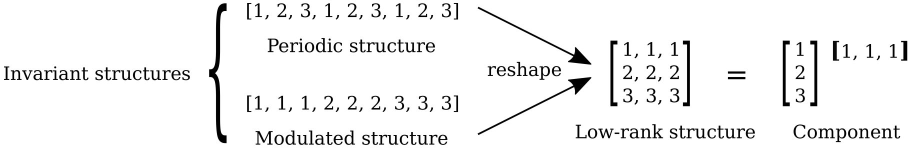
Component

Tensorization

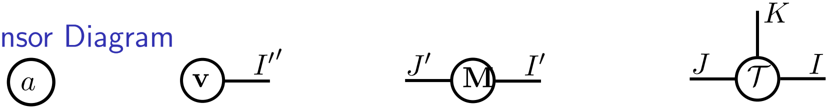
Reshape the object to a higher order object

- Identifying periodic and modulated structure by exploiting the low rank structure in the reshaped matrix

Toy Example



Tensor Diagram



Scalar $a \in \mathbb{R}$ Vector $\mathbf{v} \in \mathbb{R}^{I''}$ Matrix $\mathbf{M} \in \mathbb{R}^{I' \times J'}$ Tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$

- Reshape $\mathbf{v}$ to $\mathbf{M}$ to $\mathcal{T}$
  - $I'' = I' \times J' = I \times J \times K$

Higher Order Tensor Decompositions

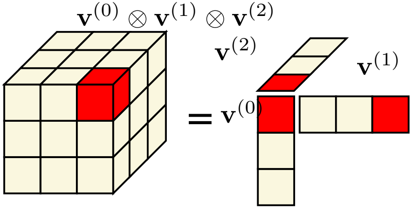
$\boxed{m\text{-order tensor } \mathcal{T} \in \mathbb{R}^{I_0 \times I_1 \times \cdots \times I_{m-1}}}$

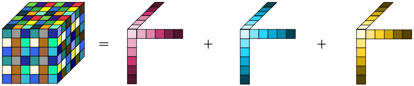
Higher Order Tensor Decompositions

$\boxed{m\text{-order tensor } \mathcal{T} \in \mathbb{R}^{I_0 \times I_1 \times \cdots \times I_{m-1}}}$

CANDECOMP/PARAFAC (CP) Decomposition

- Factorize a tensor into sum of rank-1 tensors
- Rank-1 tensor is defined as outer product of multiple vectors
- $\mathcal{T} = \sum_{r=0}^{R-1} \mathbf{v}_r^{(0)} \otimes \cdots \otimes \mathbf{v}_r^{(m-1)}$

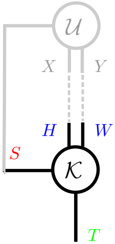




- $[\mathbf{v}^{(0)} \otimes \mathbf{v}^{(1)} \otimes \mathbf{v}^{(2)}]_{i_1, i_2, i_3} = \mathbf{v}_{i_1}^{(0)} \mathbf{v}_{i_2}^{(1)} \mathbf{v}_{i_3}^{(2)}$
- Rank-1 tensor

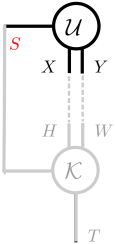
- $\sum_{r=0}^{2} \mathbf{v}_r^{(0)} \mathbf{v}_r^{(1)} \mathbf{v}_r^{(2)}$
- Rank-3 tensor

- **Convolutional Kernel**: $\mathcal{K} \in \mathbb{R}^{H \times W \times S \times T}$



Compression of Convolutional Layer w/ Tensor Decompositions

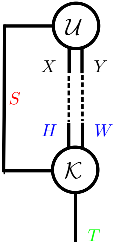
- **Convolutional Kernel**: $\mathcal{K} \in \mathbb{R}^{H \times W \times S \times T}$
- **Input** tensor: $\mathcal{U} \in \mathbb{R}^{X \times Y \times S}$



Compression of Convolutional Layer w/ Tensor Decompositions

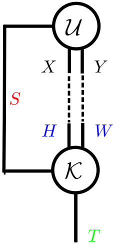
- **Convolutional Kernel**: $\mathcal{K} \in \mathbb{R}^{H \times W \times S \times T}$
- **Input** tensor: $\mathcal{U} \in \mathbb{R}^{X \times Y \times S}$
- Map the input tensor $\mathcal{U}$ to an **output** tensor $\mathcal{V} \in \mathbb{R}^{X' \times Y' \times T}$:

$$\mathcal{V}_{x,y,t} = \sum_{s=0}^{S-1} \sum_{i,j} \mathcal{K}_{i,j,s,t}\, \mathcal{U}_{x-i,y-j,s}.$$



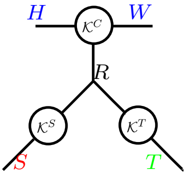
- **Convolutional Kernel**: $\mathcal{K} \in \mathbb{R}^{H \times W \times S \times T}$
- **Input** tensor: $\mathcal{U} \in \mathbb{R}^{X \times Y \times S}$
- Map the input tensor $\mathcal{U}$ to an **output** tensor $\mathcal{V} \in \mathbb{R}^{X' \times Y' \times T}$:

$$\mathcal{V}_{x,y,t} = \sum_{s=0}^{S-1} \sum_{i,j} \mathcal{K}_{i,j,s,t} \, \mathcal{U}_{x-i,y-j,s}.$$



CP Decomposition on the Kernel

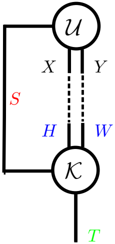
- **CP**: Decompose kernel $\mathcal{K}$ into 3 factor tensors
- $\mathcal{K}_{i,j,s,t} = \sum_{r=0}^{R-1} \mathcal{K}^S_{s,r} \, \mathcal{K}^C_{i,j,r} \, \mathcal{K}^T_{r,t}$
- # of param.: $HWST \to (HW + S + T)R$



Compression of Convolutional Layer w/ Tensor Decompositions

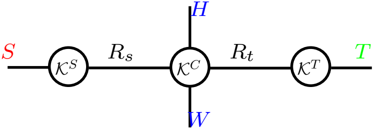
- **Convolutional Kernel**: $\mathcal{K} \in \mathbb{R}^{H \times W \times S \times T}$
- **Input** tensor: $\mathcal{U} \in \mathbb{R}^{X \times Y \times S}$
- Map the input tensor $\mathcal{U}$ to an **output** tensor $\mathcal{V} \in \mathbb{R}^{X' \times Y' \times T}$:

$$\mathcal{V}_{x,y,t} = \sum_{s=0}^{S-1} \sum_{i,j} \mathcal{K}_{i,j,s,t}\, \mathcal{U}_{x-i,y-j,s}.$$



TK Decomposition on the Kernel

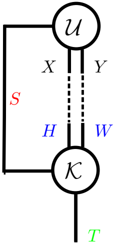
- **TK**: Decompose $\mathcal{K}$ into 1 core tensor, 2 factor tensors
- $\mathcal{K}_{i,j,s,t} = \sum_{r_s=0}^{R_s-1} \sum_{r_t=0}^{R_t-1} \mathcal{K}^S_{s,r_s}\ \mathcal{K}^C_{i,j,r_s,r_t}\ \mathcal{K}^T_{r_t,t}$
- # of param.: $HWST \to SR_s + HWR_sR_t + R_tT$



Compression of Convolutional Layer w/ Tensor Decompositions

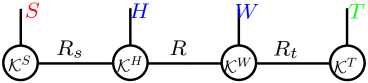
- **Convolutional Kernel**: $\mathcal{K} \in \mathbb{R}^{H\times W\times S\times T}$
- **Input** tensor: $\mathcal{U} \in \mathbb{R}^{X\times Y\times S}$
- Map the input tensor $\mathcal{U}$ to an **output** tensor $\mathcal{V} \in \mathbb{R}^{X'\times Y'\times T}$:

$$\mathcal{V}_{x,y,t} = \sum_{s=0}^{S-1} \sum_{i,j} \mathcal{K}_{i,j,s,t}\, \mathcal{U}_{x-i,y-j,s}.$$



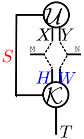
TT Decomposition on the Kernel

- **TT**: Decompose $\mathcal{K}$ into 4 factor tensors

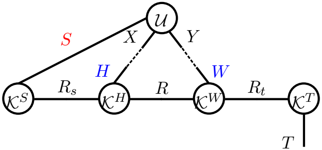


- $\mathcal{K}_{i,j,s,t} = \sum_{r_s=0}^{R_s-1} \sum_{r=0}^{R-1} \sum_{r_t=0}^{R_t-1} \mathcal{K}^S_{s,r_s} \mathcal{K}^H_{r_s,i,r}\, \mathcal{K}^W_{r,j,r_t}\, \mathcal{K}^T_{r_t,t}$
- # of param.: $HWST \rightarrow SR_s + HR_sR + WR_tR + R_tT$

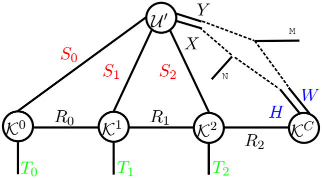
Reshaped Tensor Decomposition— Narrower & Deeper Nets



Uncompressed



Compressed via TT



Compressed via r-TT $(m = 3)$

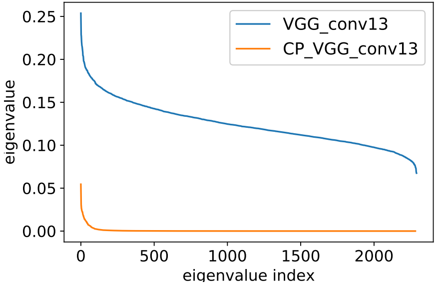
TT Decomposition on the Reshaped Kernel

- Param. $\#: HWST \rightarrow SR_s + HR_sR + WR_tR + R_tT \rightarrow (m(ST)^{\frac{1}{m}}R + HW)R$

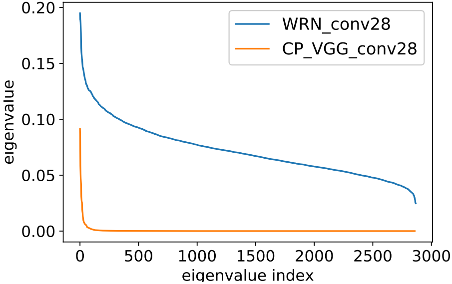
Low Rank Structure?

Comparisons of Eigenvalue Spectra

- CP-VGG and CP-WRN are TNNs



VGG (layer 13)



WRN (layer 28)

Outline

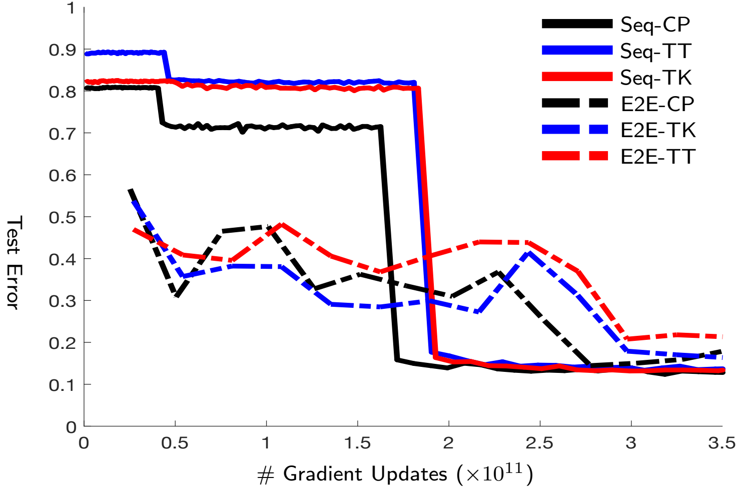
Experiments - Compress CIFAR10 Resnet-32

Successful Compression of CIFAR10 Resnet-32 Network (Su, Li, Bhattacharjee & H., 2018)

	Compression rate					Compression rate			
	5%	10%	20%	40%		2%	5%	10%	20%
SVD	83.09	87.27	89.58	90.85	r-TR†	-	80.80	-	90.60
CP	84.02	86.93	88.75	88.75	r-CP	85.7	89.86	**91.28**	-
TK	83.57	86.00	88.03	89.35	r-TK	61.06	71.34	81.59	87.11
TT	77.44	82.92	84.13	86.64	r-TT	78.95	84.26	87.89	-

- Testing accuracies of tensor methods under compression rates.
- The uncompressed network achieves 93.2% accuracy.
- CIFAR10 Resnet-32 has 0.46M parameters that have to be trained and retained during testing.

Experiments - Convergence Rate



Convergence rate for Seq vs. E2E compression on CIFAR10.

Experiments - Compress ImageNet Resnet-50

Successful Compression of ImageNet Resnet-50 Network (Su, Li, Bhattacharjee & H., 2018)

# samples	Uncompressed # params.: 25M	TT (E2E) # params.: 2.5M	r-TT (Seq) # params.: 2.5M
0.24M	4.22	2.78	44.35
0.36M	6.23	3.99	46.98
0.60M	9.01	7.48	49.92
1.20M	17.3	12.80	52.59
2.40M	30.8	18.17	**54.00**

- Testing accuracy of tensor methods compared to the uncompressed ImageNet Resnet-50.

Summary

1. Extends traditional NN via a new framework TNN which
   - naturally preserve multi-dimensional structures of the input data (such as videos)
   - effectively compress existing NN by exploiting additional invariant structures
2. Introduce a system of *generalized tensor algebra* and *generalized tensor operations* for
   - efficient learning and prediction in TNNs
   - deriving and analyze backpropagation rules for generalized tensor operations.
3. Interpretations of famous neural network architectures using TNNs.

Arxiv: 1805.10352

Code: github.com/FurongHuang/TTP-NeuralNets-Compression