Neural Networks Speed-up and Compression

Lecture 3: Tensor-based methods

When Tensor Methods help Deep Learning?

Tensor Methods are used to improve deep learning pipelines in a variety of tasks:

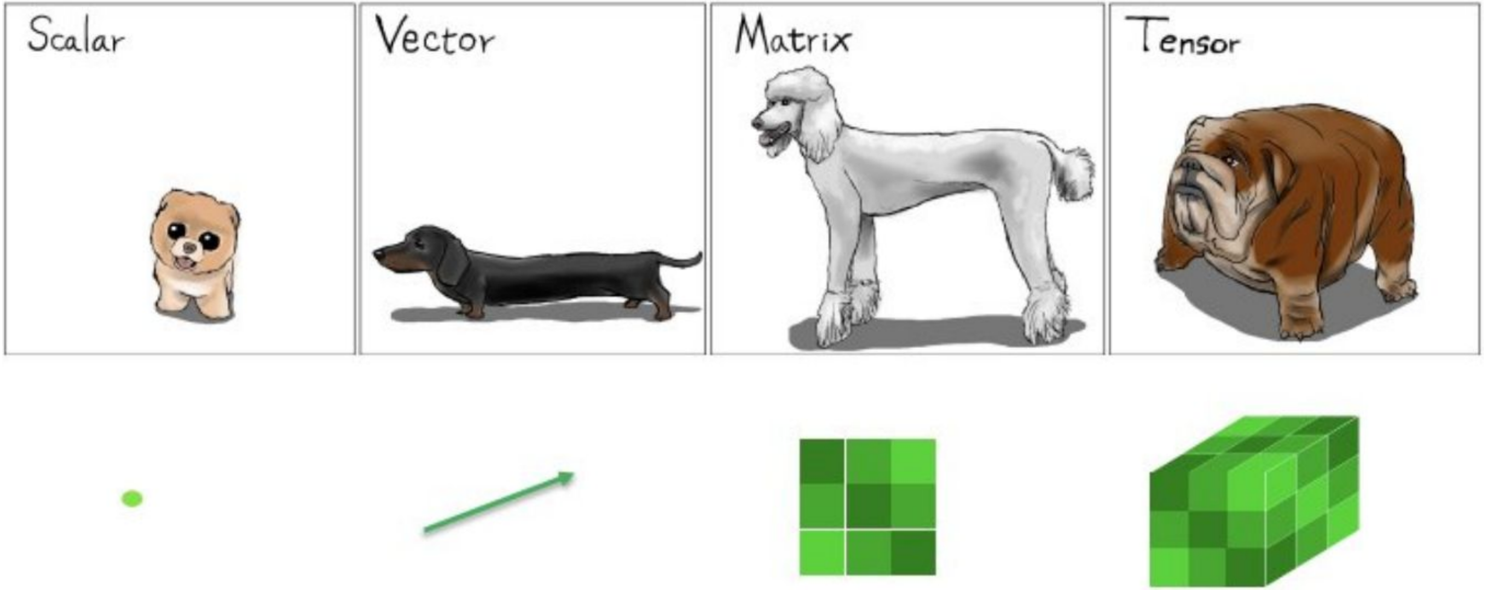
- Model compression and acceleration
- One-shot learning
- Domain adaptation
- Incremental learning
- Fusion of features
- etc.

Outline

- Tensor decompositions
- Tensor-based methods for model speed-up and compression
- Other tensor-based deep learning applications
- Practical exercises

Tensor Decompositions

Tensor: Multidimentional array



Scalar-level → Matrix-level → Block-Matrix level → Tensor level

1960’s 1980’s 2000’s Now

Level of thinking in algebra field

Low-rank Matrix Decomposition

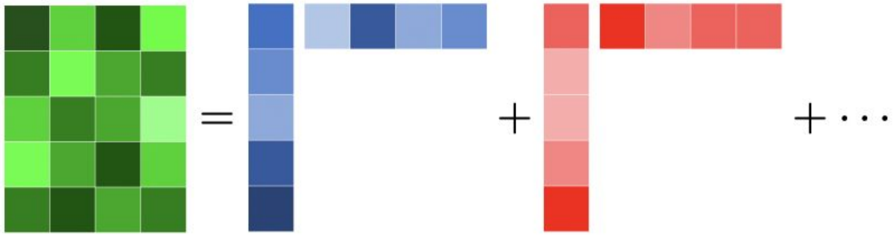


Figure 2.1: Decomposition of a matrix $M \in \mathbb{R}^{5\times 4}$ as sum of the rank-1 components. Note that each component is the product of a column vector u_j and a row vector $v_j^\top$.

Source: https://arxiv.org/pdf/2004.07984.pdf

Low-rank Matrix Decomposition

$$\begin{array}{r} \\ \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{Dave} \\ \text{Eve} \end{array} \begin{array}{c} \begin{array}{cccc} \text{Math.} & \text{Classics} & \text{Physics} & \text{Music} \end{array} \\ \begin{pmatrix} 19 & 26 & 17 & 21 \\ 8 & 17 & 9 & 12 \\ 7 & 12 & 7 & 9 \\ 15 & 29 & 16 & 21 \\ 31 & 40 & 27 & 33 \end{pmatrix} \end{array} = \overbrace{\begin{pmatrix} 4 \\ 3 \\ 2 \\ 5 \\ 6 \end{pmatrix} \begin{pmatrix} 1 \\ 5 \\ 2 \\ 3 \end{pmatrix}^{\top}}^{\text{Verbal}} + \overbrace{\begin{pmatrix} 3 \\ 1 \\ 1 \\ 2 \\ 5 \end{pmatrix} \begin{pmatrix} 5 \\ 2 \\ 3 \\ 3 \end{pmatrix}^{\top}}^{\text{Quantitative}}$$

- $$\begin{aligned} \text{Score}(\text{student}, \text{test}) = {} & \text{student}_{\text{verbal-intlg.}} \times \text{test}_{\text{verbal}} \\ & + \text{student}_{\text{quant-intlg.}} \times \text{test}_{\text{quant.}} \end{aligned}$$

- u_{verbal}, u_{quant} - vectors that describe the verbal/quantitative strength for each student
- v_{verbal}, v_{quant} - vectors that describe the requirement for each test

$$M = u_{\text{verbal}} v_{\text{verbal}}^{\top} + u_{\text{quant.}} v_{\text{quant.}}^{\top}$$

Source: https://arxiv.org/pdf/2004.07984.pdf

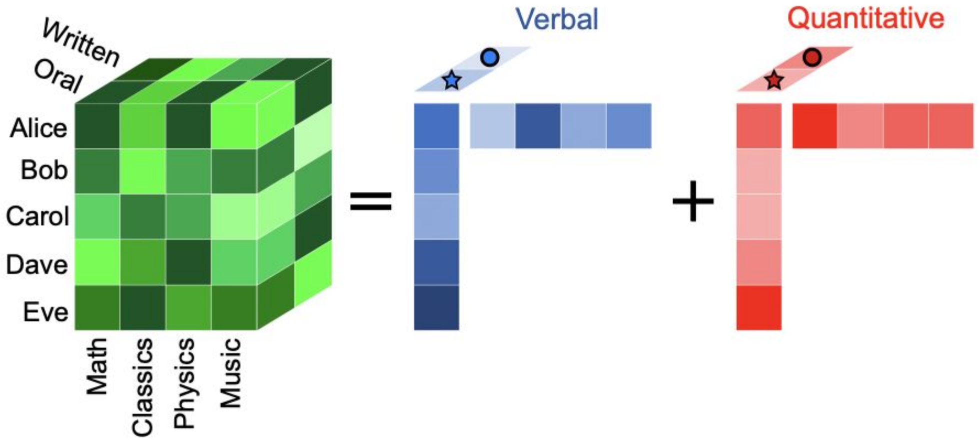
Ambiguity of Matrix Decomposition

$$
\begin{array}{r c}
 & \begin{array}{cccc} \text{Math.} & \text{Classics} & \text{Physics} & \text{Music} \end{array} \\
\begin{array}{r} \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{Dave} \\ \text{Eve} \end{array} &
\begin{pmatrix} 19 & 26 & 17 & 21 \\ 8 & 17 & 9 & 12 \\ 7 & 12 & 7 & 9 \\ 15 & 29 & 16 & 21 \\ 31 & 40 & 27 & 33 \end{pmatrix}
\end{array}
= \underbrace{\begin{pmatrix} 4 \\ 3 \\ 2 \\ 5 \\ 6 \end{pmatrix} \begin{pmatrix} 1 \\ 5 \\ 2 \\ 3 \end{pmatrix}^{\top}}_{\text{Verbal}} + \underbrace{\begin{pmatrix} 3 \\ 1 \\ 1 \\ 2 \\ 5 \end{pmatrix} \begin{pmatrix} 5 \\ 2 \\ 3 \\ 3 \end{pmatrix}^{\top}}_{\text{Quantitative}}
$$

$$
= \begin{pmatrix} 1 \\ 2 \\ 1 \\ 3 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 5 \\ 2 \\ 3 \end{pmatrix}^{\top} + \begin{pmatrix} 3 \\ 1 \\ 1 \\ 2 \\ 5 \end{pmatrix} \begin{pmatrix} 6 \\ 7 \\ 5 \\ 6 \end{pmatrix}^{\top}
$$

Note that the students verbal intelligence and tests quantitative weights are different between two decompositions.

Tensor Decomposition



- $\text{Score}(\text{student}, \text{test}, \text{format}) =$
$$\text{student}_{\text{verbal-intlg.}} \times \text{test}_{\text{verbal}} \times \text{format}_{\text{verbal}}$$
$$+\, \text{student}_{\text{quant-intlg.}} \times \text{test}_{\text{quant.}} \times \text{format}_{\text{quant}}$$
- w_{verbal}, w_{quant} correspond to verbal/quantitative importance for different formats

$$(M_{\text{written}}, M_{\text{oral}}) = u_{\text{verbal}} \otimes v_{\text{verbal}} \otimes w_{\text{verbal}}$$
$$+\, u_{\text{quant.}} \otimes v_{\text{quant.}} \otimes w_{\text{quant}}$$

Speed-up and Compression with Tensor Methods

Model compression and acceleration

Problem: Modern deep learning architectures need a lot of space to store and significant time to run

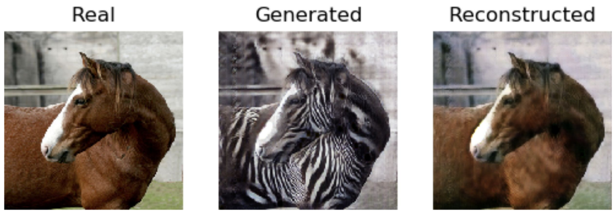
Goal: Make deep learning models compact and fast, so that they can fit to mobile/edge devices and run fast enough not to annoy the user

Tensor Methods are effective for:

- Compression and acceleration of pre-trained deep learning models
- Building new compact architectures that are trained from scratch
- Training binary neural networks

Convolutional Neural Network reminder

Increasing importance and number of practical applications of CNN applications:



Credits:
1) https://medium.com/@ismailou.sa
2) https://machinelearningmastery.com/cyclegan-tutorial-with-keras/
3) https://www.internetandtechnologylaw.com/bias-facial-recognition-flaws

Convolutional Neural Network reminder

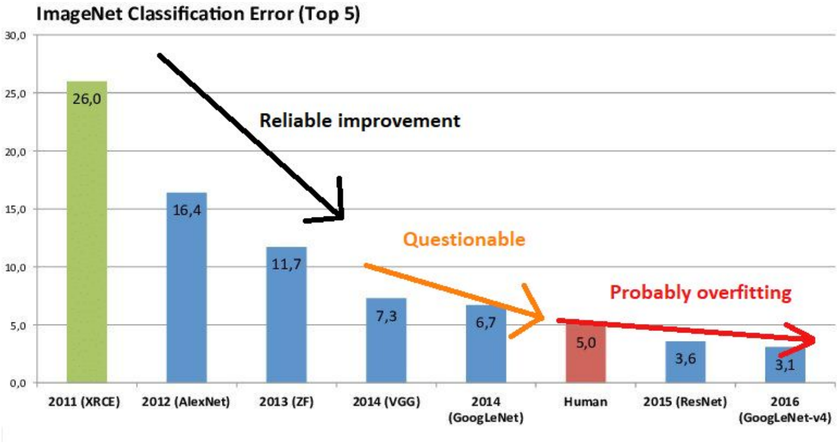


Figure 1: Imagenet classification Error diagram

Figure 2: Convolutional neural network scheme

*Left image is borrowed from: https://devopedia.org/imagenet
*Right image is borrowed from: https://www.google.com/about/main/machine-learning-qa/

CNN compression: Motivation

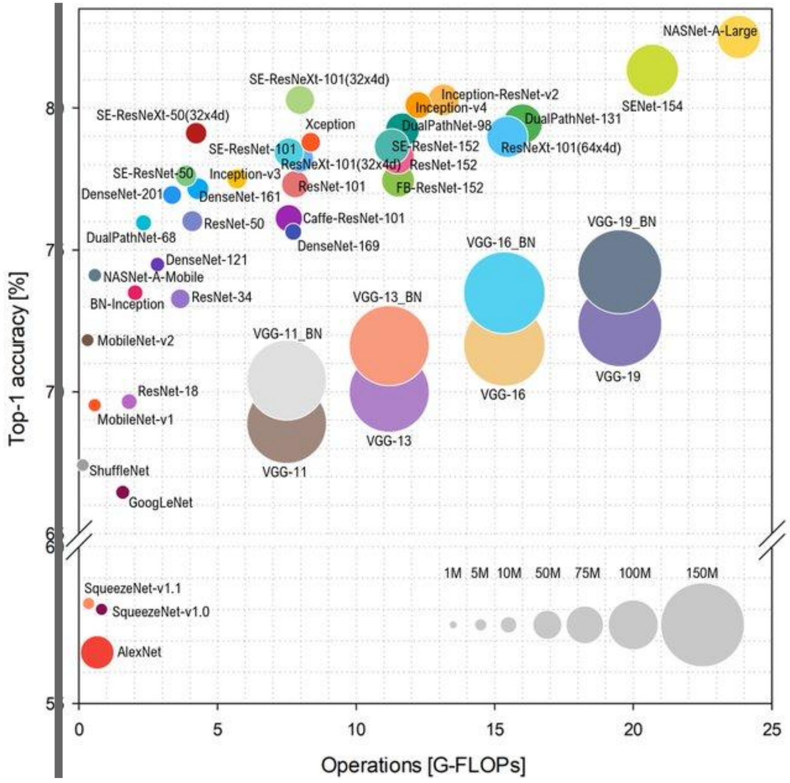
DL model limitations:

- High memory consumption
- Huge computational requirements
- Great power consumption

Difficult to deploy on portable devices

(e.g. laptops and smartphones)

Efficient architecture design is required



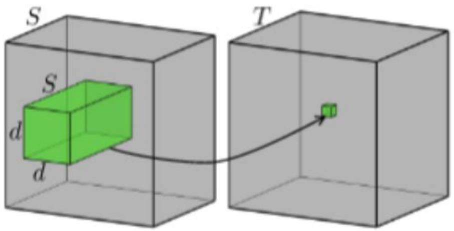
Efficient CNN zone

Credits: S. Bianco, et al,, "Benchmark Analysis of Representative Deep Neural Network Architectures," in IEEE Access

NN compression via weight approximation: Motivation

For a convolutional layer with input of size $H \times W \times S$ and kernel (weight tensor) of size $d \times d \times T \times S$ number of

- parameters: $O(d^2 ST)$
- operations: $O(HWd^2 ST)$



Source: https://arxiv.org/pdf/1412.6553.pdf

Figure: Convolutional layer.

Reducing the number of parameters in NNs is a common trick to accelerate inference time and at the same time reduce power usage and network memory.

Tensor decomposition for weight approximation

- $\underline{X}_{ijk} \cong \sum_{r=1}^{R} \lambda_r a_{ir} b_{jr} c_{kr}$

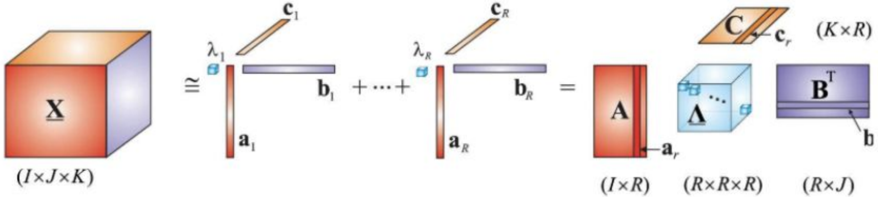


Figure: rank-R CP decomposition of 3D tensor (source: http://arxiv.org/abs/1609.00893)

Tensor decomposition for weight approximation

- $\underline{X}_{ijk} \cong \sum_{r=1}^{R} \lambda_r a_{ir} b_{jr} c_{kr}$

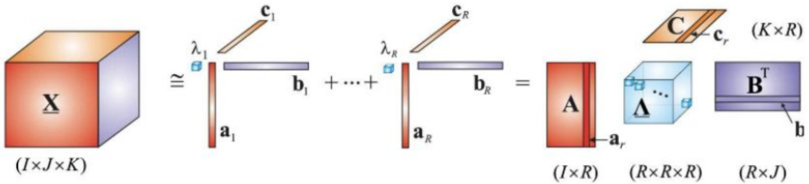


Figure: rank-R CP decomposition of 3D tensor (source: http://arxiv.org/abs/1609.00893)

- $\underline{X}_{ijk} \cong \sum_{r_1}^{R_1} \sum_{r_2}^{R_2} \sum_{r_3}^{R_3} g_{r_1 r_2 r_3} b_{ir_1}^{(1)} b_{jr_2}^{(2)} b_{kr_3}^{(3)}$

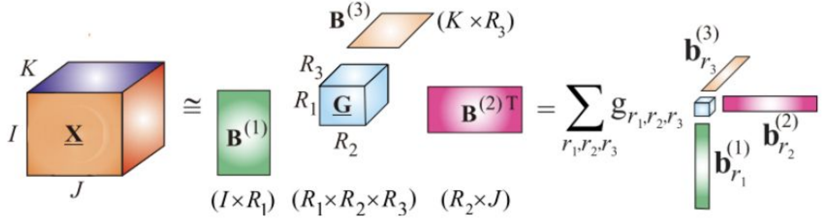
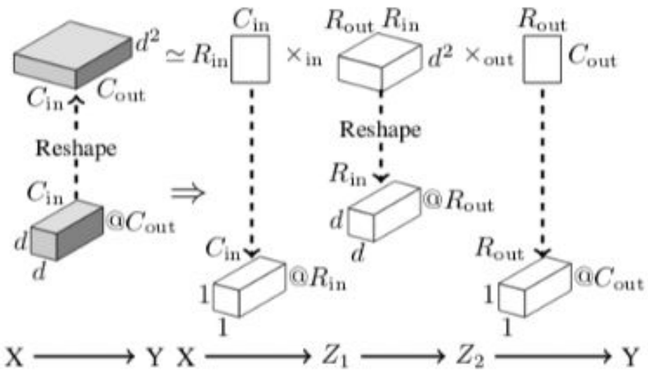


Figure: rank-(R_1, R_2, R_3) Tucker decomposition of 3D tensor

Layer compression via weight approximation



- **Top row**: low-rank approximation of 3D weight tensor.
  - Tucker: $O(d^2 C_{\text{in}} C_{\text{out}}) \to O(C_{\text{in}} R_{\text{in}} + d^2 R_{\text{out}} R_{\text{in}} + C_{\text{out}} R_{\text{out}})$ parameters,
  - CP: $O(d^2 C_{\text{in}} C_{\text{out}}) \to O\left(R(C_{\text{in}} + d^2 + C_{\text{out}})\right)$ parameters, $R = R_{out} = R_{in}$.
- **Bottom row**: initial layer is replaced with a sequence of layers.
  - Tucker: middle convolution is standard.
  - CP: middle convolution is depth-wise.

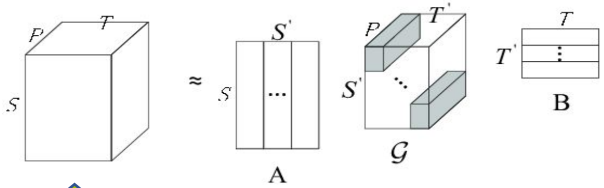
CNN compression: Description

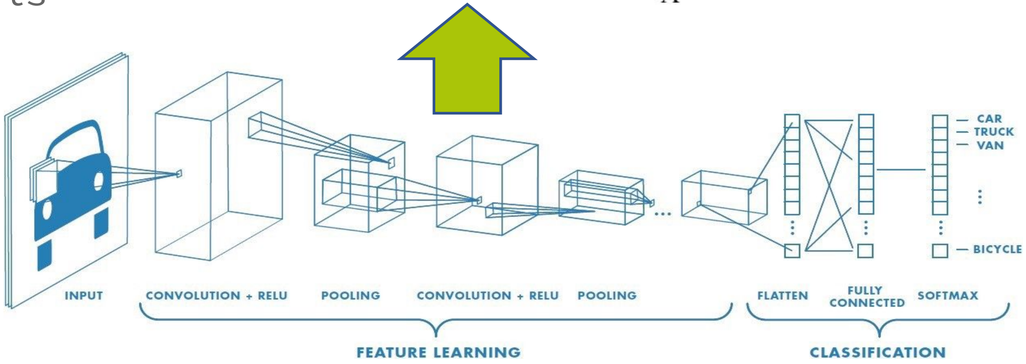
Pipeline:

1. Extract convolutional kernel
2. Decompose it into factors: A, G, B
3. Replace initial layer by sequence of layers with factors as kernels
4. Fine-tune network

Result:

1. Faster inference
2. Lower memory consumption
3. Longer battery life

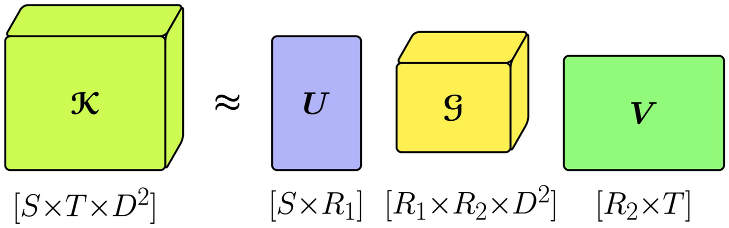


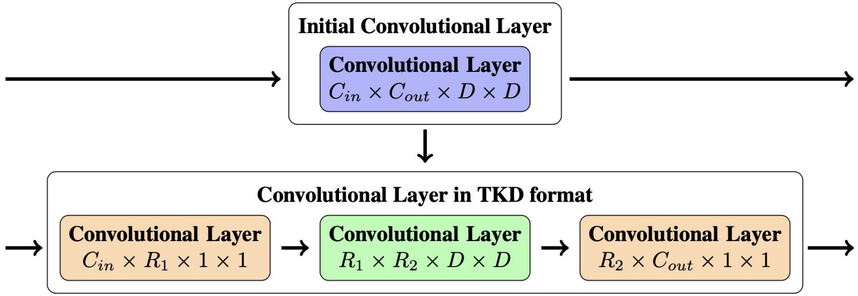


*Bottom image is borrowed from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

CNN compression: Tucker Decomposition

- Allows to reduce number of parameters
  - from ($S \times T \times D^2$)
  - to ($S \times R_1 + T \times R_2 + R_1 \times R_2 \times D^2$)

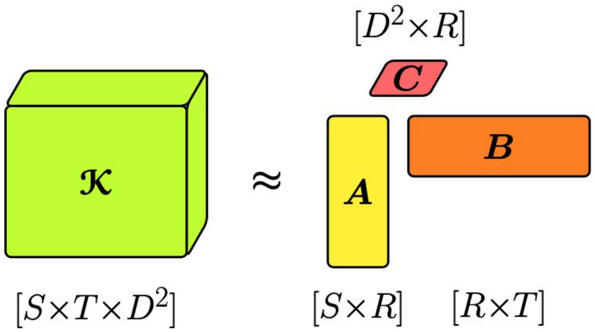




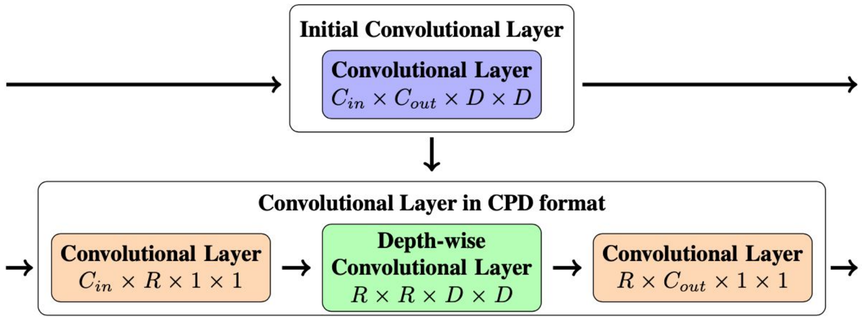
Source: https://arxiv.org/abs/1511.06530

CNN compression: CPD

- CPD allows to reduce number of parameters from *S x T x D x D* to *(S + T + D^2) x R*.
- Applying ordinary CPD to convolutional kernel leads to instability for the neural network fine-tuning.



CPD scheme: $\hat{\mathcal{K}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$



CP-decomposed Convolutional Layer

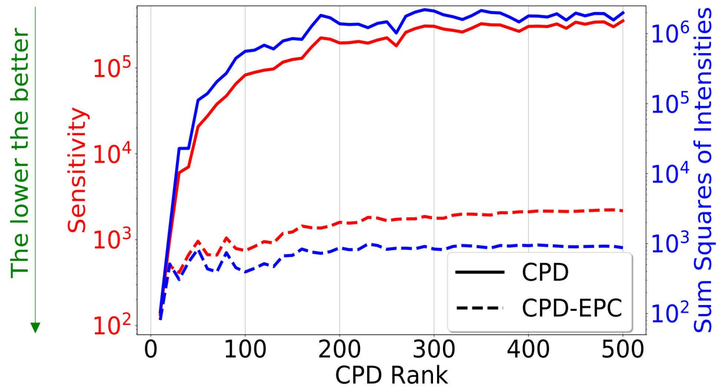
Source: https://arxiv.org/abs/1412.6553

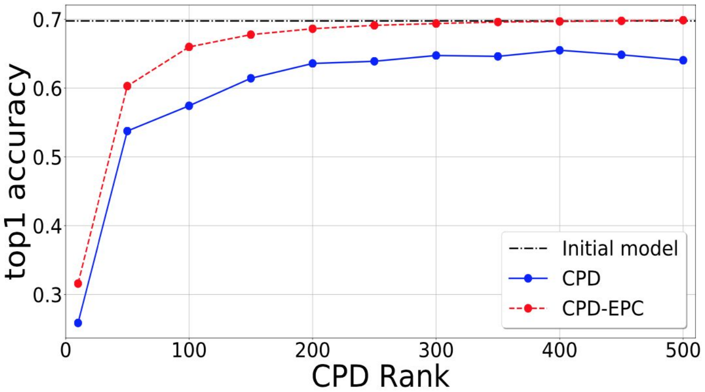
CNN compression: CPD

CPD with minimal sensitivity:

$$\min_{\{\mathbf{A},\mathbf{B},\mathbf{C}\}} \quad \mathrm{ss}([\![\mathbf{A},\mathbf{B},\mathbf{C}]\!])$$
$$\text{s.t.} \quad \|\mathcal{K} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|_F^2 \leq \delta^2$$

Bound δ^2 can be the approximation error of the CPD with diverging components.

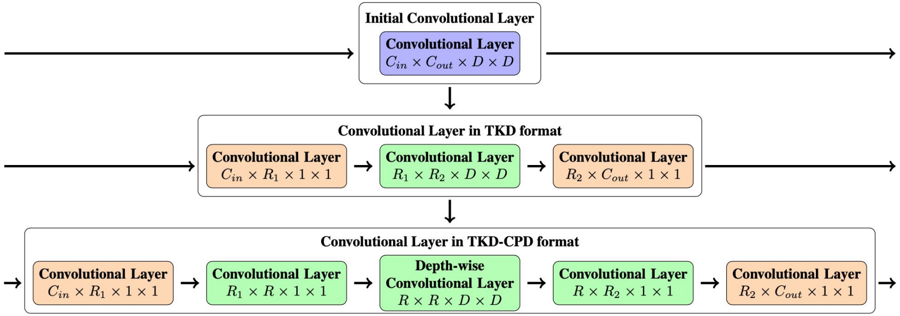


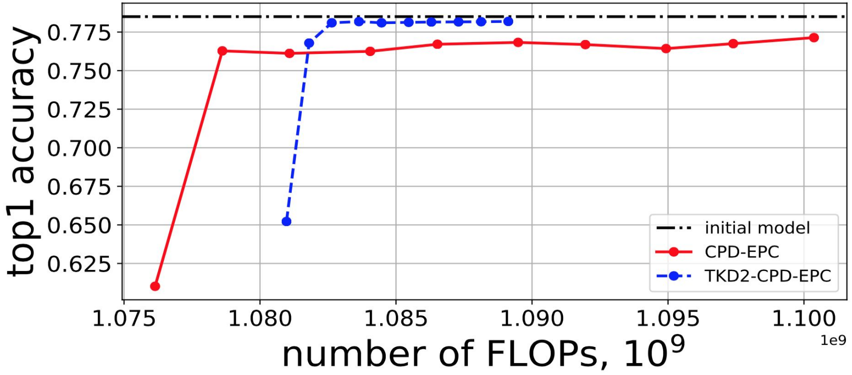


Source: https://arxiv.org/abs/2008.05441

CNN compression: TKD-CPD

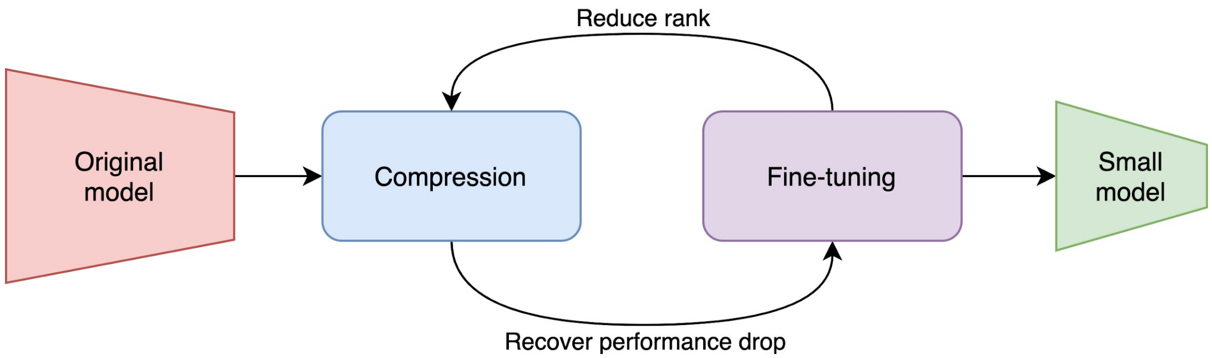
- Tucker decomposition is particularly suited as prior-compression for CPD.
- CPD is applied to core tensor in TKD, which is of smaller dimensions than the original kernels.





CNN compression: MUSCO

- Rank selection strikes a balance between compression ratio and performance degradation
- Instead of direct rank selection, we can reduce it iteratively



Source:
http://openaccess.thecvf.com/content_ICCVW_2019/papers/LPCV/Gusak_Automated_Multi-Stage_Compression_of_Neural_Networks_ICCVW_2019_paper.pdf

CNN compression: Results

Model	Method	↓ FLOPs	Δ top-1	Δ top-5
VGG-16	Asym. [6]	≈ 5.00	-	-1.00
	TKD+VBMF [4]	4.93	-	-0.50
	CPD-EPC [5] (EPS=0.005)	**5.26**	**-0.92**	**-0.34**
ResNet-18	Channel Gating NN [3]	1.61	-1.62	-1.03
	Discrimination-aware Channel Pruning [7]	1.89	-2.29	-1.38
	FBS [1]	1.98	-2.54	-1.46
	MUSCO [2]	2.42	-0.47	-0.30
	CPD-EPC [5] (EPS=0.00325)	**3.09**	**-0.69**	**-0.15**
ResNet-50	CPD-EPC [5] (EPS=0.0028)	**2.64**	**-1.47**	**-0.71**

[1] Gao,X. et al. :Dynamic channel pruning: Feature boosting and suppression. In: ICLR (2019)
[2] Gusak, J., et al.: Automated multi-stage compression of neural networks. In: ICCVW (2019)
[3] Hua, W. et al.: Channel gating neural networks. In: NeurIPS (2019)
[4] Kim, Y., et al.: Compression of deep convolutional neural networks for fast and low power mobile applications. In: ICLR (2016)
[5] Phan, A.-H., et al.: Stable Low-rank Tensor Decomposition for Compression of Convolutional Neural Network. In: ECCV (2020)
[6] Zhang,X.,et al.: Accelerating very deep convolutional networks for classification and detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 38(10) (2016)
[7] Zhuang,Z., et al.:Discrimination - aware channel pruning for deep neural networks. In: NeurIPS (2018)

Tensor layers to replace Flattening and Fully-connected layers

Problem: Flattening discards multilinear structure in the activations, and fully-connected layers require many parameters

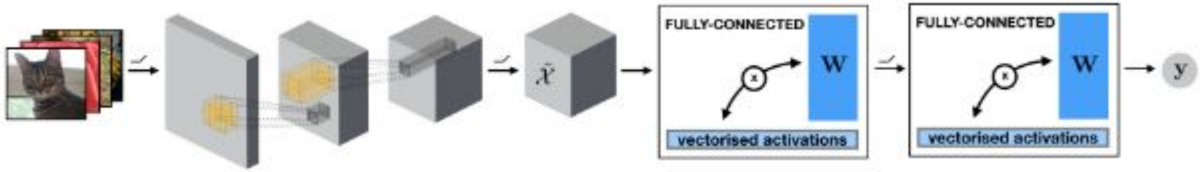
Goal: Alternative to Flattening + Fully-connected layers that preserves multilinear structure and reduce number of parameters

Tensor-based solution:

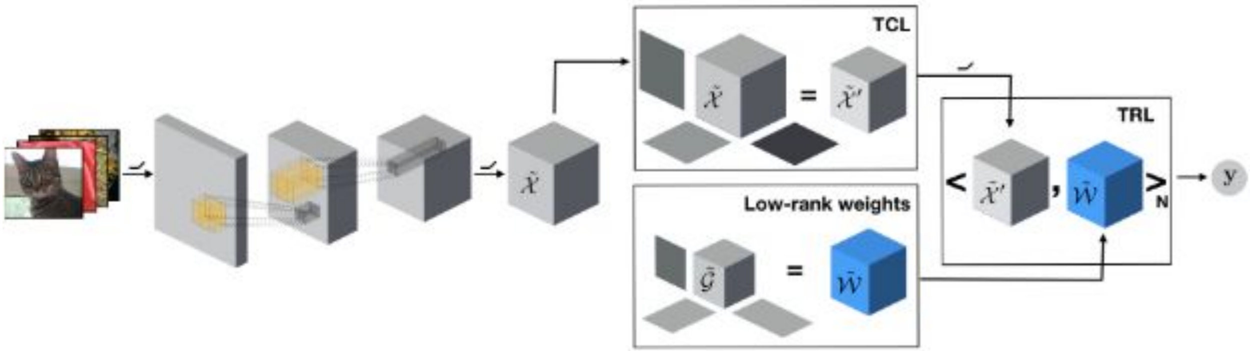
- Tensor Contraction Layers (TCLs): reduce the dimensionality of their input while preserving their multilinear structure using tensor contraction
- Tensor Regression Layers (TRLs): a low-rank multilinear mapping from a high-order activation tensor to an output tensor of arbitrary order

Tensor layers to replace Flattening and Fully-connected layers

Flattening + Fully-connected layers:



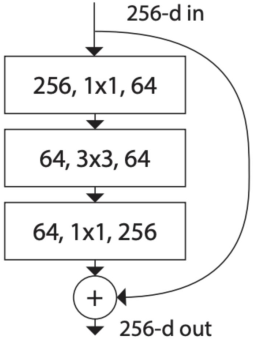
Tencor contraction + Tensor regression layers:



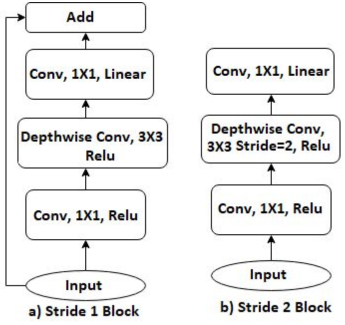
DL architectures inspired by Tensor Decompositions

Applying different decompositions to the regular convolutional layer with d x d spatial kernel we obtain modern architecture units:

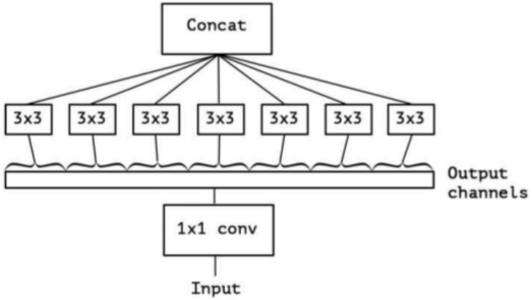
- Tucker decomposition → ResNet Bottleneck block
- CP decomposition → MobileNet block
- Block Term decomposition → ResNext and Xception blocks



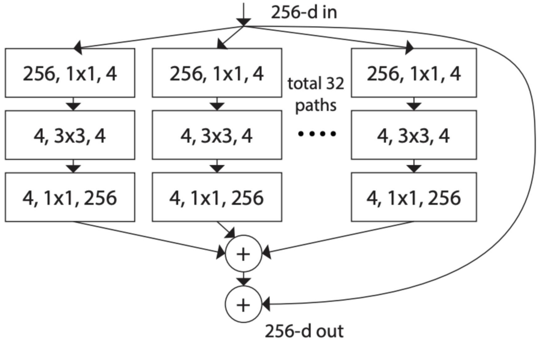
ResNet block



MobileNet block



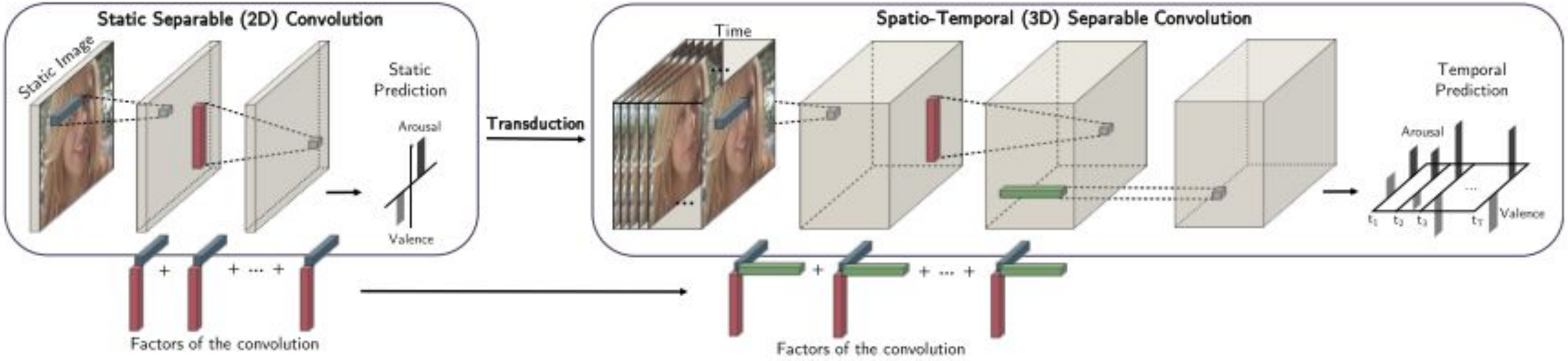
Xception block



ResNeXT block

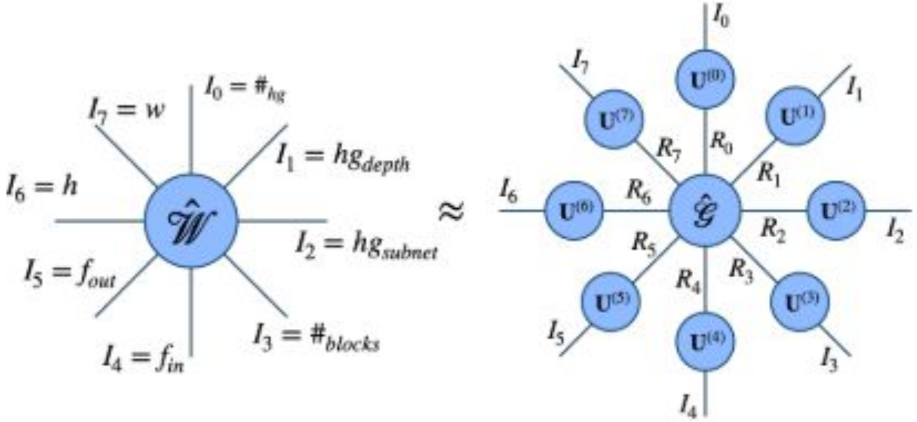
DL architectures inspired by Tensor Decompositions

- **N-D convolutions via higher-order factorization** (Kossaifi et al., CVPR'20)
- Applied to **Static affect estimation**: estimate continuous levels of valence (how positive or negative an emotional display is) and arousal (how exciting or calming is the emotional experience)



DL architectures inspired by Tensor Decompositions

- **The full structure of a neural network is parametrized with a single high-order tensor** (Kossaifi et al., 2019), the modes of which represent each of the architectural design parameters of the network (e.g. number of convolutional blocks, depth, number of stacks, input features, etc).
- Applied to: **Human pose estimation, Semantic facial part segmentation**



Tensor Decompositions to improve Training of Binary Neural Networks

- During training of Binary Neural Networks (have binary activations and binary weights):
  - Latent parameterization:
    - W = UV, where W is a layer weight tensor
  - Binarization:
    - B_i = sign(W_i)
- During inference:
  - Only binary weights are needed.
- Tensor decomposition introduces an **inter-dependency between the to-be-binarized weights through the shared factor U** either at a layer level or even more globally at a network level.
- Applied to **Human pose estimation, Image classification** (Bulat et al., 2019)

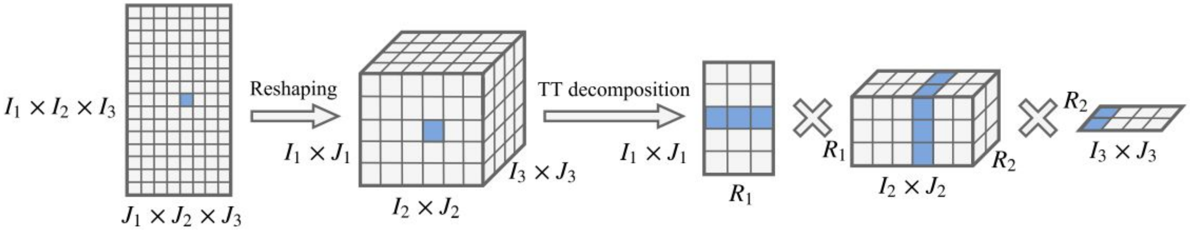
TT-embedding layer

Paper: Valentin Khrulkov, Oleksii Hrinchuk, Leyla Mirvakhabova, Elena Orlova, Ivan Oseledets (2019) “Tensorized Embedding Layers” https://arxiv.org/pdf/1901.10787.pdf

Problem: when the number of identities is large, a weight matrix of a fully-connected embedding layer might require a lot of memory.

Goal: reduce the storage memory for the embedding layer to facilitate model inference in the limited resource settings.

Tensor-based solution: parametrize the embedding layer using Tensor Train (TT) decomposition. That allows to achieve significant compression while maintaining the original performance.



TT-embedding layer

Example: sentiment analysis

Dataset	Model	Embedding shape	Test acc.	Emb compr.	Total params
IMDB	Full	25000×256	0.886	1	7.19M
	TT1	$(25, 30, 40) \times (4, 8, 8)$	0.871	93	0.86M
	TT2	$(10, 10, 15, 20) \times (4, 4, 4, 4)$	0.888	232	0.82M
	TT3	$(5, 5, 5, 5, 6, 8) \times (2, 2, 2, 2, 4, 4)$	**0.897**	441	0.81M
SST	Full	17200×256	0.374	1	5.19M
	TT1	$(24, 25, 30) \times (4, 8, 8)$	**0.415**	78	0.85M
	TT2	$(10, 10, 12, 15) \times (4, 4, 4, 4)$	0.411	182	0.82M
	TT3	$(4, 5, 5, 5, 6, 6) \times (2, 2, 2, 2, 4, 4)$	0.399	307	0.81M

source: https://arxiv.org/pdf/1901.10787.pdf

Follow-up paper: Chunxing Yin et al. (2021) “TT-Rec: Tensor Train Compression for Deep Learning Recommendation Models”
https://arxiv.org/abs/2101.11714

Python-package: musco-pytorch

MUSCO is a Python library for NNs compression via tensor/matrix approximation of weight tensors.

- **Supported layers**: convolutional (1D, 2D), fully-connected.
- **Supported decompositions**: SVD, different types of CPD, Tucker decomposition.
- **Supported rank selection**: manual, constant compression rate, Bayesian (VBMF).
- Supports multi-stage compression.
- **Source code**: `https://github.com/musco-ai/musco-pytorch/tree/develop`

Python-package: musco-pytorch

Steps to perform model compression using MUSCO package.

- Load a pre-trained model.
- Compute model statistics.
- Define a model compression schedule.
- Create a Compressor.
- Compress.

Python-package: musco-pytorch

```python
from flopco import FlopCo
from musco.pytorch import Compressor

model = resnet50(pretrained = True)
model_stats = FlopCo(model, device = device)

compressor = Compressor(model,
                        model_stats,
                        ft_every=5,
                        nglobal_compress_iters=2,
                        config_type = 'vbmf')
```

```python
while not compressor.done:
    # Compress layers
    compressor.compression_step()
    # Fine-tune compressor.compressed_model
```

For detailed instructions check *README.md* and *docs* at
https://github.com/musco-ai/musco-pytorch/tree/develop

Deep Learning applications based on Tensor Methods

One-shot learning using factorized weights

Problem: Large datasets are usually needed to train a powerful neural networks in a supervised manner. This is a significant limitation, because in many real situations only few training samples are available

Goal: From a single supervised example induce a full, deep discriminative model to recognize other instances of the same object class

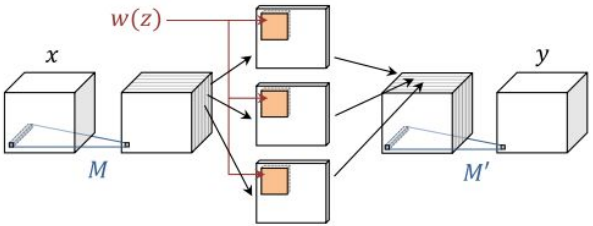
Solution: Learn a deep neural network (*hypernetwork / learnet*) that, given a single sample of a new object class, predicts the parameters of a second network (*main network*) that can recognize other objects of the same type

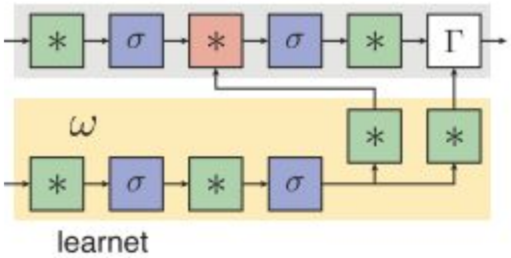
One-shot learning using factorized weights

Weights of the main network are represented in a factorized CP-format, hence

Tensor methods are effective for:

- Reducing the number of parameters that hypernetwork predicts for the main network
- Keeping the number of predicted elements to grow linearly instead of growing quadratically with the number of channels





Applied to **Character recognition, Object tracking**

Incremental multi-domain learning

Problem: Adapting the learned classification to new domains remains is a hard problem due to several arisen reasons:

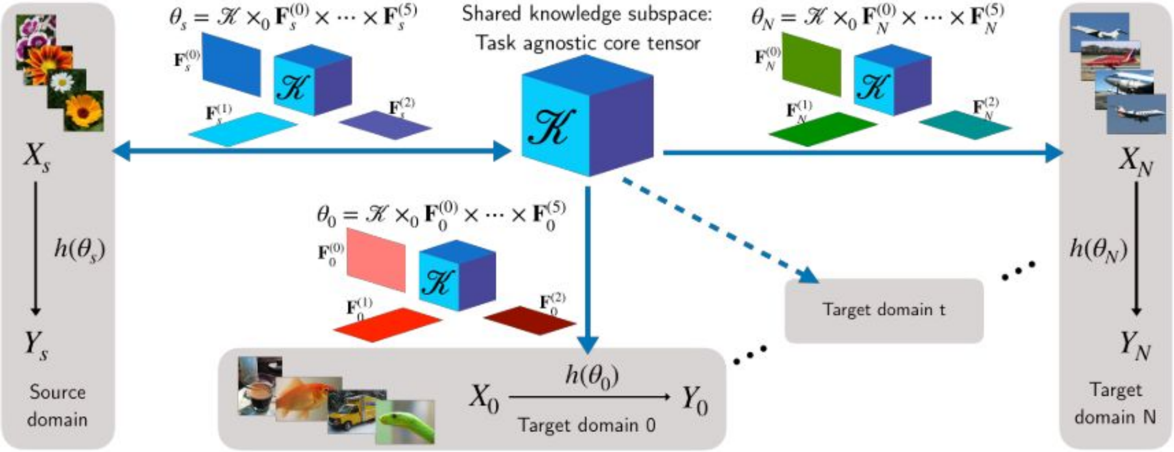
- Very different domains and tasks
- Very limited amount of annotated data on the new domain
- Full training of a new model for each new task is prohibitive in terms of memory

Goal: multi-domain/task learning without catastrophic forgetting using a fully tensorized architecture

Incremental multi-domain learning

Tensor methods are effective for:

- modeling a group of identically structured blocks within a CNN as a high-order tensor
- as a result, we leverage correlations across different layers, and obtain more compact representations for each new task/domain



Fusion of features

Problem: usually low-dimensional feature representations of different modalities (e.g. speech, images) are combined via concatenation. That accounts only first-order interactions and ignores high-order interactions.

Goal: a fusion layer that incorporates higher-order interactions.

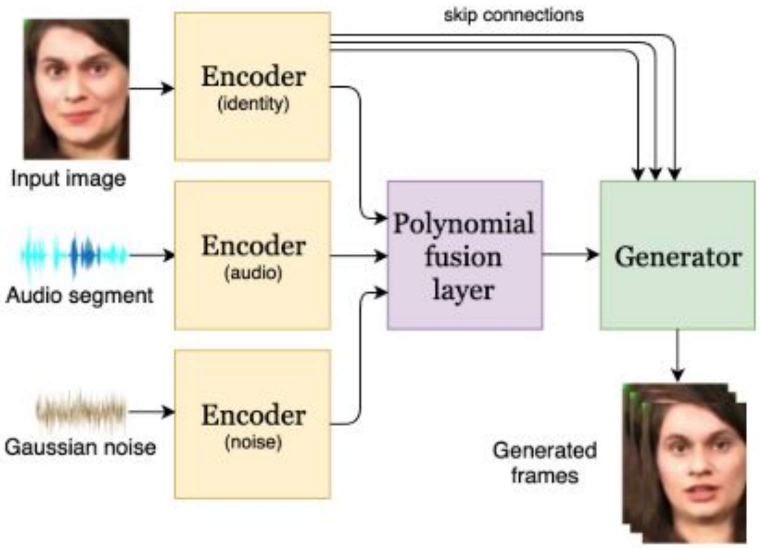
Solution: polynomial fusion layer that models the joint representation of the encodings by a higher-order polynomial

$$\tilde{\mathbf{z}} = \mathbf{b} + \mathbf{W}^{[a]}\mathbf{z}_a + \mathbf{W}^{[d]}\mathbf{z}_d + \mathcal{W}^{[a,d]} \times_2 \mathbf{z}_a \times_3 \mathbf{z}_d$$

Tensor methods are effective for: modelling parameters of higher-order polynomial by a tensor decomposition (W^[a,d] - the tensor of second-order interactions between audio and identity)

Fusion of features

- Applied to **Speech-driven facial animation**
- Fusion layer: $\tilde{\mathbf{z}} = \mathbf{b} + \mathbf{W}^{[a]}\mathbf{z}_a + \mathbf{W}^{[d]}\mathbf{z}_d + \mathcal{W}^{[a,d]} \times_2 \mathbf{z}_a \times_3 \mathbf{z}_d$
- W^[a,d] - the tensor of second-order interactions between audio and identity, W^[a], W^[d] - first-order interaction matrices for audio and identity respectively



Take home message

- Tensor methods are useful to compress and accelerate neural networks
    - Redundancy in a neural network is reduced by replacing convolutional weights with their low-rank tensor approximations,
    - As a result, we can achieve a smaller and faster network, which performs on par with the initial network
    - Convolutional layers can be approximated separately or jointly (in the second case we get better parameter reduction due to presence of shared factors)
- Many applications can benefit from using architectures, where weights are represented in a low-rank factorized format: one-shot/incremental learning, domain adaptation.
- Tensor methods is an effective way to fuse features of different modalities, while saving high-order interactions among them.

References (Speed-up and Compression)

- Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., & Lempitsky, V. (2014). Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition. http://arxiv.org/abs/1412.6553
- Kim, Y.-D., Park, E., Yoo, S., Choi, T., Yang, L., & Shin, D. (2015). Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications. http://arxiv.org/abs/1511.06530
- Kossaifi, J., Bulat, A., Tzimiropoulos, G., & Pantic, M. (2019). T-Net: Parametrizing Fully Convolutional Nets with a Single High-Order Tensor. http://arxiv.org/abs/1904.02698
- J. Kossaifi, Z. C. Lipton, A. Khanna, T. Furlanello, and A. Anandkumar (2017) “Tensor Regression Networks,” Jul. 2017. http://arxiv.org/abs/1707.08308
- Kossaifi, J., Bulat, A., Panagakis, Y., & Pantic, M. (2019). Efficient N-Dimensional Convolutions via Higher-Order Factorization. http://arxiv.org/abs/1906.06196
- Bulat, A., Kossaifi, J., Tzimiropoulos, G., & Pantic, M. (2019). Matrix and tensor decompositions for training binary neural networks. *http://arXiv.org/abs/1904.07852*
- Gusak, Julia, et al. (2019) "Automated multi-stage compression of neural networks." *Proceedings of the IEEE International Conference on Computer Vision Workshops*. http://openaccess.thecvf.com/content_ICCVW_2019/papers/LPCV/Gusak_Automated_Multi-Stage_Compression_of_Neural_Networks_ICCVW_2019_paper.pdf
- Phan, Anh-Huy, et al. (2020) "Stable Low-rank Tensor Decomposition for Compression of Convolutional Neural Network." *European Conference on Computer Vision*. *http://arxiv.org/abs/2008.05441*

References (other applications)

- L. Bertinetto, J. F. Henriques, J. Valmadre, P. H. S. Torr, and A. Vedaldi, (2016) “Learning feed-forward one-shot learners,” https://arxiv.org/abs/1606.05233
- A. Bulat, J. Kossaifi, G. Tzimiropoulos, and M. Pantic, (2019) “Incremental multi-domain learning with network latent tensor factorization,” https://arxiv.org/abs/1904.06345
- Kefalas, K. Vougioukas, Y. Panagakis, S. Petridis, J. Kossaifi, and M. Pantic, (2019) “Speech-driven facial animation using polynomial fusion of features,” https://arxiv.org/abs/1912.05833