On the Expressive Power of Deep Learning: A Tensor Analysis

Nadav Cohen

The Hebrew University of Jerusalem

GAMM 2017 Minisymposium:
Nonlinear Approximations for High-dimensional Problems

Sources

Deep SimNets
N. Cohen, O. Sharir and A. Shashua
Computer Vision and Pattern Recognition (CVPR) 2016

On the Expressive Power of Deep Learning: A Tensor Analysis
N. Cohen, O. Sharir and A. Shashua
Conference on Learning Theory (COLT) 2016

Convolutional Rectifier Networks as Generalized Tensor Decompositions
N. Cohen and A. Shashua
International Conference on Machine Learning (ICML) 2016

Inductive Bias of Deep Convolutional Networks through Pooling Geometry
N. Cohen and A. Shashua
International Conference on Learning Representations (ICLR) 2017

Boosting Dilated Convolutional Networks with Mixed Tensor Decompositions
N. Cohen, R. Tamari and A. Shashua
arXiv preprint 2017

Outline

Expressiveness

The driving force behind deep networks is their expressiveness

Fundamental theoretical questions:

- What kind of functions can different network architectures represent?
- Why are these functions suitable for real-world tasks?
- What is the representational benefit of depth?
- Can other architectural features deliver representational benefits?

Expressiveness – Basic Concepts

Universality:
Network can realize any func if its size (width) is unlimited

Efficiency:
Architecture A is efficient w.r.t. architecture B if:

(1) $\forall$func realized by B w/size r_B can be realized by A w/size $r_A \in \mathcal{O}(r_B)$

(2) $\exists$func realized by A w/size r_A requiring B to have size $r_B \in \Omega(f(r_A))$, where $f(\cdot)$ is super-linear

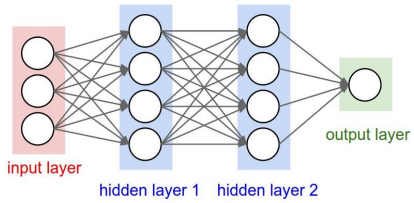
Complete efficiency:
Set of func realized by A for which (2) does not hold has measure zero

Inductive bias:
Relaxation in requirements, based on assumptions regarding task at hand

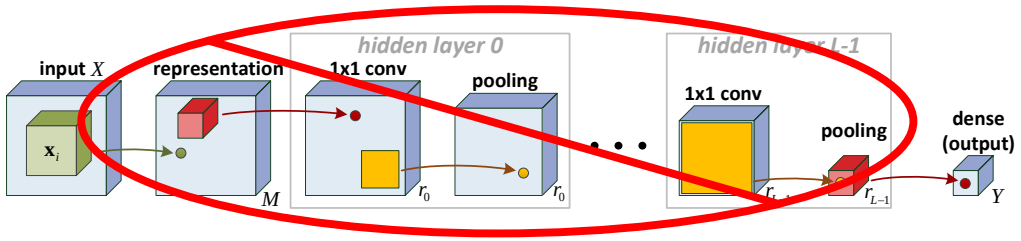
Expressiveness – Prior Works

Existing results:

- Prove (universality and) that efficiency of depth exists
  - Do not provide any information on how frequent it is
  - Do not consider other forms of efficiency
  - Do not treat inductive bias
- Apply only to fully-connected networks, not the architectures commonly used in practice (e.g. convolutional networks)



fully-connected

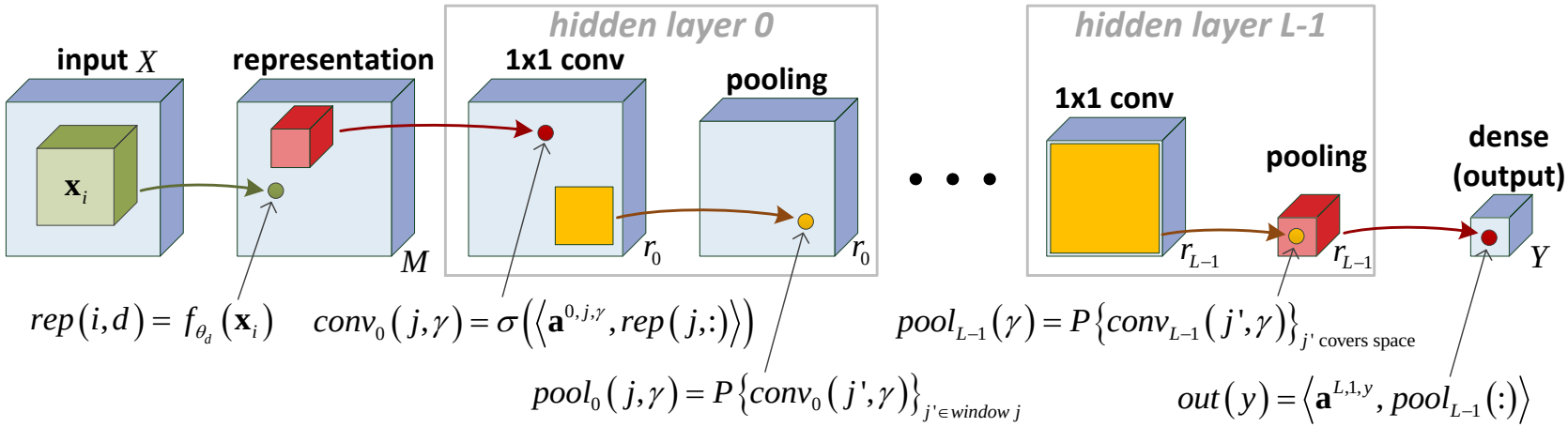


convolutional

Outline

Convolutional Arithmetic Circuits

Convolutional networks – locality, weight sharing, pooling:



$\sigma(\cdot)$ – point-wise activation

$P\{\cdot\}$ – pooling operator

Convolutional arithmetic circuits are a special case:

- linear activation: $\sigma(z) = z$
- product pooling: $P\{c_j\} = \prod_j c_j$

Computation in log-space leads to **SimNets** – new deep learning architecture showing promising empirical performance [1]

[1] *Deep SimNets, Cohen-Sharir-Shashua, CVPR'16*

Outline

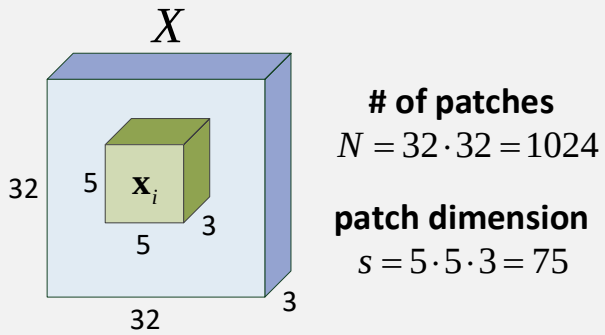
Tensorial Function Spaces

Represent instances as N-tuples of vectors ("**patches**"):

$$X = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \in (\mathbb{R}^s)^N$$

Example

32x32 RGB image represented via 5x5 patches around all pixels:



# of patches

$N = 32 \cdot 32 = 1024$

patch dimension

$s = 5 \cdot 5 \cdot 3 = 75$

Let $f_{\theta_1} \ldots f_{\theta_M} : \mathbb{R}^s \to \mathbb{R}$ be func over patches; denote $\mathcal{F} := span\{f_{\theta_1} \ldots f_{\theta_M}\}$

Extension of $\mathcal{F}$ from patches to instances:

$$\mathcal{F}^{\otimes N} := span\left\{(\mathbf{x}_1, \ldots, \mathbf{x}_N) \mapsto \prod_{i=1}^{N} f_{\theta_{d_i}}(\mathbf{x}_i) \;\; : \;\; d_1 \ldots d_N \in [M]\right\}$$

(tensor product of $\mathcal{F}$ with itself N times)

Coefficient Tensors

$$\mathcal{F}^{\otimes N} := span\left\{(\mathbf{x}_1, \ldots, \mathbf{x}_N) \mapsto \prod_{i=1}^{N} f_{\theta_{d_i}}(\mathbf{x}_i) \;:\; d_1 \ldots d_N \in [M]\right\}$$

General func $h(\cdot) \in \mathcal{F}^{\otimes N}$ can be written as:

$$h(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{d_1 \ldots d_N = 1}^{M} \mathcal{A}_{d_1, \ldots, d_N} \prod_{i=1}^{N} f_{\theta_{d_i}}(\mathbf{x}_i)$$

where $\mathcal{A} \in \mathbb{R}^{M \times \cdots \times M}$ is the **coefficient tensor** of $h(\cdot)$

Naïve computation of $h(\cdot)$ is intractable – exponential # (M^N) of terms!

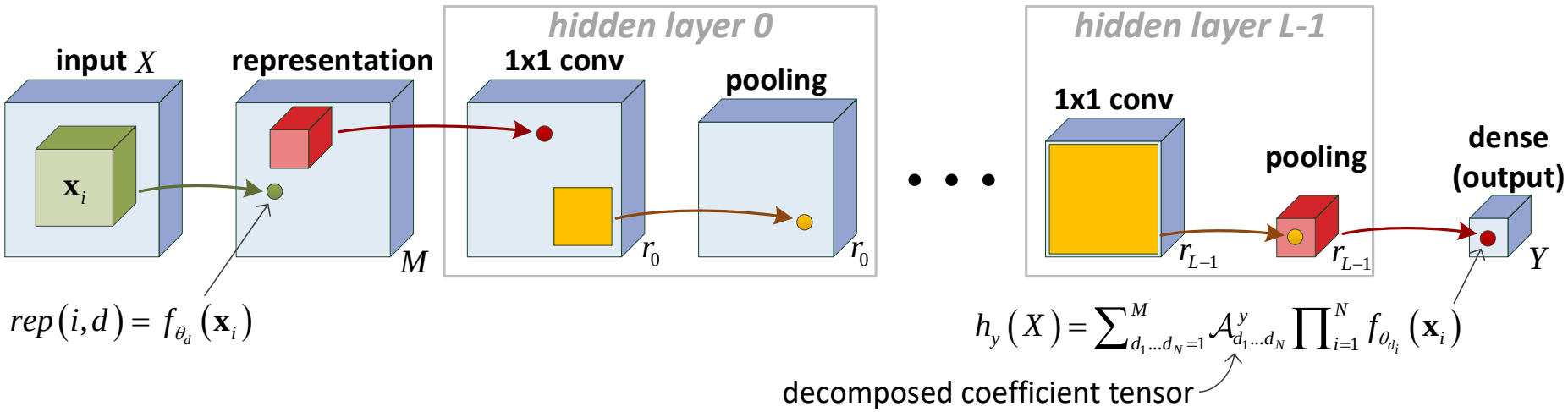
Can be made tractable by decomposing (approximating) coefficient tensor

Computing Functions by Decomposing Coefficient Tensors

$h_1 \ldots h_Y$ – set of func over instances:

$$h_y(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{d_1 \ldots d_N = 1}^{M} \mathcal{A}^y_{d_1, \ldots, d_N} \prod_{i=1}^{N} f_{\theta_{d_i}}(\mathbf{x}_i)$$

With tensor decompositions applied to $\{\mathcal{A}^y\}_y$, the func $\{h_y(\cdot)\}_y$ are computed by convolutional arithmetic circuits!



1-1 correspondence between type of tensor decomposition and structure of network (# of layers, pooling schemes, layer widths etc)

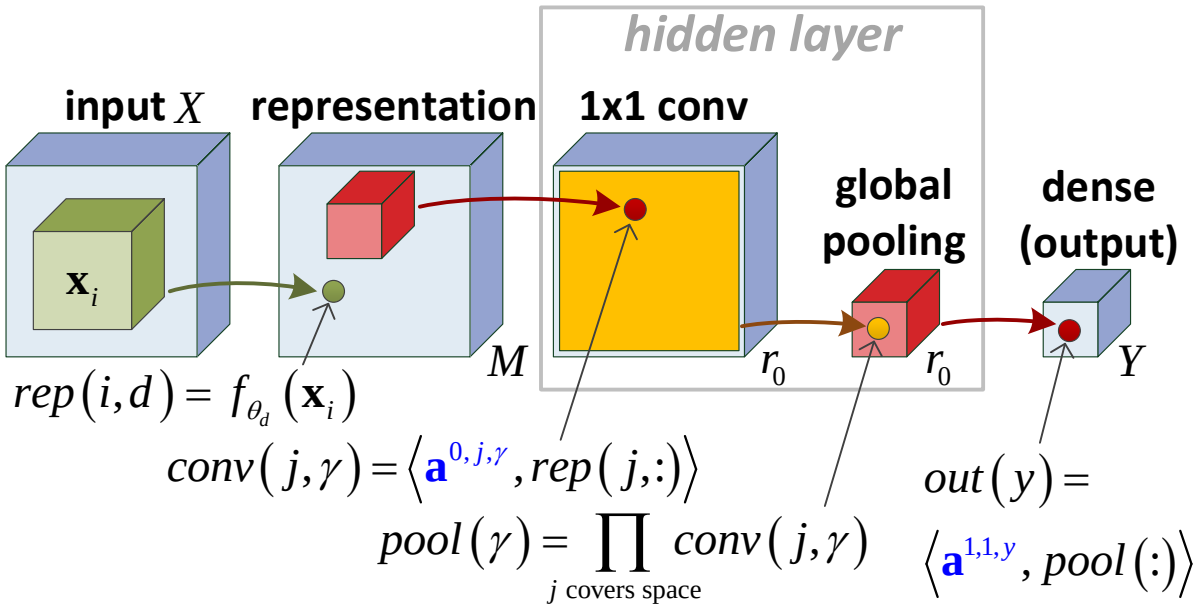
CP (CANDECOMP/PARAFAC) Decomposition $\longleftrightarrow$ Shallow Convolutional Arithmetic Circuit

Classic **CP decomposition** of coefficient tensors $\{\mathcal{A}^y\}_y$:

$$\mathcal{A}^y = \sum_{\gamma=1}^{r_0} a_\gamma^{1,1,y} \cdot \underbrace{\mathbf{a}^{0,1,\gamma} \otimes \mathbf{a}^{0,2,\gamma} \otimes \cdots \otimes \mathbf{a}^{0,N,\gamma}}_{\text{rank-1 tensor}}$$

$$(rank(\mathcal{A}^y) \leq r_0)$$

corresponds to shallow network (single hidden layer, global pooling):

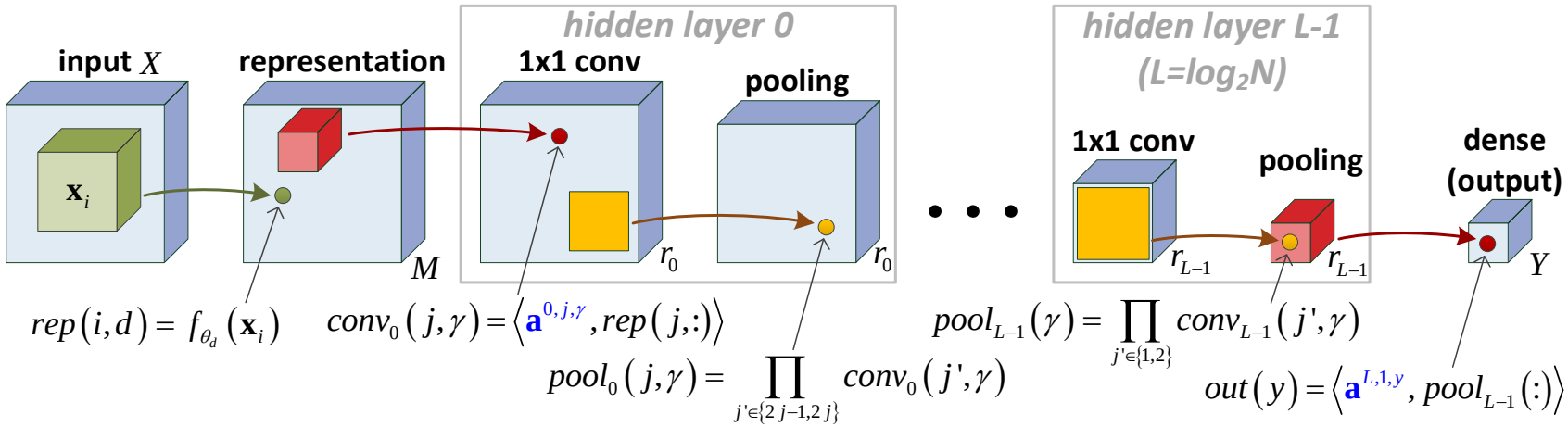


Hierarchical Tucker Decomposition $\longleftrightarrow$ Deep Convolutional Arithmetic Circuit

Hierarchical Tucker decomposition of coefficient tensors $\{\mathcal{A}^y\}_y$:

$$\begin{aligned}
\phi^{1,j,\gamma} &= \sum_{\alpha=1}^{r_0} a_\alpha^{1,j,\gamma} \cdot \mathbf{a}^{0,2j-1,\alpha} \otimes \mathbf{a}^{0,2j,\alpha} \\
&\cdots \\
\phi^{l,j,\gamma} &= \sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,j,\gamma} \cdot \phi^{l-1,2j-1,\alpha} \otimes \phi^{l-1,2j,\alpha} \\
&\cdots \\
\mathcal{A}^y &= \sum_{\alpha=1}^{r_{L-1}} a_\alpha^{L,1,y} \cdot \phi^{L-1,1,\alpha} \otimes \phi^{L-1,2,\alpha}
\end{aligned}$$

corresponds to deep network ($L = \log_2 N$ hidden layers, size-2 pooling):



Outline

Universality

Fact:
CP decomposition can realize any tensors $\{\mathcal{A}^y\}_y$ given M^N terms

Implies:
Shallow network can realize any func (in $\mathcal{F}^{\otimes N}$) given M^N hidden channels

Fact:
Hierarchical Tucker decomposition is a superset of CP decomposition if each level has matching number of terms

Implies:
Deep network can realize any func (in $\mathcal{F}^{\otimes N}$) given M^N channels in each of its hidden layers

convolutional arithmetic circuits are universal

Efficiency of Depth

Theorem

The rank of tensor $\mathcal{A}^y$ given by Hierarchical Tucker decomposition is exponential (in N) almost everywhere w.r.t. decomposition parameters

Since rank of $\mathcal{A}^y$ generated by CP decomposition is no more than the number of terms (# of hidden channels in shallow network):

Corollary

Almost all functions realizable by deep network cannot be approximated by shallow network with less than exponentially many hidden channels

w/convolutional arithmetic circuits efficiency of depth is complete!

Efficiency of Depth Theorem – Proof Sketch

- $[\![\mathcal{A}]\!]$ – arrangement of tensor $\mathcal{A}$ as matrix (*matricization*)
- $\odot$ – Kronecker product for matrices. Holds: $rank(A \odot B) = rank(A) \cdot rank(B)$
- Relation between tensor and Kronecker products: $[\![\mathcal{A} \otimes \mathcal{B}]\!] = [\![\mathcal{A}]\!] \odot [\![\mathcal{B}]\!]$
- Implies: $\mathcal{A} = \sum_{z=1}^{Z} \lambda_z \mathbf{v}_1^{(z)} \otimes \cdots \otimes \mathbf{v}_{2^L}^{(z)} \implies rank[\![\mathcal{A}]\!] \leq Z$
- By induction over $l = 1 \ldots L$, almost everywhere w.r.t. $\{\mathbf{a}^{l,j,\gamma}\}_{l,j,\gamma}$:

$$\forall j \in [N/2^l], \gamma \in [r_l] : rank[\![\phi^{l,j,\gamma}]\!] \geq \left(\min\{r_0, M\}\right)^{2^l/2}$$

  - Base: "SVD has maximal rank almost everywhere"
  - Step: $rank[\![\mathcal{A} \otimes \mathcal{B}]\!] = rank([\![\mathcal{A}]\!] \odot [\![\mathcal{B}]\!]) = rank[\![\mathcal{A}]\!] \cdot rank[\![\mathcal{B}]\!]$, and "linear combination preserves rank almost everywhere"

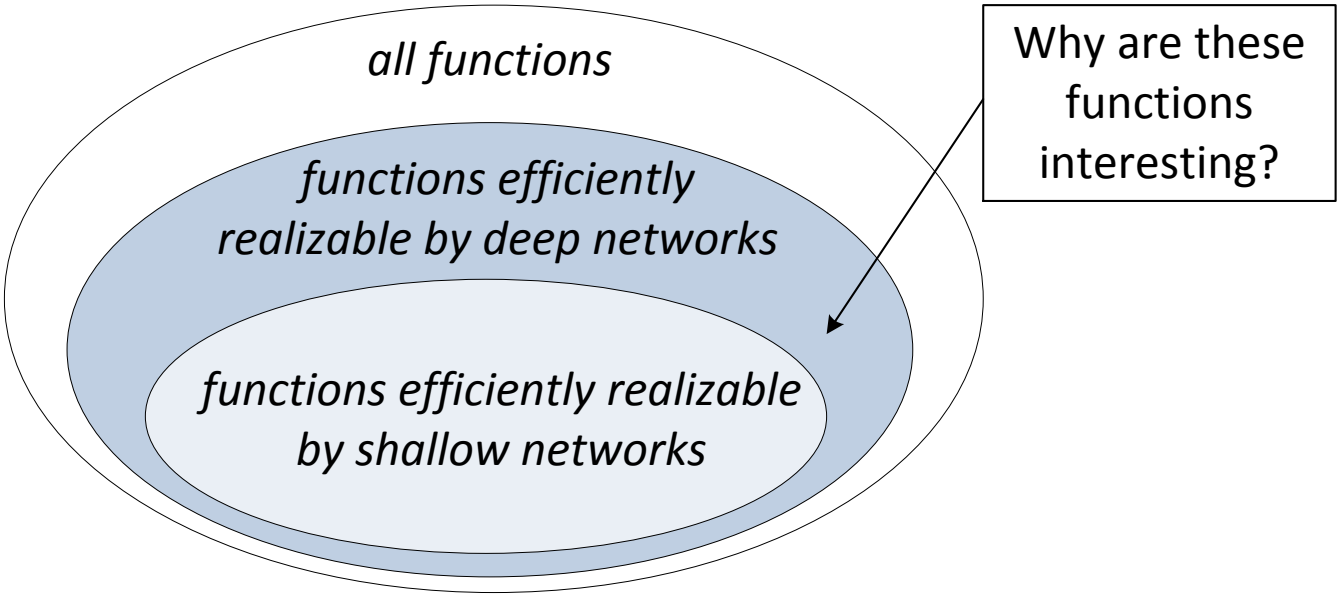
Outline

Beyond Efficiency of Depth

Efficiency of depth $\Longrightarrow$

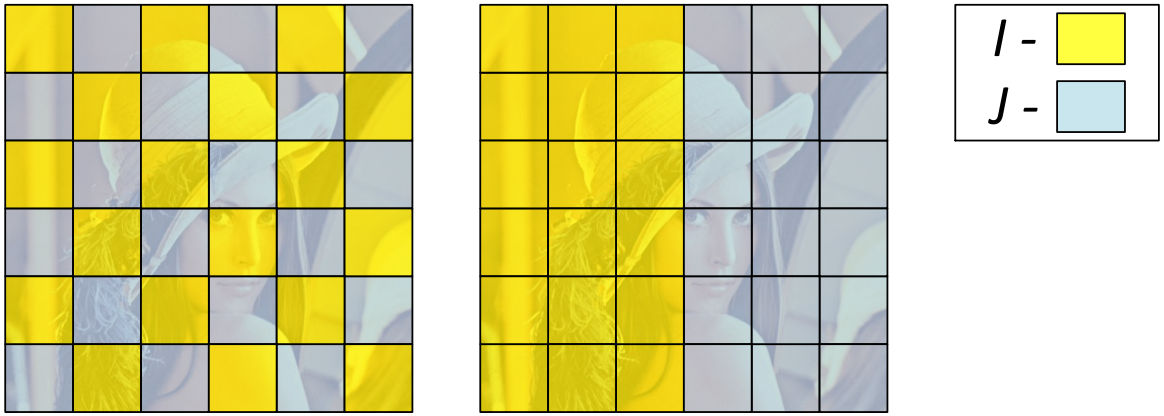
$\exists$func efficiently realizable by deep networks but not by shallow ones

Does not explain why these func are effective:



To address this, we must consider the **inductive bias** of deep architectures

Separation Rank – A Measure of Input Correlations



Partition A *Partition B*

The **separation rank** of func $h(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ w.r.t. partition $I \dot{\cup} J = [N]$:

$$sep(h; I, J) := \min \Big\{ R : \exists g_1 \ldots g_R \ , \ g'_1 \ldots g'_R \ \ s.t. \ \ h(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{\nu=1}^{R} g_\nu((\mathbf{x}_i)_{i \in I}) \cdot g'_\nu((\mathbf{x}_j)_{j \in J}) \Big\}$$

- $sep(h; I, J) = 1 \implies$ no interaction between $(\mathbf{x}_i)_{i \in I}$ and $(\mathbf{x}_j)_{j \in J}$
- $sep(h; I, J) \ \nearrow \implies$ more interaction between $(\mathbf{x}_i)_{i \in I}$ and $(\mathbf{x}_j)_{j \in J}$

Separation Ranks of Convolutional Arithmetic Circuits

Let:

- h_y – func realized by convolutional arithmetic circuit
- $\mathcal{A}^y$ – its coefficient tensor

Denote:

$$[\![\mathcal{A}^y]\!]_{I,J} \text{ – } \textbf{matricization of } \mathcal{A}^y \textbf{ according to partition } \mathbf{I \dot{\cup} J = [N]}$$

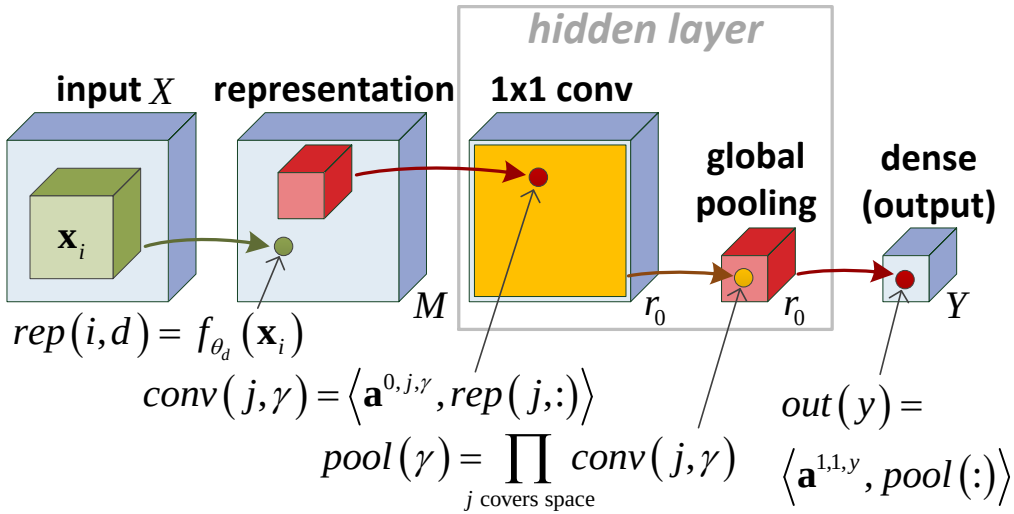
Claim

$$sep(h_y; I, J) = rank[\![\mathcal{A}^y]\!]_{I,J}$$

We thus study correlations modeled by convolutional arithmetic circuits through ranks of matricized coefficient tensors

Shallow Separation Ranks

Shallow convolutional arithmetic circuit (CP decomposition):



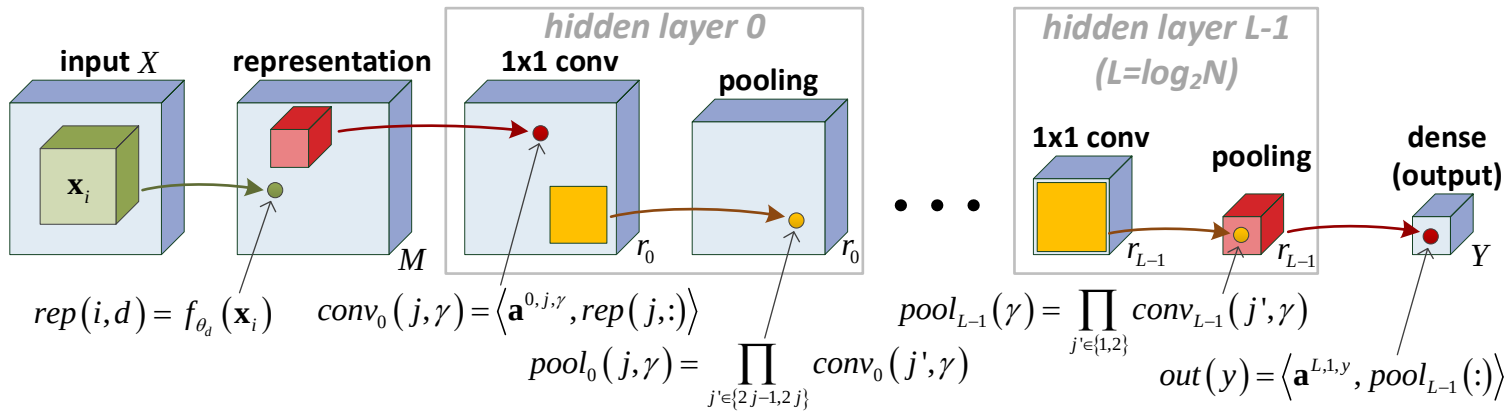
Claim

$$rank[\![\mathcal{A}^y]\!]_{I,J} \leq r_0$$

shallow network only realizes separation ranks (correlations) linear in its size

Deep Separation Ranks

Deep convolutional arithmetic circuit (Hierarchical Tucker decomposition):



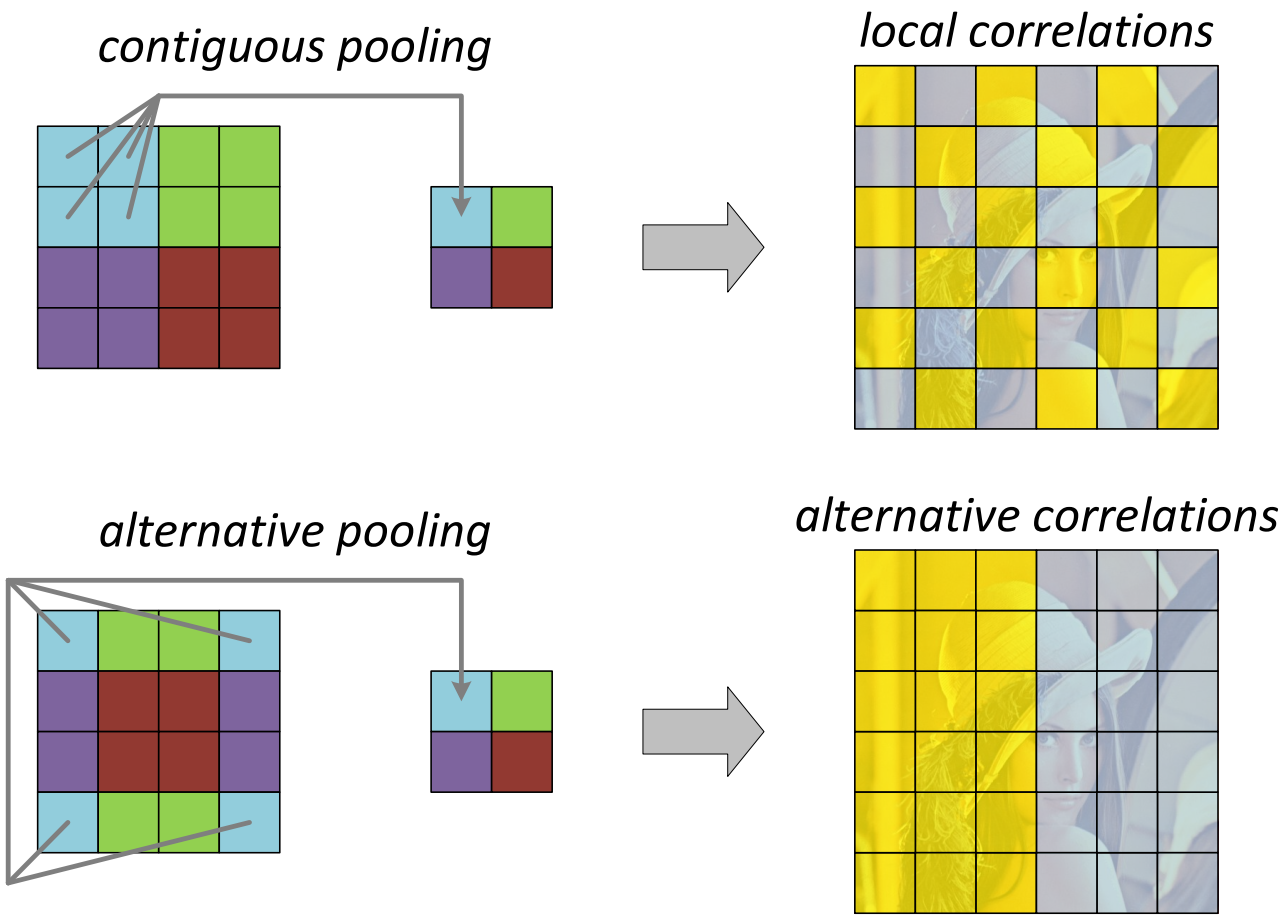
Theorem

Maximal rank that $[\![\mathcal{A}^y]\!]_{I,J}$ can take is:

- *Exponential (in N) for "interleaved" partitions*
- *Polynomial (in network size) for "coarse" partitions*

deep network realizes exponential separation ranks (correlations) for favored partitions, polynomial (in network size) for others

Inductive Bias through Pooling Geometry

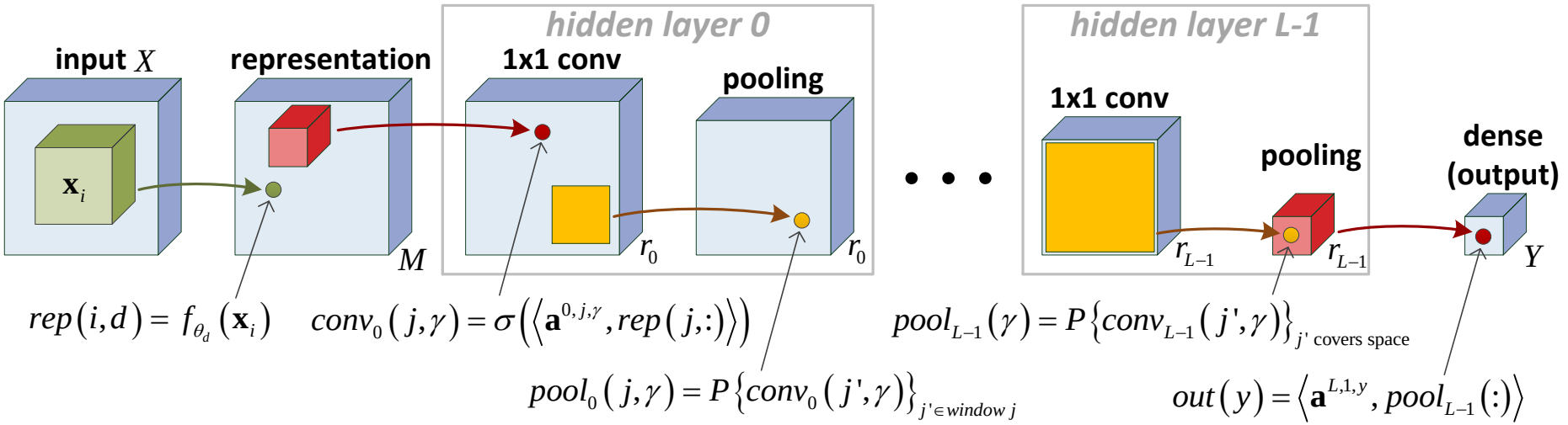


deep network's pooling geometry determines which input patterns can have high separation ranks (correlations), thus controls inductive bias

Outline

1. The Expressive Power of Deep Learning
2. Convolutional Arithmetic Circuits *(COLT'16, ICLR'17)*
   - Equivalence to Tensor Decompositions
   - Universality and Efficiency of Depth
   - Inductive Bias
3. **Convolutional Rectifier Networks** *(ICML'16)*
   - Equivalence to Generalized Tensor Decompositions
   - Universality and Efficiency of Depth
4. Dilated Convolutional Networks *(arXiv'17)*
   - Mode Trees and Dilations
   - Mixing Decompositions and Networks
   - Efficiency of Interconnectivity

From Convolutional Arithmetic Circuits to Convolutional Rectifier Networks



Transform convolutional arithmetic circuits into convolutional rectifier networks:

linear activation $\longrightarrow$ **ReLU activation**: $\sigma(z) = \max\{z, 0\}$

product pooling $\longrightarrow$ **max/average pooling**: $P\{c_j\} = max\{c_j\}/mean\{c_j\}$

Most successful deep learning architecture to date!

Outline

Generalized Tensor Decompositions

Convolutional arithmetic circuits correspond to tensor decompositions based on tensor product $\otimes$:

$$(\mathcal{A} \otimes \mathcal{B})_{d_1,\ldots,d_{P+Q}} = \mathcal{A}_{d_1,\ldots,d_P} \cdot \mathcal{B}_{d_{P+1},\ldots,d_{P+Q}}$$

For an operator $g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, the **generalized tensor product** $\otimes_g$:

$$(\mathcal{A} \otimes_g \mathcal{B})_{d_1,\ldots,d_{P+Q}} := g(\mathcal{A}_{d_1,\ldots,d_P}, \mathcal{B}_{d_{P+1},\ldots,d_{P+Q}})$$

(same as $\otimes$ but with $g(\cdot)$ instead of multiplication)

Generalized tensor decompositions are obtained by replacing $\otimes$ with $\otimes_g$

Convolutional Rectifier Networks $\longleftrightarrow$ Generalized Tensor Decompositions

Define the **activation-pooling operator**:

$$\rho_{\sigma/P}(a, b) := P\{\sigma(a), \sigma(b)\}$$

Convolutional rectifier networks are equivalent to generalized tensor decompositions with $g(\cdot) \equiv \rho_{\sigma/P}(\cdot)$:

Shallow network $\longleftrightarrow$ *Generalized CP decomposition*

Deep network $\longleftrightarrow$ *Generalized Hierarchical Tucker decomposition*

Outline

Results

Universality:

Claim

Convolutional rectifier networks are universal with max pooling, but not with average pooling

Efficiency of depth:

Claim

With convolutional rectifier networks efficiency of depth exists, but it is not complete

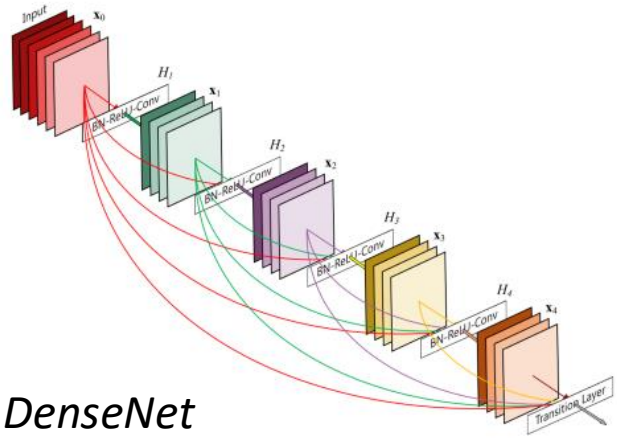
expressiveness of convolutional rectifier networks inferior to that of convolutional arithmetic circuits!

Outline

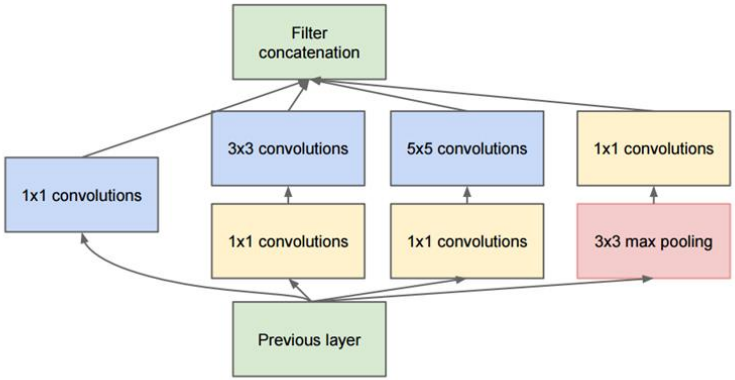
Connectivity

To date, only efficiency of depth was treated

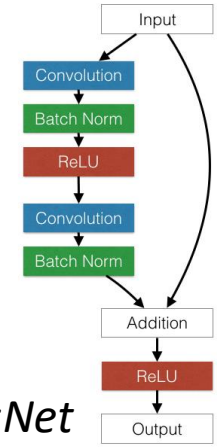
This overlooks architectural feature of **connectivity**, present in nearly all state of the art networks



DenseNet



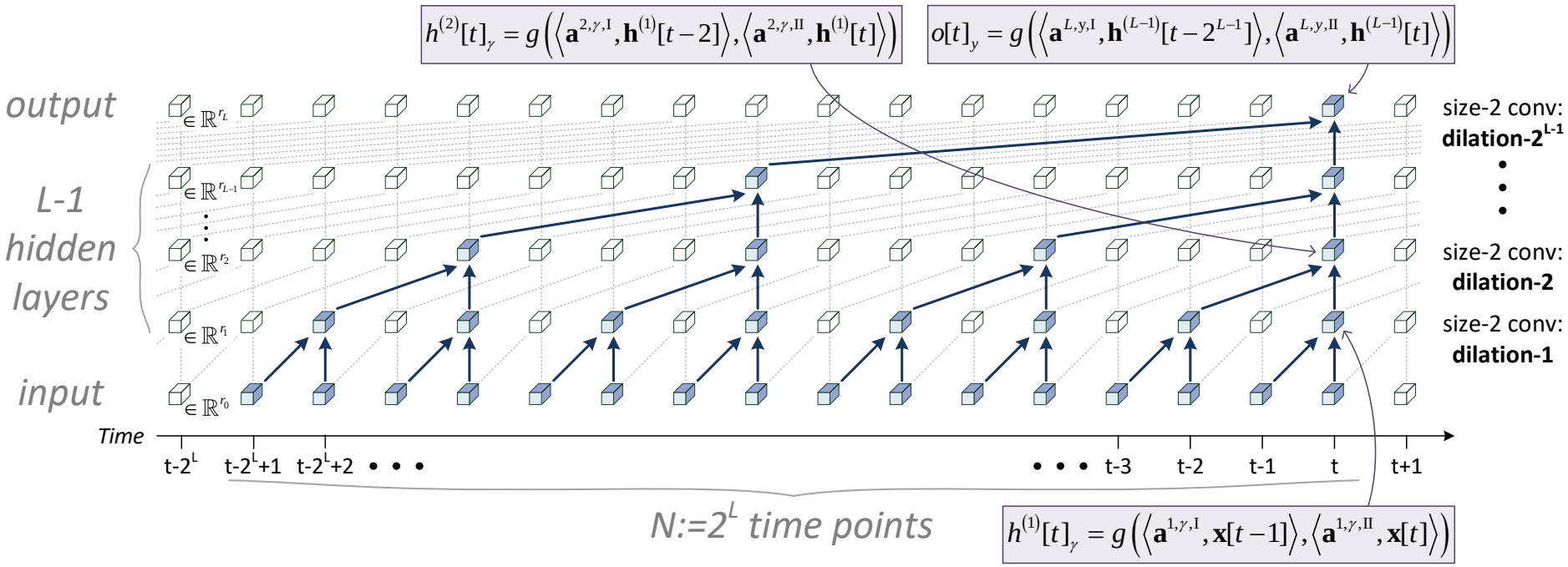
Inception (GoogLeNet)



ResNet

We study efficiency of connectivity in **dilated convolutional networks** – state of the art in audio and text processing tasks

Baseline Dilated Convolutional Network



1D convolutional network with:

- **dilation** (gap) 2^{l-1} in layer $l = 1, \ldots, L$
- no pooling

Underlies Google's WaveNet – state of the art in audio processing

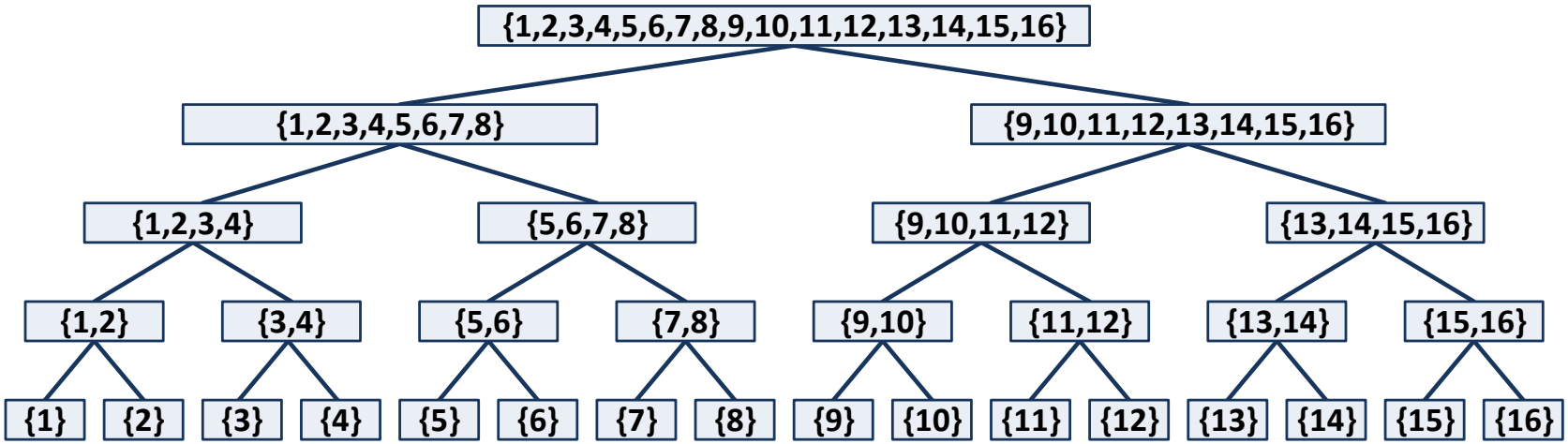
Outline

Baseline Mode Tree

Baseline network corresponds to Hierarchical Tucker decomposition:

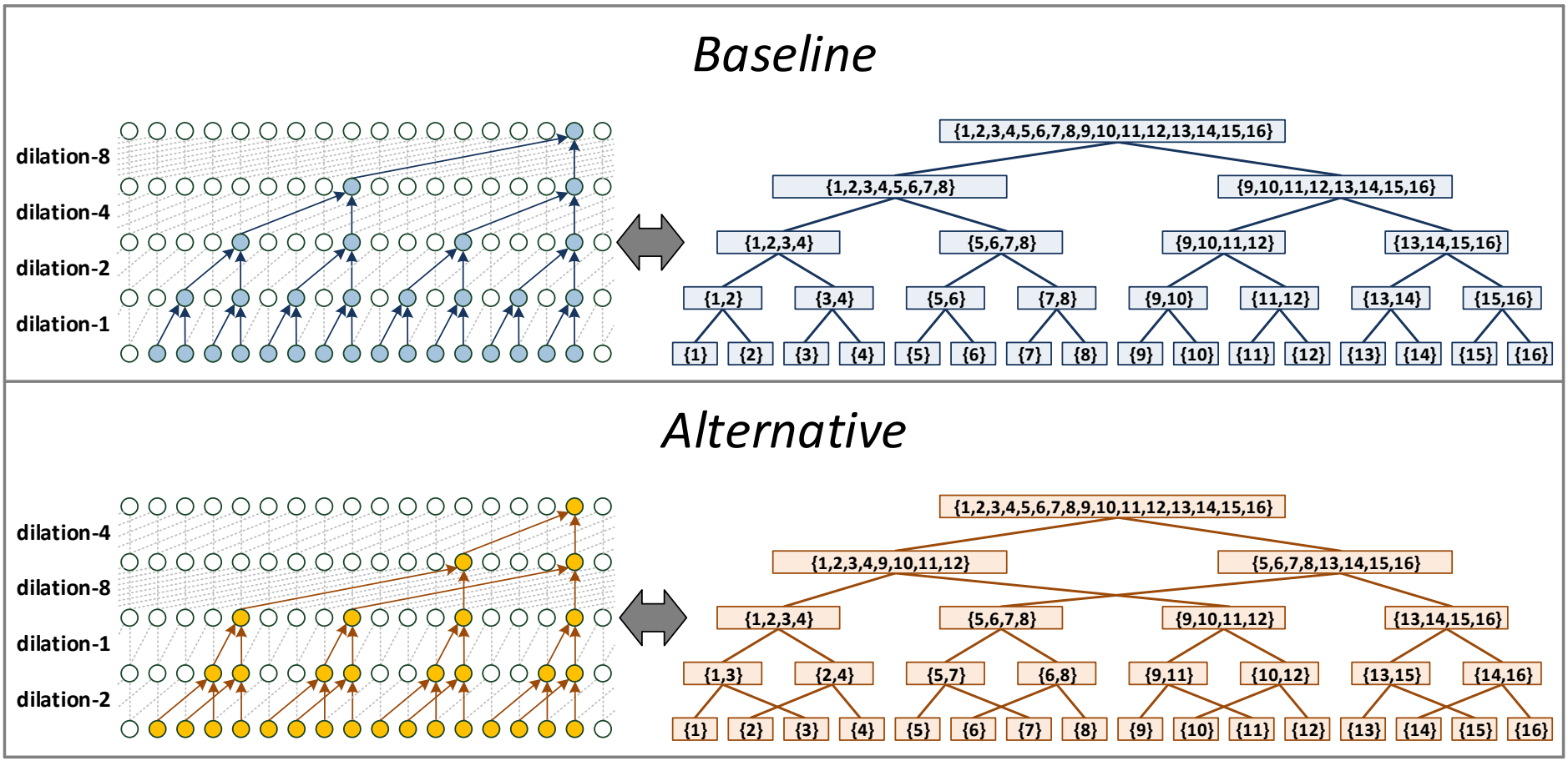
$$\begin{aligned}
\phi^{1,j,\gamma} &= \sum_{\alpha=1}^{r_0} a_\alpha^{1,j,\gamma} \cdot \mathbf{a}^{0,2j-1,\alpha} \otimes \mathbf{a}^{0,2j,\alpha} \\
&\cdots \\
\phi^{l,j,\gamma} &= \sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,j,\gamma} \cdot \phi^{l-1,2j-1,\alpha} \otimes \phi^{l-1,2j,\alpha} \\
&\cdots \\
\mathcal{A}^{y} &= \sum_{\alpha=1}^{r_{L-1}} a_\alpha^{L,1,y} \cdot \phi^{L-1,1,\alpha} \otimes \phi^{L-1,2,\alpha}
\end{aligned}$$

which adheres to a particular **mode tree** (tree over tensor modes):



Different Mode Trees $\longleftrightarrow$ Different Dilations

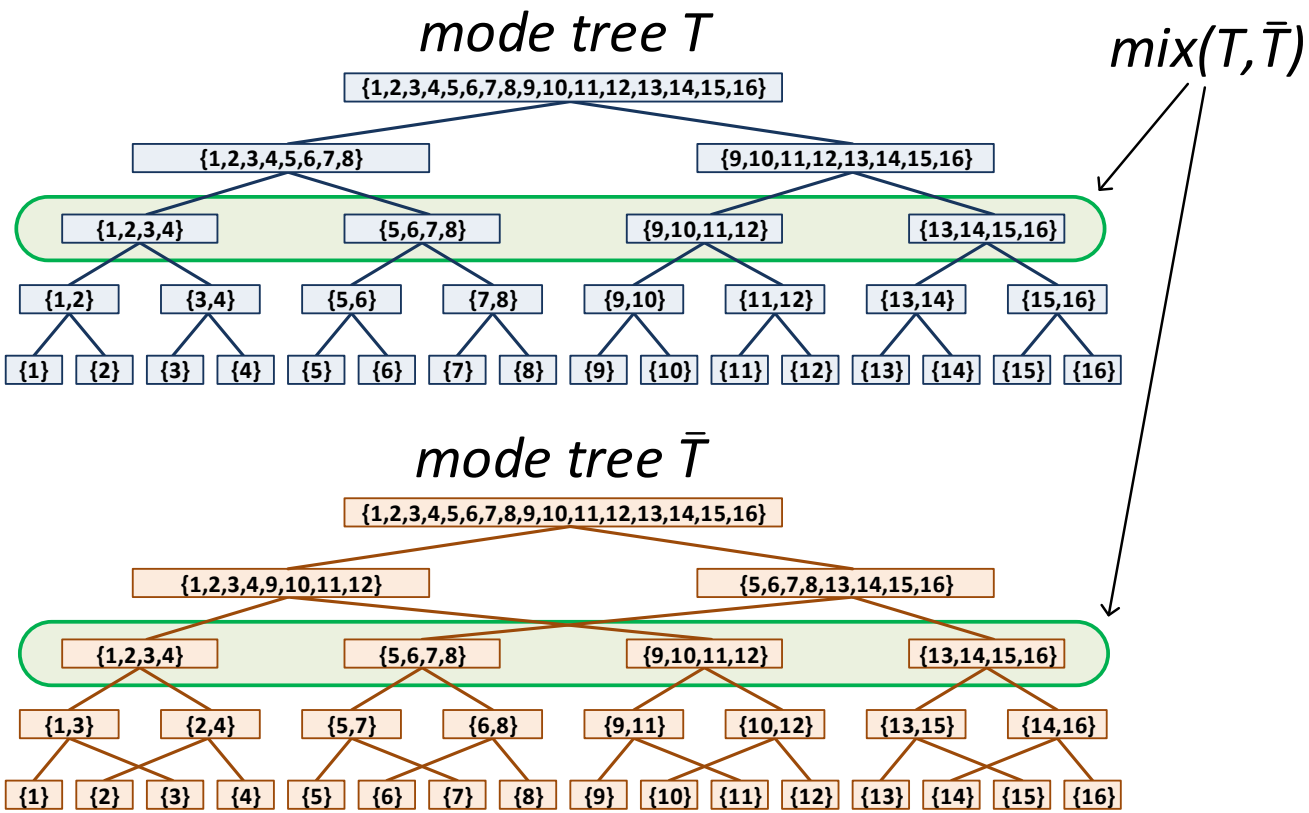
Changing underlying mode tree gives decompositions corresponding to networks with different dilations:



Outline

Mixed Tensor Decompositions

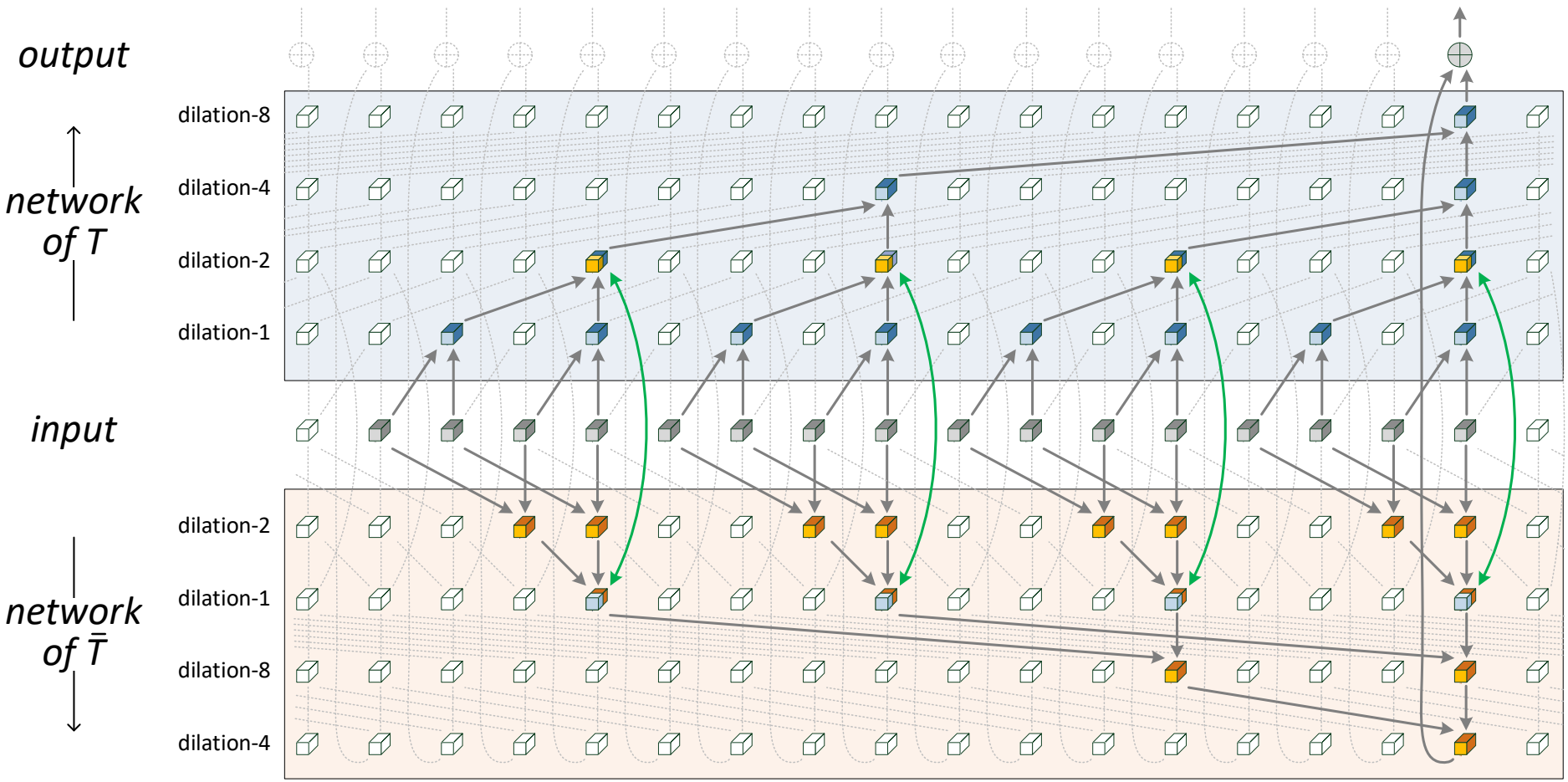
$T, \bar{T}$ – two mode trees ; $mix(T, \bar{T})$ – set of nodes present in both trees



A **mixed tensor decomposition** blends together T and $\bar{T}$ by running their decompositions in parallel, exchanging tensors in each node of $mix(T, \bar{T})$

Mixed Dilated Convolutional Networks

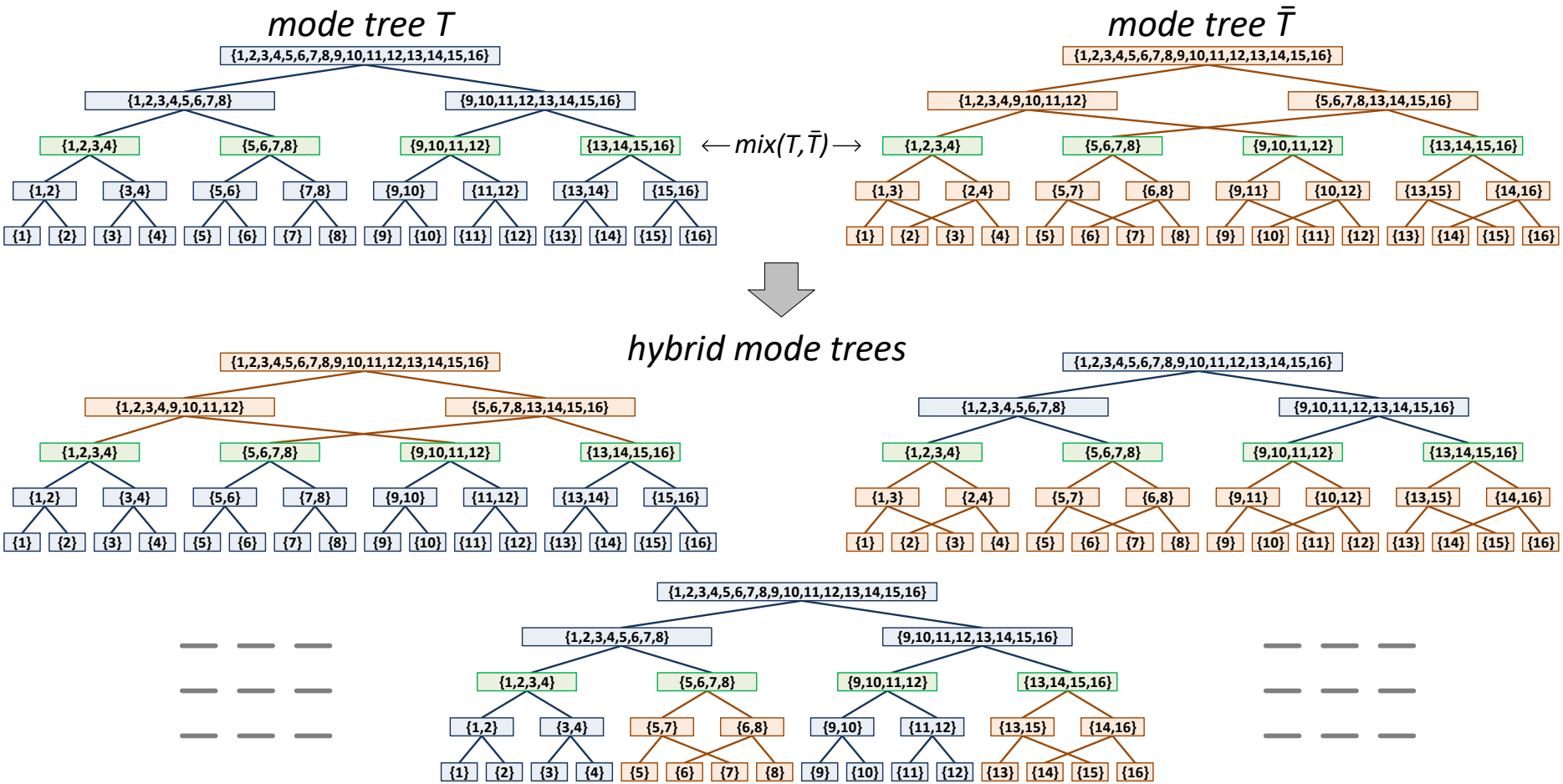
Mixed tensor decomposition corresponds to **mixed dilated convolutional network**, formed by interconnecting the networks of T and $\bar{T}$:



Outline

Hybrid Mode Trees

Mode trees T and $\bar{T}$ give rise to an exponential # of **hybrid mode trees**



Results

Claim

Any tensor generated by decomposition of a hybrid mode tree can be realized by mixed decomposition with no more than linear growth in size

Theorem

There exist hybrid mode trees whose decompositions generate tensors requiring those of T and $\bar{T}$ to grow at least quadratically

This implies:

Corollary

Mixed dilated convolutional network is efficient w.r.t. networks of T and $\bar{T}$

interconnectivity leads to expressive efficiency!

Outline

1. The Expressive Power of Deep Learning
2. Convolutional Arithmetic Circuits *(COLT'16, ICLR'17)*
   - Equivalence to Tensor Decompositions
   - Universality and Efficiency of Depth
   - Inductive Bias
3. Convolutional Rectifier Networks *(ICML'16)*
   - Equivalence to Generalized Tensor Decompositions
   - Universality and Efficiency of Depth
4. Dilated Convolutional Networks *(arXiv'17)*
   - Mode Trees and Dilations
   - Mixing Decompositions and Networks
   - Efficiency of Interconnectivity

Conclusion

- **Convolutional networks** $\longleftrightarrow$ **tensor decompositions**:

arithmetic circuits	$\longleftrightarrow$	standard decompositions
rectifier networks	$\longleftrightarrow$	generalized decompositions
interconnected networks	$\longleftrightarrow$	mixed decompositions

- **Equivalence used to analyze expressiveness**:
  - Universality
  - Efficiency of depth
  - Inductive bias: pooling geometry determines modeled correlations
  - Efficiency of interconnectivity
- Future work: use equivalence to analyze generalization/optimization

Thank You