Inductive Bias and Depth Efficiency of Convolutional Networks

Amnon Shashua

The Hebrew University of Jerusalem

Mathematics of Deep Learning, CVPR 2016

Joint work with Nadav Cohen and Or Sharir

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
arXiv preprint, May 2016

Expressiveness

The driving force behind deep networks is their expressiveness

Fundamental theoretical questions:

- What kind of functions can different network architectures represent?
- What is the representational benefit of depth?
- Why do functions realized by convolutional networks suit images?

Expressiveness: Basic Concepts

Universality:
Network is able to realize any function if its size (width) is unlimited

Expressiveness: Basic Concepts

Universality:
Network is able to realize any function if its size (width) is unlimited

Depth efficiency:
Function realized by deep network of polynomial size requires super-polynomial size for being realized (or approximated) by shallow network

Expressiveness: Basic Concepts

Universality:
Network is able to realize any function if its size (width) is unlimited

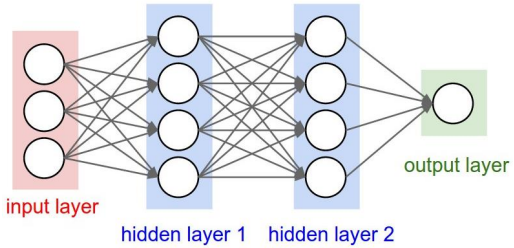
Depth efficiency:
Function realized by deep network of polynomial size requires super-polynomial size for being realized (or approximated) by shallow network

Complete depth efficiency:
The set of functions realizable by deep network for which depth efficiency does not hold is negligible (has measure zero)

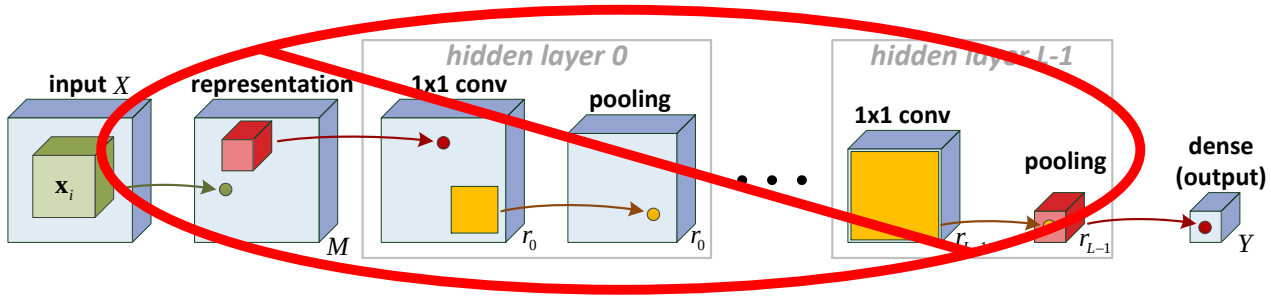
Prior Works on Expressiveness

Prior works on the expressiveness of deep networks:

- Study universality and only *existence* of depth efficiency
- Consider only fully-connected networks, not the architectures commonly used in practice (e.g. convolutional networks)



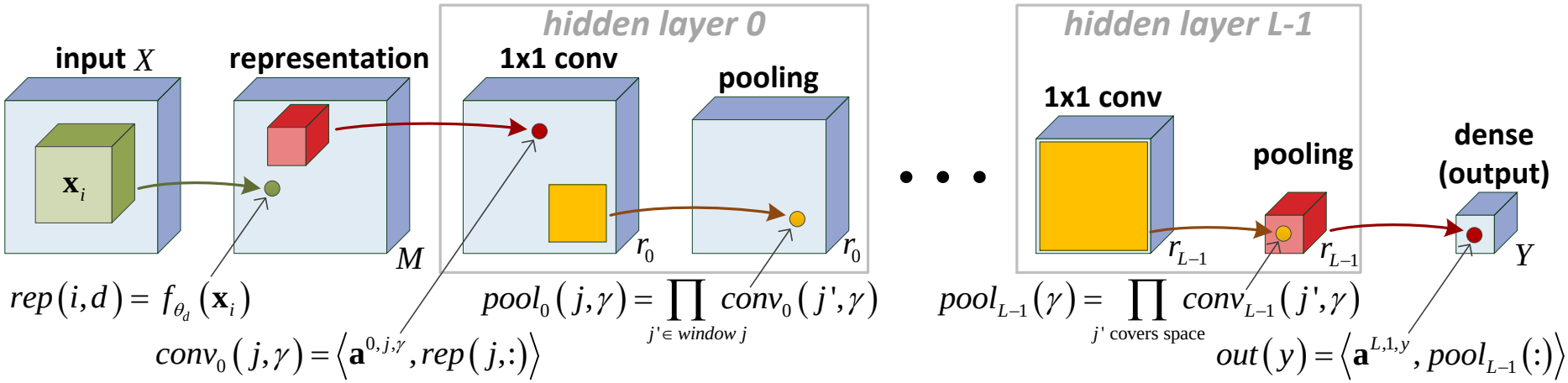
fully-connected



convolutional

Outline

Convolutional Arithmetic Circuits



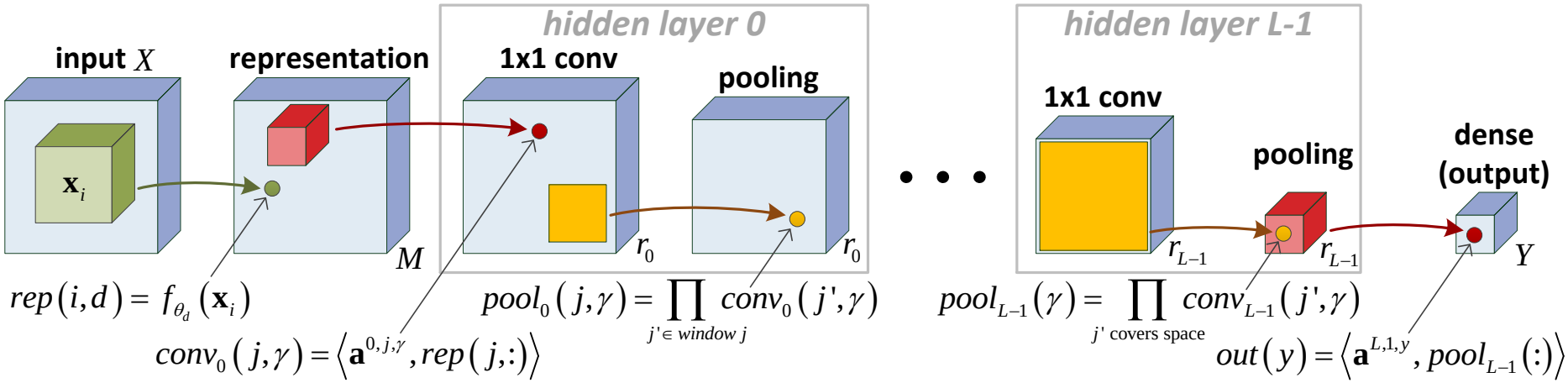
Convolutional networks:

- locality
- weight sharing (optional)
- linear activation, product pooling

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

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

Coefficient Tensor



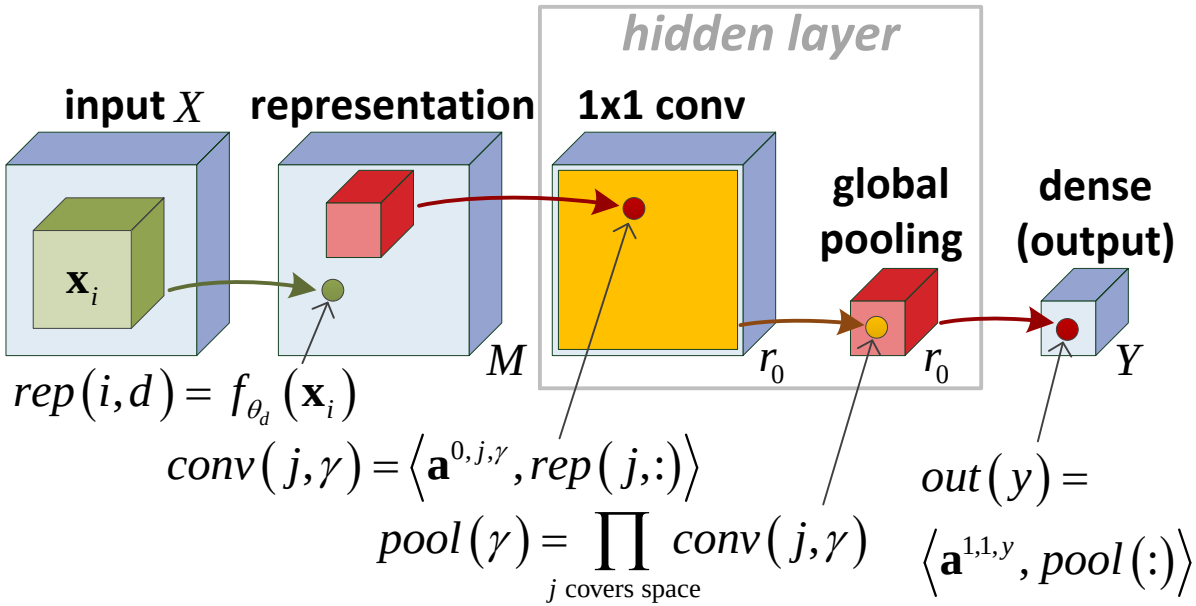
Function realized by output y:

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

- $\mathbf{x}_1 \ldots \mathbf{x}_N$ – input patches
- $f_{\theta_1} \ldots f_{\theta_M}$ – representation layer functions
- $\mathcal{A}^y$ – **coefficient tensor** (M^N entries, polynomials in weights $\mathbf{a}^{l,j,\gamma}$)

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

Shallow network (single hidden layer, global pooling):



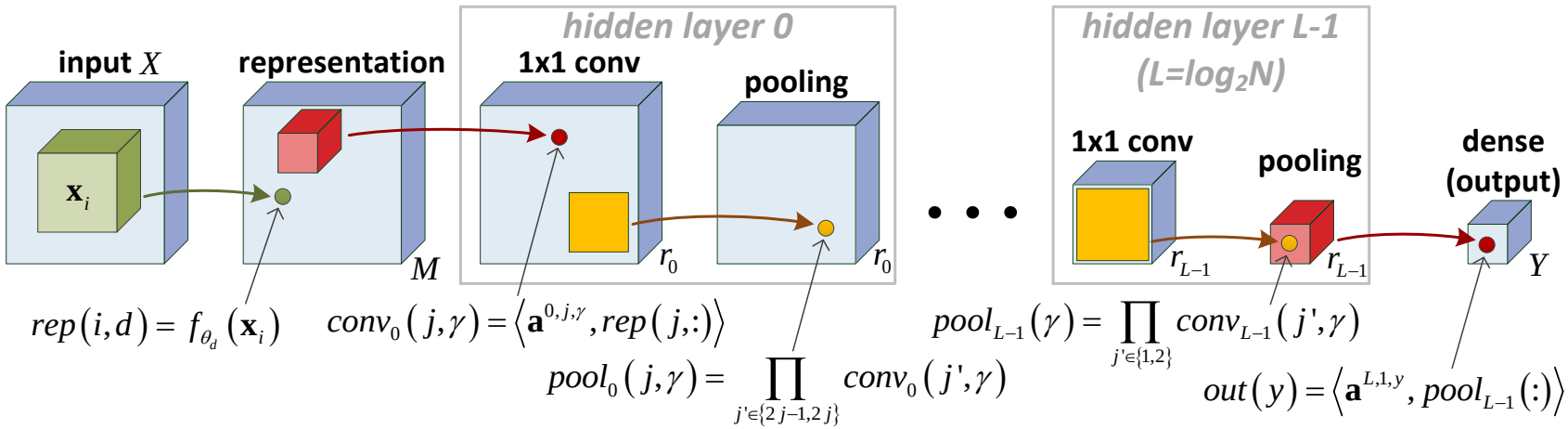
Coefficient tensor $\mathcal{A}^y$ given by classic **CP decomposition**:

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

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

Deep Convolutional Arithmetic Circuit $\longleftrightarrow$ Hierarchical Tucker Decomposition

Deep network ($L = \log_2 N$ hidden layers, size-2 pooling windows):



Coefficient tensor $\mathcal{A}^y$ given by **Hierarchical Tucker decomposition**:

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

Universality

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

Implies:
Shallow network can realize any function 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 function given M^N channels in each of its hidden layers

Convolutional arithmetic circuits are universal

Depth Efficiency

Theorem

The rank of tensor $\mathcal{A}^y$ given by Hierarchical Tucker decomposition is at least $\min\{r_0, M\}^{N/2}$ *almost everywhere w.r.t. decomposition parameters*

Depth Efficiency

Theorem

The rank of tensor $\mathcal{A}^y$ given by Hierarchical Tucker decomposition is at least $\min\{r_0, M\}^{N/2}$ *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 $\min\{r_0, M\}^{N/2}$ *hidden channels*

Depth Efficiency

Theorem

The rank of tensor $\mathcal{A}^y$ given by Hierarchical Tucker decomposition is at least $\min\{r_0, M\}^{N/2}$ 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 $\min\{r_0, M\}^{N/2}$ hidden channels

Convolutional arithmetic circuits are completely depth efficient!

Depth Efficiency Theorem – Proof Sketch

- $[\![\mathcal{A}]\!]$ – arrangement of tensor $\mathcal{A}$ as matrix (*matricization*)
- Relation between tensor and Kronecker products: $[\![\mathcal{A} \otimes \mathcal{B}]\!] = [\![\mathcal{A}]\!] \odot [\![\mathcal{B}]\!]$
- $\odot$ – Kronecker product for matrices. Holds: $rank(A \odot B) = rank(A) \cdot rank(B)$
- 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"

A Note about Measure Zero

- Depth Efficiency occurs with probability 1, i..e, besides a set of measure zero, all functions that can be implemented by a deep network of polynomial size, require exponential size in order to be realized (or even approximated) by a shallow network.

A Note about Measure Zero

- Depth Efficiency occurs with probability 1, i..e, besides a set of measure zero, all functions that can be implemented by a deep network of polynomial size, require exponential size in order to be realized (or even approximated) by a shallow network.
- The set is a zero set of a certain polynomial (based on determinants).
- The zero set of a polynomial is closed, i.e., cannot approximate anything that is not included in the set.
- In other words, the *closure* of the set is also of *measure zero*.

A Note about Measure Zero

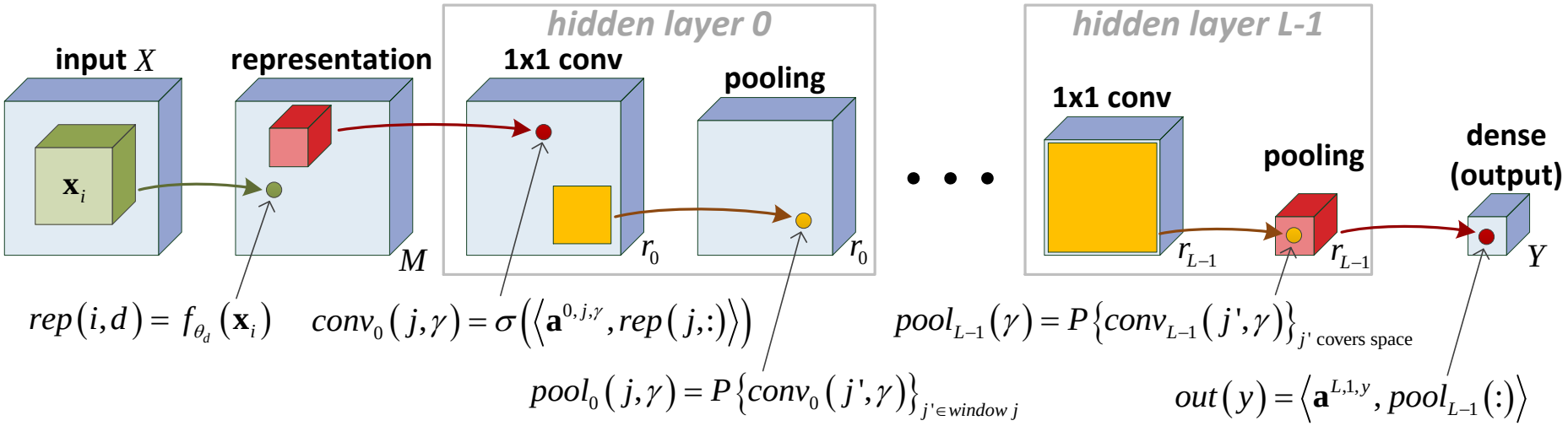
- Depth Efficiency occurs with probability 1, i..e, besides a set of measure zero, all functions that can be implemented by a deep network of polynomial size, require exponential size in order to be realized (or even approximated) by a shallow network.
- The set is a zero set of a certain polynomial (based on determinants).
- The zero set of a polynomial is closed, i.e., cannot approximate anything that is not included in the set.
- In other words, the *closure* of the set is also of *measure zero*.
- For example, the set of Rational numbers is of measure zero, but the closure of the set is **not** of measure zero. It actually fills the entire space.

A Note about Measure Zero

- Depth Efficiency occurs with probability 1, i..e, besides a set of measure zero, all functions that can be implemented by a deep network of polynomial size, require exponential size in order to be realized (or even approximated) by a shallow network.
- The set is a zero set of a certain polynomial (based on determinants).
- The zero set of a polynomial is closed, i.e., cannot approximate anything that is not included in the set.
- In other words, the *closure* of the set is also of *measure zero*.
- For example, the set of Rational numbers is of measure zero, but the closure of the set is **not** of measure zero. It actually fills the entire space.
- Therefore, the set of functions that do not satisfy depth efficiency should be viewed as a low-dimensional manifold rather than a scattered set in space.

Outline

From Convolutional Arithmetic Circuits to General Convolutional Networks



Transform convolutional arithmetic circuits into general convolutional networks:

liner activation $\longrightarrow$ general point-wise activation $\sigma(\cdot)$

product pooling $\longrightarrow$ general pooling operator $P\{\cdot\}$

Generalized Tensor Decompositions

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

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

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 associative and commutative operator $g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, the **generalized tensor product** $\otimes_g$ is defined by:

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

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

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 associative and commutative operator $g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, the **generalized tensor product** $\otimes_g$ is defined by:

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

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

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

General Convolutional Networks $\longleftrightarrow$ Generalized Tensor Decompositions

Define the **activation-pooling operator**:

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

General Convolutional Networks $\longleftrightarrow$ Generalized Tensor Decompositions

Define the **activation-pooling operator**:

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

If $\rho_{\sigma/P}$ is associative and commutative:

Shallow ConvNet with activation $\sigma(\cdot)$ and pooling $P\{\cdot\}$ $\longleftrightarrow$ Generalized *CP* decomposition with $\otimes_{\rho_{\sigma/P}}$

Deep ConvNet with activation $\sigma(\cdot)$ and pooling $P\{\cdot\}$ $\longleftrightarrow$ Generalized *Hierarchical Tucker* decomposition with $\otimes_{\rho_{\sigma/P}}$

Convolutional Rectifier Networks

Convolutional networks with:

- **ReLU activation**: $\sigma(z) = [z]_+ := \max\{z, 0\}$
- **max/average pooling**: $P\{c_j\} = max\{c_j\}/mean\{c_j\}$

Most successful deep learning architecture to date

[1] Sum and average pooling are equivalent in terms of expressiveness

Convolutional Rectifier Networks

Convolutional networks with:

- **ReLU activation**: $\sigma(z) = [z]_+ := \max\{z, 0\}$
- **max/average pooling**: $P\{c_j\} = max\{c_j\}/mean\{c_j\}$

Most successful deep learning architecture to date

Corresponding activation-pooling operators associative and commutative:

- $\rho_{ReLU/max}(a, b) := \max\{[a]_+, [b]_+\} = \max\{a, b, 0\}$
- $\rho_{ReLU/sum}(a, b) := [a]_+ + [b]_+$ [1]

Equivalence with generalized tensor decompositions thus holds!

[1] Sum and average pooling are equivalent in terms of expressiveness

Universality

Claim

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

Proof idea

- Generalized CP and Hierarchical Tucker decompositions with $\otimes_{\rho_{ReLU/max}}$ can realize any tensor given sufficient number of terms
- With $\otimes_{\rho_{ReLU/sum}}$ the decompositions generate tensors that have low rank when arranged as matrices

Depth Efficiency

Claim

Convolutional rectifier networks realize depth efficient functions

Proof idea

With $\otimes_{\rho_{ReLU/max}}$ the rank of a generated tensor arranged as a matrix is:

- *Generalized CP decomposition*:
  Linear in number of terms (# of hidden channels in shallow network)
- *Generalized Hierarchical Tucker decomposition*:
  Exponentially high for appropriately chosen weight settings

Depth Efficiency (cont')

Claim

*Convolutional rectifier networks are **not** completely depth efficient*

Proof idea

With $\otimes_{\rho_{ReLU/max}}$ there exist weight settings $\mathbf{a}^{l,j,\gamma}$ for generalized Hierarchical Tucker decomposition such that:

- Realized tensor can be implemented by generalized CP decomposition with single term (single hidden channel in shallow network)
- The same holds if $\mathbf{a}^{l,j,\gamma}$ are subject to small perturbations

Convolutional Rectifier Networks vs. Convolutional Arithmetic Circuits

	convolutional rectifier networks	*convolutional arithmetic circuits*
universality	*applies w/max pool not w/average pool*	*applies*
depth efficiency	*incomplete*	*complete*
optimization methods	*well studied and developed*	*addressed only recently* [1]

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

Convolutional Rectifier Networks vs. Convolutional Arithmetic Circuits

	convolutional rectifier networks	*convolutional arithmetic circuits*
universality	*applies w/max pool not w/average pool*	*applies*
depth efficiency	*incomplete*	*complete*
optimization methods	*well studied and developed*	*addressed only recently* [1]

Developing optimization methods for convolutional arithmetic circuits may give rise to an architecture that is provably superior but has so far been overlooked

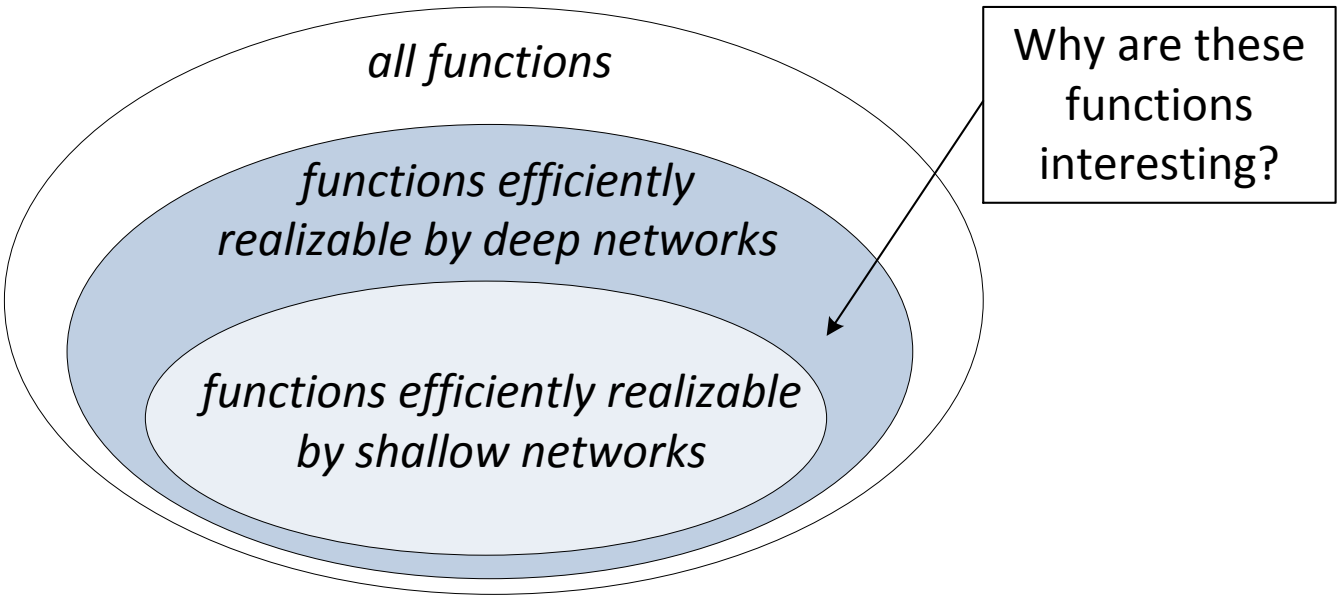
[1] *Deep SimNets, Cohen-Sharir-Shashua, CVPR'16*

Outline

Limitations of Depth Efficiency

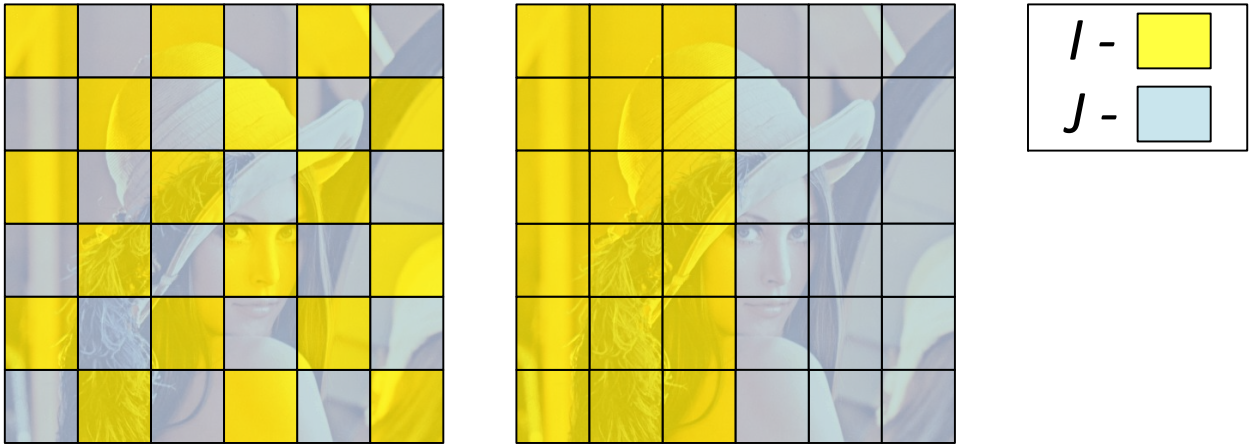
Depth efficiency implies that there are functions efficiently realizable by deep networks but not by shallow ones.

It does not explain why these functions are effective.



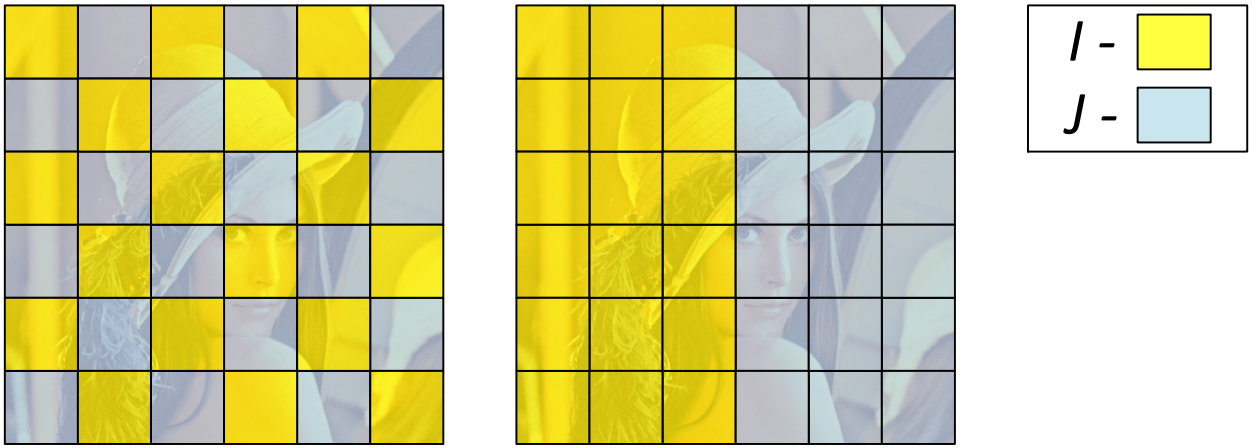
Moreover, it does not compare different networks of the same depth, and thus does not shed light on the **inductive bias** of deep architectures.

Separation Rank – A Measure of Input Correlations



Partition A *Partition B*

Separation Rank – A Measure of Input Correlations



Partition A *Partition B*

The **separation rank** of a function $h(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ w.r.t. the partition $I \cup J = [N], I = \{i_1, \ldots, i_{|I|}\}, J = \{j_1, \ldots, j_{|J|}\}$:

$$sep(h; I, J) := \min \Big\{ R \in \mathbb{N} : \exists g_1 \ldots g_R : (\mathbb{R}^s)^{|I|} \to \mathbb{R}, g'_1 \ldots g'_R : (\mathbb{R}^s)^{|J|} \to \mathbb{R} \;\; s.t. \; h(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{\nu=1}^{R} g_\nu(\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_{|I|}}) g'_\nu(\mathbf{x}_{j_1}, \ldots, \mathbf{x}_{j_{|J|}}) \Big\}$$

Measures correlation induced by h between $(\mathbf{x}_{i_1} \ldots \mathbf{x}_{i_{|I|}})$ and $(\mathbf{x}_{j_1} \ldots \mathbf{x}_{j_{|J|}})$.

We analyze the separation ranks of convolutional arithmetic circuits.

Separation Ranks of Convolutional Arithmetic Circuits

Recall expression for function realized by convolutional arithmetic circuit:

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

- $\mathbf{x}_1\ldots\mathbf{x}_N\in\mathbb{R}^s$ – local image patches
- $\mathcal{A}^y\in\mathbb{R}^{M\times\cdots\times M}$ – *coefficient tensor*, N **modes** (indexing entries)

Separation Ranks of Convolutional Arithmetic Circuits

Recall expression for function realized by convolutional arithmetic circuit:

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

- $\mathbf{x}_1\ldots\mathbf{x}_N\in\mathbb{R}^s$ – local image patches
- $\mathcal{A}^y\in\mathbb{R}^{M\times\cdots\times M}$ – *coefficient tensor*, N **modes** (indexing entries)

Define $[\![\mathcal{A}^y]\!]_{I,J}$ – **matricization of $\mathcal{A}^y$ w.r.t. the partition $I\dot{\cup}J=[N]$**:

- Arrangement of $\mathcal{A}^y$ as matrix
- Rows correspond to modes indexed by I
- Columns correspond to modes indexed by J

Separation Ranks of Convolutional Arithmetic Circuits

Recall expression for function realized by convolutional arithmetic circuit:

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

- $\mathbf{x}_1\ldots\mathbf{x}_N\in\mathbb{R}^s$ – local image patches
- $\mathcal{A}^y\in\mathbb{R}^{M\times\cdots\times M}$ – *coefficient tensor*, N **modes** (indexing entries)

Define $[\![\mathcal{A}^y]\!]_{I,J}$ – **matricization of $\mathcal{A}^y$ w.r.t. the partition $I\dot\cup J=[N]$**:

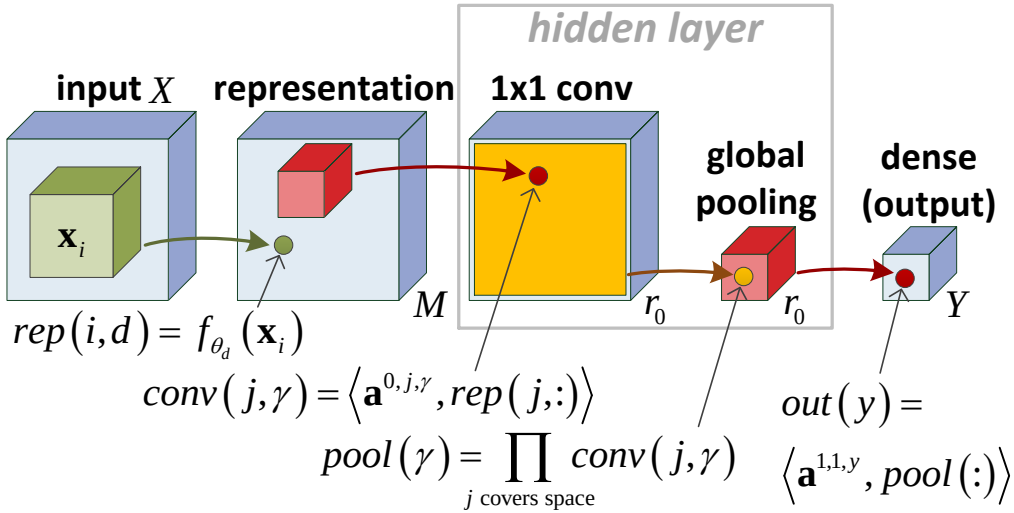
- Arrangement of $\mathcal{A}^y$ as matrix
- Rows correspond to modes indexed by I
- Columns correspond to modes indexed by J

Claim

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

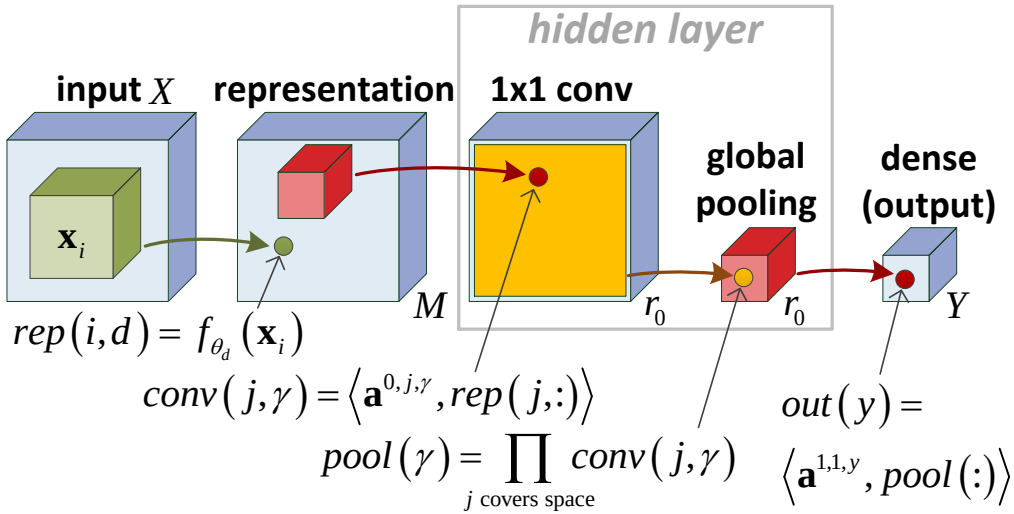
Shallow Separation Ranks

Shallow convolutional arithmetic circuit (single hidden layer):



Shallow Separation Ranks

Shallow convolutional arithmetic circuit (single hidden layer):



Claim

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

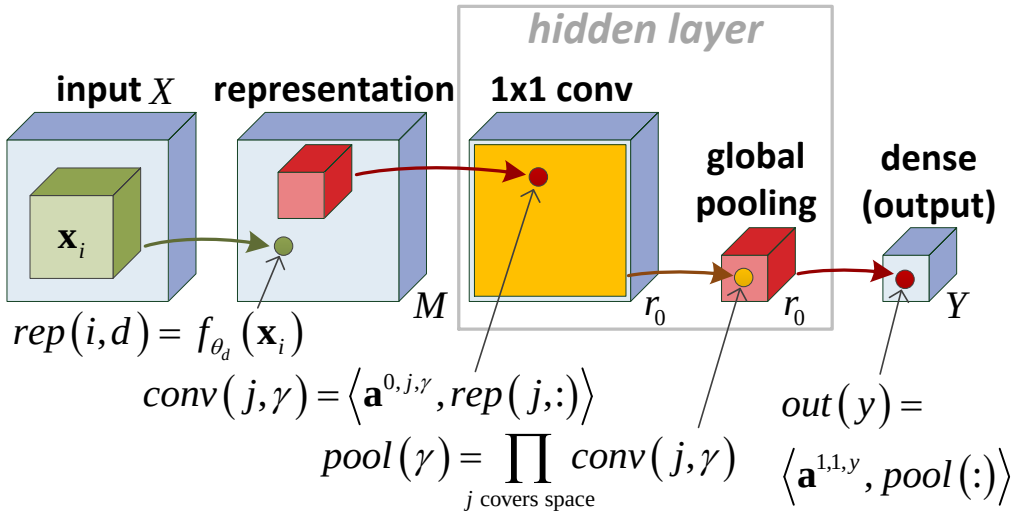
Proof sketch

Matricizing CP decomposition gives ($\odot$ – Kronecker product):

$$[\![\mathcal{A}^y]\!]_{I,J} = \sum_{\gamma=1}^{r_0} a_\gamma^{1,1,y} \cdot \left(\odot_{t=1}^{|I|} \mathbf{a}^{0,i_t,\gamma}\right) \left(\odot_{t=1}^{|J|} \mathbf{a}^{0,j_t,\gamma}\right)^\top$$

Shallow Separation Ranks

Shallow convolutional arithmetic circuit (single hidden layer):



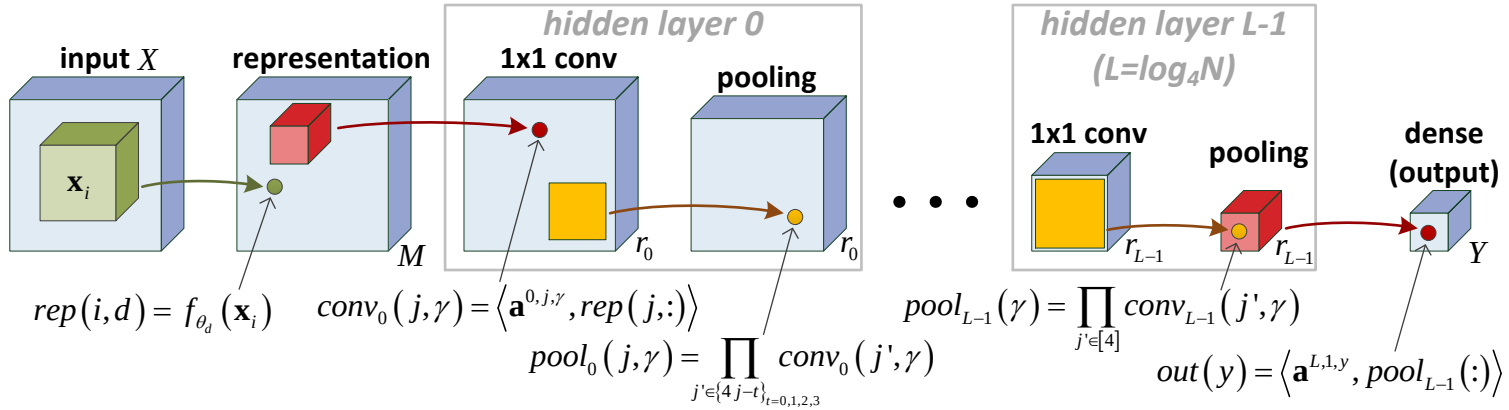
Claim

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

Shallow networks only realize separation ranks (correlations) linear in their size

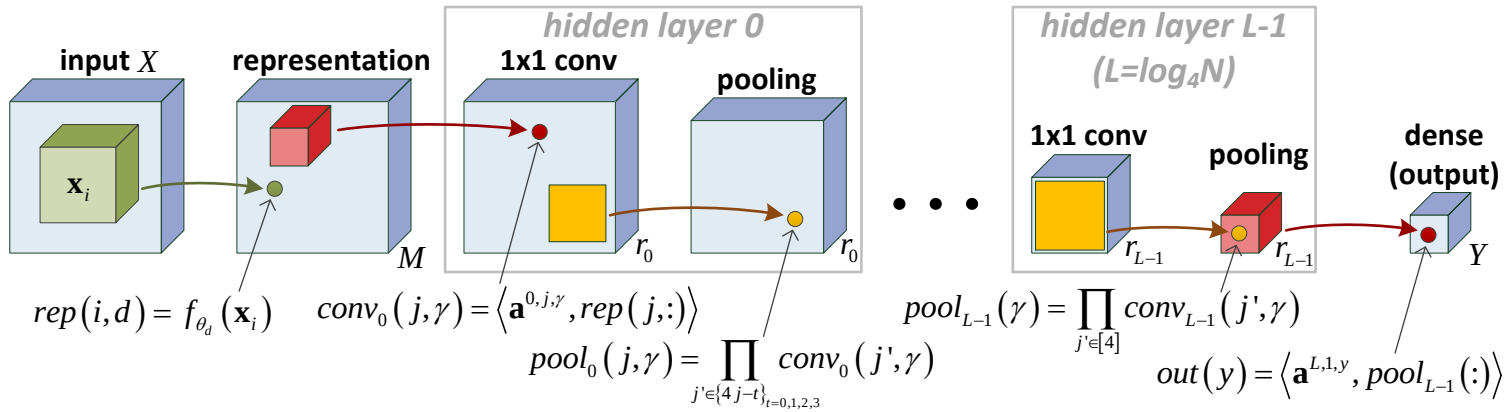
Deep Separation Ranks

Deep convolutional arithmetic circuit ($L = \log_4 N$ hidden layers):



Deep Separation Ranks

Deep convolutional arithmetic circuit ($L = \log_4 N$ hidden layers):



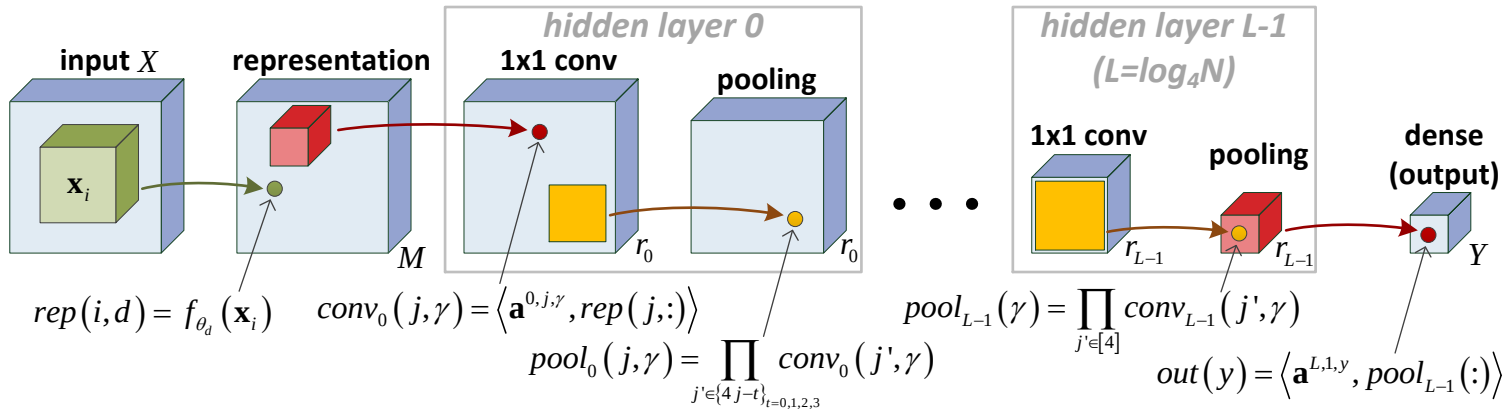
Theorem

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

- *Exponential (in N) for "interleaved" partitions,*
  e.g. $\geq \min\{r_0, M\}^{N/4}$ *for* $I = \{1, 3, \ldots, N-1\}, J = \{2, 4, \ldots, N\}$
- *Polynomial (in network size) for "coarse" partitions,*
  e.g. $\leq r_{L-1}$ *for* $I = \{1, \ldots, N/2\}, J = \{N/2+1, \ldots, N\}$

Deep Separation Ranks

Deep convolutional arithmetic circuit ($L = \log_4 N$ hidden layers):



Theorem

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

- *Exponential (in N) for "interleaved" partitions,*
  e.g. $\geq \min\{r_0, M\}^{N/4}$ for $I = \{1, 3, \ldots, N-1\}, J = \{2, 4, \ldots, N\}$
- *Polynomial (in network size) for "coarse" partitions,*
  e.g. $\leq r_{L-1}$ for $I = \{1, \ldots, N/2\}, J = \{N/2+1, \ldots, N\}$

Deep networks realize exponential separation ranks (correlations) for favored partitions, polynomial (in network size) for others

Deep Separation Ranks Theorem – Proof Sketch

- For tensors $\mathcal{B}, \mathcal{C}$ with P, Q modes resp, and a partition $T \cup S = [P+Q]$:

$$[\![\mathcal{B} \otimes \mathcal{C}]\!]_{T,S} = [\![\mathcal{B}]\!]_{T\cap[P],S\cap[P]} \odot [\![\mathcal{C}]\!]_{(T-P)\cap[Q],(S-P)\cap[Q]}$$

- Recursive application to the hierarchical decomposition of $\mathcal{A}^y$ gives:

$$\begin{aligned}
[\![\phi^{1,k,\gamma}]\!]_{I_{1,k},J_{1,k}} &= \sum_{\alpha=1}^{r_0} a_\alpha^{1,\gamma} \cdot \bigodot_{t=1}^{4} [\![\mathbf{a}^{0,4(k-1)+t,\alpha}]\!]_{I_{0,4(k-1)+t},J_{0,4(k-1)+t}} \\
&\cdots \\
[\![\phi^{l,k,\gamma}]\!]_{I_{l,k},J_{l,k}} &= \sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,\gamma} \cdot \bigodot_{t=1}^{4} [\![\phi^{l-1,4(k-1)+t,\alpha}]\!]_{I_{l-1,4(k-1)+t},J_{l-1,4(k-1)+t}} \\
&\cdots \\
[\![\mathcal{A}^y]\!]_{I,J} &= \sum_{\alpha=1}^{r_{L-1}} a_\alpha^{L,y} \cdot \bigodot_{t=1}^{4} [\![\phi^{L-1,t,\alpha}]\!]_{I_{L-1,t},J_{L-1,t}}
\end{aligned}$$

where $I \cup J = [N]$ is an arbitrary partition and:

$$I_{l,k} := (I - (k-1)\cdot 4^l) \cap [4^l]$$

$$J_{l,k} := (J - (k-1)\cdot 4^l) \cap [4^l]$$

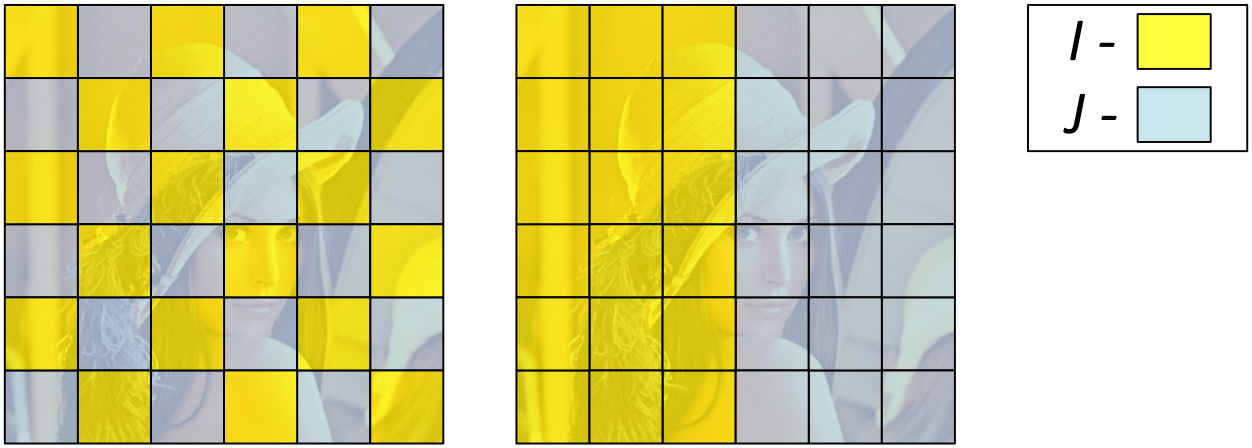
Deep Separation Ranks Theorem – Proof Sketch (cont')

$$
\begin{aligned}
[\![\phi^{1,k,\gamma}]\!]_{I_{1,k},J_{1,k}} &= \sum_{\alpha=1}^{r_0} a_\alpha^{1,\gamma} \cdot \bigodot_{t=1}^{4} [\![\mathbf{a}^{0,4(k-1)+t,\alpha}]\!]_{I_{0,4(k-1)+t},J_{0,4(k-1)+t}} \\
&\cdots \\
[\![\phi^{l,k,\gamma}]\!]_{I_{l,k},J_{l,k}} &= \sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,\gamma} \cdot \bigodot_{t=1}^{4} [\![\phi^{l-1,4(k-1)+t,\alpha}]\!]_{I_{l-1,4(k-1)+t},J_{l-1,4(k-1)+t}} \\
&\cdots \\
[\![\mathcal{A}^{y}]\!]_{I,J} &= \sum_{\alpha=1}^{r_{L-1}} a_\alpha^{L,y} \cdot \bigodot_{t=1}^{4} [\![\phi^{L-1,t,\alpha}]\!]_{I_{L-1,t},J_{L-1,t}}
\end{aligned}
$$

$$
I_{l,k} := (I - (k-1)\cdot 4^l) \cap [4^l] \;,\; J_{l,k} := (J - (k-1)\cdot 4^l) \cap [4^l]
$$

- Recall the rank-multiplicative property of the Kronecker product:
$$rank(A \odot B) = rank(A) \cdot rank(B)$$
- If partition $I \dot\cup J = [N]$ is "interleaved" then $I_{l,k} \approx J_{l,k}$ and $[\![\phi^{l,k,\gamma}]\!]_{I_{l,k},J_{l,k}}$ are approximately square, allowing ranks to grow double-exponentially fast.
- If partition $I \dot\cup J = [N]$ is "coarse" then either $I_{l,k}$ or $J_{l,k}$ is small, and the ranks of $[\![\phi^{l,k,\gamma}]\!]_{I_{l,k},J_{l,k}}$ are limited.

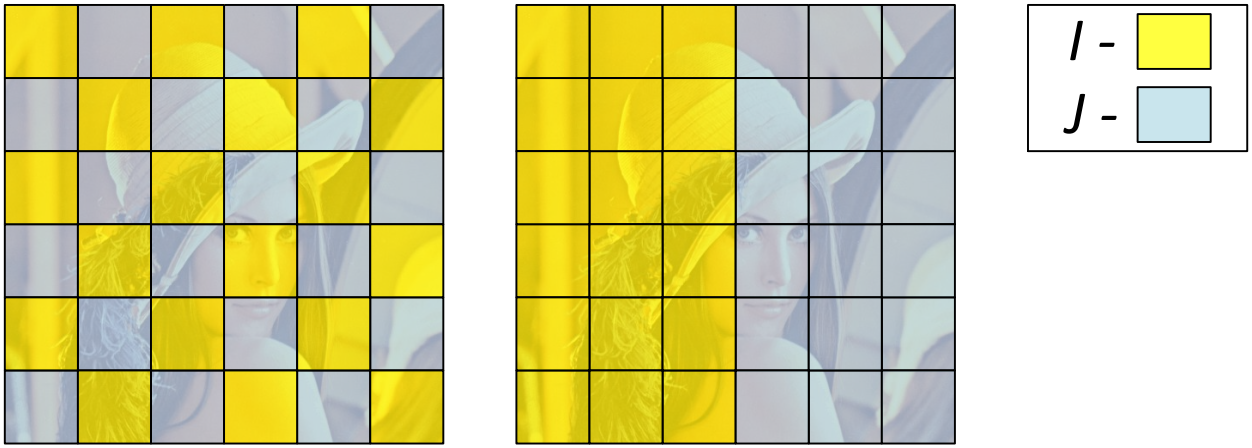
Inductive Bias through Pooling Geometry



Partition A *Partition B*

The **pooling geometry** of a deep network links partitions $I \dot{\cup} J = [N]$ to spatial input patterns, thereby controlling the inductive bias:

Inductive Bias through Pooling Geometry

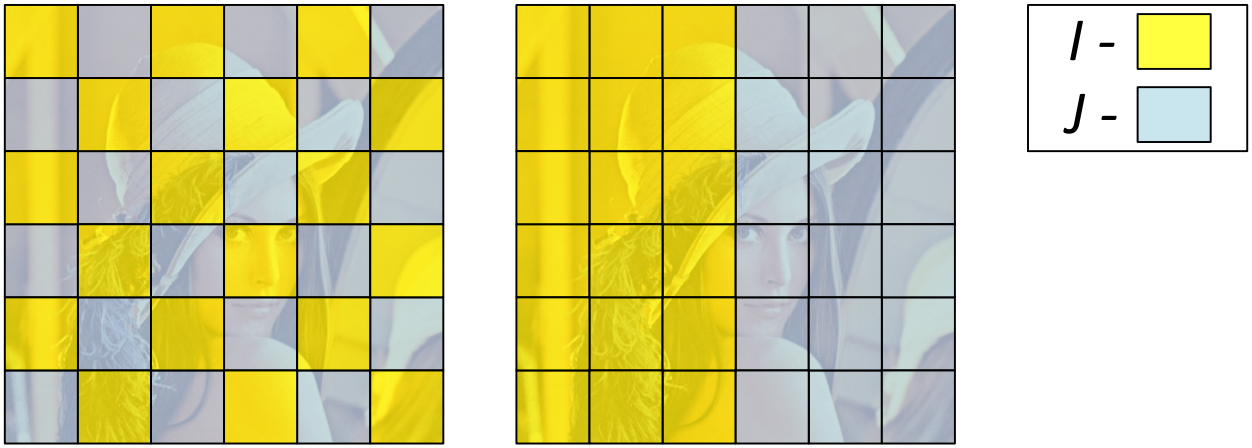


Partition A *Partition B*

The **pooling geometry** of a deep network links partitions $I \dot{\cup} J = [N]$ to spatial input patterns, thereby controlling the inductive bias:

- Contiguous pooling supports strong correlation between entangled regions (e.g. A), at the expense of low correlation between distinct ones (e.g. B)

Inductive Bias through Pooling Geometry

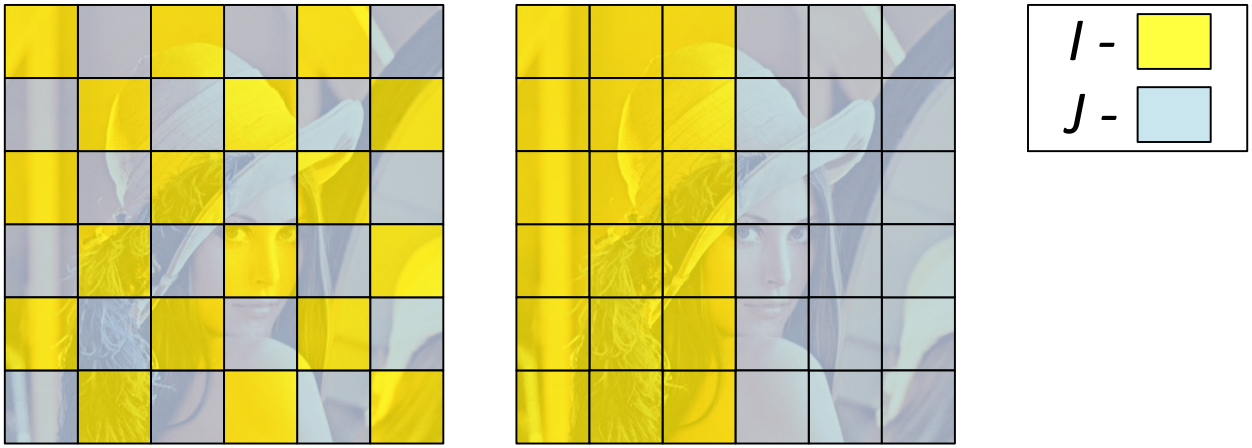


Partition A *Partition B*

The **pooling geometry** of a deep network links partitions $I \dot{\cup} J = [N]$ to spatial input patterns, thereby controlling the inductive bias:

- Contiguous pooling supports strong correlation between entangled regions (e.g. A), at the expense of low correlation between distinct ones (e.g. B)
- Other pooling schemes will lead to different correlation preferences

Inductive Bias through Pooling Geometry



Partition A *Partition B*

The **pooling geometry** of a deep network links partitions $I \dot{\cup} J = [N]$ to spatial input patterns, thereby controlling the inductive bias:

- Contiguous pooling supports strong correlation between entangled regions (e.g. A), at the expense of low correlation between distinct ones (e.g. B)
- Other pooling schemes will lead to different correlation preferences

Standard convolutional network design orients inductive bias towards statistics of natural images

Conclusion

- Depth efficiency is complete with convolutional arithmetic circuits (SimNets), incomplete with convolutional rectifier networks
- Shallow networks cannot model high correlation between input regions – require exponential size for exponential separation ranks
- Deep networks can model with polynomial size exponential separation ranks for "favorable" partitions
- Pooling geometry of a deep network determines which partitions are favorable, standard contiguous windows favor entangled partitions

Thank You