Inductive Bias of Deep Convolutional Networks through Pooling Geometry

Nadav Cohen

(joint work with Amnon Shashua)

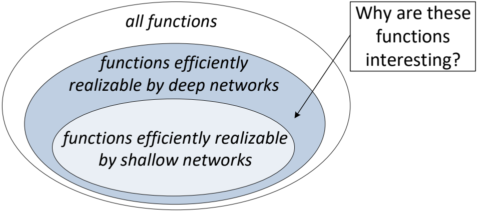
The Hebrew University of Jerusalem

NIPS 2016 Tensor Learning Workshop

Inductive Bias of Deep Convolutional Networks

Convolutional networks exhibit **depth efficiency**: [1] [2]
Functions realized by deep networks of polynomial size require super-polynomial size for being realized (or approximated) by shallow networks

This does not explain why functions brought forth by depth are effective

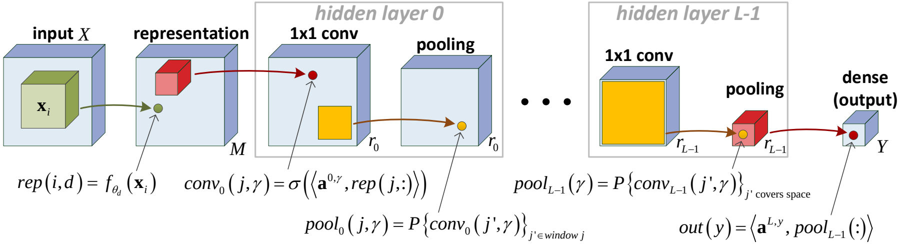


An explanation must consider the **inductive bias**, i.e. the assumptions encoded into functions, and their suitability for real-world tasks

[1] *On the Expressive Power of Deep Learning: A Tensor Analysis, COLT'16*
[2] *Convolutional Rectifier Networks as Generalized Tensor Decompositions, ICML'16*

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: $P\{c_j\} = \prod_j c_j$

Convolutional Arithmetic Circuits (cont')

Convolutional arithmetic circuits are theoretically appealing:

- Algebraic in nature (sums and products)
- Exhibit *complete* depth efficiency [1] – *almost all* functions realizable by deep networks cannot be efficiently realized by shallow ones

Also deliver promising results in practice:

- Excel in computationally constrained settings [2]
- Classify optimally with missing data [3]

We analyze convolutional arithmetic circuits, and empirically validate our findings with ReLU activation and max/average pooling as well (adapting analysis as done with depth efficiency [4] is left for future work)

[1] *On the Expressive Power of Deep Learning: A Tensor Analysis, COLT'16*

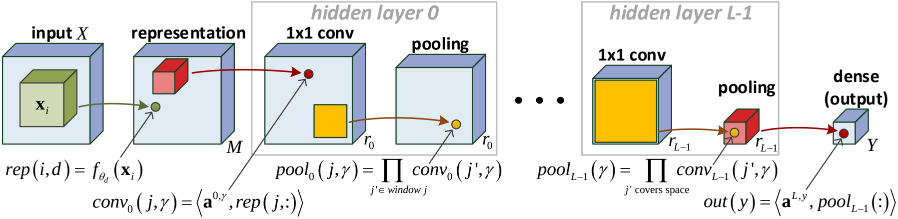
[2] *Deep SimNets, CVPR'16*

[3] *Tensorial Mixture Models, arXiv*

[4] *Convolutional Rectifier Networks as Generalized Tensor Decompositions, ICML'16*

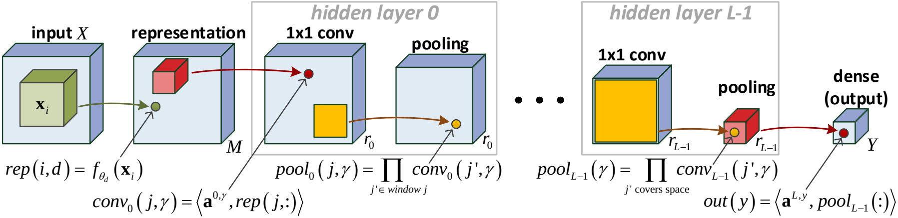
Equivalence to Tensor Decompositions

Convolutional arithmetic circuit:



- $\mathbf{x}_1 \ldots \mathbf{x}_N$ – input patches
- $f_{\theta_1} \ldots f_{\theta_M} : \mathbb{R}^s \to \mathbb{R}$ – **representation functions**
- $\{\mathbf{a}^{l,\gamma}\}$ – linear (conv/output) weights

Convolutional arithmetic circuit:



Function realized by network output y:

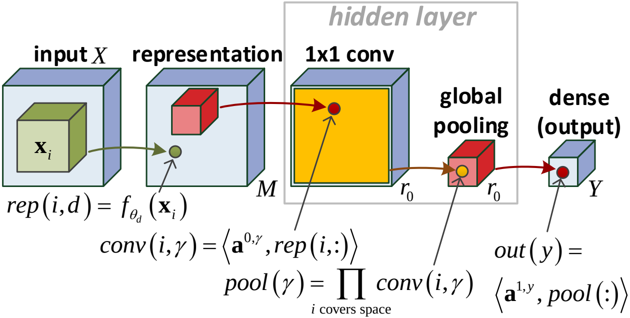
$$h_y(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{d_1 \ldots d_N = 1}^{M} \mathcal{A}^y_{d_1 \ldots d_N} \prod_{i=1}^{N} f_{\theta_{d_i}}(\mathbf{x}_i)$$

$\mathcal{A}^y$ – **coefficient tensor**:

- Order N (# of patches), dim M (# of rep funcs) in each mode
- Entries are polynomials in network's linear weights ($\mathbf{a}^{l,\gamma}$)

Shallow Network

Single hidden layer with global pooling:

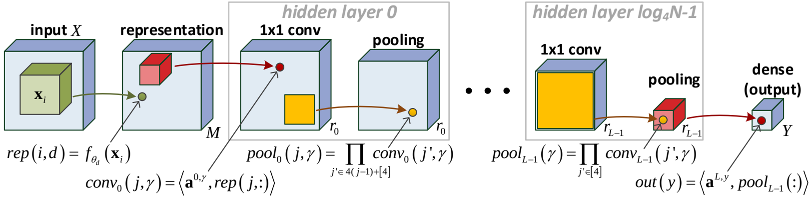


Coefficient tensors given by CP (rank-1) decomposition:

$$\mathcal{A}^y = \sum_{\gamma=1}^{r_0} a_\gamma^{1,y} \cdot \otimes^N \mathbf{a}^{0,\gamma}$$

Deep Network

Size-4 pooling windows, $L = \log_4 N$ hidden layers:



Coefficient tensors given by hierarchical decomposition:

$$\underbrace{\phi^{1,\gamma}}_{\text{order } 4} = \sum_{\alpha=1}^{r_0} a_\alpha^{1,\gamma} \cdot \otimes^4 \mathbf{a}^{0,\alpha} \quad , \gamma \in [r_1]$$

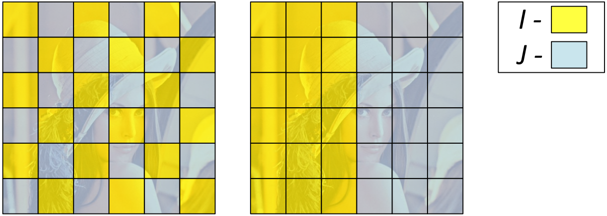
$$\cdots$$

$$\underbrace{\phi^{l,\gamma}}_{\text{order } 4^l} = \sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,\gamma} \cdot \otimes^4 \phi^{l-1,\alpha} \quad , l \in \{2 \ldots L-1\}, \gamma \in [r_l]$$

$$\cdots$$

$$\underbrace{\mathcal{A}^y}_{\text{order } 4^L = N} = \sum_{\alpha=1}^{r_{L-1}} a_\alpha^{L,y} \cdot \otimes^4 \phi^{L-1,\alpha}$$

Separation Rank



Partition A *Partition B*

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

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

In words, $sep(h; I, J)$ is the minimal number of summands that together give h, where each summand is separable w.r.t. (I, J)

Separation Rank – Interpretation

If $sep(h; I, J) = 1$ then h is separable w.r.t. (I, J):

- No interaction between $(\mathbf{x}_i)_{i \in I}$ and $(\mathbf{x}_j)_{j \in J}$ is modeled
- In probabilistic setup: $(\mathbf{x}_i)_{i \in I}$ and $(\mathbf{x}_j)_{j \in J}$ are statistically independent

The **higher** $\mathbf{sep}(\mathbf{h}; \mathbf{I}, \mathbf{J})$ is, the farther h is from separability, i.e. the **more correlation** is modeled between $(\mathbf{x}_i)_{i \in I}$ and $(\mathbf{x}_j)_{j \in J}$

Formally (details in the paper):

- Define $D(h; I, J)$ – L^2 distance of $h / \|h\|$ from the set of separable functions w.r.t. (I, J)
- It holds that $D(h; I, J) \leq \sqrt{1 - sep(h; I, J)^{-1}}$
- Inequality holds with equality in cases of interest

Separation Ranks of Convolutional Arithmetic Circuits

Function realized by convolutional arithmetic circuit:

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

- $\mathbf{x}_1 \ldots \mathbf{x}_N$ – input patches
- $\mathcal{A}^y$ – coefficient tensor (N modes)

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

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

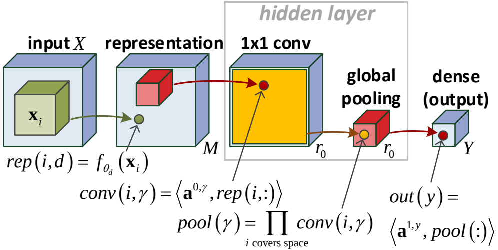
Claim

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

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

Separation Ranks of Shallow Network

Shallow network (single hidden layer):



Matricize CP decomposition of coefficient tensor ($\odot$ – Kronecker product):

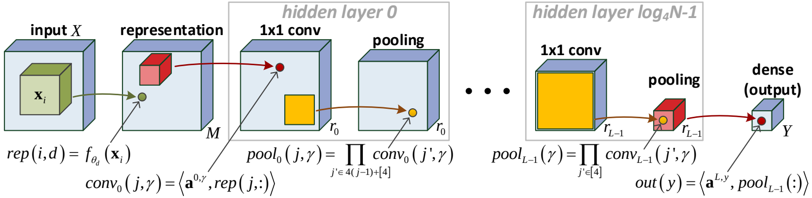
$$[\![\mathcal{A}^y]\!]_{I,J} = \sum_{\gamma=1}^{r_0} a_\gamma^{1,y} \cdot \left(\odot^{|I|}\mathbf{a}^{0,\gamma}\right)\left(\odot^{|J|}\mathbf{a}^{0,\gamma}\right)^\top$$

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

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

Separation Ranks of Deep Network

Deep network ($L = \log_4 N$ hidden layers):



Matricize hierarchical decomposition of coefficient tensor:

$$[\![\phi^{1,\gamma}]\!]_{I_{1,k},J_{1,k}} = \sum_{\alpha=1}^{r_0} a_\alpha^{1,\gamma} \cdot \bigodot_{t=1}^{4} [\![\mathbf{a}^{0,\alpha}]\!]_{I_{0,4(k-1)+t},J_{0,4(k-1)+t}}$$

$$\cdots$$

$$[\![\phi^{l,\gamma}]\!]_{I_{l,k},J_{l,k}} = \sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,\gamma} \cdot \bigodot_{t=1}^{4} [\![\phi^{l-1,\alpha}]\!]_{I_{l-1,4(k-1)+t},J_{l-1,4(k-1)+t}}$$

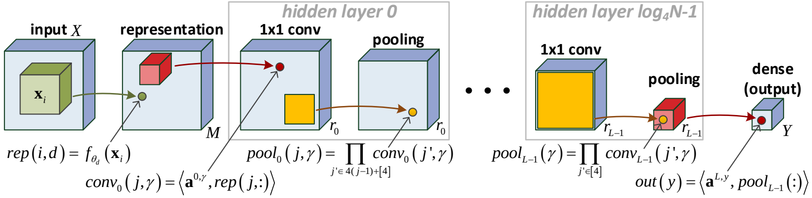
$$\cdots$$

$$[\![\mathcal{A}^y]\!]_{I,J} = \sum_{\alpha=1}^{r_{L-1}} a_\alpha^{L,y} \cdot \bigodot_{t=1}^{4} [\![\phi^{L-1,\alpha}]\!]_{I_{L-1,t},J_{L-1,t}}$$

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

Separation Ranks of Deep Network (cont')

Deep network ($L = \log_4 N$ hidden layers):



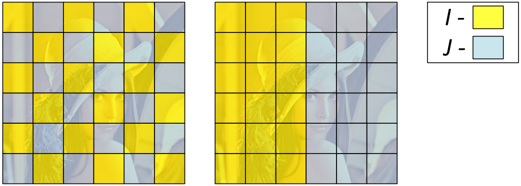
Theorem

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

Inductive Bias through Pooling Geometry



Partition A *Partition B*

Pooling geometry of deep network links partitions $I \dot\cup J = [N]$ to spatial input patterns, determining which patterns enjoy high separation ranks:

- Contiguous (2×2) pooling supports entangled patterns (e.g. A) at the expense of coarse ones (e.g. B), as required for natural images
- Other pooling schemes lead to different preferences, and this allows tailoring network to alternative types of data

> ***Pooling geometry controls inductive bias of deep network. Standard design suits natural images, other possibilities available.***

Experiments

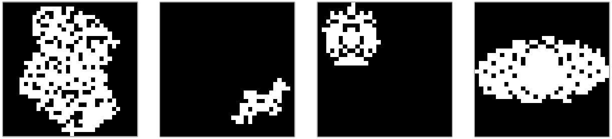
Goal

Demonstrate empirically that different pooling geometries lead to superior performance in different tasks

Dataset

Synthetic classification benchmark inspired by medical imaging tasks:

- 32-by-32 binary images, displaying random blobs with missing pixels
- Two binary (high/low) labels per image, reflecting symmetry and morphological closedness
- Predicting closedness requires local correlations, symmetry requires correlations across distances



closedness: low / symmetry: low — closedness: high / symmetry: low — closedness: low / symmetry: high — closedness: high / symmetry: high

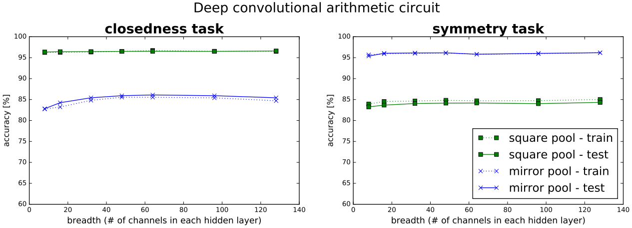
Experiments (cont')

Evaluated model

Deep network with two (size-4) pooling geometries:

square (standard 2×2 windows) **mirror** (reflections pooled together)

Results

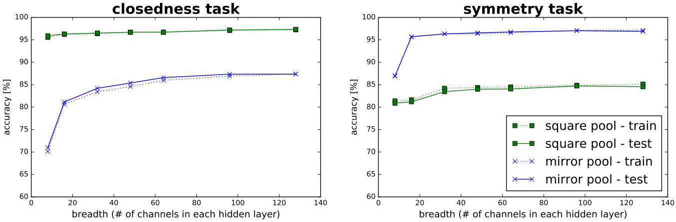


Standard square pooling superior for task of local nature, alternative mirror pooling better for symmetry detection

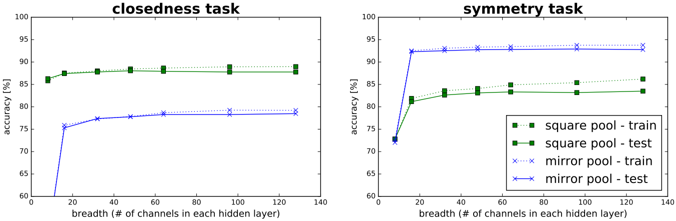
Experiments (cont')

Same trends obtained with ReLU activation and max/average pooling (instead of linear activation and product pooling):

Deep convolutional rectifier network (average pooling)



Deep convolutional rectifier network (max pooling)

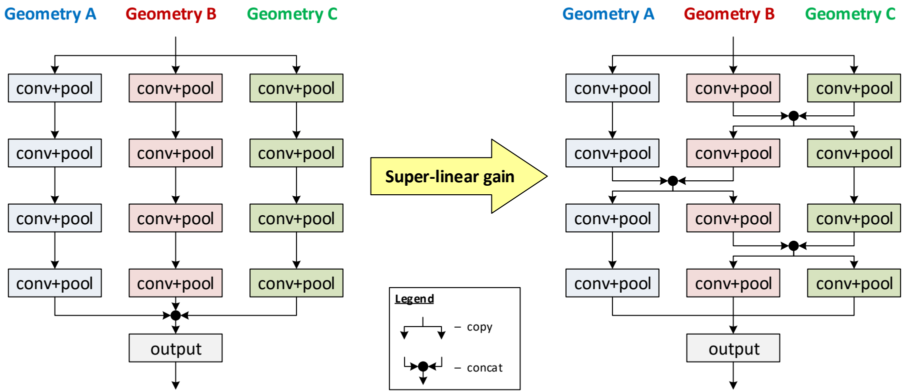


Conclusion

- **Separation rank** of function w.r.t. partition of its input measures strength of correlation modeled between sides of the partition
- We analyzed separation ranks of convolutional arithmetic circuits:
  - **Deep networks**: with polynomial size separation ranks are exponential for certain input partitions, polynomial for others
  - **Shallow networks**: separation ranks are exponential only if size is exponential (implies depth efficiency, with insight into benefit of depth)
- Deep network's **pooling geometry** determines which partitions are favored in terms of separation rank, thus **controls inductive bias**:
  - Standard contiguous pooling favors interleaved partitions, orienting inductive bias towards statistics of natural images
  - Other pooling schemes lead to different preferences, and this **allows tailoring network to data that departs from natural imagery**

Future Work

Blend together multiple pooling geometries for super-linear gain in number of favored input partitions



Thank You