Expressive Efficiency and Inductive Bias of Convolutional Networks:

Analysis & Design via Hierarchical Tensor Decompositions

Nadav Cohen

The Hebrew University of Jerusalem

Conference on Computer Vision and Pattern Recognition (CVPR) 2017

Workshop on Tensor Methods in Computer Vision

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

Tensorial Mixture Models
O. Sharir. R. Tamari, N. Cohen and A. Shashua
arXiv preprint 2017

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

Deep Learning and Quantum Entanglement:
Fundamental Connections with Implications to Network Design
Y. Levine, D. Yakira, N. Cohen and A. Shashua
arXiv preprint 2017

Collaborators

Or Sharir

Yoav Levine

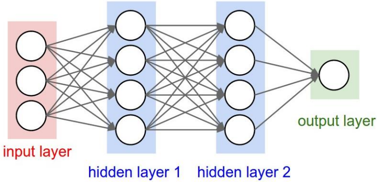
Amnon Shashua

Ronen Tamari

David Yakira

Classic vs. State of the Art Deep Learning

Classic



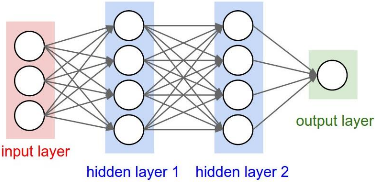
Multilayer Perceptron (MLP)

Architectural choices:

- depth
- layer widths
- activation types

Classic vs. State of the Art Deep Learning

Classic

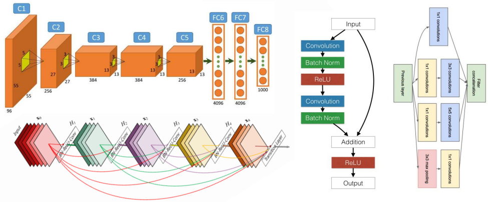


Multilayer Perceptron (MLP)

Architectural choices:

- depth
- layer widths
- activation types

State of the Art



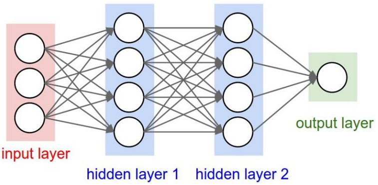
Convolutional Networks (ConvNets)

Architectural choices:

- depth
- layer widths
- activation types
- pooling types
- convolution/pooling windows
- convolution/pooling strides
- dilation factors
- connectivity
- and more...

Classic vs. State of the Art Deep Learning

Classic

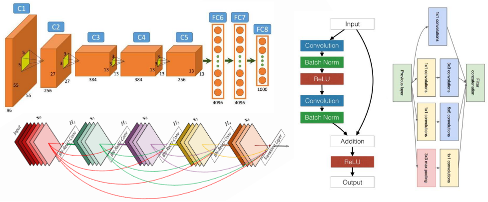


Multilayer Perceptron (MLP)

Architectural choices:

- depth
- layer widths
- activation types

State of the Art



Convolutional Networks (ConvNets)

Architectural choices:

- depth
- layer widths
- activation types
- pooling types
- convolution/pooling windows
- convolution/pooling strides

Can the architectural choices of state of the art ConvNets be theoretically analyzed?

Outline

Expressiveness

Expressiveness:

- Ability to compactly represent rich and effective classes of func
- The driving force behind deep networks

Expressiveness

Expressiveness:

- Ability to compactly represent rich and effective classes of func
- The driving force behind deep networks

Fundamental theoretical questions:

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

Expressive Efficiency

Expressive efficiency compares network arch in terms of their ability to compactly represent func

Expressive Efficiency

Expressive efficiency compares network arch in terms of their ability to compactly represent func

Let:

- $\mathcal{H}_A$ – space of func compactly representable by network arch A
- $\mathcal{H}_B$ – -"- network arch B

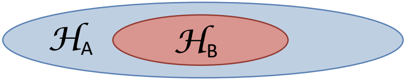
Expressive Efficiency

Expressive efficiency compares network arch in terms of their ability to compactly represent func

Let:

- $\mathcal{H}_A$ – space of func compactly representable by network arch A
- $\mathcal{H}_B$ – -”- network arch B

A is **efficient** w.r.t. B if $\mathcal{H}_A$ is a strict superset of $\mathcal{H}_B$



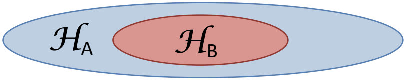
Expressive Efficiency

Expressive efficiency compares network arch in terms of their ability to compactly represent func

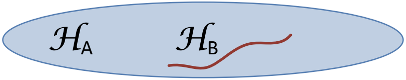
Let:

- $\mathcal{H}_A$ – space of func compactly representable by network arch A
- $\mathcal{H}_B$ – -"- network arch B

A is **efficient** w.r.t. B if $\mathcal{H}_A$ is a strict superset of $\mathcal{H}_B$



A is **completely efficient** w.r.t. B if $\mathcal{H}_B$ has zero "volume" inside $\mathcal{H}_A$



Expressive Efficiency – Formal Definition

Network arch A is **efficient** w.r.t. network arch B if:

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

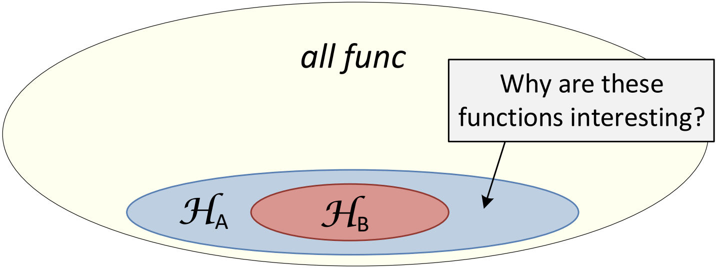
(2) $\exists$func realized by A w/size r_A requiring B to have size $r_B \in \Omega(f(r_A))$, where $f(\cdot)$ is super-linear

A is **completely efficient** w.r.t. B if (2) holds for all its func but a set of Lebesgue measure zero (in weight space)

Inductive Bias

Networks of reasonable size can only realize a fraction of all possible func

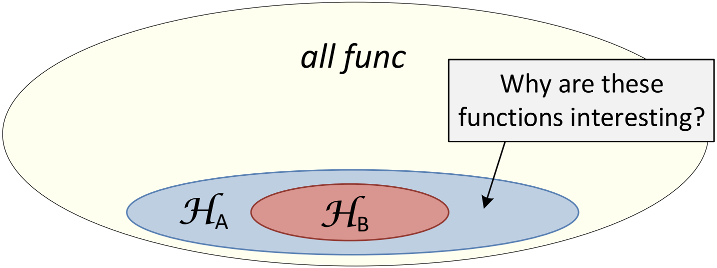
Efficiency does not explain why this fraction is effective



Inductive Bias

Networks of reasonable size can only realize a fraction of all possible func

Efficiency does not explain why this fraction is effective



To explain the effectiveness, one must consider the **inductive bias**:

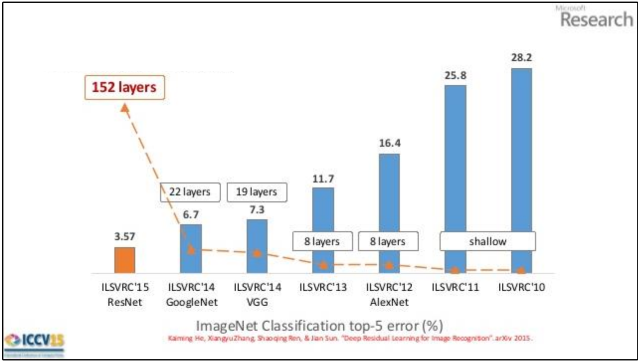
- Not all func are equally useful for a given task
- Network only needs to represent useful func

Outline

Efficiency of Depth

Longstanding conjecture, proven for MLP:

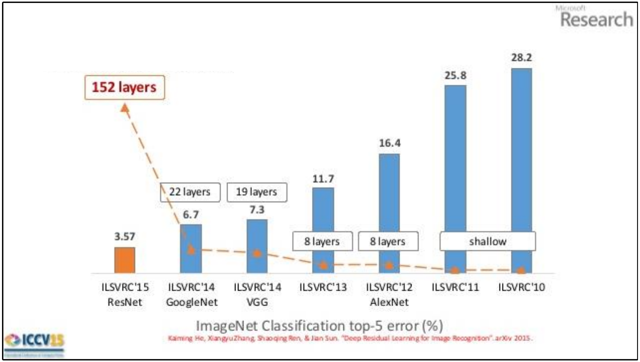
deep networks are expressively efficient w.r.t. shallow ones



Efficiency of Depth

Longstanding conjecture, proven for MLP:

deep networks are expressively efficient w.r.t. shallow ones

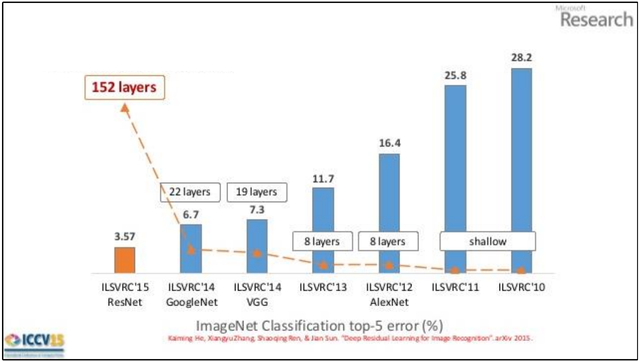


Q: *Can this be proven for ConvNets?*

Efficiency of Depth

Longstanding conjecture, proven for MLP:

deep networks are expressively efficient w.r.t. shallow ones

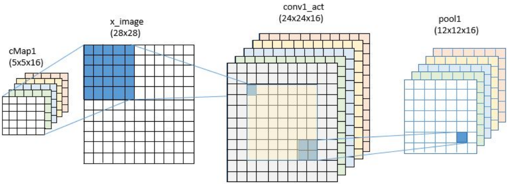


Q: *Can this be proven for ConvNets?*

Q: *Is their efficiency of depth complete? (no such results for MLP)*

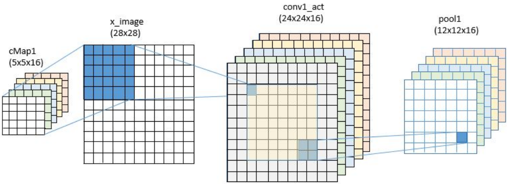
Inductive Bias of Convolution/Pooling Geometry

ConvNets typically employ square conv/pool windows

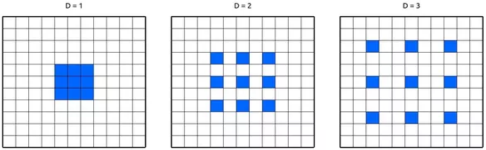


Inductive Bias of Convolution/Pooling Geometry

ConvNets typically employ square conv/pool windows

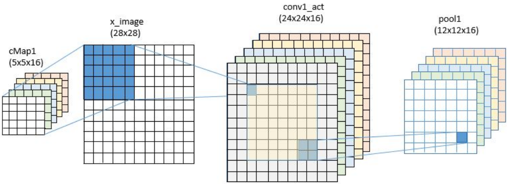


Recently, dilated windows have also become popular

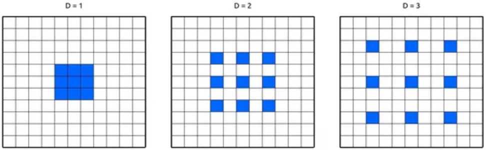


Inductive Bias of Convolution/Pooling Geometry

ConvNets typically employ square conv/pool windows



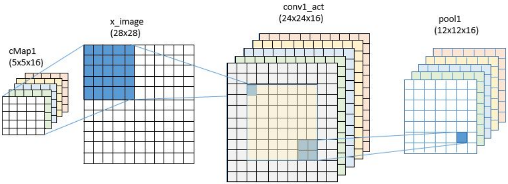
Recently, dilated windows have also become popular



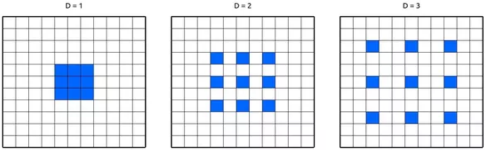
Q: *What is the inductive bias of conv/pool window geometry?*

Inductive Bias of Convolution/Pooling Geometry

ConvNets typically employ square conv/pool windows



Recently, dilated windows have also become popular

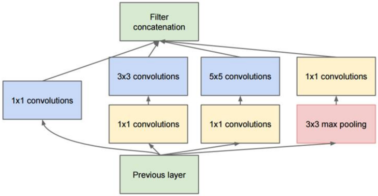


Q: *What is the inductive bias of conv/pool window geometry?*

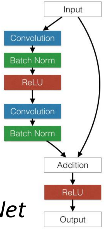
Q: *Can the geometries be tailored for a given task?*

Efficiency of Connectivity Schemes

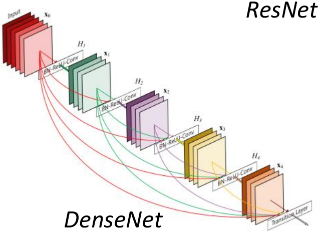
Nearly all state of the art ConvNets employ elaborate connectivity schemes



Inception (GoogLeNet)



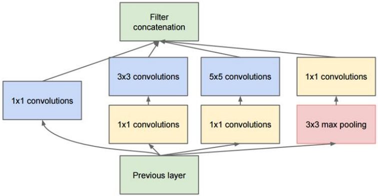
ResNet



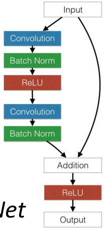
DenseNet

Efficiency of Connectivity Schemes

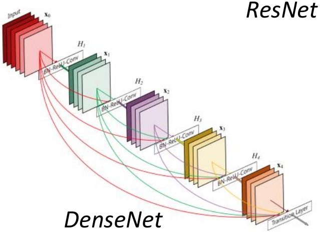
Nearly all state of the art ConvNets employ elaborate connectivity schemes



Inception (GoogLeNet)



ResNet

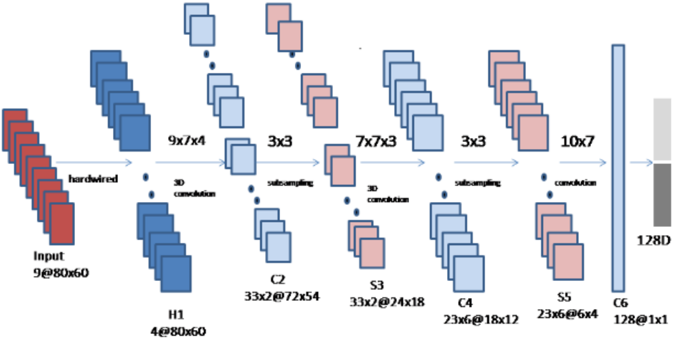


DenseNet

Q: *Can this be justified in terms of expressive efficiency?*

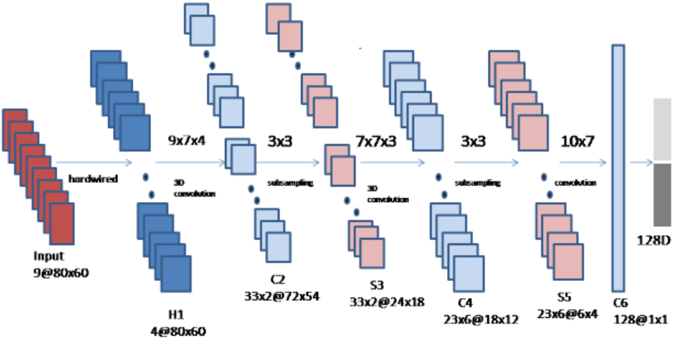
Inductive Bias of Layer Widths

No clear principle for setting widths (# of channels) of ConvNet layers



Inductive Bias of Layer Widths

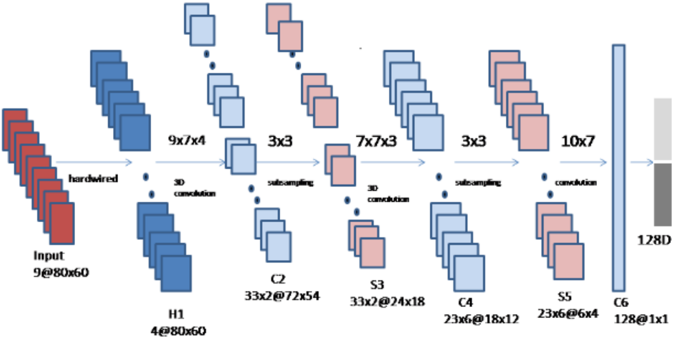
No clear principle for setting widths (# of channels) of ConvNet layers



Q: *What is the inductive bias of one layer's width vs. another's?*

Inductive Bias of Layer Widths

No clear principle for setting widths (# of channels) of ConvNet layers



Q: *What is the inductive bias of one layer's width vs. another's?*

Q: *Can the widths be tailored for a given task?*

Outline

Convolutional Arithmetic Circuits

To address raised Qs, we begin with a special case of ConvNets:

Convolutional Arithmetic Circuits (ConvACs)

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

[2] *Deep SimNets, CVPR'16*

[3] *Tensorial Mixture Models, arXiv'17*

Convolutional Arithmetic Circuits

To address raised Qs, we begin with a special case of ConvNets:

Convolutional Arithmetic Circuits (ConvACs)

ConvACs are equivalent to **hierarchical tensor decompositions**:

- May be analyzed w/various mathematical tools
- Tools may be extended to additional types of ConvNets (e.g. ReLU) [1]

[1] *Convolutional Rectifier Networks as Generalized Tensor Decompositions, ICML'16*
[2] *Deep SimNets, CVPR'16*
[3] *Tensorial Mixture Models, arXiv'17*

Convolutional Arithmetic Circuits

To address raised Qs, we begin with a special case of ConvNets:

Convolutional Arithmetic Circuits (ConvACs)

ConvACs are equivalent to **hierarchical tensor decompositions**:

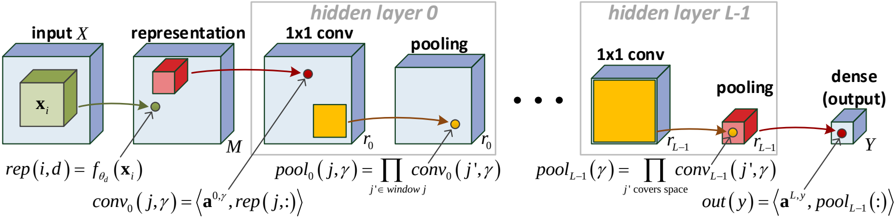
- May be analyzed w/various mathematical tools
- Tools may be extended to additional types of ConvNets (e.g. ReLU) [1]

Besides theoretical merits, ConvACs deliver promising results in practice:

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

[1] *Convolutional Rectifier Networks as Generalized Tensor Decompositions, ICML'16*
[2] *Deep SimNets, CVPR'16*
[3] *Tensorial Mixture Models, arXiv'17*

Baseline Architecture



Baseline ConvAC architecture:

- 2D ConvNet: conv $\longrightarrow$ $L \times$ (conv $\rightarrow$ pool) $\longrightarrow$ dense
- *Linear activation* $(\sigma(z) = z)$, *product pooling* $(P\{c_j\} = \prod_j c_j)$

Grid Tensors

ConvNets realize func over many local elements:

$$f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$$

$\mathbf{x}_i$ – image pixels (2D network) / sequence samples (1D network)

Grid Tensors

ConvNets realize func over many local elements:

$$f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$$

$\mathbf{x}_i$ – image pixels (2D network) / sequence samples (1D network)

$f(\cdot)$ may be studied by *discretizing* each $\mathbf{x}_i$ into one of $\{\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(M)}\}$:

$$\mathcal{A}_{d_1 \ldots d_N} = f(\mathbf{v}^{(d_1)} \ldots \mathbf{v}^{(d_N)}) \quad , d_1 \ldots d_N \in \{1, \ldots, M\}$$

Grid Tensors

ConvNets realize func over many local elements:

$$f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$$

$\mathbf{x}_i$ – image pixels (2D network) / sequence samples (1D network)

$f(\cdot)$ may be studied by *discretizing* each $\mathbf{x}_i$ into one of $\{\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(M)}\}$:

$$\mathcal{A}_{d_1 \ldots d_N} = f(\mathbf{v}^{(d_1)} \ldots \mathbf{v}^{(d_N)}) \quad , d_1 \ldots d_N \in \{1, \ldots, M\}$$

The lookup table $\mathcal{A}$ is:

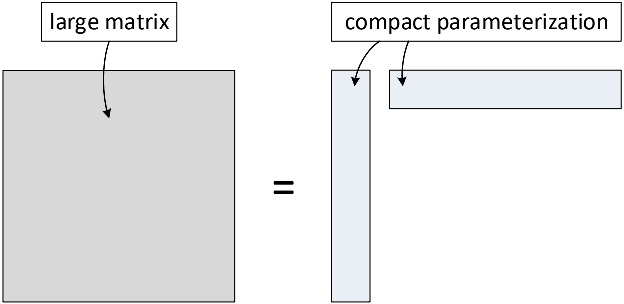
- an N-dim array (tensor) w/length M in each axis (mode)
- referred to as the **grid tensor** of $f(\cdot)$

Tensor Decompositions – Compact Parameterizations

High-dim tensors (arrays) are exponentially large – cannot be used directly

May be represented and manipulated via **tensor decompositions**:

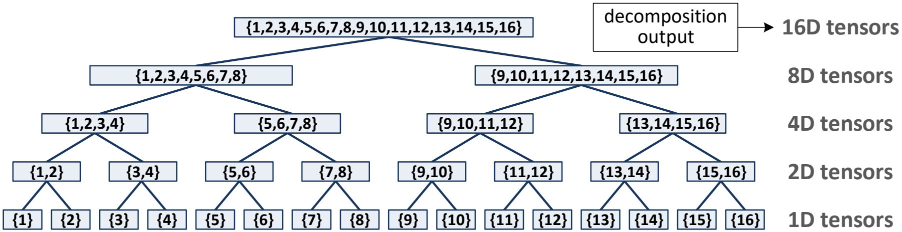
- Compact algebraic parameterizations
- Generalizations of low-rank matrix decomposition



Hierarchical Tensor Decompositions

Hierarchical tensor decompositions represent high-dim tensors by incrementally generating intermediate tensors of increasing dim

Generation process can be described by a tree over tensor modes (axes)



Convolutional Arithmetic Circuits $\longleftrightarrow$ Hierarchical Tensor Decompositions

Key observation

Grid tensors of func realized by ConvACs are given by hierarchical tensor decompositions. 1-to-1 correspondence:

network structure (depth, width, pooling etc)	$\longleftrightarrow$	decomposition type (mode tree, internal ranks etc)
network weights	$\longleftrightarrow$	decomposition parameters

Convolutional Arithmetic Circuits $\longleftrightarrow$ Hierarchical Tensor Decompositions

Key observation

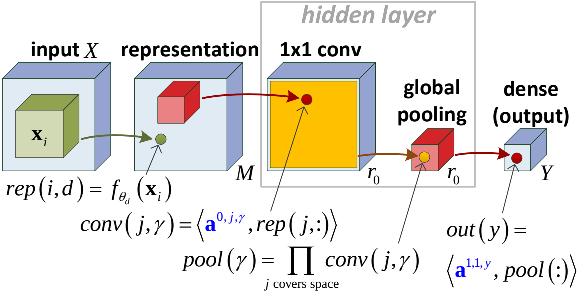
Grid tensors of func realized by ConvACs are given by hierarchical tensor decompositions. 1-to-1 correspondence:

network structure (depth, width, pooling etc)	$\longleftrightarrow$	decomposition type (mode tree, internal ranks etc)
network weights	$\longleftrightarrow$	decomposition parameters

We can study networks through corresponding decompositions!

Example 1: Shallow Network $\longleftrightarrow$ CP Decomposition

Shallow network (single hidden layer, global pooling):



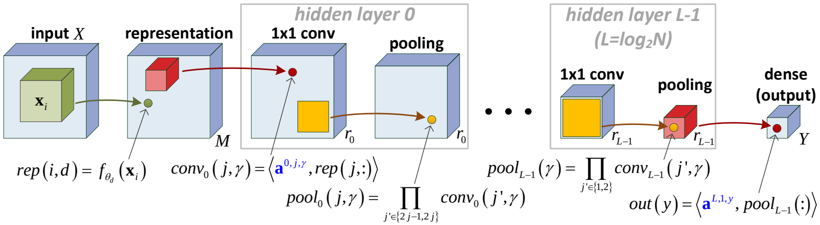
corresponds to classic **CP decomposition**:

$$\mathcal{A}^y = \sum_{\gamma=1}^{r_0} a_\gamma^{1,1,y} \cdot \mathbf{a}^{0,1,\gamma} \otimes \mathbf{a}^{0,2,\gamma} \otimes \cdots \otimes \mathbf{a}^{0,N,\gamma}$$

($\otimes$ – outer product)

Example 2: Deep Network $\longleftrightarrow$ HT Decomposition

Deep network with size-2 pooling:



corresponds to **Hierarchical Tucker (HT) decomposition**:

$$\phi^{1,j,\gamma} = \sum_{\alpha=1}^{r_0} a_\alpha^{1,j,\gamma} \cdot \mathbf{a}^{0,2j-1,\alpha} \otimes \mathbf{a}^{0,2j,\alpha}$$

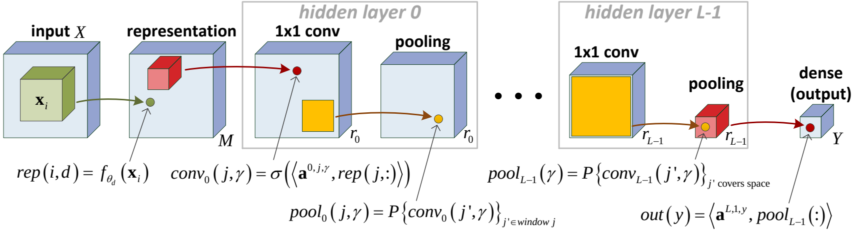
$$\cdots$$

$$\phi^{l,j,\gamma} = \sum_{\alpha=1}^{r_{l-1}} a_\alpha^{l,j,\gamma} \cdot \phi^{l-1,2j-1,\alpha} \otimes \phi^{l-1,2j,\alpha}$$

$$\cdots$$

$$\mathcal{A}^y = \sum_{\alpha=1}^{r_{L-1}} a_\alpha^{L,1,y} \cdot \phi^{L-1,1,\alpha} \otimes \phi^{L-1,2,\alpha}$$

From Convolutional Arithmetic Circuits to Convolutional Rectifier Networks



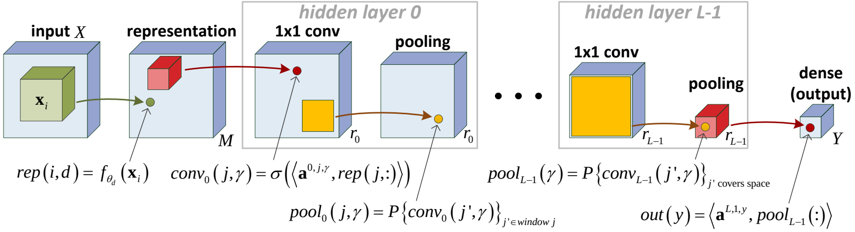
Transform ConvACs to **Convolutional Rectifier Networks (R-ConvNets)**:

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

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

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

From Convolutional Arithmetic Circuits to Convolutional Rectifier Networks



Transform ConvACs to **Convolutional Rectifier Networks (R-ConvNets)**:

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

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

Observation

Transforming ConvAC to R-ConvNet turns corresponding hierarchical tensor decomposition to a generalized one [1]

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

Outline

Efficiency of Depth *(C.Sharir.Shashua@COLT'16, C.Shashua@ICML'16)*

By analyzing ranks of matricized grid tensors, we show:

Efficiency of Depth *(C.Sharir.Shashua@COLT'16, C.Shashua@ICML'16)*

By analyzing ranks of matricized grid tensors, we show:

Theorem

Almost all func realizable by deep ConvAC cannot be replicated by shallow network with less than exponentially many hidden channels

Efficiency of Depth *(C.Sharir.Shashua@COLT'16, C.Shashua@ICML'16)*

By analyzing ranks of matricized grid tensors, we show:

Theorem

Almost all func realizable by deep ConvAC cannot be replicated by shallow network with less than exponentially many hidden channels

Theorem

There exist func realizable by deep R-ConvNet requiring shallow networks to be exponentially large, but this does not happen almost always

Efficiency of Depth *(C.Sharir.Shashua@COLT'16, C.Shashua@ICML'16)*

By analyzing ranks of matricized grid tensors, we show:

Theorem

Almost all func realizable by deep ConvAC cannot be replicated by shallow network with less than exponentially many hidden channels

Theorem

There exist func realizable by deep R-ConvNet requiring shallow networks to be exponentially large, but this does not happen almost always

W/ConvACs efficiency of depth is complete, w/R-ConvNets it's not!

Efficiency of Depth *(C.Sharir.Shashua@COLT'16, C.Shashua@ICML'16)*

By analyzing ranks of matricized grid tensors, we show:

Theorem

Almost all func realizable by deep ConvAC cannot be replicated by shallow network with less than exponentially many hidden channels

Theorem

There exist func realizable by deep R-ConvNet requiring shallow networks to be exponentially large, but this does not happen almost always

W/ConvACs efficiency of depth is complete, w/R-ConvNets it's not!

Developing optimization methods for ConvACs may give rise to an arch that is provably superior but has so far been overlooked

Inductive Bias of Pooling Geometry *(C.Shashua@ICLR'17)*

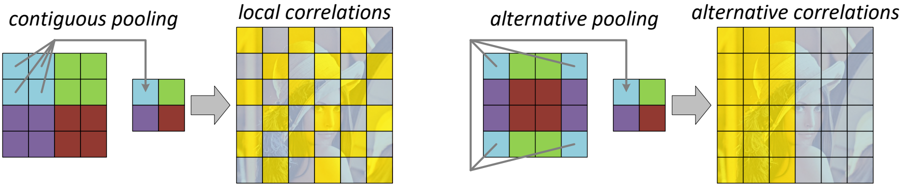
We study ability of ConvACs to model correlations between input regions

Inductive Bias of Pooling Geometry *(C.Shashua@ICLR'17)*

We study ability of ConvACs to model correlations between input regions

Theorem

Deep network effectively models some (favored) correlations, but not all. What determines which correlations are favored is the pooling geometry.

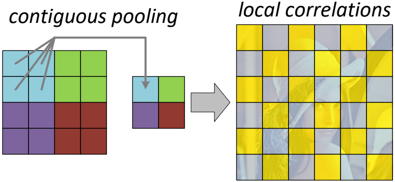


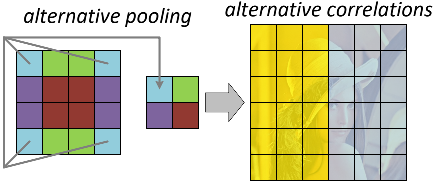
Inductive Bias of Pooling Geometry *(C.Shashua@ICLR'17)*

We study ability of ConvACs to model correlations between input regions

Theorem

Deep network effectively models some (favored) correlations, but not all.
What determines which correlations are favored is the pooling geometry.

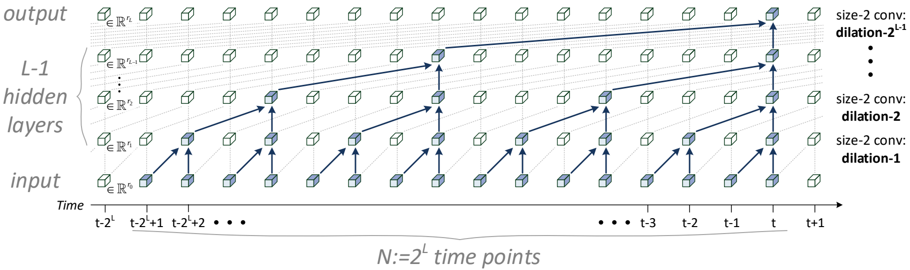




Pooling geometry controls correlation profile (inductive bias)!
Can be used to tailor networks according to needs of given task!

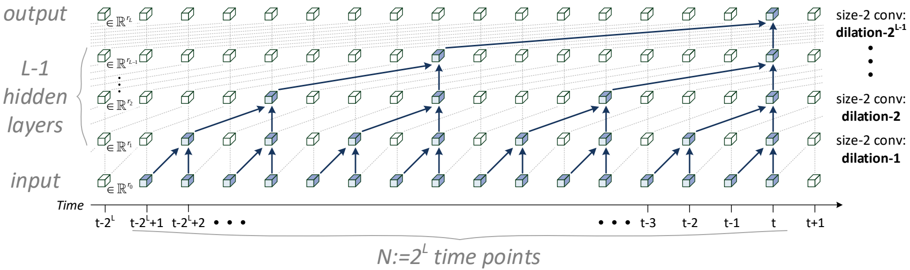
Efficiency of Interconnectivity *(C.Tamari.Shashua@arXiv'17)*

Dilated convolutional networks (state of the art for audio & text):



Efficiency of Interconnectivity *(C.Tamari.Shashua@arXiv'17)*

Dilated convolutional networks (state of the art for audio & text):



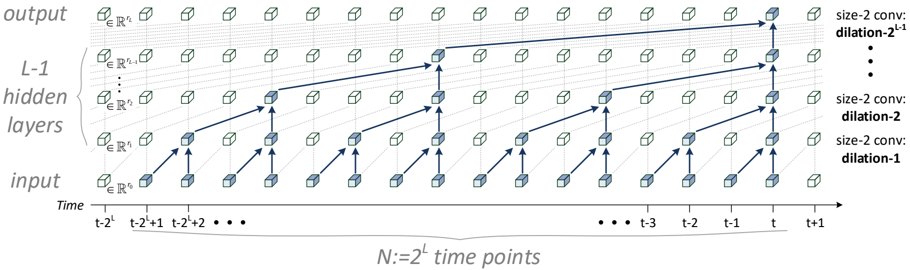
By introducing the notion of **mixed tensor decompositions**, we prove:

Theorem

Interconnected dilated ConvNets realize func that cannot be realized by individual networks unless these are quadratically larger

Efficiency of Interconnectivity *(C.Tamari.Shashua@arXiv'17)*

Dilated convolutional networks (state of the art for audio & text):



By introducing the notion of **mixed tensor decompositions**, we prove:

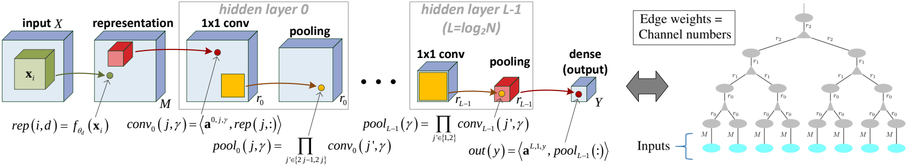
Theorem

Interconnected dilated ConvNets realize func that cannot be realized by individual networks unless these are quadratically larger

W/dilated ConvNets interconnectivity brings efficiency!

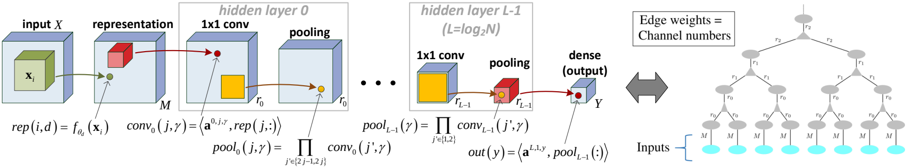
Inductive Bias of Layer Widths *(Levine.Yakira.C.Shashua@arXiv'17)*

ConvACs can be cast as **tensor networks** (graphs) from quantum physics:



Inductive Bias of Layer Widths *(Levine.Yakira.C.Shashua@arXiv'17)*

ConvACs can be cast as **tensor networks** (graphs) from quantum physics:

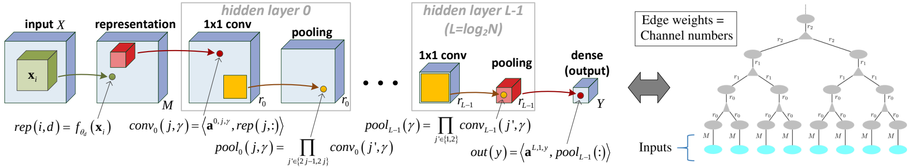


Theorem

Input correlation strengths supported by ConvAC are equal to min-cuts in graph whose edge weights are layer widths

Inductive Bias of Layer Widths *(Levine.Yakira.C.Shashua@arXiv'17)*

ConvACs can be cast as **tensor networks** (graphs) from quantum physics:



Theorem

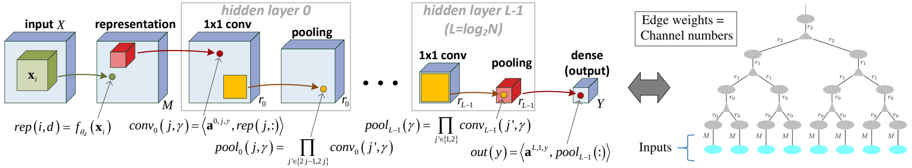
Input correlation strengths supported by ConvAC are equal to min-cuts in graph whose edge weights are layer widths

Corollary

Deep layer widths important for long-rang correlations, early layer for short

Inductive Bias of Layer Widths *(Levine.Yakira.C.Shashua@arXiv'17)*

ConvACs can be cast as **tensor networks** (graphs) from quantum physics:



Theorem

Input correlation strengths supported by ConvAC are equal to min-cuts in graph whose edge weights are layer widths

Corollary

Deep layer widths important for long-rang correlations, early layer for short

Layer widths also affect correlation profile (inductive bias)!
Can be used to tailor networks according to needs of given task!

Outline

1. Expressiveness
2. Expressiveness of Convolutional Networks – Questions
3. Analysis via Hierarchical Tensor Decompositions
4. Results

Conclusion

- **Expressiveness** – the driving force behind deep networks

Conclusion

- **Expressiveness** – the driving force behind deep networks
- Formal concepts for treating expressiveness:
  - **Expressive efficiency** – network arch realizes func requiring alternative arch to be much larger
  - **Inductive bias** – prioritization of some func over others given prior knowledge on task at hand

Conclusion

- **Expressiveness** – the driving force behind deep networks
- Formal concepts for treating expressiveness:
  - **Expressive efficiency** – network arch realizes func requiring alternative arch to be much larger
  - **Inductive bias** – prioritization of some func over others given prior knowledge on task at hand
- **ConvNets** $\longleftrightarrow$ **hierarchical tensor decompositions**

Conclusion

- **Expressiveness** – the driving force behind deep networks
- Formal concepts for treating expressiveness:
  - **Expressive efficiency** – network arch realizes func requiring alternative arch to be much larger
  - **Inductive bias** – prioritization of some func over others given prior knowledge on task at hand
- **ConvNets** $\longleftrightarrow$ **hierarchical tensor decompositions**
- We **analyzed arch features of ConvNets** (depth, width, pooling, interconnectivity) in terms of expressive efficiency and inductive bias

Conclusion

- **Expressiveness** – the driving force behind deep networks
- Formal concepts for treating expressiveness:
  - **Expressive efficiency** – network arch realizes func requiring alternative arch to be much larger
  - **Inductive bias** – prioritization of some func over others given prior knowledge on task at hand
- **ConvNets** $\longleftrightarrow$ **hierarchical tensor decompositions**
- We **analyzed arch features of ConvNets** (depth, width, pooling, interconnectivity) in terms of expressive efficiency and inductive bias
- Results not only explanatory – provide **new tools for network design**

Thank You