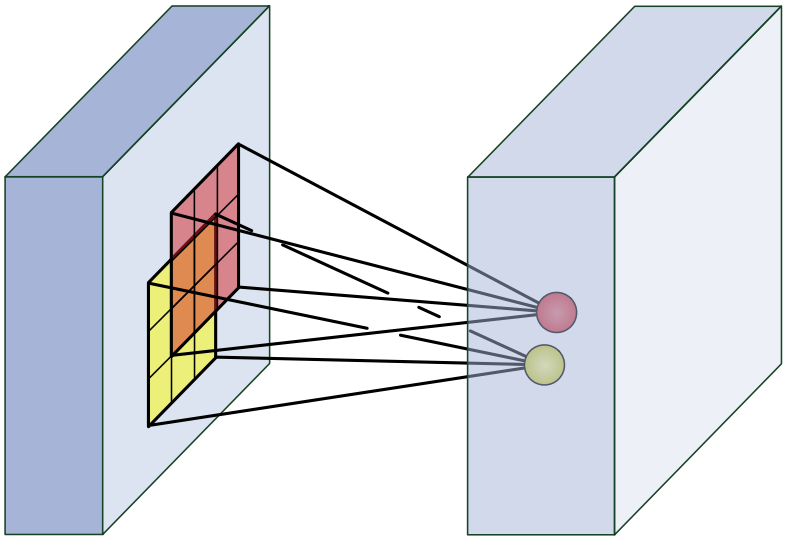
On the Expressive Efficiency of Overlapping Architectures of Deep Learning

Or Sharir Amnon Shashua

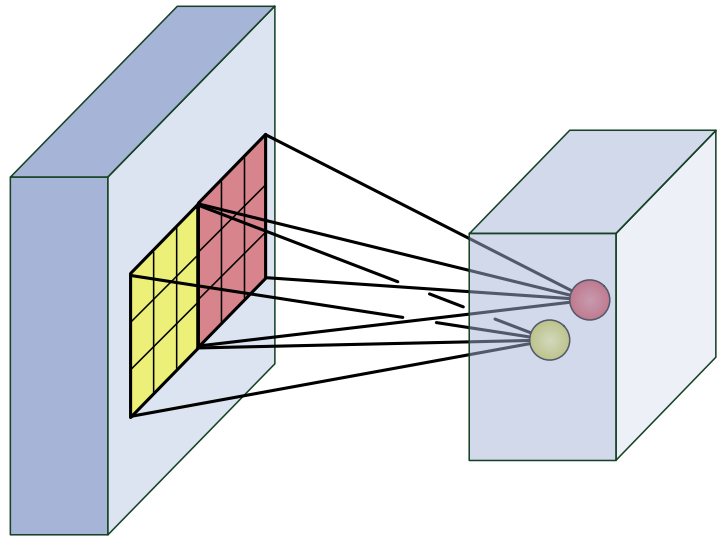
The Hebrew University of Jerusalem

June 30, 2017
Deep Learning Summer School

Overlapping vs Non-Overlapping Architectures



Receptive Field > Stride
⇔ Overlapping



Receptive Field = Stride
⇔ Non-Overlapping

The Merits of Non-overlapping Architectures

Non-overlapping arch's have theoretical merit:

- **Universality:** can approximate any func given sufficient resources
- **Optimization:** better convergence guarantees than overlapping arch[1]

[1] Alon Brutzkus & Amir Globerson. **Globally Optimal Gradient Descent for a ConvNet with Gaussian Inputs.** ICML 2017.

The Merits of Non-overlapping Architectures

Non-overlapping arch's have theoretical merit:

- **Universality:** can approximate any func given sufficient resources
- **Optimization:** better convergence guarantees than overlapping arch[1]

In practice:

- Non-overlapping arch's are used in some applications, **but only few!**
- Modern arch's use ever smaller receptive fields, including many non-overlapping layers, but never all layers!

[1] Alon Brutzkus & Amir Globerson. **Globally Optimal Gradient Descent for a ConvNet with Gaussian Inputs.** ICML 2017.

The Merits of Non-overlapping Architectures

Non-overlapping arch's have theoretical merit:

- **Universality:** can approximate any func given sufficient resources
- **Optimization:** better convergence guarantees than overlapping arch[1]

In practice:

- Non-overlapping arch's are used in some applications, **but only few!**
- Modern arch's use ever smaller receptive fields, including many non-overlapping layers, but never all layers!

Questions

1) Why are non-overlapping arch's so uncommon?

2) Why is having just a bit of overlapping sufficient for most tasks?

[1] Alon Brutzkus & Amir Globerson. **Globally Optimal Gradient Descent for a ConvNet with Gaussian Inputs.** ICML 2017.

Outline

Efficiency

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

Efficiency

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

Let:

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

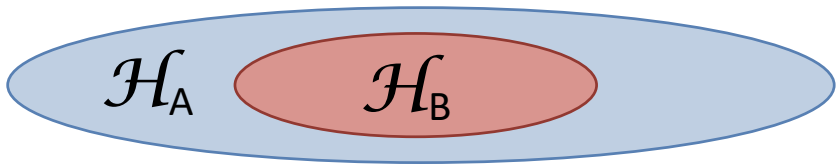
Efficiency

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

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 $\mathcal{H}_B$



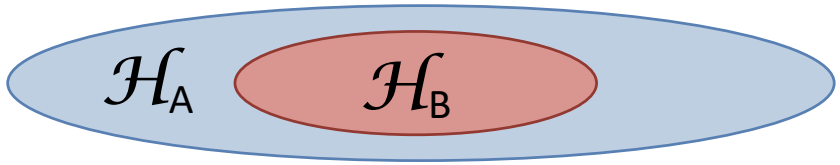
Efficiency

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

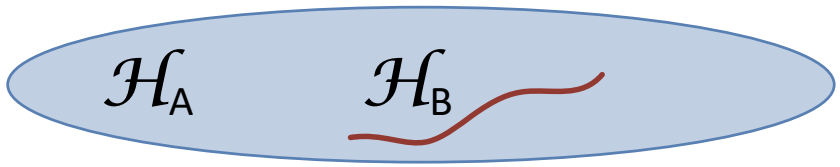
Let:

- $\mathcal{H}_A$ – 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 $\mathcal{H}_A$



Efficiency – Formal Definition

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

(1) $\forall$func realized by B w/size[1] r_B can be realized by A w/size $r_A \in \mathcal{O}(g(r_B))$, where g is polynomial.

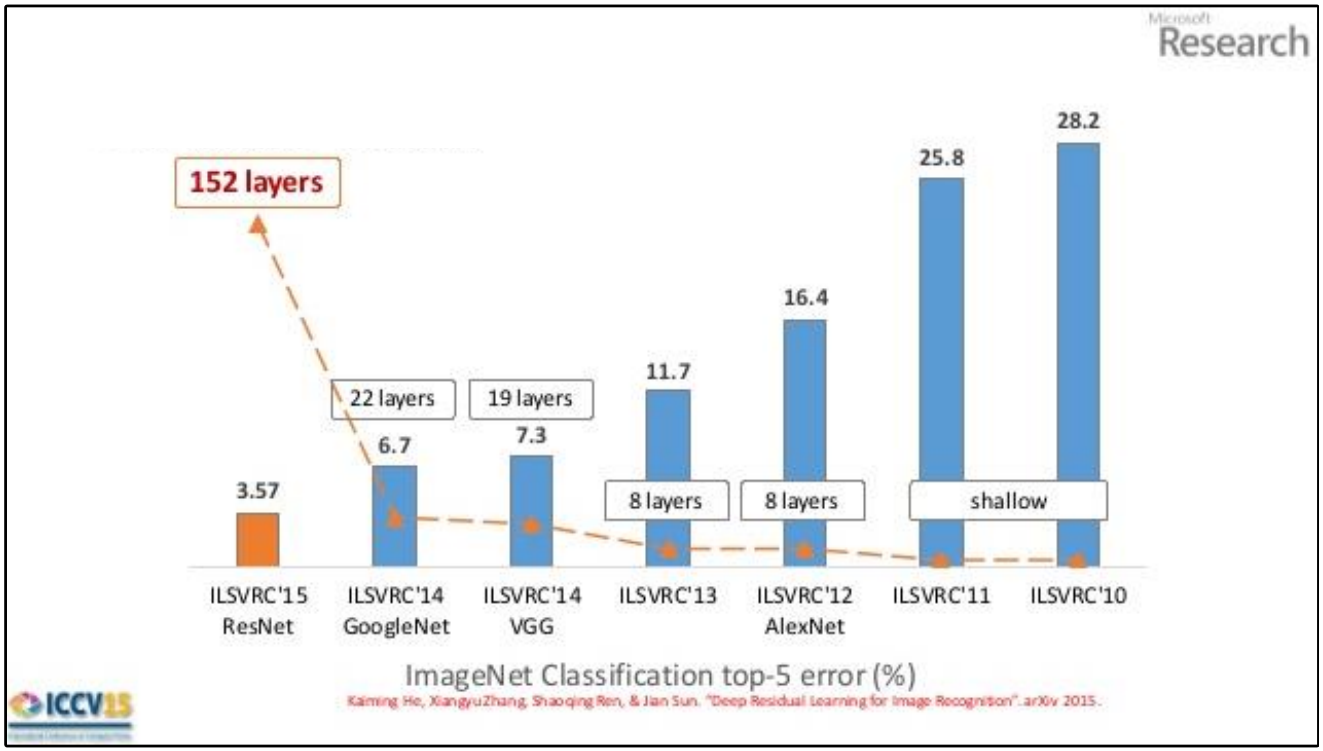
(2) $\exists$func realized by A w/size r_A requiring B to have size $r_B \in \Omega(f(r_A))$, where f is super-polynomial.

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

[1] Size depends on the measure of interest, e.g. # of neurons or # of parameters

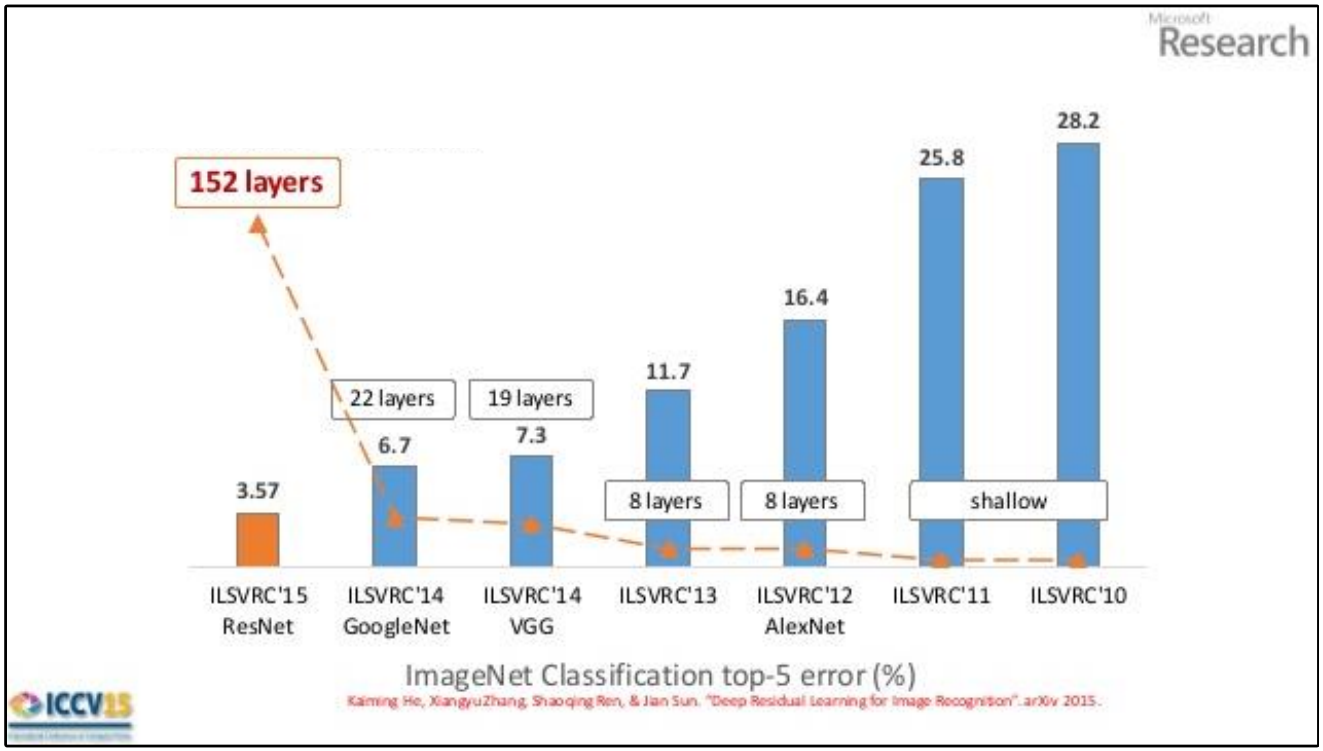
Example: Efficiency of Depth

Empirical Results: deep networks have an advantage



Example: Efficiency of Depth

Empirical Results: deep networks have an advantage



Theory

Deep nets are exponentially efficient w.r.t. shallow ones

Other Works of Our Group

Depth Efficiency:

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 Connectivity Patterns:

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

Inductive Bias of the Widths of Layers:

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

Outline

Convolutional Arithmetic Circuits

To address raised Qs, we consider 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 consider 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 consider 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]

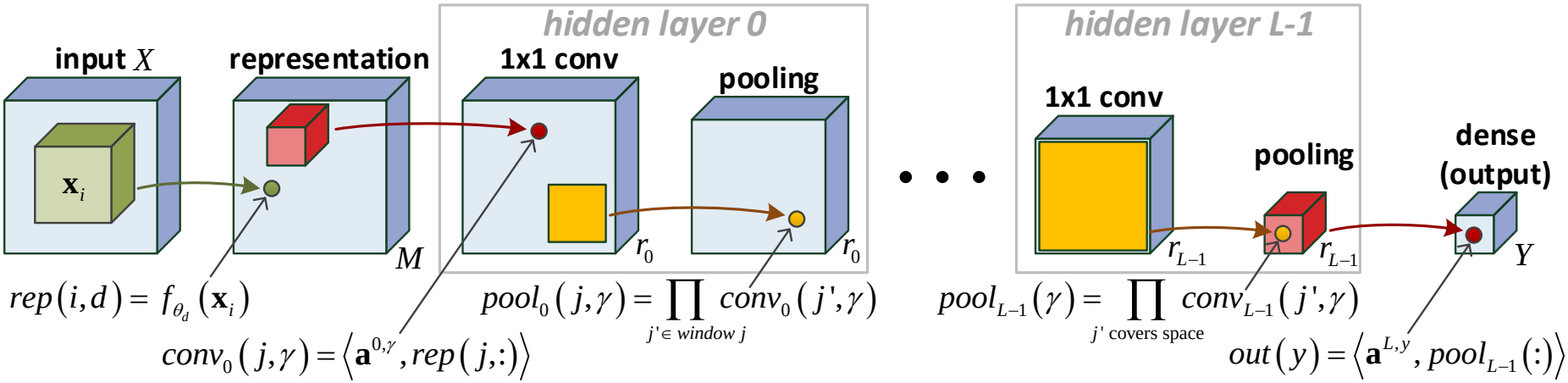
---

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

[2] *Deep SimNets, CVPR'16*

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

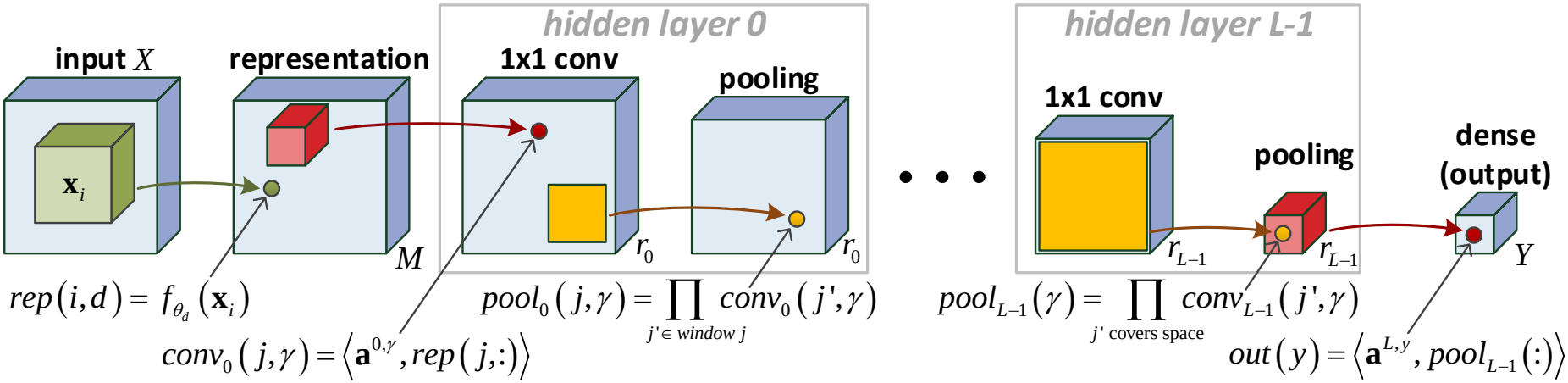
Baseline Architecture



Baseline ConvAC architecture:

- 2D ConvNet: conv $\longrightarrow$ $L \times$ (conv $\rightarrow$ pool) $\longrightarrow$ dense
- 1×1 convolutions, followed by linear activations ($\sigma(z) = z$)
- product pooling: $P\{c_j\} = \prod_j c_j$ (non-overlapping windows)

Baseline Architecture

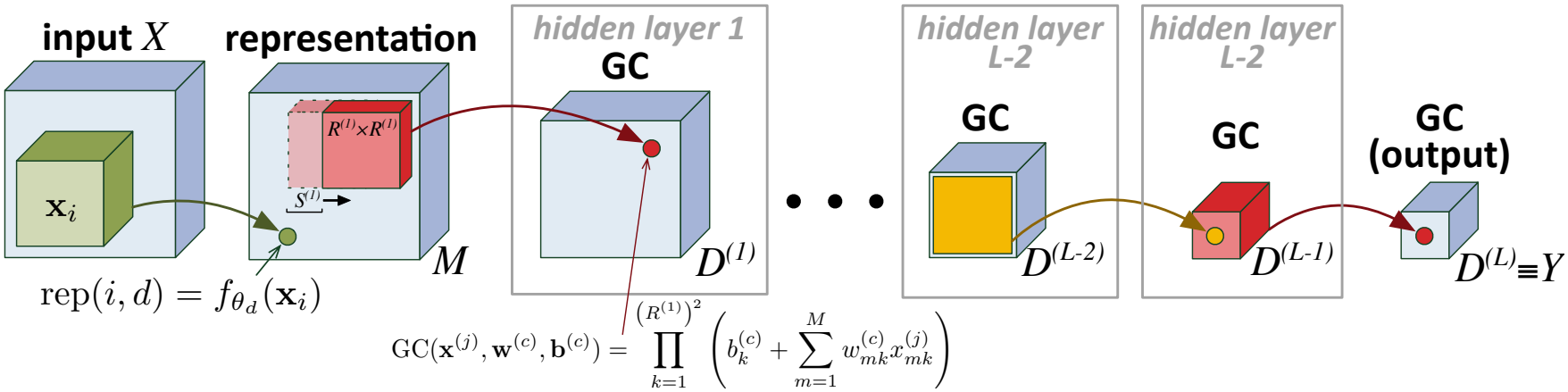


Baseline ConvAC architecture:

- 2D ConvNet: conv $\longrightarrow$ $L \times$ (conv $\rightarrow$ pool) $\longrightarrow$ dense
- 1×1 convolutions, followed by linear activations ($\sigma(z) = z$)
- product pooling: $P\{c_j\} = \prod_j c_j$ (non-overlapping windows)
- **Limitation**: supports only non-overlapping architectures!

Generalized Convolutional Arithmetic Circuits

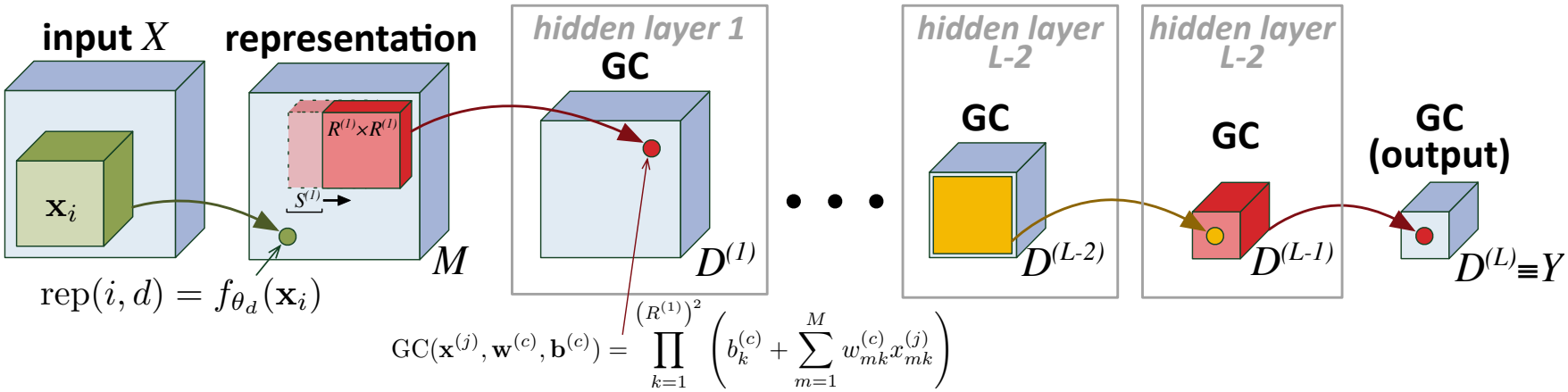
Generalizing ConvACs to overlapping arch's:



- **Generalized Convolution:** generalizes 1×1-conv and pooling

Generalized Convolutional Arithmetic Circuits

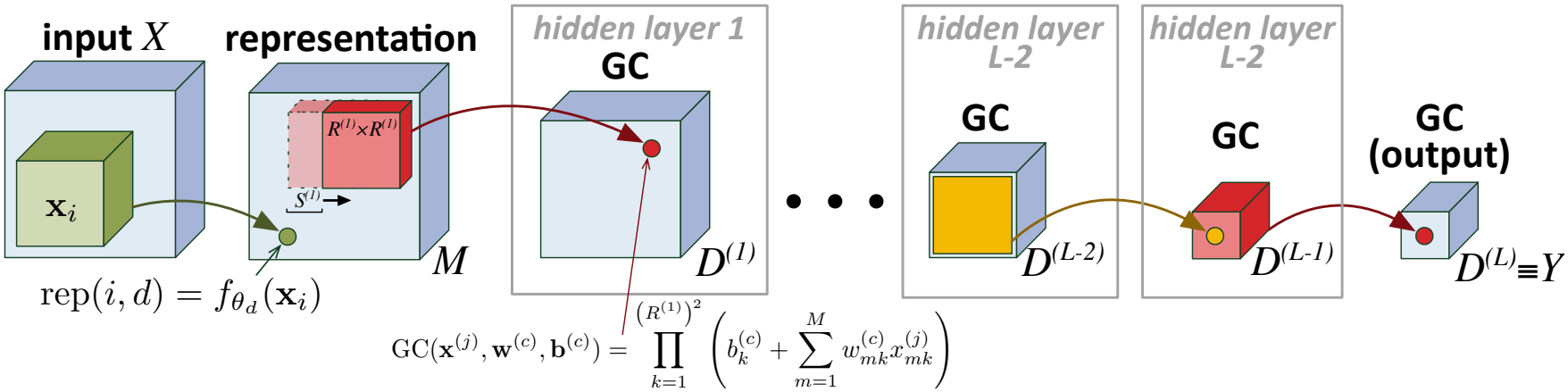
Generalizing ConvACs to overlapping arch's:



- **Generalized Convolution:** generalizes 1×1-conv and pooling
- Inspired by All Convolutional Net (pooling via stride > 1)

Generalized Convolutional Arithmetic Circuits

Generalizing ConvACs to overlapping arch's:



- **Generalized Convolution:** generalizes 1×1-conv and pooling
- Inspired by All Convolutional Net (pooling via stride > 1)
- Non-overlapping case is equivalent to standard ConvACs

Outline

Overlapping Architectures Are Just As Expressive

Claim

An overlapping arch can replicate any func realizable by a non-overlapping arch of similar size and same sequence of strides

Overlapping Architectures Are Just As Expressive

Claim

An overlapping arch can replicate any func realizable by a non-overlapping arch of similar size and same sequence of strides

Conclusion

Overlapping arch's are just as expressive as non-overlapping arch's!

Overlapping Architectures Are Just As Expressive

Claim

An overlapping arch can replicate any func realizable by a non-overlapping arch of similar size and same sequence of strides

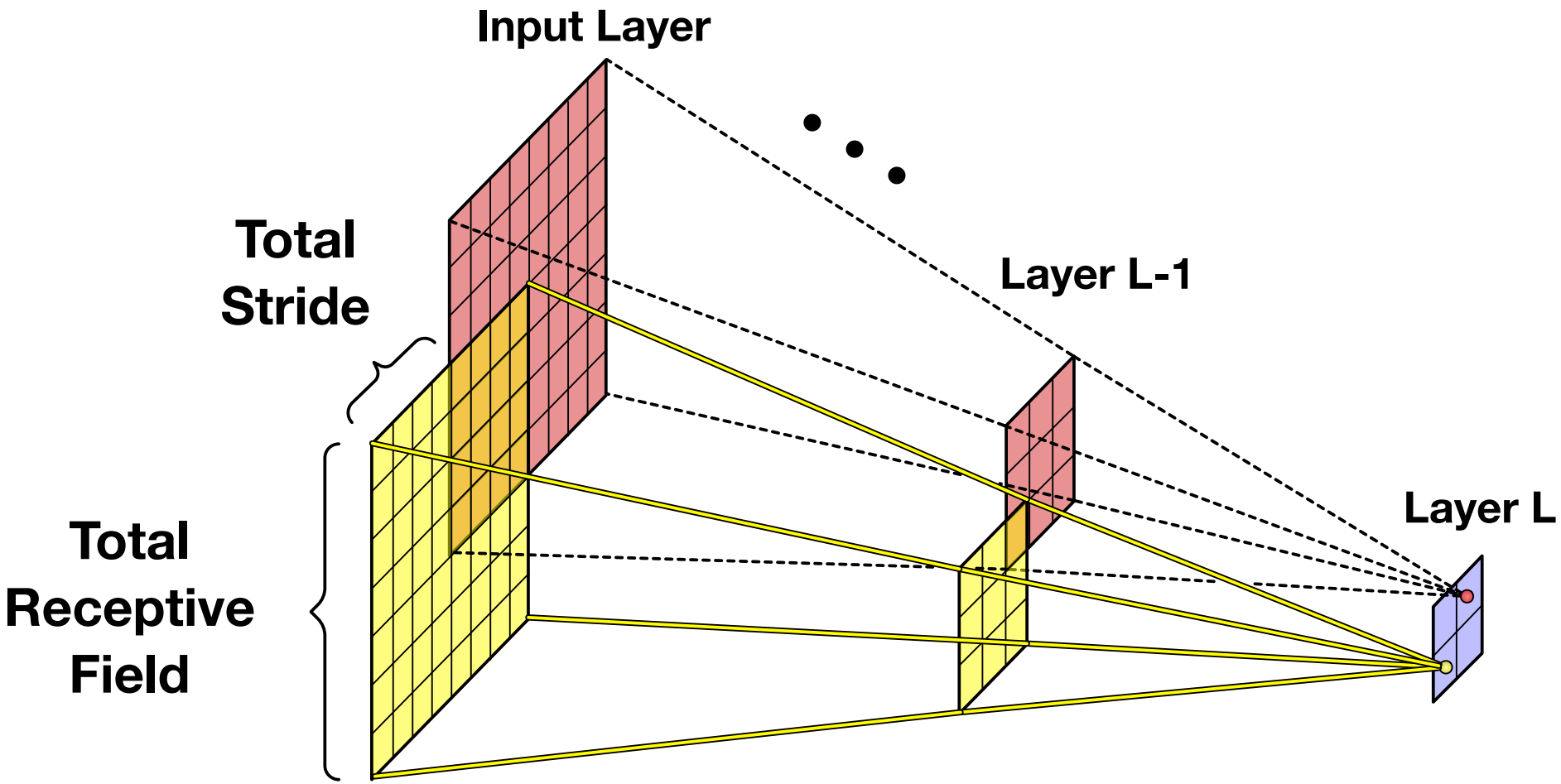
Conclusion

Overlapping arch's are just as expressive as non-overlapping arch's!

Question

Could it be that overlapping arch's are in fact more expressive?

Degree of Overlapping



Overlapping Efficiency

Theorem

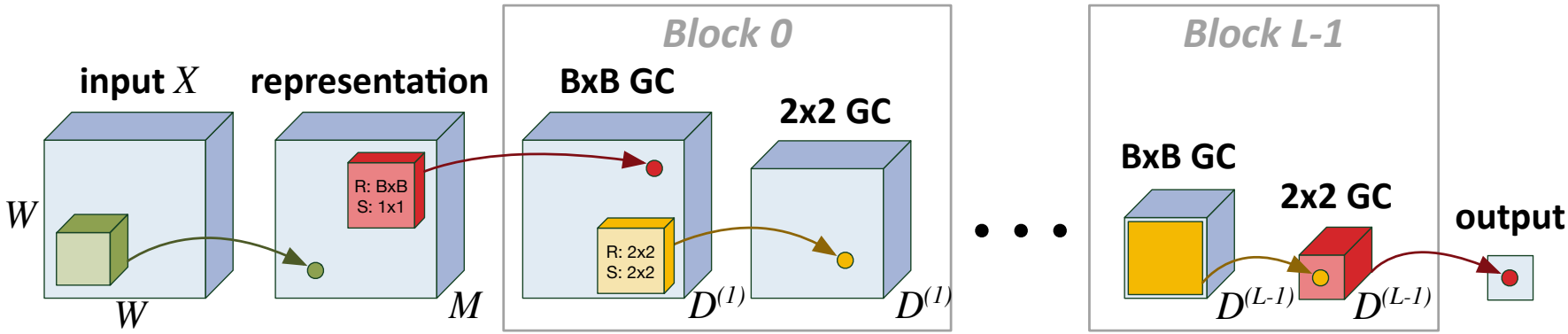
Almost all func's realizable by an overlapping arch cannot be replicated by a non-overlapping arch unless its size is exponential in the overlapping degree

Overlapping Efficiency

Theorem

Almost all func's realizable by an overlapping arch cannot be replicated by a non-overlapping arch unless its size is exponential in the overlapping degree

Common Case: alternating $B \times B$-conv and 2×2-pooling



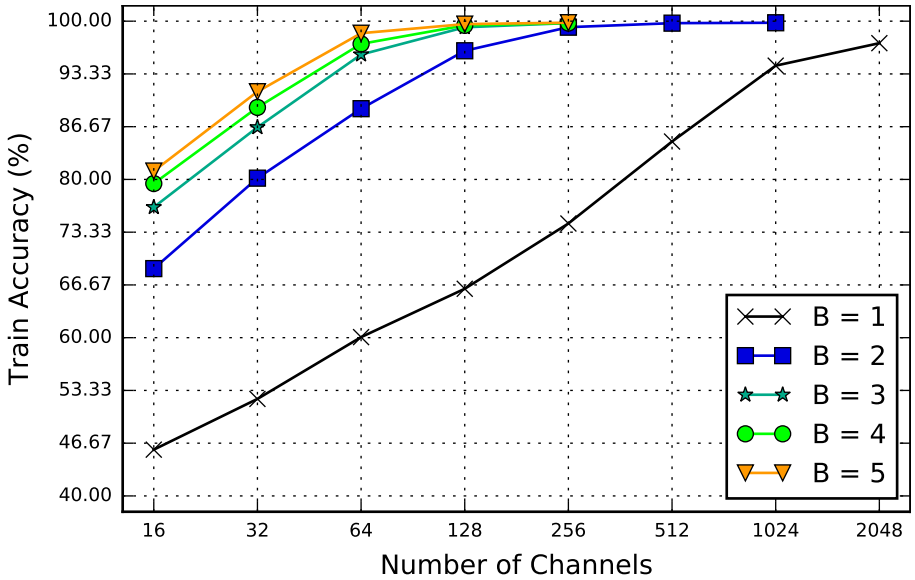
Claim

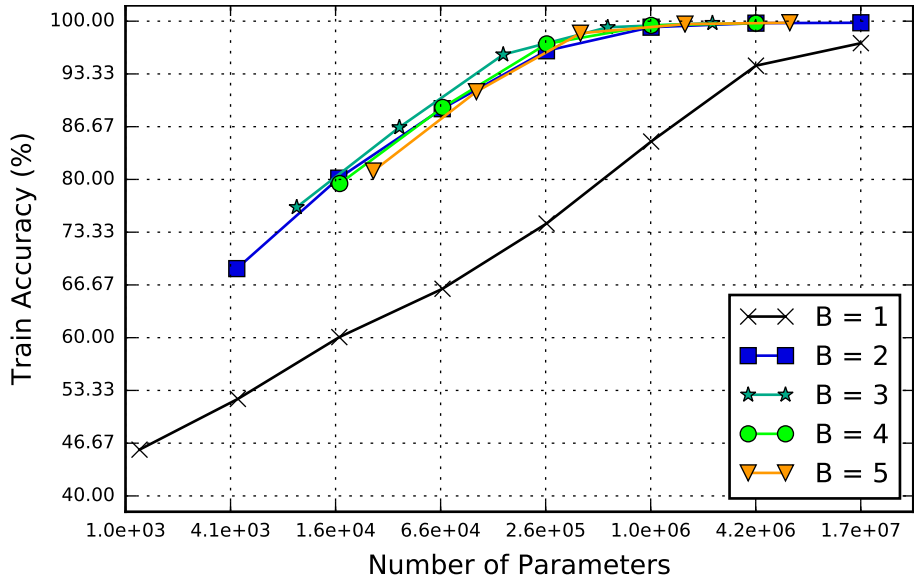
Almost all func's realizable by the above arch, cannot be replicated by a non-overlapping arch unless its size is at least $M^{\frac{(2B-1)^2}{4}}$

Outline

Experiments on Standard ConvNets

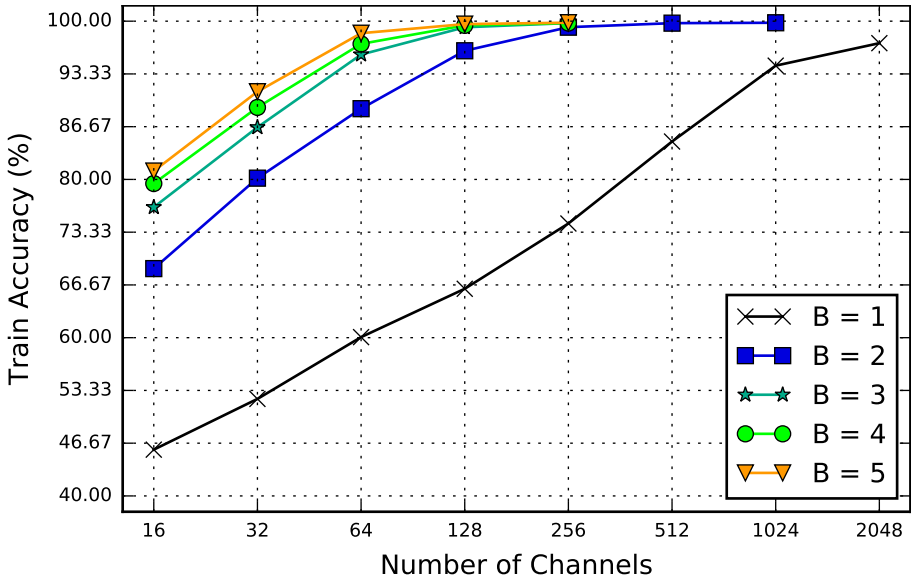
ConvNets following the arch of last claim were trained on CIFAR10, while varying the number channels and size of receptive field, denoted by B.

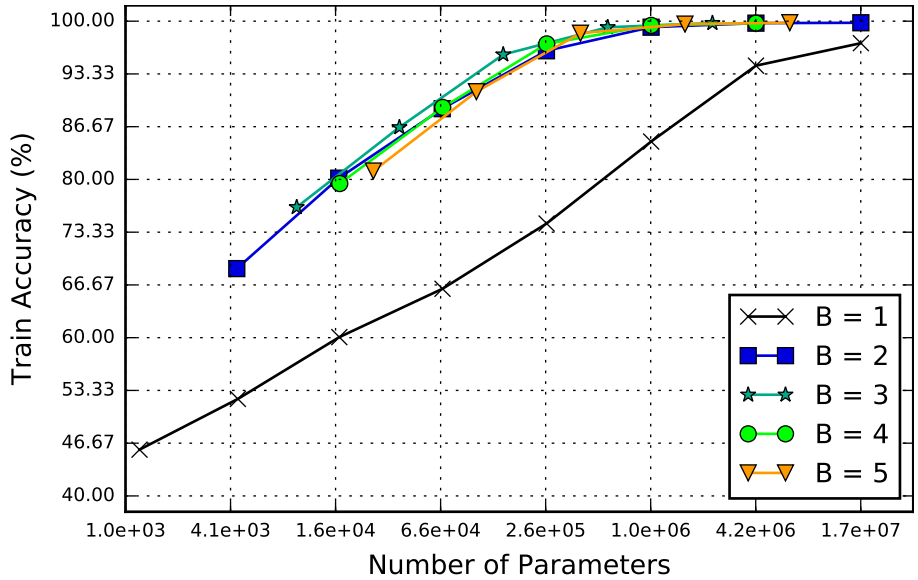




Experiments on Standard ConvNets

ConvNets following the arch of last claim were trained on CIFAR10, while varying the number channels and size of receptive field, denoted by B.





Conjecture

Increasing the overlapping degree beyond a certain point brings little to no gains in expressive efficiency!

Outline

Summary

- Comparing different arch's through **expressive efficiency**.

Summary

- Comparing different arch's through **expressive efficiency**.
- Overlapping arch's are efficient w.r.t. non-overlapping ones:

Summary

- Comparing different arch's through **expressive efficiency**.
- Overlapping arch's are efficient w.r.t. non-overlapping ones:
  - Proven in the case of ConvACs
  - Holds even for arch's of small overlapping degree
  - Experiments suggest analysis holds for standard ConvNets as well.

Summary

- Comparing different arch's through **expressive efficiency**.
- Overlapping arch's are efficient w.r.t. non-overlapping ones:
  - Proven in the case of ConvACs
  - Holds even for arch's of small overlapping degree
  - Experiments suggest analysis holds for standard ConvNets as well.
- **Concolusions:**
  - Non-overlapping arch's are uncommon out of lack of efficiency
  - **Conjecture:** Small overlapping degree *might be* all we need

Thank You

Backup Slides

Measure Efficiency via Grid Tensors

- Comparing functions directly can be ill-defined.
- Instead, compare functions via the **grid tensors** they induce:
  - Denote by $f(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ the function realized by the network.
  - $f(\cdot)$ may be studied by *discretizing* each $\mathbf{x}_i$ into one of $\{\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(M)}\}$:

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

- **Efficiency:** the minimal size required to induce a given grid-tensor.
- **Universality of ConvACs:** Any arch can induce any grid tensor, given sufficient number of channels.
- $\Rightarrow$ Efficiency via grid tensors is well-defined!

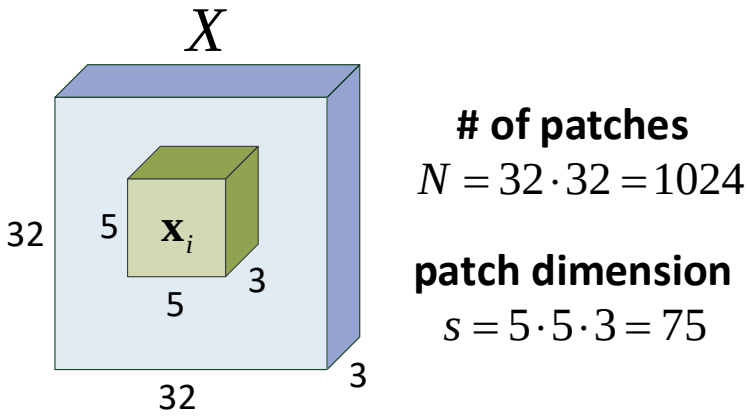
Tensorial Function Spaces

- We represent instances (images) 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 a basis of functions over patches, e.g. neurons:

$$f_{\theta_d=(\mathbf{w}_d, b_d)}(\mathbf{x}) = \sigma(\mathbf{w}_d^\top \mathbf{x} + b_d)$$

Denote $\mathcal{F} = span\{f_{\theta_1} \ldots f_{\theta_M}\}$

Tensorial Function Spaces (cont')

- $\mathcal{F}^{\otimes N}$ – extension of $\mathcal{F}$ from patches to images, i.e. the space of functions over images spanned by:

$$(\mathbf{x}_1, \ldots, \mathbf{x}_N) \mapsto \prod_{i=1}^{N} f_{\theta_{d_i}}(\mathbf{x}_i) \quad , \quad d_1 \ldots d_N \in [M]$$

(formally known as the tensor product of $\mathcal{F}$ with itself N times)

- General function $h \in \mathcal{F}^{\otimes N}$ 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

Tensor Decompositions

Tensor – multi-dimensional array:

$$\mathcal{A}_{d_1 \ldots d_N} \in \mathbb{R} \quad , d_1 \ldots d_N \in [M]$$

Tensor Decompositions

Tensor – multi-dimensional array:

$$\mathcal{A}_{d_1 \ldots d_N} \in \mathbb{R} \quad , d_1 \ldots d_N \in [M]$$

Suppose we would like to draw an entry from tensor $\mathcal{A}$:

approach	computation complexity	storage complexity
naïve (lookup table)	constant	exponential (in N)

Tensor Decompositions

Tensor – multi-dimensional array:

$$\mathcal{A}_{d_1 \ldots d_N} \in \mathbb{R} \quad , d_1 \ldots d_N \in [M]$$

Suppose we would like to draw an entry from tensor $\mathcal{A}$:

approach	computation complexity	storage complexity
naïve (lookup table)	constant	exponential (in N)
tensor decomposition	polynomial	polynomial

Tensor Decompositions

Tensor – multi-dimensional array:

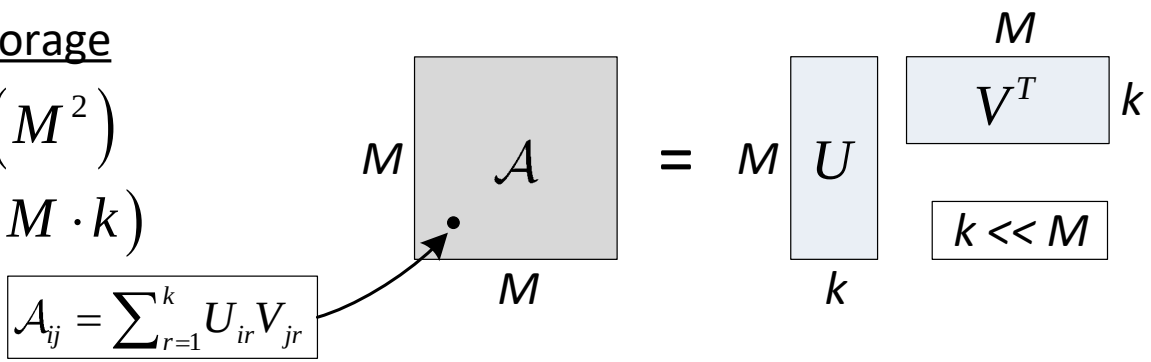
$$\mathcal{A}_{d_1 \ldots d_N} \in \mathbb{R} \quad , d_1 \ldots d_N \in [M]$$

Suppose we would like to draw an entry from tensor $\mathcal{A}$:

approach	computation complexity	storage complexity
naïve (lookup table)	constant	exponential (in N)
tensor decomposition	polynomial	polynomial

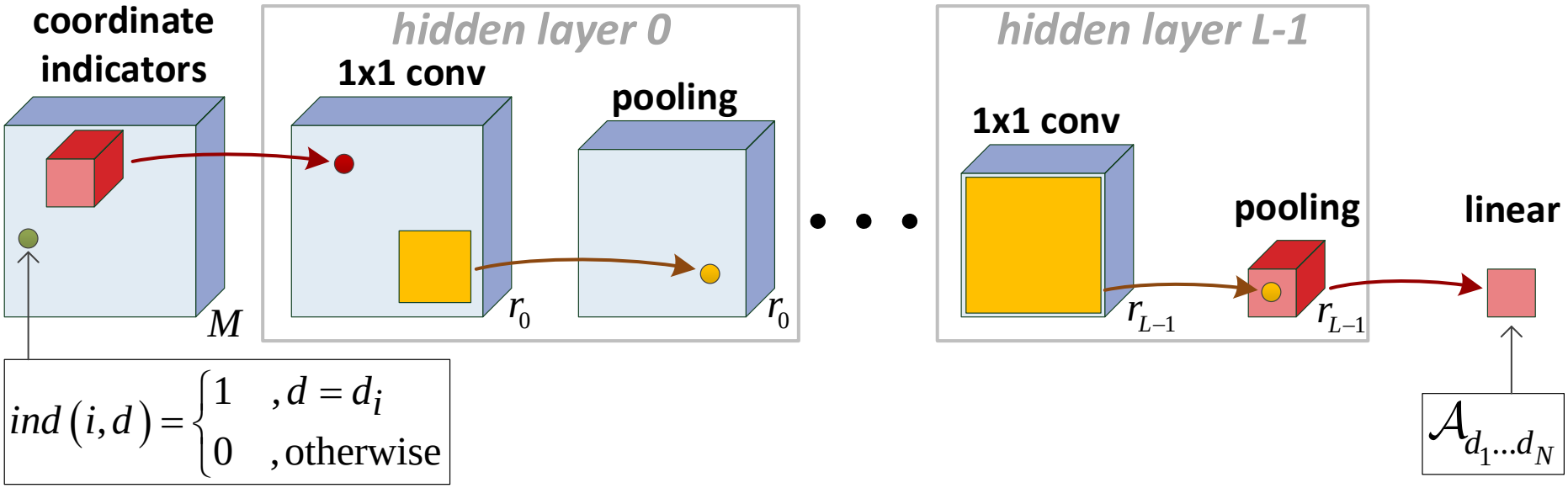
Special case $N = 2$ – low-rank matrix decomposition:

	computation	storage
lookup table	$\mathcal{O}(1)$	$\mathcal{O}(M^2)$
decomposition	$\mathcal{O}(k)$	$\mathcal{O}(M \cdot k)$



Tensor Decompositions (cont')

For general order N, tensor decomposition is realized by convolutional arithmetic circuit over coordinate $(d_1 \ldots d_N)$ indicators:



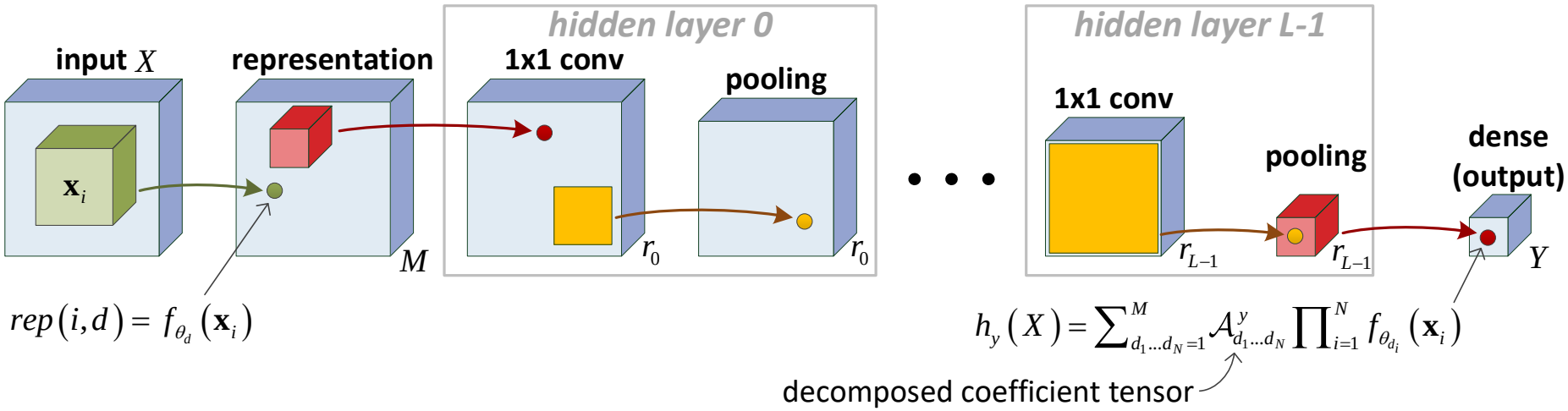
1-1 correspondence between type of tensor decomposition and structure of network (# of layers, pooling schemes, layer widths etc)

Computing Functions by Decomposing Coefficient Tensors

$h_1 \ldots h_Y$ – functions over images:

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

With tensor decomposition applied to $\mathcal{A}^y$, functions h_y are computed by convolutional arithmetic circuit over $\{f_{\theta_d}(\mathbf{x}_i)\}_{d \in [M], i \in [N]}$ (**representation**):



Again:

1-1 correspondence between decomposition type and network structure

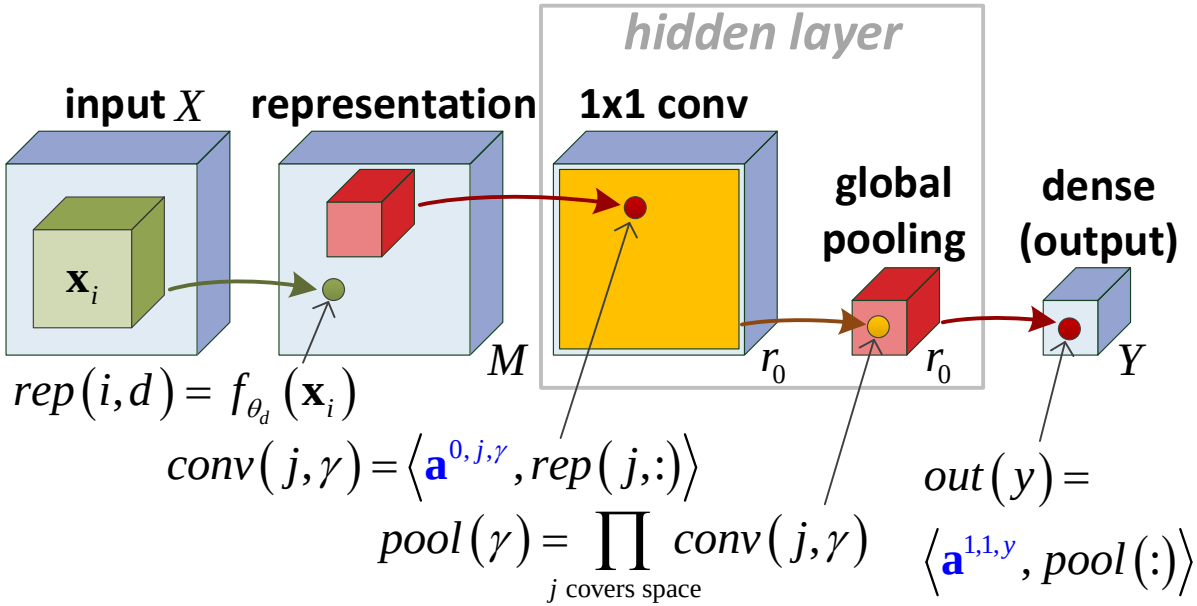
CP (CANDECOMP/PARAFAC) Decomposition $\longleftrightarrow$ Shallow Convolutional Arithmetic Circuit

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

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

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

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



Hierarchical Tucker Decomposition $\longleftrightarrow$ Deep Convolutional Arithmetic Circuit

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

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

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