On Expressiveness and Optimization in Deep Learning

Nadav Cohen

Institute for Advanced Study

(funding provided by Eric and Wendy Schmidt)

School of Mathematics Members' Seminar

2 April 2018

Sources

Deep SimNets
C + Or Sharir + Amnon Shashua
Computer Vision and Pattern Recognition (CVPR) 2016

On the Expressive Power of Deep Learning: A Tensor Analysis
C + Or Sharir + Amnon Shashua
Conference on Learning Theory (COLT) 2016

Convolutional Rectifier Networks as Generalized Tensor Decompositions
C + Amnon Shashua
International Conference on Machine Learning (ICML) 2016

Inductive Bias of Deep Convolutional Networks through Pooling Geometry
C + Amnon Shashua
International Conference on Learning Representations (ICLR) 2017

Boosting Dilated Convolutional Networks with Mixed Tensor Decompositions
C + Ronen Tamari + Amnon Shashua
International Conference on Learning Representations (ICLR) 2018

Deep Learning and Quantum Entanglement:
Fundamental Connections with Implications to Network Design
Yoav Levine + David Yakira + *C* + Amnon Shashua
International Conference on Learning Representations (ICLR) 2018

On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization
Sanjeev Arora + *C* + Elad Hazan
arXiv preprint 2018

Collaborators

Ronen Tamari

Or Sharir

Amnon Shashua

Yoav Levine

David Yakira

האוניברסיטה העברית בירושלים
THE HEBREW UNIVERSITY OF JERUSALEM

Sanjeev Arora

PRINCETON UNIVERSITY

Elad Hazan

Google

Outline

Statistical Learning Setup

Statistical Learning Setup

$\mathcal{X}$ – instance space (e.g. $\mathbb{R}^{100\times100}$ for 100-by-100 grayscale images)

Statistical Learning Setup

$\mathcal{X}$ – instance space (e.g. $\mathbb{R}^{100\times100}$ for 100-by-100 grayscale images)

$\mathcal{Y}$ – label space (e.g. $\mathbb{R}$ for regression or $[k] := \{1,\ldots,k\}$ for classification)

Statistical Learning Setup

$\mathcal{X}$ – instance space (e.g. $\mathbb{R}^{100 \times 100}$ for 100-by-100 grayscale images)

$\mathcal{Y}$ – label space (e.g. $\mathbb{R}$ for regression or $[k] := \{1, \ldots, k\}$ for classification)

$\mathcal{D}$ – distribution over $\mathcal{X} \times \mathcal{Y}$ (unknown)

Statistical Learning Setup

$\mathcal{X}$ – instance space (e.g. $\mathbb{R}^{100\times 100}$ for 100-by-100 grayscale images)

$\mathcal{Y}$ – label space (e.g. $\mathbb{R}$ for regression or $[k] := \{1, \ldots, k\}$ for classification)

$\mathcal{D}$ – distribution over $\mathcal{X} \times \mathcal{Y}$ (unknown)

$\ell : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}_{\geq 0}$ – loss func (e.g. $\ell(y, \hat{y}) = (y - \hat{y})^2$ for $\mathcal{Y} = \mathbb{R}$)

Statistical Learning Setup

$\mathcal{X}$ – instance space (e.g. $\mathbb{R}^{100\times100}$ for 100-by-100 grayscale images)

$\mathcal{Y}$ – label space (e.g. $\mathbb{R}$ for regression or $[k] := \{1,\ldots,k\}$ for classification)

$\mathcal{D}$ – distribution over $\mathcal{X} \times \mathcal{Y}$ (unknown)

$\ell : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}_{\geq 0}$ – loss func (e.g. $\ell(y,\hat{y}) = (y-\hat{y})^2$ for $\mathcal{Y} = \mathbb{R}$)

Task

Given training sample $S = \{(X_1, y_1), \ldots, (X_m, y_m)\}$ drawn i.i.d. from $\mathcal{D}$, return hypothesis (predictor) $h : \mathcal{X} \to \mathcal{Y}$ that minimizes population loss:

$$L_{\mathcal{D}}(h) := \mathbb{E}_{(X,y)\sim\mathcal{D}}[\ell(y, h(X))]$$

Statistical Learning Setup

$\mathcal{X}$ – instance space (e.g. $\mathbb{R}^{100\times 100}$ for 100-by-100 grayscale images)

$\mathcal{Y}$ – label space (e.g. $\mathbb{R}$ for regression or $[k] := \{1, \ldots, k\}$ for classification)

$\mathcal{D}$ – distribution over $\mathcal{X} \times \mathcal{Y}$ (unknown)

$\ell : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}_{\geq 0}$ – loss func (e.g. $\ell(y, \hat{y}) = (y - \hat{y})^2$ for $\mathcal{Y} = \mathbb{R}$)

Task

Given training sample $S = \{(X_1, y_1), \ldots, (X_m, y_m)\}$ drawn i.i.d. from $\mathcal{D}$, return hypothesis (predictor) $h : \mathcal{X} \to \mathcal{Y}$ that minimizes population loss:

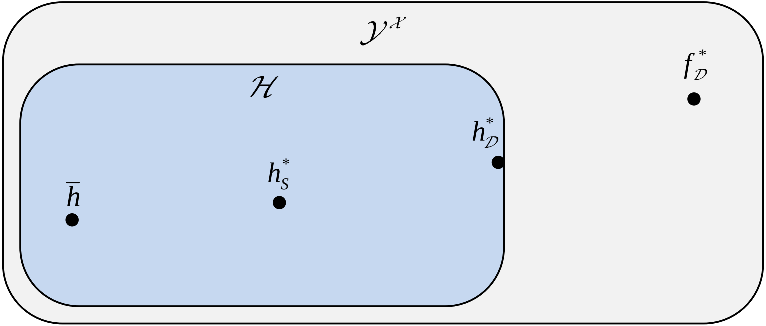
$$L_{\mathcal{D}}(h) := \mathbb{E}_{(X,y)\sim\mathcal{D}}[\ell(y, h(X))]$$

Approach

Predetermine hypotheses space $\mathcal{H} \subset \mathcal{Y}^{\mathcal{X}}$, and return hypothesis $h \in \mathcal{H}$ that minimizes empirical loss:

$$L_S(h) := \mathbb{E}_{(X,y)\sim S}[\ell(y, h(X))] = \frac{1}{m}\sum_{i=1}^{m} \ell(y_i, h(X_i))$$

Three Pillars of Statistical Learning Theory: Expressiveness, Generalization and Optimization



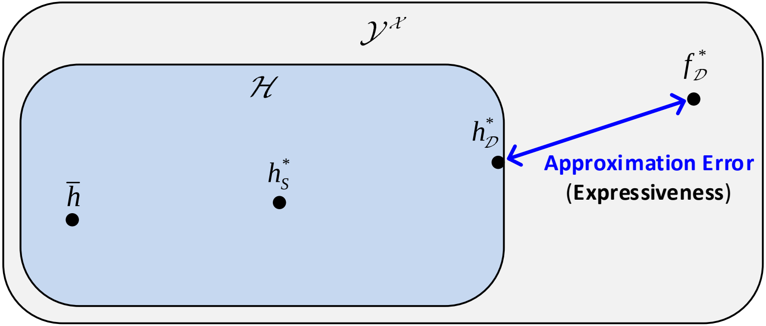
$f^*_{\mathcal{D}}$ – ground truth ($\text{argmin}_{f \in \mathcal{Y}^{\mathcal{X}}} L_{\mathcal{D}}(f)$)

$h^*_{\mathcal{D}}$ – optimal hypothesis ($\text{argmin}_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$)

h^*_S – empirically optimal hypothesis ($\text{argmin}_{h \in \mathcal{H}} L_S(h)$)

$\bar{h}$ – returned hypothesis

Three Pillars of Statistical Learning Theory: Expressiveness, Generalization and Optimization



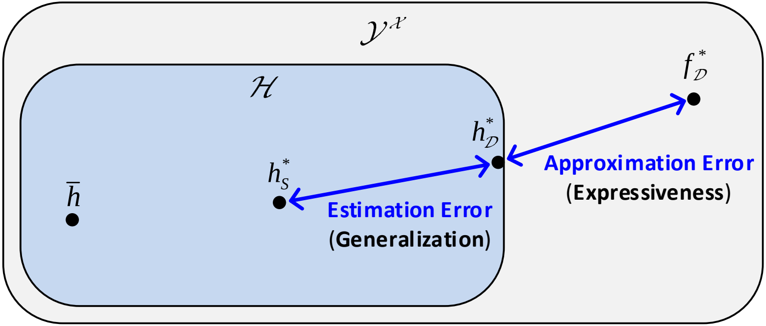
$f_{\mathcal{D}}^*$ – ground truth ($\text{argmin}_{f \in \mathcal{Y}^{\mathcal{X}}} L_{\mathcal{D}}(f)$)

$h_{\mathcal{D}}^*$ – optimal hypothesis ($\text{argmin}_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$)

h_S^* – empirically optimal hypothesis ($\text{argmin}_{h \in \mathcal{H}} L_S(h)$)

$\bar{h}$ – returned hypothesis

Three Pillars of Statistical Learning Theory: Expressiveness, Generalization and Optimization



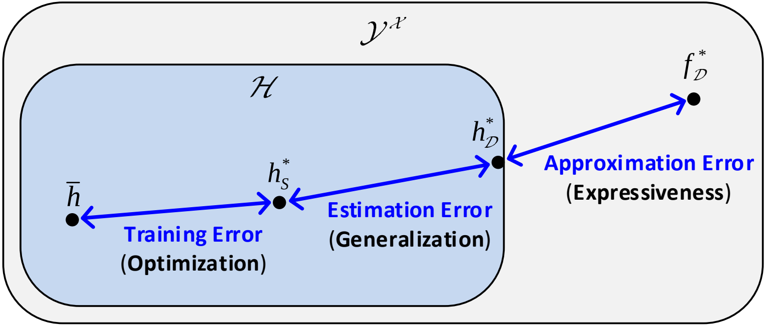
$f^*_{\mathcal{D}}$ – ground truth ($\text{argmin}_{f \in \mathcal{Y}^{\mathcal{X}}} L_{\mathcal{D}}(f)$)

$h^*_{\mathcal{D}}$ – optimal hypothesis ($\text{argmin}_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$)

h^*_S – empirically optimal hypothesis ($\text{argmin}_{h \in \mathcal{H}} L_S(h)$)

$\bar{h}$ – returned hypothesis

Three Pillars of Statistical Learning Theory: Expressiveness, Generalization and Optimization



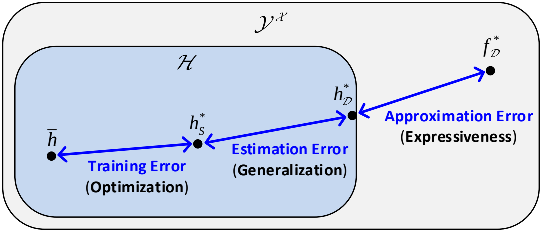
$f^*_{\mathcal{D}}$ – ground truth ($\text{argmin}_{f \in \mathcal{Y}^{\mathcal{X}}} L_{\mathcal{D}}(f)$)

$h^*_{\mathcal{D}}$ – optimal hypothesis ($\text{argmin}_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$)

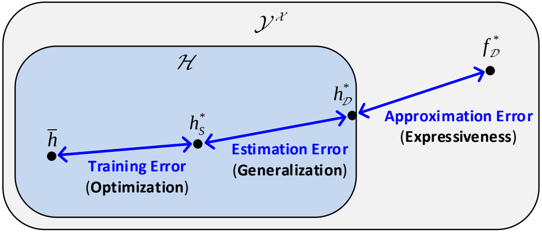
h^*_S – empirically optimal hypothesis ($\text{argmin}_{h \in \mathcal{H}} L_S(h)$)

$\bar{h}$ – returned hypothesis

Classical Machine Learning



Classical Machine Learning

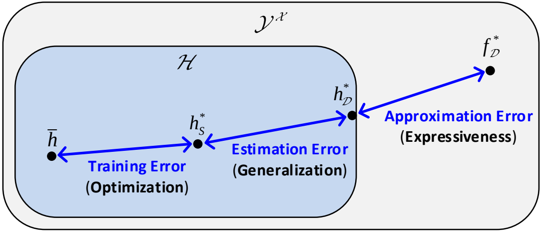


Optimization

Empirical loss minimization is a convex program:

$$\bar{h} \approx h^*_S \quad (\text{ training err} \approx 0 \text{ })$$

Classical Machine Learning



Optimization

Empirical loss minimization is a convex program:

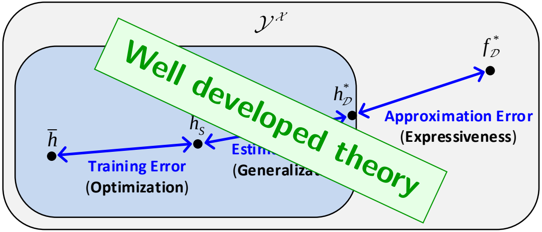
$$\bar{h} \approx h_S^* \quad (\text{ training err} \approx 0 \text{ })$$

Expressiveness & Generalization

Bias-variance trade-off:

$\mathcal{H}$	approximation err	estimation err
expands	$\searrow$	$\nearrow$
shrinks	$\nearrow$	$\searrow$

Classical Machine Learning



Optimization

Empirical loss minimization is a convex program:

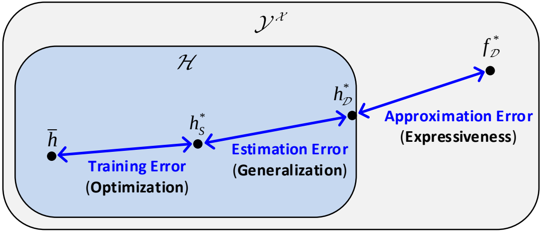
$$\bar{h} \approx h^*_S \quad (\text{ training err} \approx 0 \text{ })$$

Expressiveness & Generalization

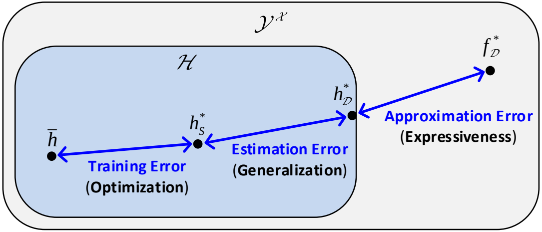
Bias-variance trade-off:

$\mathcal{H}$	approximation err	estimation err
expands	$\searrow$	$\nearrow$
shrinks	$\nearrow$	$\searrow$

Deep Learning



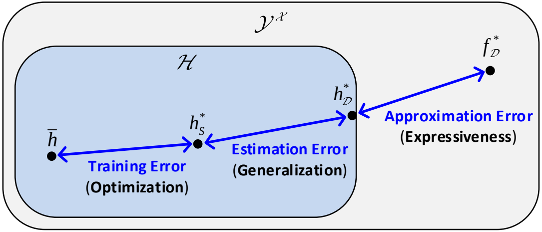
Deep Learning



Optimization

Empirical loss minimization is a non-convex program:

Deep Learning

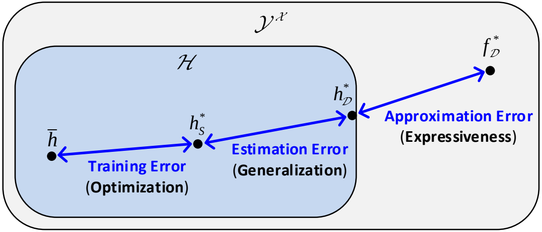


Optimization

Empirical loss minimization is a non-convex program:

- h^*_S is not unique – many hypotheses have low training err

Deep Learning

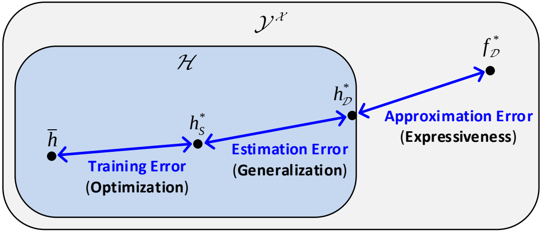


Optimization

Empirical loss minimization is a non-convex program:

- h^*_S is not unique – many hypotheses have low training err
- Stochastic Gradient Descent somehow reaches one of these

Deep Learning



Optimization

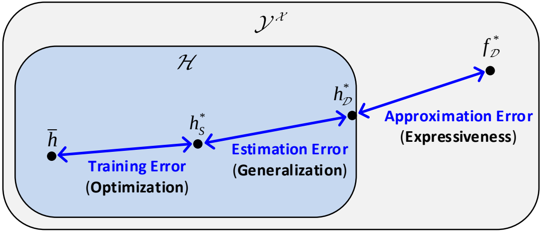
Empirical loss minimization is a non-convex program:

- h^*_S is not unique – many hypotheses have low training err
- Stochastic Gradient Descent somehow reaches one of these

Expressiveness & Generalization

Vast difference from classical ML:

Deep Learning



Optimization

Empirical loss minimization is a non-convex program:

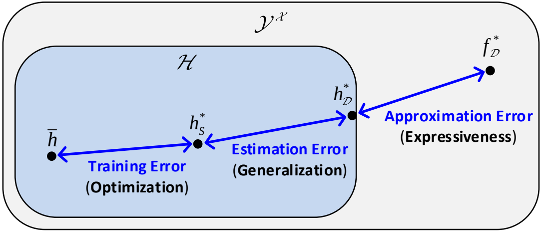
- h_S^* is not unique – many hypotheses have low training err
- Stochastic Gradient Descent somehow reaches one of these

Expressiveness & Generalization

Vast difference from classical ML:

- Some low training err hypotheses generalize well, others don't

Deep Learning



Optimization

Empirical loss minimization is a non-convex program:

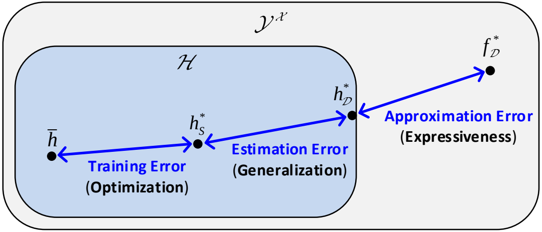
- h_S^* is not unique – many hypotheses have low training err
- Stochastic Gradient Descent somehow reaches one of these

Expressiveness & Generalization

Vast difference from classical ML:

- Some low training err hypotheses generalize well, others don't
- W/typical data, solution returned by SGD often generalizes well

Deep Learning



Optimization

Empirical loss minimization is a non-convex program:

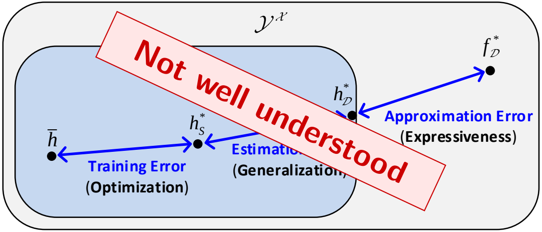
- h^*_S is not unique – many hypotheses have low training err
- Stochastic Gradient Descent somehow reaches one of these

Expressiveness & Generalization

Vast difference from classical ML:

- Some low training err hypotheses generalize well, others don't
- W/typical data, solution returned by SGD often generalizes well
- Expanding $\mathcal{H}$ reduces approximation err, but also estimation err!

Deep Learning



Optimization

Empirical loss minimization is a non-convex program:

- h_S^* is not unique – many hypotheses have low training err
- Stochastic Gradient Descent somehow reaches one of these

Expressiveness & Generalization

Vast difference from classical ML:

- Some low training err hypotheses generalize well, others don't
- W/typical data, solution returned by SGD often generalizes well
- Expanding $\mathcal{H}$ reduces approximation err, but also estimation err!

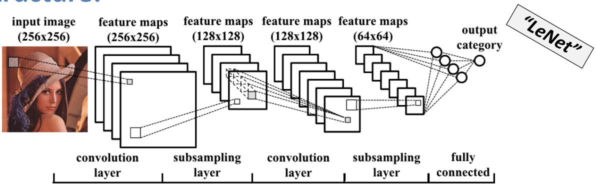
Outline

1. Deep Learning Theory: Expressiveness, Optimization and Generalization
2. Convolutional Networks as Hierarchical Tensor Decompositions
3. Expressiveness of Convolutional Networks
   - Efficiency of Depth *(C|Sharir|Shashua@COLT'16, C|Shashua@ICML'16)*
   - Modeling Interactions *(Levine|Yakira|C|Shashua@ICLR'18, C|Shashua@ICLR'17)*
   - Efficiency of Interconnectivity *(C|Tamari|Shashua@ICLR'18)*
4. Towards Optimization
   - Analysis for Linear Networks *(Arora|C|Hazan@arXiv'18)*
5. Conclusion

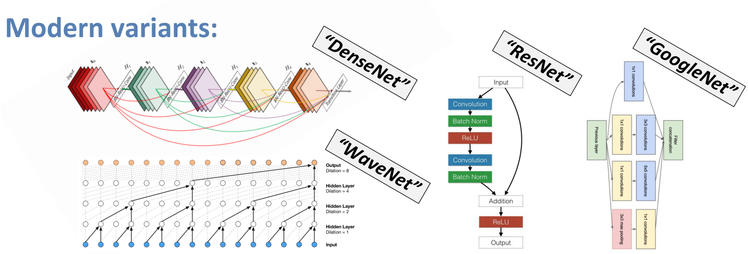
Convolutional Networks

Most successful deep learning arch to date!

Classic structure:



Modern variants:



Traditionally used for images/video, nowadays for audio and text as well

Tensor Product of L^2 Spaces

ConvNets realize func over many local elements (e.g. pixels, audio samples)

[1] Set of linearly independent func w/dense span

Tensor Product of L^2 Spaces

ConvNets realize func over many local elements (e.g. pixels, audio samples)

Let $\mathbb{R}^s$ be the space of such elements (e.g. $\mathbb{R}^3$ for RGB pixels)

[1]Set of linearly independent func w/dense span

Tensor Product of L^2 Spaces

ConvNets realize func over many local elements (e.g. pixels, audio samples)

Let $\mathbb{R}^s$ be the space of such elements (e.g. $\mathbb{R}^3$ for RGB pixels)

Consider:

- $L^2(\mathbb{R}^s)$ – space of func over single element
- $L^2((\mathbb{R}^s)^N)$ – space of func over N elements

[1]Set of linearly independent func w/dense span

Tensor Product of L^2 Spaces

ConvNets realize func over many local elements (e.g. pixels, audio samples)

Let $\mathbb{R}^s$ be the space of such elements (e.g. $\mathbb{R}^3$ for RGB pixels)

Consider:

- $L^2(\mathbb{R}^s)$ – space of func over single element
- $L^2((\mathbb{R}^s)^N)$ – space of func over N elements

Fact

$L^2((\mathbb{R}^s)^N)$ is equal to the tensor product of $L^2(\mathbb{R}^s)$ with itself N times:

$$L^2((\mathbb{R}^s)^N) = \underbrace{L^2(\mathbb{R}^s) \otimes \cdots \otimes L^2(\mathbb{R}^s)}_{N \text{ times}}$$

[1] Set of linearly independent func w/dense span

Tensor Product of L^2 Spaces

ConvNets realize func over many local elements (e.g. pixels, audio samples)

Let $\mathbb{R}^s$ be the space of such elements (e.g. $\mathbb{R}^3$ for RGB pixels)

Consider:

- $L^2(\mathbb{R}^s)$ – space of func over single element
- $L^2((\mathbb{R}^s)^N)$ – space of func over N elements

Fact

$L^2((\mathbb{R}^s)^N)$ is equal to the tensor product of $L^2(\mathbb{R}^s)$ with itself N times:

$$L^2((\mathbb{R}^s)^N) = \underbrace{L^2(\mathbb{R}^s) \otimes \cdots \otimes L^2(\mathbb{R}^s)}_{N \text{ times}}$$

Implication

If $\{f_d(\mathbf{x})\}_{d=1}^{\infty}$ is a basis[1] for $L^2(\mathbb{R}^s)$, the following is a basis for $L^2((\mathbb{R}^s)^N)$:

$$\left\{(\mathbf{x}_1, \ldots, \mathbf{x}_N) \mapsto \prod\nolimits_{i=1}^{N} f_{d_i}(\mathbf{x}_i)\right\}_{d_1 \ldots d_N = 1}^{\infty}$$

[1] Set of linearly independent func w/dense span

Coefficient Tensor

For practical purposes, restrict $L^2(\mathbb{R}^s)$ basis to a finite set: $f_1(\mathbf{x})\ldots f_M(\mathbf{x})$

We call $f_1(\mathbf{x})\ldots f_M(\mathbf{x})$ **descriptors**

Coefficient Tensor

For practical purposes, restrict $L^2(\mathbb{R}^s)$ basis to a finite set: $f_1(\mathbf{x})...f_M(\mathbf{x})$

We call $f_1(\mathbf{x})...f_M(\mathbf{x})$ **descriptors**

General func over N elements can now be written as:

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

w/func fully determined by the **coefficient tensor**:

$$\mathcal{A} \in \mathbb{R}^{\overbrace{M \times \cdots \times M}^{N \text{ times}}}$$

Coefficient Tensor

For practical purposes, restrict $L^2(\mathbb{R}^s)$ basis to a finite set: $f_1(\mathbf{x})...f_M(\mathbf{x})$

We call $f_1(\mathbf{x})...f_M(\mathbf{x})$ **descriptors**

General func over N elements can now be written as:

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

w/func fully determined by the **coefficient tensor**:

$$\mathcal{A}\in\mathbb{R}^{\overbrace{M\times\cdots\times M}^{N\text{ times}}}$$

Example

- 100-by-100 images ($N = 10^4$)
- pixels represented by 256 descriptors ($M = 256$)

Coefficient Tensor

For practical purposes, restrict $L^2(\mathbb{R}^s)$ basis to a finite set: $f_1(\mathbf{x})\ldots f_M(\mathbf{x})$

We call $f_1(\mathbf{x})\ldots f_M(\mathbf{x})$ **descriptors**

General func over N elements can now be written as:

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

w/func fully determined by the **coefficient tensor**:

$$\mathcal{A} \in \mathbb{R}^{\overbrace{M \times \cdots \times M}^{N \text{ times}}}$$

Example

- 100-by-100 images ($N = 10^4$)
- pixels represented by 256 descriptors ($M = 256$)

Then, func over images correspond to coeff tensors of:

- order 10^4
- dim 256 in each mode

Decomposing Coefficient Tensor $\longrightarrow$ Convolutional Arithmetic Circuit

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

Coeff tensor $\mathcal{A}$ is exponential (in # of elements N)

$\Longrightarrow$ directly computing a general func is intractable

Decomposing Coefficient Tensor $\longrightarrow$ Convolutional Arithmetic Circuit

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

Coeff tensor $\mathcal{A}$ is exponential (in # of elements N)

$\Longrightarrow$ directly computing a general func is intractable

Observation

Applying hierarchical decomposition to coeff tensor gives ConvNet w/linear activation and product pooling (**Convolutional Arithmetic Circuit**)!

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

Example 1: CP Decomposition $\longrightarrow$ Shallow Network

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

W/CP decomposition applied to coeff tensor:

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

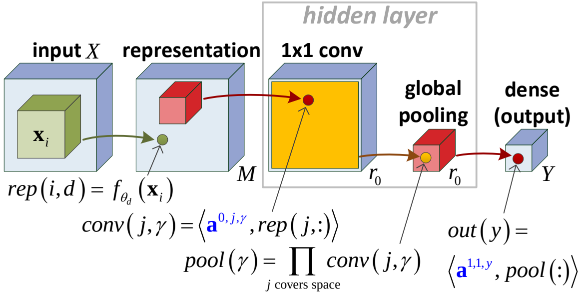
Example 1: CP Decomposition $\longrightarrow$ Shallow Network

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

W/CP decomposition applied to coeff tensor:

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

func is computed by shallow network (single hidden layer, global pooling):



Example 2: HT Decomposition $\longrightarrow$ Deep Network

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

W/Hierarchical Tucker (HT) decomposition applied to coeff tensor:

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

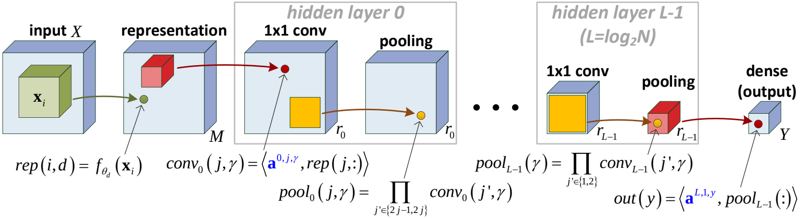
Example 2: HT Decomposition $\longrightarrow$ Deep Network

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

W/Hierarchical Tucker (HT) decomposition applied to coeff tensor:

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

func is computed by deep network w/size-2 pooling windows:



Generalization to Other Types of Convolutional Networks

We established equivalence:

hierarchical tensor decompositions $\longleftrightarrow$ conv arith circuits (ConvACs)

[1] *Deep SimNets, CVPR'16*

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

Generalization to Other Types of Convolutional Networks

We established equivalence:

hierarchical tensor decompositions $\longleftrightarrow$ conv arith circuits (ConvACs)

ConvACs deliver promising empirical results,[1] but other types of ConvNets (e.g. w/ReLU activation and max/ave pooling) are much more common

[1] *Deep SimNets, CVPR'16*

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

Generalization to Other Types of Convolutional Networks

We established equivalence:

hierarchical tensor decompositions $\longleftrightarrow$ conv arith circuits (ConvACs)

ConvACs deliver promising empirical results,[1] but other types of ConvNets (e.g. w/ReLU activation and max/ave pooling) are much more common

The equivalence extends to other types of ConvNets if we generalize the notion of tensor product:[2]

Tensor product:

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

Generalized tensor product:

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

(same as $\otimes$ but w/general $g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ instead of mult)

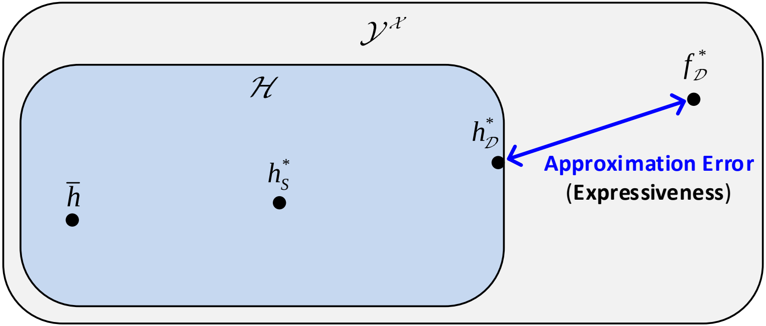
[1] *Deep SimNets, CVPR'16*

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

Outline

1. Deep Learning Theory: Expressiveness, Optimization and Generalization
2. Convolutional Networks as Hierarchical Tensor Decompositions
3. Expressiveness of Convolutional Networks
   - Efficiency of Depth *(C|Sharir|Shashua@COLT'16, C|Shashua@ICML'16)*
   - Modeling Interactions *(Levine|Yakira|C|Shashua@ICLR'18, C|Shashua@ICLR'17)*
   - Efficiency of Interconnectivity *(C|Tamari|Shashua@ICLR'18)*
4. Towards Optimization
   - Analysis for Linear Networks *(Arora|C|Hazan@arXiv'18)*
5. Conclusion

Expressiveness



$f^*_{\mathcal{D}}$ – ground truth ($\text{argmin}_{f \in \mathcal{Y}^{\mathcal{X}}} L_{\mathcal{D}}(f)$)

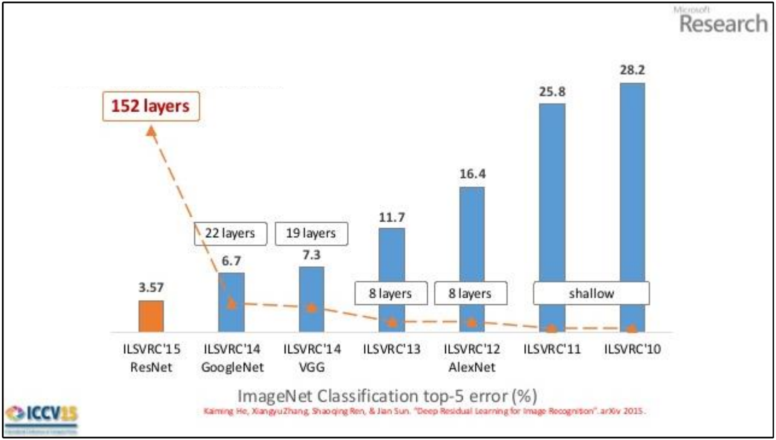
$h^*_{\mathcal{D}}$ – optimal hypothesis ($\text{argmin}_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$)

h^*_S – empirically optimal hypothesis ($\text{argmin}_{h \in \mathcal{H}} L_S(h)$)

$\bar{h}$ – returned hypothesis

Outline

Efficiency of Depth



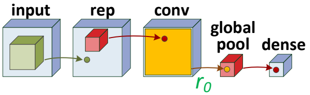
Longstanding conjecture

Efficiency of depth: deep ConvNets realize func that require shallow ConvNets to have exponential size (width)

Tensor Decomposition Viewpoint

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

Shallow Network

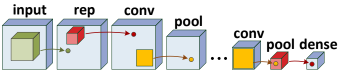


$\longleftrightarrow$

CP Decomposition

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

Deep Network



$\longleftrightarrow$

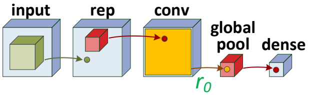
HT Decomposition

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

Tensor Decomposition Viewpoint

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

Shallow Network

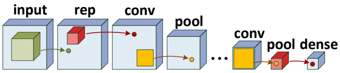


$\longleftrightarrow$

CP Decomposition

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

Deep Network



$\longleftrightarrow$

HT Decomposition

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

Efficiency of depth

HT decomposition realizes tensors that require CP decomposition to have exponential rank (r_0 exponential in N)

HT vs. CP Analysis

Theorem

Besides a negligible (zero measure) set, all parameter settings for HT decomposition lead to tensors w/CP-rank exponential in N

HT Decomposition

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

$$\cdots$$

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

$$\cdots$$

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

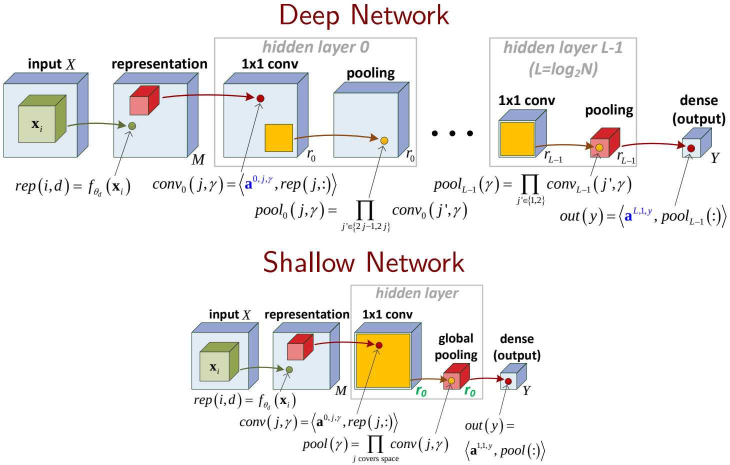
CP Decomposition

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

HT vs. CP Analysis (cont'd)

Corollary

Randomizing weights of deep ConvAC by a cont distribution leads, w.p. 1, to func that require shallow ConvAC to have exponential # of channels



HT vs. CP Analysis (cont'd)

Theorem proof sketch

- $[\![\mathcal{A}]\!]$ – matricization of $\mathcal{A}$ (arrangement of tensor as matrix)
- $\odot$ – Kronecker product for matrices. Holds: $rank(A \odot B) = rank(A) \cdot rank(B)$
- Relation between tensor and Kronecker products: $[\![\mathcal{A} \otimes \mathcal{B}]\!] = [\![\mathcal{A}]\!] \odot [\![\mathcal{B}]\!]$
- Implies: $rank[\![\mathcal{A}]\!] \leq CP\text{-}rank(\mathcal{A})$
- By induction over levels of HT, $rank[\![\mathcal{A}]\!]$ is exponential almost always:

HT Decomposition

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

  - Base: "SVD has maximal rank almost always"
  - 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 always"

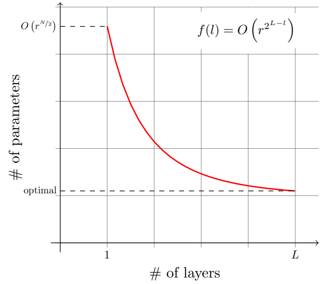
HT vs. CP Analysis – Generalizations

HT vs. CP analysis may be generalized in various ways, e.g.:

HT vs. CP Analysis – Generalizations

HT vs. CP analysis may be generalized in various ways, e.g.:

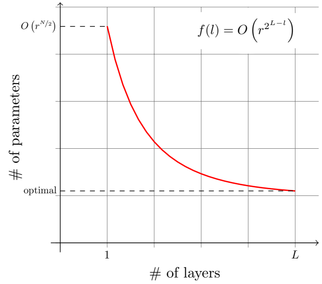
- **Comparison between arbitrary depths**
  Penalty in resources is double-exponential w.r.t. # of layers cut-off



HT vs. CP Analysis – Generalizations

HT vs. CP analysis may be generalized in various ways, e.g.:

- **Comparison between arbitrary depths**
  Penalty in resources is double-exponential w.r.t. # of layers cut-off

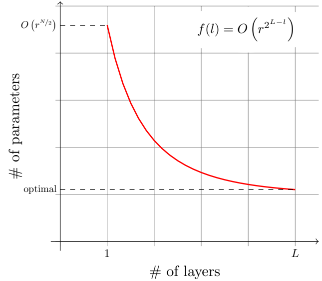


- **Adaptation to other types of ConvNets**
  W/ReLU activation and max pooling, deep nets realize func requiring shallow nets to be exponentially large, but not almost always

HT vs. CP Analysis – Generalizations

HT vs. CP analysis may be generalized in various ways, e.g.:

- **Comparison between arbitrary depths**
  Penalty in resources is double-exponential w.r.t. # of layers cut-off



- **Adaptation to other types of ConvNets**
  W/ReLU activation and max pooling, deep nets realize func requiring shallow nets to be exponentially large, but not almost always

Efficiency of depth proven!

Outline

Modeling Interactions

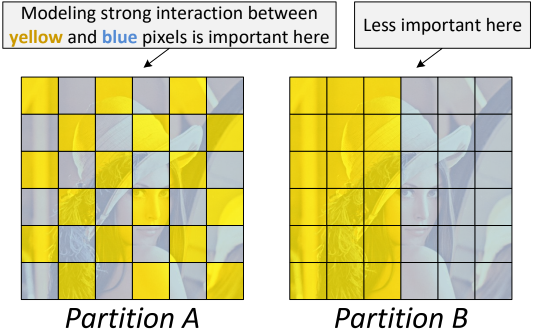
ConvNets realize func over many local elements (e.g. pixels, audio samples)

Modeling Interactions

ConvNets realize func over many local elements (e.g. pixels, audio samples)

Key property of such func:

interactions modeled between different sets of elements

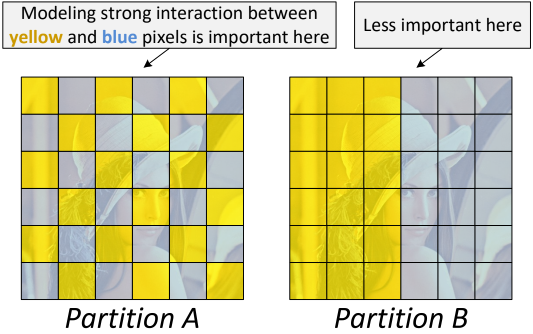


Modeling Interactions

ConvNets realize func over many local elements (e.g. pixels, audio samples)

Key property of such func:

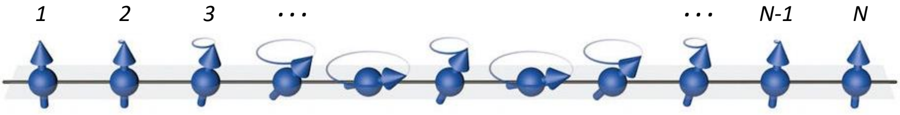
interactions modeled between different sets of elements



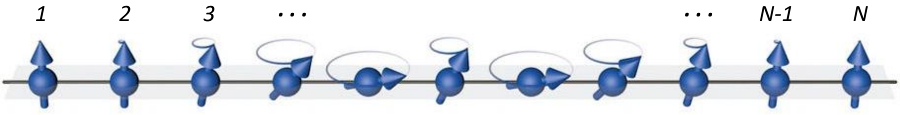
Questions

- What kind of interactions do ConvNets model?
- How do these depend on network structure?

Quantum Entanglement



Quantum Entanglement



In quantum physics, state of particle is represented as vec in Hilbert space:

$$|\text{particle state}\rangle = \sum_{d=1}^{M} \underbrace{a_d}_{\text{coeff}} \cdot \underbrace{|\psi_d\rangle}_{\text{basis}} \in \mathbf{H}$$

Quantum Entanglement

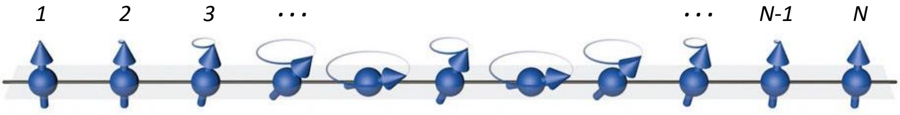
In quantum physics, state of particle is represented as vec in Hilbert space:

$$|\text{particle state}\rangle = \sum_{d=1}^{M} \underbrace{a_d}_{\text{coeff}} \cdot \underbrace{|\psi_d\rangle}_{\text{basis}} \in \mathbf{H}$$

System of N particles is represented as vec in tensor product space:

$$|\text{system state}\rangle = \sum_{d_1 \ldots d_N=1}^{M} \underbrace{\mathcal{A}_{d_1 \ldots d_N}}_{\text{coeff tensor}} \cdot |\psi_{d_1}\rangle \otimes \cdots \otimes |\psi_{d_N}\rangle \in \underbrace{\mathbf{H} \otimes \cdots \otimes \mathbf{H}}_{N \text{ times}}$$

Quantum Entanglement



In quantum physics, state of particle is represented as vec in Hilbert space:

$$|\text{particle state}\rangle = \sum_{d=1}^{M} \underbrace{a_d}_{\text{coeff}} \cdot \underbrace{|\psi_d\rangle}_{\text{basis}} \in \mathbf{H}$$

System of N particles is represented as vec in tensor product space:

$$|\text{system state}\rangle = \sum_{d_1 \ldots d_N=1}^{M} \underbrace{\mathcal{A}_{d_1 \ldots d_N}}_{\text{coeff tensor}} \cdot |\psi_{d_1}\rangle \otimes \cdots \otimes |\psi_{d_N}\rangle \in \underbrace{\mathbf{H} \otimes \cdots \otimes \mathbf{H}}_{N \text{ times}}$$

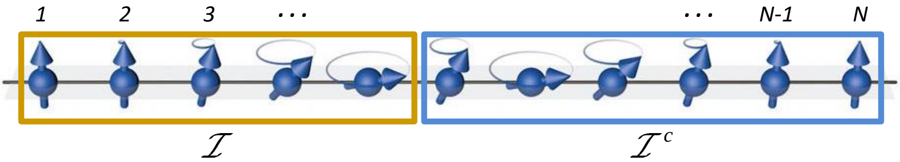
Quantum entanglement measures quantify interactions that a system state models between sets of particles

Quantum Entanglement (cont'd)

$$|\text{system state}\rangle = \sum_{d_1 \ldots d_N=1}^{M} \mathcal{A}_{d_1 \ldots d_N} \cdot |\psi_{d_1}\rangle \otimes \cdots \otimes |\psi_{d_N}\rangle$$

Quantum Entanglement (cont'd)

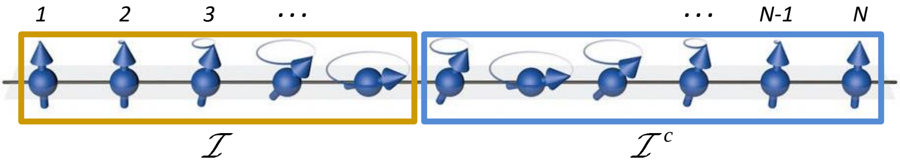
$$|\text{system state}\rangle = \sum_{d_1 \ldots d_N = 1}^{M} \mathcal{A}_{d_1 \ldots d_N} \cdot |\psi_{d_1}\rangle \otimes \cdots \otimes |\psi_{d_N}\rangle$$



Consider partition of the N particles into sets $\mathcal{I}$ and $\mathcal{I}^c$

Quantum Entanglement (cont'd)

$$|\text{system state}\rangle = \sum_{d_1 \ldots d_N=1}^{M} \mathcal{A}_{d_1 \ldots d_N} \cdot |\psi_{d_1}\rangle \otimes \cdots \otimes |\psi_{d_N}\rangle$$

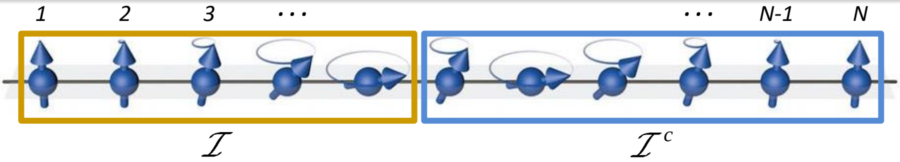


Consider partition of the N particles into sets $\mathcal{I}$ and $\mathcal{I}^c$

$[\![\mathcal{A}]\!]_{\mathcal{I}}$ – matricization of coeff tensor $\mathcal{A}$ w.r.t. $\mathcal{I}$:

- arrangement of $\mathcal{A}$ as matrix
- rows/cols correspond to modes indexed by $\mathcal{I}/\mathcal{I}^c$

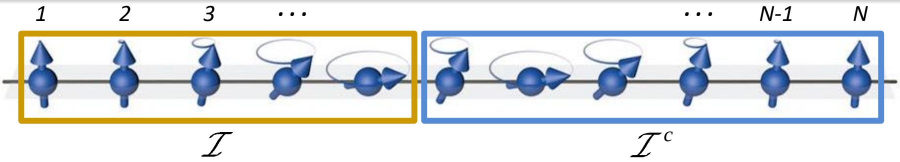
Quantum Entanglement (cont'd)



$$|\text{system state}\rangle = \sum_{d_1 \ldots d_N=1}^{M} \mathcal{A}_{d_1 \ldots d_N} \cdot |\psi_{d_1}\rangle \otimes \cdots \otimes |\psi_{d_N}\rangle$$

$[\![\mathcal{A}]\!]_{\mathcal{I}}$ – matricization of $\mathcal{A}$ w.r.t. $\mathcal{I}$

Quantum Entanglement (cont'd)

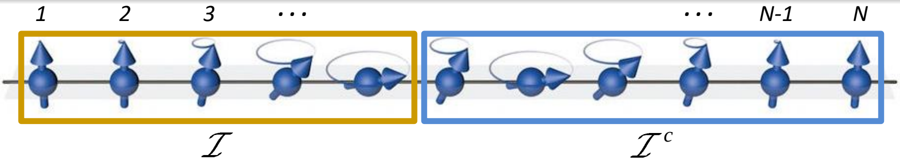


$$|\text{system state}\rangle = \sum_{d_1 \ldots d_N = 1}^{M} \mathcal{A}_{d_1 \ldots d_N} \cdot |\psi_{d_1}\rangle \otimes \cdots \otimes |\psi_{d_N}\rangle$$

$[\![\mathcal{A}]\!]_{\mathcal{I}}$ – matricization of $\mathcal{A}$ w.r.t. $\mathcal{I}$

Let $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_R)$ be the singular vals of $[\![\mathcal{A}]\!]_{\mathcal{I}}$

Quantum Entanglement (cont'd)



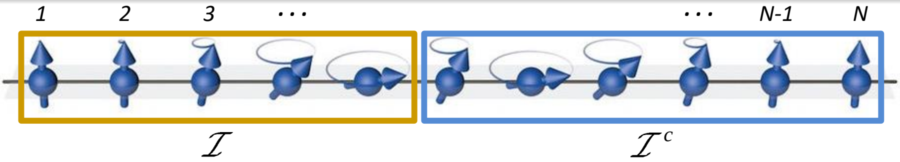
$$|\text{system state}\rangle = \sum_{d_1 \ldots d_N = 1}^{M} \mathcal{A}_{d_1 \ldots d_N} \cdot |\psi_{d_1}\rangle \otimes \cdots \otimes |\psi_{d_N}\rangle$$

$[\![\mathcal{A}]\!]_{\mathcal{I}}$ – matricization of $\mathcal{A}$ w.r.t. $\mathcal{I}$

Let $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_R)$ be the singular vals of $[\![\mathcal{A}]\!]_{\mathcal{I}}$

Entanglement measures between particles of $\mathcal{I}$ and of $\mathcal{I}^c$ are based on $\boldsymbol{\sigma}$:

Quantum Entanglement (cont'd)



$$|\text{system state}\rangle = \sum_{d_1 \ldots d_N=1}^{M} \mathcal{A}_{d_1 \ldots d_N} \cdot |\psi_{d_1}\rangle \otimes \cdots \otimes |\psi_{d_N}\rangle$$

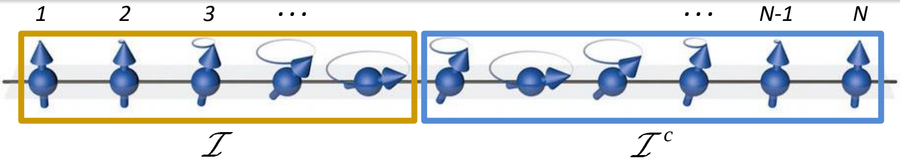
$[\![\mathcal{A}]\!]_{\mathcal{I}}$ – matricization of $\mathcal{A}$ w.r.t. $\mathcal{I}$

Let $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_R)$ be the singular vals of $[\![\mathcal{A}]\!]_{\mathcal{I}}$

Entanglement measures between particles of $\mathcal{I}$ and of $\mathcal{I}^c$ are based on $\boldsymbol{\sigma}$:

- Entanglement Entropy: entropy of $(\sigma_1^2, \ldots, \sigma_R^2) / \|\boldsymbol{\sigma}\|_2^2$

Quantum Entanglement (cont'd)



$$|\text{system state}\rangle = \sum_{d_1 \ldots d_N = 1}^{M} \mathcal{A}_{d_1 \ldots d_N} \cdot |\psi_{d_1}\rangle \otimes \cdots \otimes |\psi_{d_N}\rangle$$

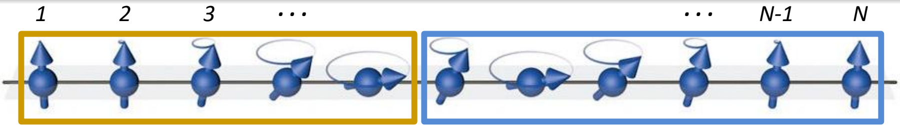
$[\![\mathcal{A}]\!]_{\mathcal{I}}$ – matricization of $\mathcal{A}$ w.r.t. $\mathcal{I}$

Let $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_R)$ be the singular vals of $[\![\mathcal{A}]\!]_{\mathcal{I}}$

Entanglement measures between particles of $\mathcal{I}$ and of $\mathcal{I}^c$ are based on $\boldsymbol{\sigma}$:

- Entanglement Entropy: entropy of $(\sigma_1^2, \ldots, \sigma_R^2) / \|\boldsymbol{\sigma}\|_2^2$
- Geometric Measure: $1 - \sigma_1^2 / \|\boldsymbol{\sigma}\|_2^2$

Quantum Entanglement (cont'd)



$\mathcal{I}$ $\mathcal{I}^c$

$$|\text{system state}\rangle = \sum_{d_1 \dots d_N = 1}^{M} \mathcal{A}_{d_1 \dots d_N} \cdot |\psi_{d_1}\rangle \otimes \cdots \otimes |\psi_{d_N}\rangle$$

$[\![\mathcal{A}]\!]_{\mathcal{I}}$ – matricization of $\mathcal{A}$ w.r.t. $\mathcal{I}$

Let $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \dots, \sigma_R)$ be the singular vals of $[\![\mathcal{A}]\!]_{\mathcal{I}}$

Entanglement measures between particles of $\mathcal{I}$ and of $\mathcal{I}^c$ are based on $\boldsymbol{\sigma}$:

- Entanglement Entropy: entropy of $(\sigma_1^2, \dots, \sigma_R^2) / \|\boldsymbol{\sigma}\|_2^2$
- Geometric Measure: $1 - \sigma_1^2 / \|\boldsymbol{\sigma}\|_2^2$
- **Schmidt Number**: $\|\boldsymbol{\sigma}\|_0 = rank[\![\mathcal{A}]\!]_{\mathcal{I}}$

Entanglement with Convolutional Arithmetic Circuits

Structural equivalence:

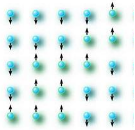
quantum system (many-body) state

$$|\text{system state}\rangle = \sum_{d_1 \ldots d_N = 1}^{M} \underbrace{\mathcal{A}_{d_1 \ldots d_N}}_{\text{coeff tensor}} \cdot |\psi_{d_1}\rangle \otimes \cdots \otimes |\psi_{d_N}\rangle$$

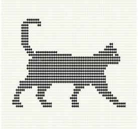
func realized by ConvAC

$$h(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{d_1 \ldots d_N = 1}^{M} \underbrace{\mathcal{A}_{d_1 \ldots d_N}}_{\text{coeff tensor}} \cdot f_{d_1}(\mathbf{x}_1) \cdots f_{d_N}(\mathbf{x}_N)$$

state of many particles



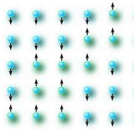
func over many pixels



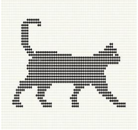
Entanglement with Convolutional Arithmetic Circuits

Structural equivalence:

state of many particles



func over many pixels



quantum system (many-body) state

$$|\text{system state}\rangle = \sum_{d_1 \ldots d_N = 1}^{M} \underbrace{\mathcal{A}_{d_1 \ldots d_N}}_{\text{coeff tensor}} \cdot |\psi_{d_1}\rangle \otimes \cdots \otimes |\psi_{d_N}\rangle$$

func realized by ConvAC

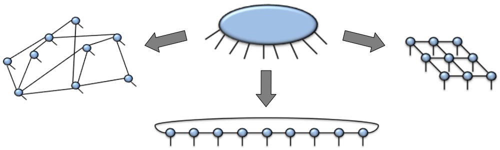
$$h(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{d_1 \ldots d_N = 1}^{M} \underbrace{\mathcal{A}_{d_1 \ldots d_N}}_{\text{coeff tensor}} \cdot f_{d_1}(\mathbf{x}_1) \cdots f_{d_N}(\mathbf{x}_N)$$

We may quantify interactions ConvAC models between input sets by applying entanglement measures to its coeff tensor!

Quantum Tensor Networks

Coeff tensors of quantum many-body states are simulated via:

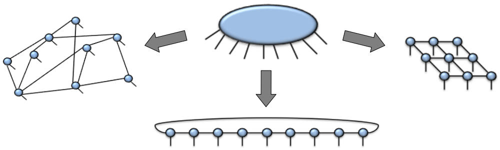
Tensor Networks



Quantum Tensor Networks

Coeff tensors of quantum many-body states are simulated via:

Tensor Networks



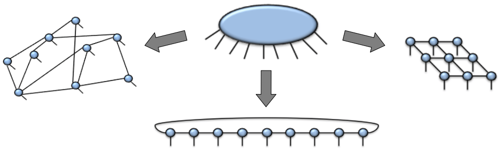
Tensor Networks (TNs):

- Graphs in which: vertices $\longleftrightarrow$ tensors edges $\longleftrightarrow$ modes

Quantum Tensor Networks

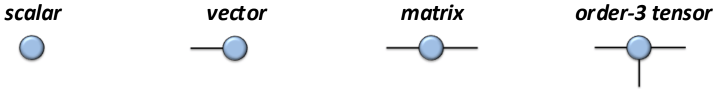
Coeff tensors of quantum many-body states are simulated via:

Tensor Networks

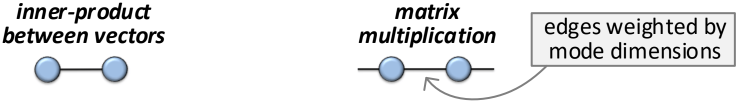


Tensor Networks (TNs):

- Graphs in which: vertices ⟷ tensors edges ⟷ modes

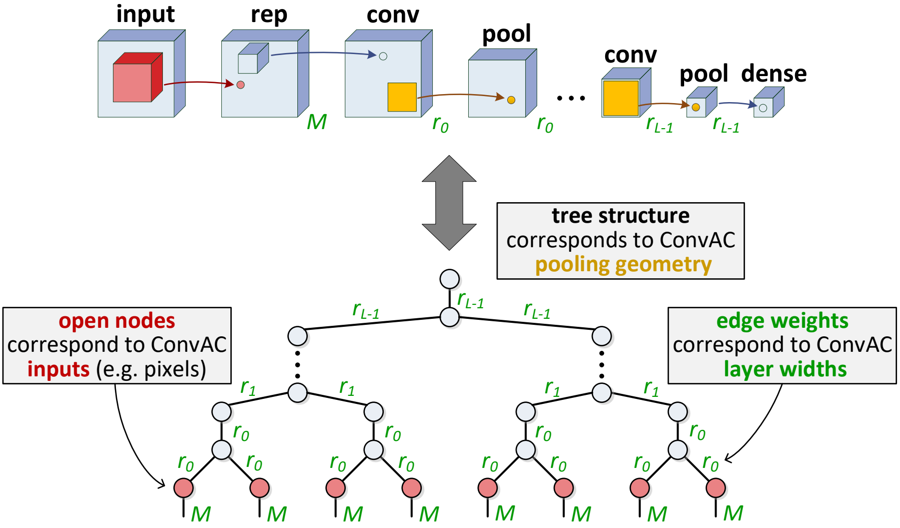


- Edge (mode) connecting two vertices (tensors) represents contraction



Convolutional Arithmetic Circuits as Tensor Networks

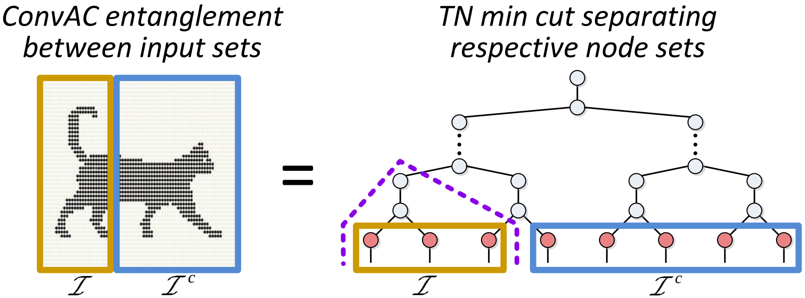
Coeff tensor of ConvAC may be represented via TN:



Entanglement via Minimal Cuts

Theorem

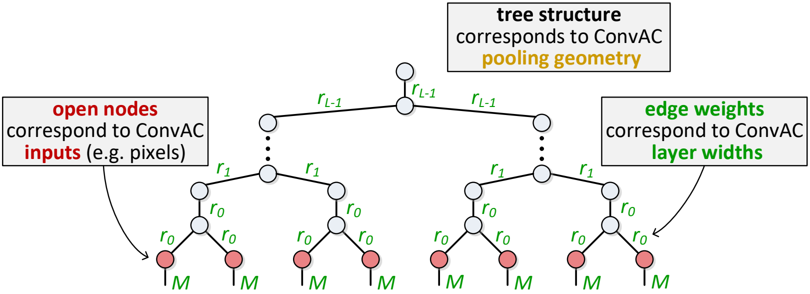
Maximal Schmidt entanglement ConvAC models between input sets $\mathcal{I}/\mathcal{I}^c$ is equal to min cut in respective TN separating nodes of $\mathcal{I}/\mathcal{I}^c$



Controlling Entanglement (Interactions)

Corollary

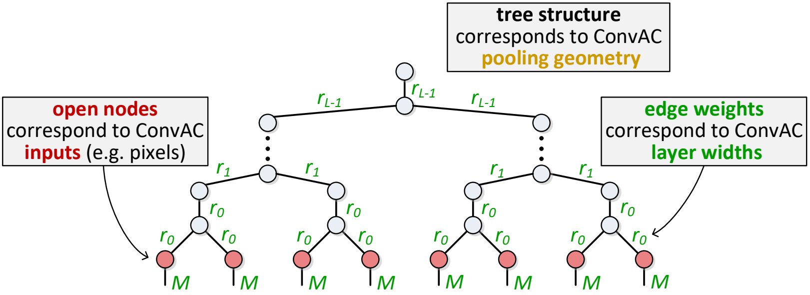
Controlling entanglement (interactions) modeled by ConvAC is equivalent to controlling min cuts in respective TN



Controlling Entanglement (Interactions)

Corollary

Controlling entanglement (interactions) modeled by ConvAC is equivalent to controlling min cuts in respective TN

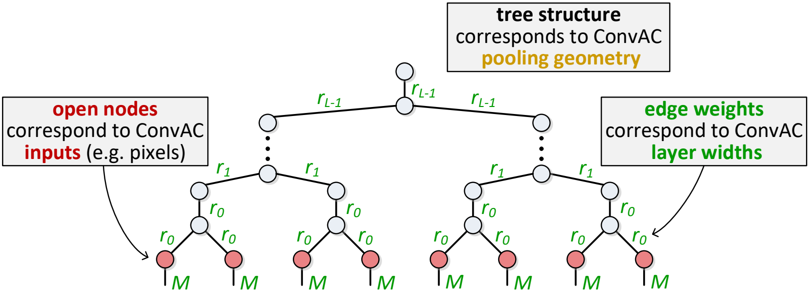


Two sources of control: layer widths, pooling geometry

Controlling Entanglement (Interactions)

Corollary

Controlling entanglement (interactions) modeled by ConvAC is equivalent to controlling min cuts in respective TN



Two sources of control: layer widths, pooling geometry

We may analyze the effect of ConvAC arch on the interactions (entanglement) it can model!

Controlling Interactions – Layer Widths

Claim

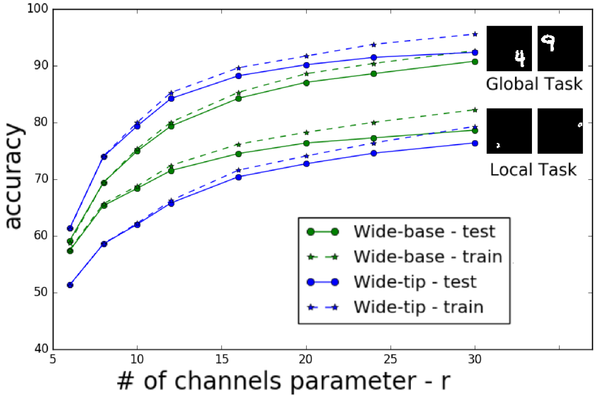
Deep (early) layer widths are important for long (short)-range interactions

Controlling Interactions – Layer Widths

Claim

Deep (early) layer widths are important for long (short)-range interactions

Experiment



Controlling Interactions – Pooling Geometry

Claim

Input elements pooled together early have stronger interaction

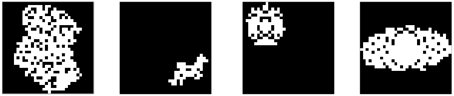
Controlling Interactions – Pooling Geometry

Claim

Input elements pooled together early have stronger interaction

Experiment

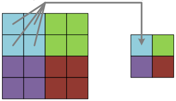
data



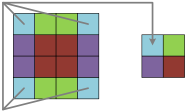
closedness: low	closedness: high	closedness: low	closedness: high
symmetry: low	symmetry: low	symmetry: high	symmetry: high

archs

square pooling
(local interactions)

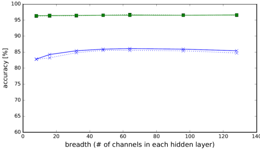


mirror pooling
(interactions between reflections)

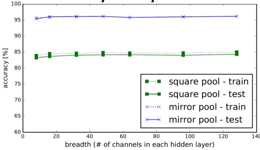


results

closedness task



symmetry task

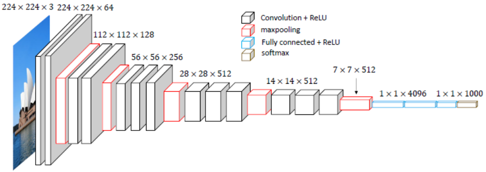


Outline

1. Deep Learning Theory: Expressiveness, Optimization and Generalization
2. Convolutional Networks as Hierarchical Tensor Decompositions
3. Expressiveness of Convolutional Networks
   - Efficiency of Depth *(C|Sharir|Shashua@COLT'16, C|Shashua@ICML'16)*
   - Modeling Interactions *(Levine|Yakira|C|Shashua@ICLR'18, C|Shashua@ICLR'17)*
   - Efficiency of Interconnectivity *(C|Tamari|Shashua@ICLR'18)*
4. Towards Optimization
   - Analysis for Linear Networks *(Arora|C|Hazan@arXiv'18)*
5. Conclusion

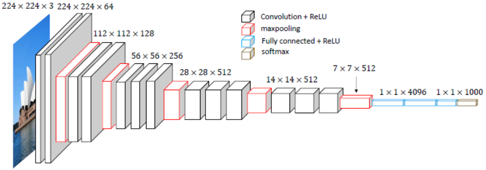
Efficiency of Interconnectivity

Classic ConvNets have feed-forward (chain) structure:

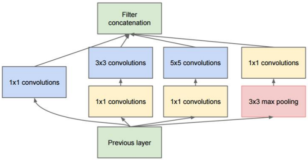


Efficiency of Interconnectivity

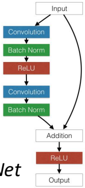
Classic ConvNets have feed-forward (chain) structure:



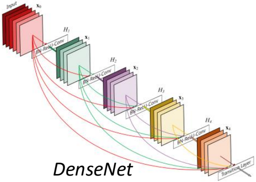
Modern ConvNets employ elaborate connectivity schemes:



Inception (GoogLeNet)



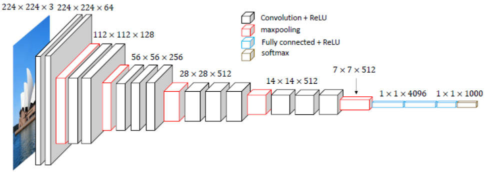
ResNet



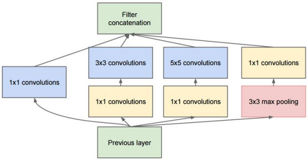
DenseNet

Efficiency of Interconnectivity

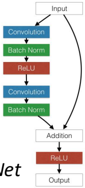
Classic ConvNets have feed-forward (chain) structure:



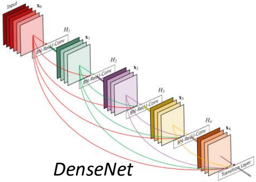
Modern ConvNets employ elaborate connectivity schemes:



Inception (GoogLeNet)



ResNet



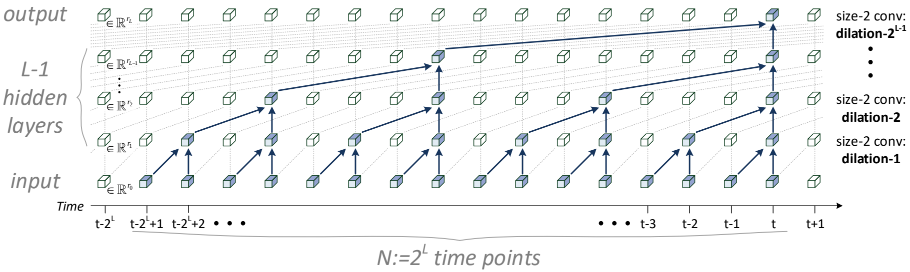
DenseNet

<u>Question</u>

Can such connectivities lead to more efficient representation of func?

Dilated Convolutional Networks

We focus on dilated ConvNets (D-ConvNets) for sequence data:

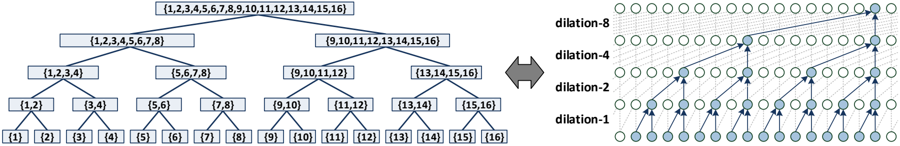


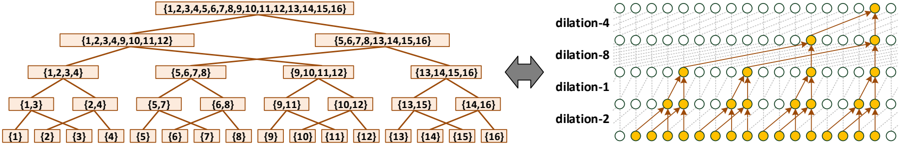
- 1D ConvNets
- No pooling
- Dilated (gapped) conv windows

Underlie Google's WaveNet & ByteNet – state of the art for audio & text!

Dilations and Mode Trees

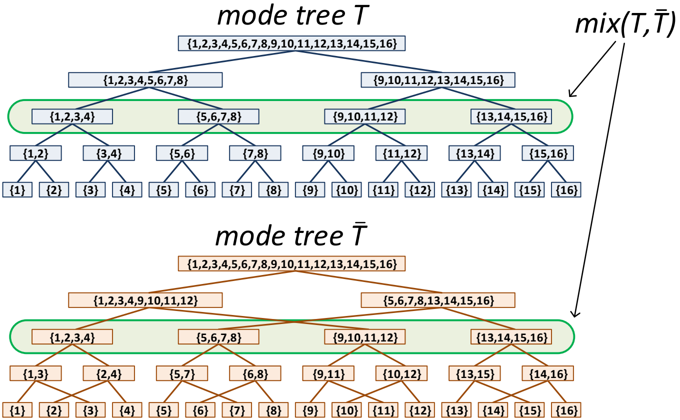
W/D-ConvNet, mode tree underlying corresponding tensor decomposition determines dilation scheme





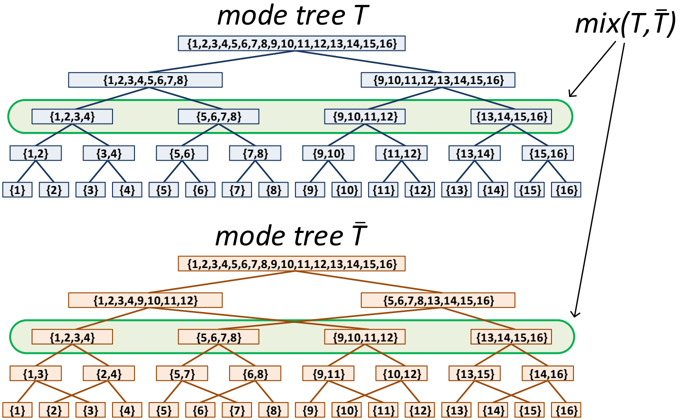
Mixed Tensor Decompositions

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



Mixed Tensor Decompositions

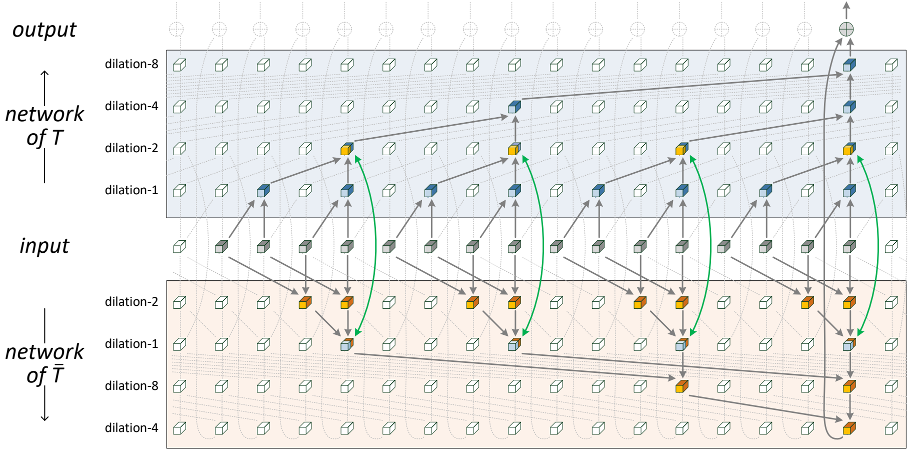
Let: $T, \bar{T}$ – mode trees ; $mix(T, \bar{T})$ – set of nodes present in both trees



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

Mixed Dilated Convolutional Networks

Mixed tensor decomposition corresponds to **mixed D-ConvNet**, formed by interconnecting the networks of T and $\bar{T}$:



Mixture $\longrightarrow$ Expressive Efficiency

Theorem

Mixed tensor decomposition of T and $\bar{T}$ can generate tensors that require individual decompositions to grow quadratically (in terms of their ranks)

Mixture $\longrightarrow$ Expressive Efficiency

Theorem

Mixed tensor decomposition of $\mathcal{T}$ and $\bar{\mathcal{T}}$ can generate tensors that require individual decompositions to grow quadratically (in terms of their ranks)

Corollary

Mixed D-ConvNet can realize func that require individual networks to grow quadratically (in terms of layer widths)

Mixture $\longrightarrow$ Expressive Efficiency

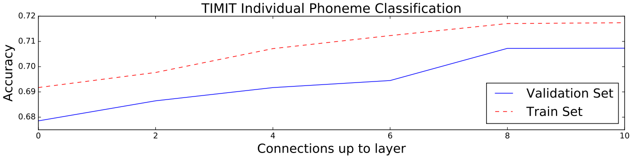
Theorem

Mixed tensor decomposition of $\mathcal{T}$ and $\bar{\mathcal{T}}$ can generate tensors that require individual decompositions to grow quadratically (in terms of their ranks)

Corollary

Mixed D-ConvNet can realize func that require individual networks to grow quadratically (in terms of layer widths)

Experiment



Mixture $\longrightarrow$ Expressive Efficiency

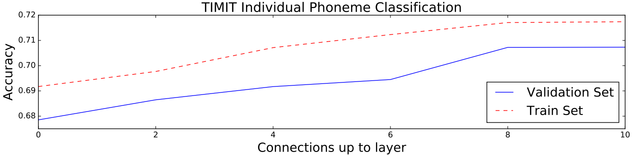
Theorem

Mixed tensor decomposition of T and $\bar{T}$ can generate tensors that require individual decompositions to grow quadratically (in terms of their ranks)

Corollary

Mixed D-ConvNet can realize func that require individual networks to grow quadratically (in terms of layer widths)

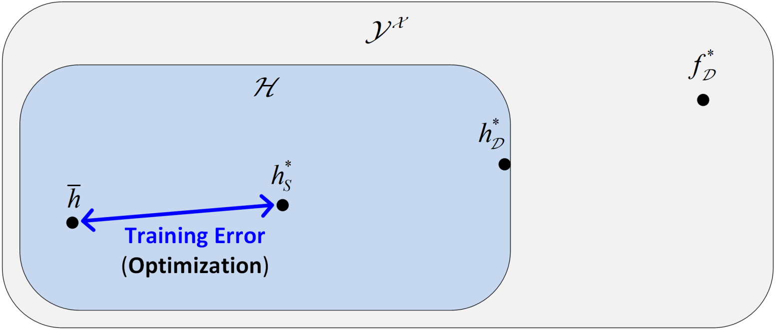
Experiment



Interconnectivity can lead to more efficient representation!

Outline

Optimization



$f^*_{\mathcal{D}}$ – ground truth ($\text{argmin}_{f \in \mathcal{Y}^{\mathcal{X}}} L_{\mathcal{D}}(f)$)

$h^*_{\mathcal{D}}$ – optimal hypothesis ($\text{argmin}_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$)

h^*_S – empirically optimal hypothesis ($\text{argmin}_{h \in \mathcal{H}} L_S(h)$)

$\bar{h}$ – returned hypothesis

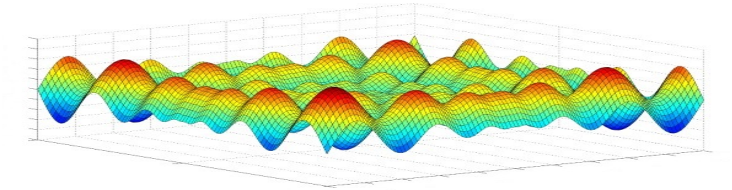
The Effect of Depth – Conventional Wisdom

Depth introduces non-convexity $\implies$ complicates optimization

The Effect of Depth – Conventional Wisdom

Depth introduces non-convexity $\implies$ complicates optimization

However...



Local minima are typically as good as global

$\implies$ gradient-based algorithms reach optimum

Outline

1. Deep Learning Theory: Expressiveness, Optimization and Generalization

2. Convolutional Networks as Hierarchical Tensor Decompositions

3. Expressiveness of Convolutional Networks
   - Efficiency of Depth *(C|Sharir|Shashua@COLT'16, C|Shashua@ICML'16)*
   - Modeling Interactions *(Levine|Yakira|C|Shashua@ICLR'18, C|Shashua@ICLR'17)*
   - Efficiency of Interconnectivity *(C|Tamari|Shashua@ICLR'18)*

4. Towards Optimization
   - Analysis for Linear Networks *(Arora|C|Hazan@arXiv'18)*

5. Conclusion

Linear Networks

Linear network of depth N:

$$\mathbf{x} \mapsto W_N W_{N-1} \cdots W_1 \mathbf{x} \qquad (W_j \text{ -- weight matrices})$$

Linear Networks

Linear network of depth N:

$$\mathbf{x} \mapsto W_N W_{N-1} \cdots W_1 \mathbf{x} \qquad (W_j \text{ – weight matrices})$$

Expressiveness oblivious to depth $\implies$ effect on optimization isolated!

Linear Networks

Linear network of depth N:

$$\mathbf{x} \mapsto W_N W_{N-1} \cdots W_1 \mathbf{x} \qquad (W_j \text{ – weight matrices})$$

Expressiveness oblivious to depth $\implies$ effect on optimization isolated!

Define the **end-to-end weight matrix**:

$$W_e := W_N W_{N-1} \cdots W_1$$

Linear Networks

Linear network of depth N:

$$\mathbf{x} \mapsto W_N W_{N-1} \cdots W_1 \mathbf{x} \qquad (W_j \text{ – weight matrices})$$

Expressiveness oblivious to depth $\Longrightarrow$ effect on optimization isolated!

Define the **end-to-end weight matrix**:

$$W_e := W_N W_{N-1} \cdots W_1$$

Given loss $L(\cdot)$ over linear model, we have **overparameterized loss**:

$$L^N(W_1, \ldots, W_N) := L(W_e)$$

Linear Networks

Linear network of depth N:

$$\mathbf{x} \mapsto W_N W_{N-1} \cdots W_1 \mathbf{x} \qquad (W_j \text{ – weight matrices})$$

Expressiveness oblivious to depth $\implies$ effect on optimization isolated!

Define the **end-to-end weight matrix**:

$$W_e := W_N W_{N-1} \cdots W_1$$

Given loss $L(\cdot)$ over linear model, we have **overparameterized loss**:

$$L^N(W_1, \ldots, W_N) := L(W_e)$$

Question

How does W_e behave during gradient descent over $W_1 \ldots W_N$?

Implicit Dynamics of Gradient Descent

Theorem

*When $W_1 \ldots W_N$ are optimized by gradient descent, W_e follows the **end-to-end update rule**:*

$$W_e^{(t+1)} \hookleftarrow W_e^{(t)} - \eta \sum_{j=1}^{N} \left[W_e^{(t)} (W_e^{(t)})^\top \right]^{\frac{j-1}{N}} \frac{dL}{dW}(W_e^{(t)}) \left[(W_e^{(t)})^\top W_e^{(t)} \right]^{\frac{N-j}{N}}$$

Implicit Dynamics of Gradient Descent

Theorem

When $W_1 \dots W_N$ are optimized by gradient descent, W_e follows the ***end-to-end update rule****:*

$$W_e^{(t+1)} \leftarrow W_e^{(t)} - \eta \sum_{j=1}^{N} \left[W_e^{(t)} (W_e^{(t)})^\top \right]^{\frac{j-1}{N}} \frac{dL}{dW}(W_e^{(t)}) \left[(W_e^{(t)})^\top W_e^{(t)} \right]^{\frac{N-j}{N}}$$

We show:

- End-to-end update rule is a preconditioning scheme, that promotes movement along directions already taken

Implicit Dynamics of Gradient Descent

Theorem

*When $W_1 \ldots W_N$ are optimized by gradient descent, W_e follows the **end-to-end update rule**:*

$$W_e^{(t+1)} \leftarrow W_e^{(t)} - \eta \sum_{j=1}^{N} \left[W_e^{(t)} (W_e^{(t)})^\top \right]^{\frac{j-1}{N}} \frac{dL}{dW}(W_e^{(t)}) \left[(W_e^{(t)})^\top W_e^{(t)} \right]^{\frac{N-j}{N}}$$

We show:

- End-to-end update rule is a preconditioning scheme, that promotes movement along directions already taken
- Can be seen as combination of adaptive learning rate and momentum

Implicit Dynamics of Gradient Descent

Theorem

When $W_1 \ldots W_N$ are optimized by gradient descent, W_e follows the ***end-to-end update rule****:*

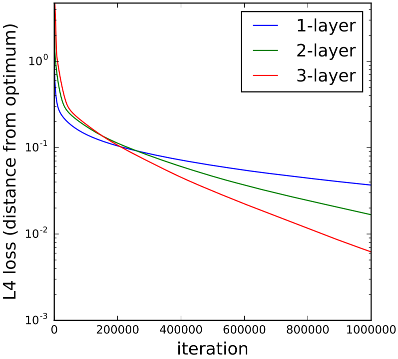
$$W_e^{(t+1)} \hookleftarrow W_e^{(t)} - \eta \sum_{j=1}^{N} \left[W_e^{(t)} (W_e^{(t)})^\top \right]^{\frac{j-1}{N}} \frac{dL}{dW}(W_e^{(t)}) \left[(W_e^{(t)})^\top W_e^{(t)} \right]^{\frac{N-j}{N}}$$

We show:

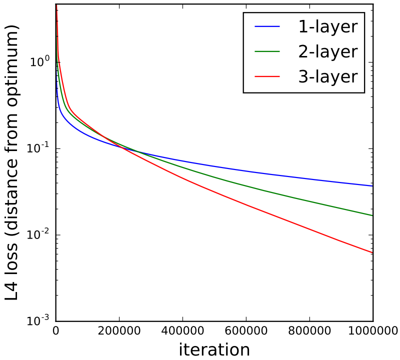
- End-to-end update rule is a preconditioning scheme, that promotes movement along directions already taken
- Can be seen as combination of adaptive learning rate and momentum

W/linear nets depth induces on gradient descent a certain acceleration scheme!

Experiment

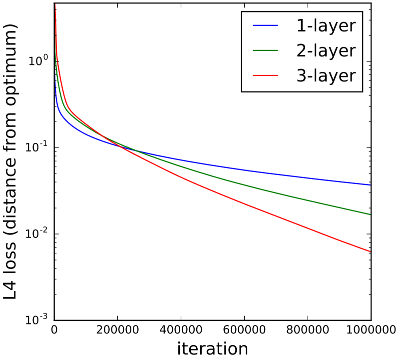

1-layer
2-layer
3-layer
L4 loss (distance from optimum)
10^{0}
10^{-1}
10^{-2}
10^{-3}
0
200000
400000
600000
800000
1000000
iteration

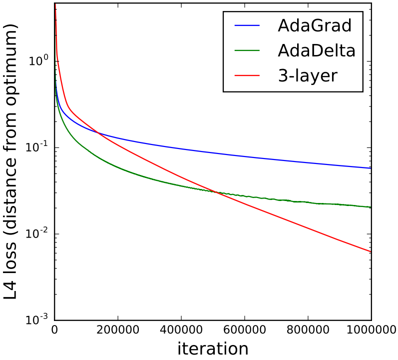
Experiment



Depth can speed-up gradient descent, even w/o any expressiveness gain, and despite introducing non-convexity!

Experiment





Depth can speed-up gradient descent, even w/o any expressiveness gain, and despite introducing non-convexity!

This speed-up can outperform popular acceleration methods designed for convex problems!

Outline

1. Deep Learning Theory: Expressiveness, Optimization and Generalization

2. Convolutional Networks as Hierarchical Tensor Decompositions

3. Expressiveness of Convolutional Networks
   - Efficiency of Depth *(C|Sharir|Shashua@COLT'16, C|Shashua@ICML'16)*
   - Modeling Interactions *(Levine|Yakira|C|Shashua@ICLR'18, C|Shashua@ICLR'17)*
   - Efficiency of Interconnectivity *(C|Tamari|Shashua@ICLR'18)*

4. Towards Optimization
   - Analysis for Linear Networks *(Arora|C|Hazan@arXiv'18)*

5. **Conclusion**

Conclusion

- Three pillars of statistical learning theory:

Expressiveness   Generalization   Optimization

Conclusion

- Three pillars of statistical learning theory:

  Expressiveness Generalization Optimization

  - Well developed theory for classical ML
  - Limited understanding for Deep Learning

Conclusion

- Three pillars of statistical learning theory:

  Expressiveness   Generalization   Optimization

  - Well developed theory for classical ML
  - Limited understanding for Deep Learning

- We derive equivalence:

  ConvNets $\longleftrightarrow$ hierarchical tensor decompositions

Conclusion

- Three pillars of statistical learning theory:

  Expressiveness Generalization Optimization

  - Well developed theory for classical ML
  - Limited understanding for Deep Learning

- We derive equivalence:

  ConvNets $\longleftrightarrow$ hierarchical tensor decompositions

- We use equivalence to **analyze expressiveness of ConvNets**:

Conclusion

- Three pillars of statistical learning theory:

  Expressiveness Generalization Optimization

  - Well developed theory for classical ML
  - Limited understanding for Deep Learning

- We derive equivalence:

  ConvNets $\longleftrightarrow$ hierarchical tensor decompositions

- We use equivalence to **analyze expressiveness of ConvNets**:
  - Representational efficiency of depth

Conclusion

- Three pillars of statistical learning theory:

  Expressiveness   Generalization   Optimization

  - Well developed theory for classical ML
  - Limited understanding for Deep Learning

- We derive equivalence:

  ConvNets $\longleftrightarrow$ hierarchical tensor decompositions

- We use equivalence to **analyze expressiveness of ConvNets**:
  - Representational efficiency of depth
  - Input interaction (entanglement) modeling

Conclusion

- Three pillars of statistical learning theory:

  Expressiveness Generalization Optimization

  - Well developed theory for classical ML
  - Limited understanding for Deep Learning

- We derive equivalence:

  ConvNets $\longleftrightarrow$ hierarchical tensor decompositions

- We use equivalence to **analyze expressiveness of ConvNets**:
  - Representational efficiency of depth
  - Input interaction (entanglement) modeling
  - Efficiency of interconnectivity schemes

Conclusion

- Three pillars of statistical learning theory:

  Expressiveness   Generalization   Optimization

  - Well developed theory for classical ML
  - Limited understanding for Deep Learning

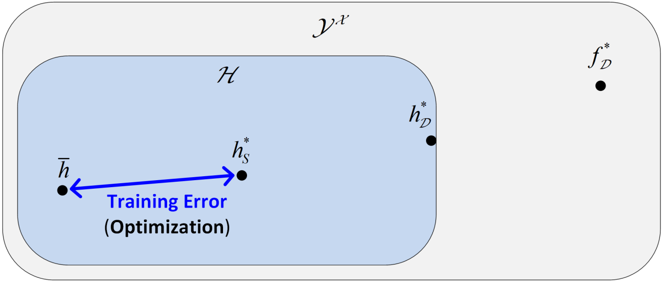
- We derive equivalence:

  ConvNets $\longleftrightarrow$ hierarchical tensor decompositions

- We use equivalence to **analyze expressiveness of ConvNets**:
  - Representational efficiency of depth
  - Input interaction (entanglement) modeling
  - Efficiency of interconnectivity schemes

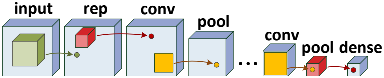
- **Analysis for linear nets shows depth can accelerate optimization**

Ongoing Work – Optimization of Convolutional Networks



+

ConvNets



$\longleftrightarrow$

Tensor Decompositions

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

Outline

Thank You