Deep Neural Networks: A Universal Classification Strategy?

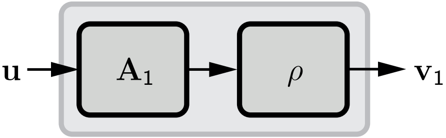
Alex Bronstein

School of Electrical Engineering
Tel Aviv University

February 3, 2016

Based on joint work with Raja Giryes and Guillermo Sapiro

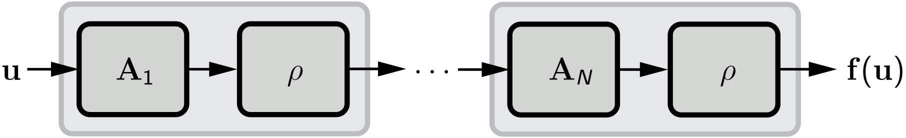
(Deep) neural network



Single layer:

$$\mathbf{v}_1 = \rho(\mathbf{A}_1\mathbf{u})$$

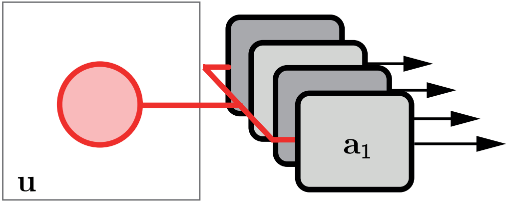
(Deep) neural network



Whole net response:

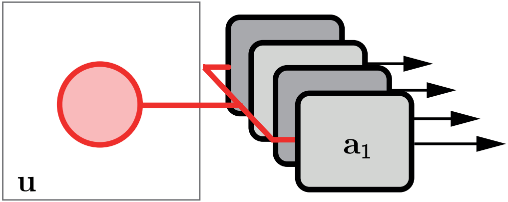
$$\mathbf{f}(\mathbf{u}) = \rho(\mathbf{A}_N\, \rho(\mathbf{A}_{N-1}\, \rho(\cdots \rho(\mathbf{A}_1 \mathbf{u}) \cdots)))$$

Linear part



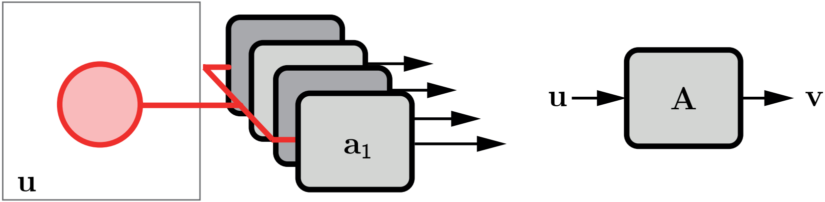
- Convolutional layer: shift-invariant filter bank $\mathbf{v}_i = \mathbf{a}_i * \mathbf{u}_i$

Linear part



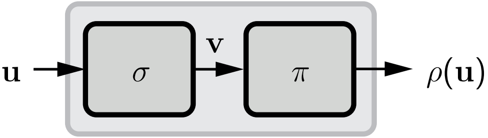
- Convolutional layer: shift-invariant filter bank $\mathbf{v}_i = \mathbf{a}_i * \mathbf{u}_i$
  $\mathbf{A}$ is block-Töplitz

Linear part

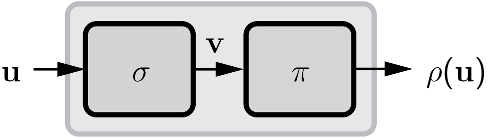


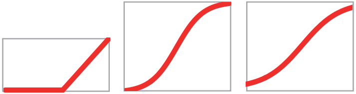
- Convolutional layer: shift-invariant filter bank $\mathbf{v}_i = \mathbf{a}_i * \mathbf{u}_i$
  $\mathbf{A}$ is block-Töplitz
- Fully-connected layer: $\mathbf{v} = \mathbf{A}\mathbf{u}$

Non-linear part



Non-linear part

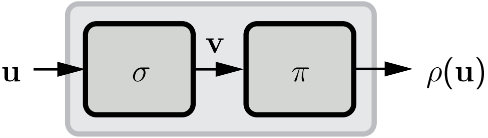




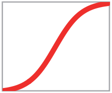
ReLU $\max\{u, 0\}$ | **tanh** $\tanh u$ | **Sigmoid** $\dfrac{1}{1+e^{-u}}$

- Element-wise activation function $\sigma(u)$

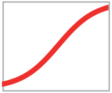
Non-linear part



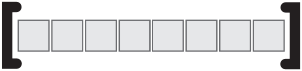
ReLU
$\max\{u, 0\}$



tanh
$\tanh u$



Sigmoid
$$\frac{1}{1+e^{-u}}$$

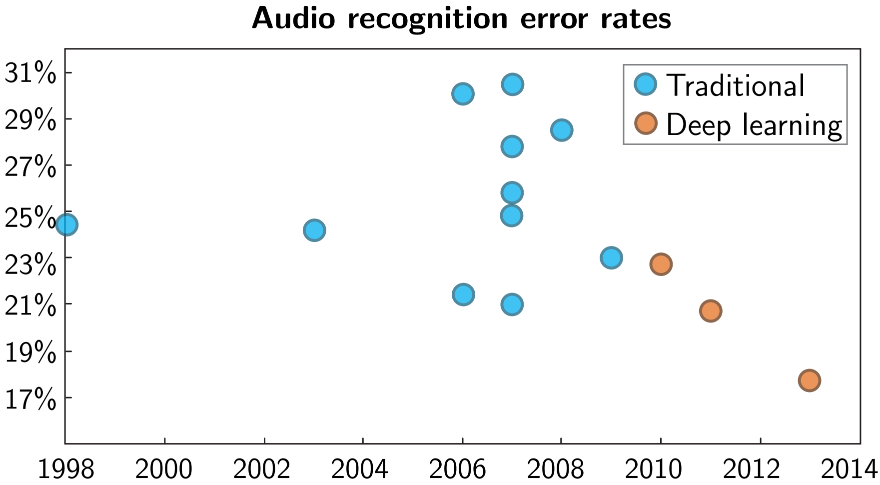


Max
$$\max_i v_i$$

Mean
$$\frac{1}{m}\sum_{i=1} v_i$$

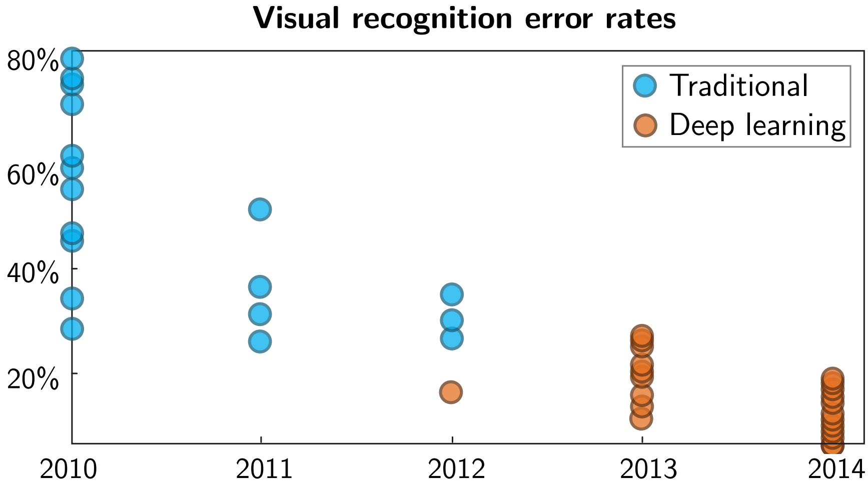
- Element-wise activation function $\sigma(u)$
- Pooling or aggregation operator $\pi(\mathbf{v})$

Impact of deep learning



Source: Clarifi

Impact of deep learning



Source: Clarifi

Why DNNs work?

Why DNNs work?

- Representation power?

Why DNNs work?

- Representation power?
- Role of depth?

Why DNNs work?

- Representation power?
- Role of depth?
- Role of pooling?

Why DNNs work?

- Representation power?
- Role of depth?
- Role of pooling?
- Role of nonlinearity?

Why DNNs work?

- Representation power?
- Role of depth?
- Role of pooling?
- Role of nonlinearity?
- How to train?

Why DNNs work?

- Representation power?
- Role of depth?
- Role of pooling?
- Role of nonlinearity?
- How to train?
- How much training data are needed?

Representation power

- DNNs are universal approximators of any Borel function[1]

[1]Cybenko 1989; Hornik 1991; [2]Barron 1992

Representation power

- DNNs are universal approximators of any Borel function[1]
- Estimation error of a function $\mathbf{f}$ by DNN is[2]

$$\mathcal{O}\left(\frac{C_f}{K}\right) + \mathcal{O}\left(\frac{nK}{T}\log T\right)$$

$C_f =$ smoothness of f
$K = \#$ of degrees of freedom
$n =$ input dimension
$T = \#$ of training samples

[1]Cybenko 1989; Hornik 1991; [2]Barron 1992

Representation power

- DNNs represent restricted Boltzmann machines with number of parameters exponentially greater than the number of degrees of freedom of the network[1]

[1]Montúfar & Morton, 2014; [2]Montúfar *et al.*, 2014

Representation power

- DNNs represent restricted Boltzmann machines with number of parameters exponentially greater than the number of degrees of freedom of the network[1]

- Deep network with the same number of degrees of freedom divides the input space into exponentially greater number of sets[2]

[1] Montúfar & Morton, 2014; [2] Montúfar *et al.*, 2014

Representation power

- DNNs represent restricted Boltzmann machines with number of parameters exponentially greater than the number of degrees of freedom of the network[1]
- Deep network with the same number of degrees of freedom divides the input space into exponentially greater number of sets[2]
- **Depth is important!**

[1]Montúfar & Morton, 2014; [2]Montúfar *et al.*, 2014

Role of pooling

- Pooling provides shift invariance[1]

[1]Bruna, LeCun & Szlam, 2013,2014; [2]Bruna & Mallat, 2013

Role of pooling

- Pooling provides shift invariance[1]
- Scattering networks[2] : a cascade of wavelet transform, modulus and averaging

[1]Bruna, LeCun & Szlam, 2013,2014; [2]Bruna & Mallat, 2013

Role of pooling

- Pooling provides shift invariance[1]
- Scattering networks[2] : a cascade of wavelet transform, modulus and averaging
- Deeper network provides invariance to more complex transformations

[1]Bruna, LeCun & Szlam, 2013,2014; [2]Bruna & Mallat, 2013

Role of pooling

- Pooling provides shift invariance[1]
- Scattering networks[2] : a cascade of wavelet transform, modulus and averaging
- Deeper network provides invariance to more complex transformations
- **Depth is important!**

[1]Bruna, LeCun & Szlam, 2013,2014; [2]Bruna & Mallat, 2013

Training

- Supervised training by back propagation (chain rule) results in non-convex optimization problem

Saxe, McClelland & Ganguli, 2014; Dauphin *et al.*, 2014; Choromanska *et al.*, 2015; Haeffele & Vidal, 2015

Training

- Supervised training by back propagation (chain rule) results in non-convex optimization problem
- In deep networks, local minima are nearly as good as global ones

Saxe, McClelland & Ganguli, 2014; Dauphin *et al.*, 2014; Choromanska *et al.*, 2015; Haeffele & Vidal, 2015

Training

- Supervised training by back propagation (chain rule) results in non-convex optimization problem

- In deep networks, local minima are nearly as good as global ones

- Deep networks have less saddle points

Saxe, McClelland & Ganguli, 2014; Dauphin *et al.*, 2014; Choromanska *et al.*, 2015; Haeffele & Vidal, 2015

Training

- Supervised training by back propagation (chain rule) results in non-convex optimization problem
- In deep networks, local minima are nearly as good as global ones
- Deep networks have less saddle points
- Random initialization works well

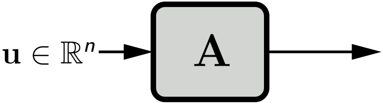
Saxe, McClelland & Ganguli, 2014; Dauphin *et al.*, 2014; Choromanska *et al.*, 2015; Haeffele & Vidal, 2015

Training

- Supervised training by back propagation (chain rule) results in non-convex optimization problem
- In deep networks, local minima are nearly as good as global ones
- Deep networks have less saddle points
- Random initialization works well
- Extreme learning strategies rely only on randomization

Saxe, McClelland & Ganguli, 2014; Dauphin *et al.*, 2014; Choromanska *et al.*, 2015; Haeffele & Vidal, 2015

Assumptions



$\mathbf{A} \in \mathbb{R}^{m \times n}$

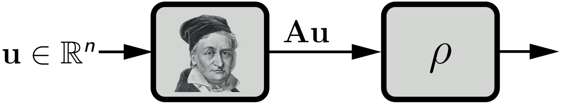
- Fully-connected linear layers

Assumptions

$$a_{ij} \sim \mathcal{N}(0, \tfrac{1}{m})$$

- Fully-connected linear layers with random Gaussian weights

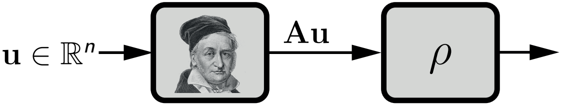
Assumptions



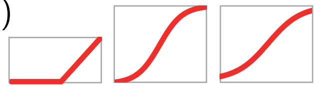
$a_{ij} \sim \mathcal{N}(0, \frac{1}{m})$

- Fully-connected linear layers with random Gaussian weights
- Element-wise (approximately) truncated linear activation

Assumptions



$a_{ij} \sim \mathcal{N}(0, \frac{1}{m})$

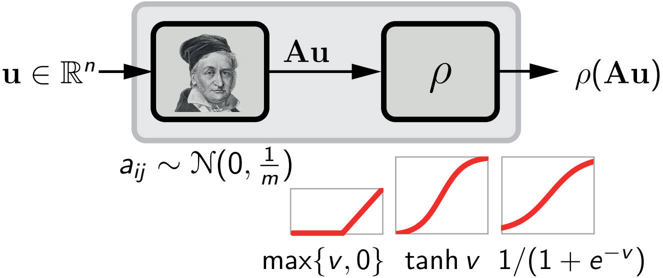


$\max\{v, 0\}$ $\tanh v$ $1/(1+e^{-v})$

- Fully-connected linear layers with random Gaussian weights
- Element-wise (approximately) truncated linear activation

$$\rho|_{[a,b]} \text{ linear} \qquad \rho|_{\mathbb{R}\setminus[a,b]} = \text{const}$$

Assumptions

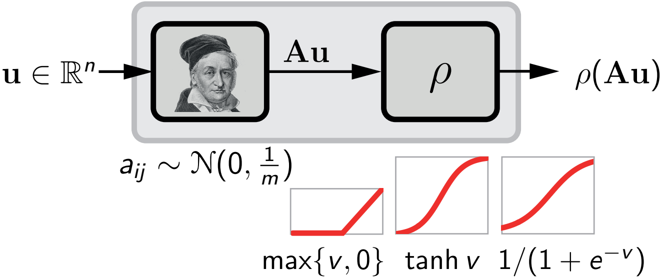


- Fully-connected linear layers with random Gaussian weights
- Element-wise (approximately) truncated linear activation

$$\rho|_{[a,b]} \text{ linear} \qquad \rho|_{\mathbb{R}\setminus[a,b]} = \text{const}$$

- No pooling

Assumptions

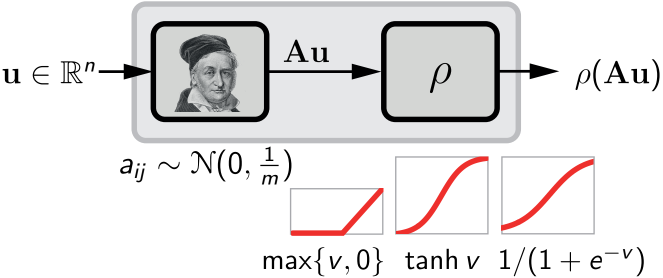


- Fully-connected linear layers with random Gaussian weights
- Element-wise (approximately) truncated linear activation

$$\rho|_{[a,b]} \text{ linear} \qquad \rho|_{\mathbb{R}\setminus[a,b]} = \mathrm{const}$$

- No pooling (pooling = invariance)

Assumptions

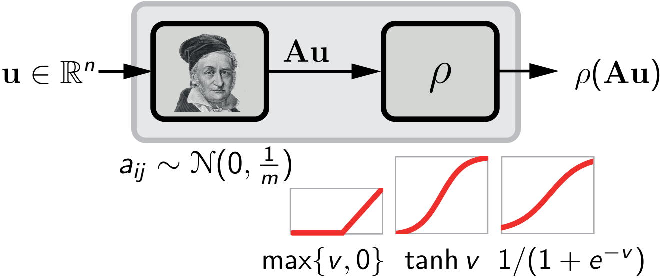


- Fully-connected linear layers with random Gaussian weights
- Element-wise (approximately) truncated linear activation

$$\rho|_{[a,b]} \text{ linear} \qquad \rho|_{\mathbb{R}\setminus[a,b]} = \text{const}$$

- No pooling: input already invariant

Assumptions



- Fully-connected linear layers with random Gaussian weights
- Element-wise (approximately) truncated linear activation

$$\rho|_{[a,b]} \text{ linear} \qquad \rho|_{\mathbb{R}\setminus[a,b]} = \mathrm{const}$$

- No pooling: input already invariant
- Low dimensional input data

Random Gaussian weights

5406 IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 52, NO. 12, DECEMBER 2006

Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies?

Emmanuel J. Candes and Terence Tao

Candés, Romberg, Tao, 2006; Candés & Tao, 2006

Random Gaussian weights

5406 IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 52, NO. 12, DECEMBER 2006

Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies?

Emmanuel J. Candes and Terence Tao

$$\mathbf{v} = \Bigg[$$  $$\Bigg]\mathbf{u}$$

$$a_{ij} \sim \mathcal{N}(0, \tfrac{1}{m})$$

Candés, Romberg, Tao, 2006; Candés & Tao, 2006

Random Gaussian weights

$\mathbf{v} = \Bigg[$  $\Bigg]\mathbf{u}$

$a_{ij} \sim \mathcal{N}(0, \frac{1}{m})$

A k**-sparse vector** $\mathrm{u} \in \mathbb{R}^n$ **can be reconstructed from** $m = \mathcal{O}(k \log(n/k))$ **random projections**

Candés, Romberg, Tao, 2006; Candés & Tao, 2006

Random Gaussian weights

$\mathbf{v} = \Bigg[$  $\Bigg]\mathbf{u}$

$a_{ij} \sim \mathcal{N}(0, \frac{1}{m})$

A k-sparse vector $\mathrm{u} \in \mathbb{R}^n$ can be reconstructed from $m = \mathcal{O}(k \log(n/k))$ random projections

Restricted isometry property (RIP)

Candés, Romberg, Tao, 2006; Candés & Tao, 2006

Random Gaussian weights

$$\mathbf{v} = \Big[\quad\Big]\mathbf{u}$$

$$a_{ij} \sim \mathcal{N}(0, \tfrac{1}{m})$$

A k-sparse vector $\mathrm{u} \in \mathbb{R}^n$ can be reconstructed from $m = \mathcal{O}(k \log(n/k))$ random projections

Restricted isometry property (RIP)

Random projection is universally good

Candés, Romberg, Tao, 2006; Candés & Tao, 2006

Low-dimensional input

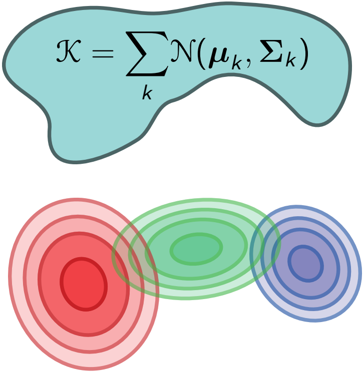
Input data have a small number of degrees of freedom

Low-dimensional input

Input data have a small number of degrees of freedom but may be embedded in a high-dimensional space

Low-dimensional input

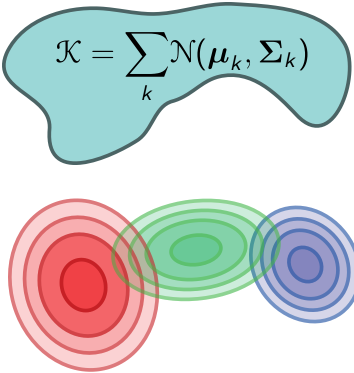
Input data have a small number of degrees of freedom but may be embedded in a high-dimensional space



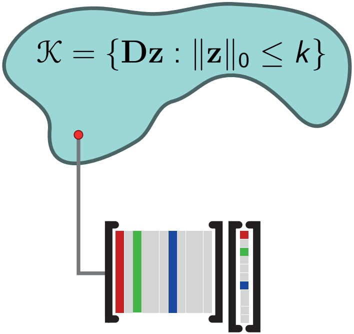
Gaussian mixture model

Low-dimensional input

Input data have a small number of degrees of freedom but may be embedded in a high-dimensional space



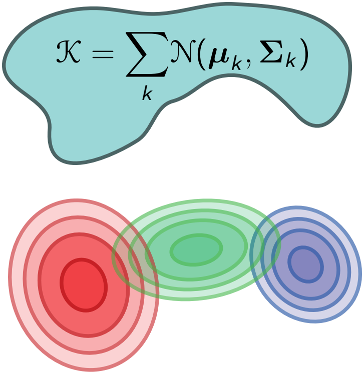
Gaussian mixture model



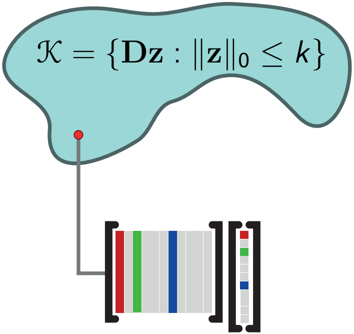
Sparse representation

Low-dimensional input

Violated at the output due to DNN nonlinearity!

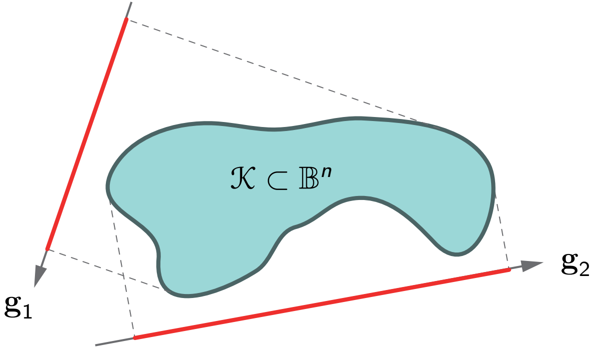


Gaussian mixture model



Sparse representation

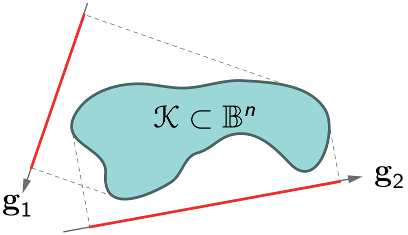
Gaussian mean width



$$\omega(\mathcal{K}) = \mathbb{E} \sup_{\mathbf{u},\mathbf{v} \in \mathcal{K}} \langle \mathbf{u} - \mathbf{v}, \mathbf{g} \rangle \qquad \mathbf{g} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Plan & Vershynin, 2012

Gaussian mean width

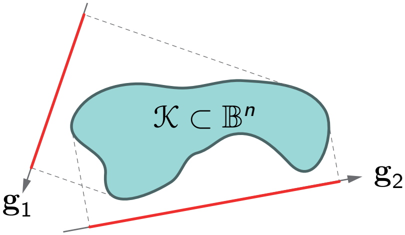


$$\omega(\mathcal{K}) = \mathbb{E} \sup_{\mathbf{u},\mathbf{v}\in\mathcal{K}} \langle \mathbf{u}-\mathbf{v}, \mathbf{g}\rangle \qquad \mathbf{g} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

- $\omega^2(\mathcal{K})$ measures intrinsic data dimension

Plan & Vershynin, 2012

Gaussian mean width

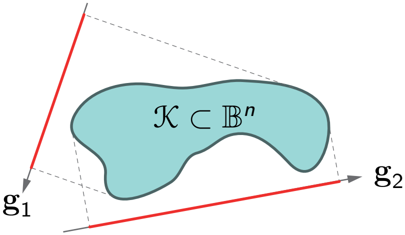


$$\omega(\mathcal{K}) = \mathbb{E} \sup_{\mathbf{u},\mathbf{v}\in\mathcal{K}} \langle \mathbf{u} - \mathbf{v}, \mathbf{g} \rangle \qquad \mathbf{g} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

- $\omega^2(\mathcal{K})$ measures intrinsic data dimension
- $\mathcal{K}$ is GMM with k Gaussians: $\omega^2(\mathcal{K}) = \mathcal{O}(k)$

Plan & Vershynin, 2012

Gaussian mean width

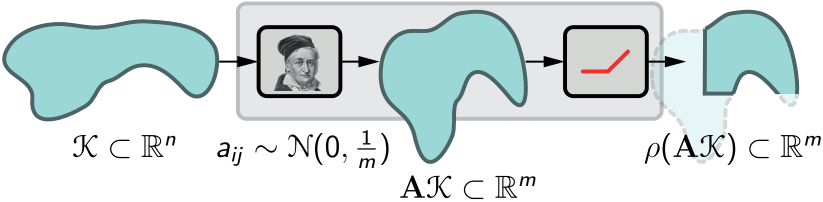


$$\omega(\mathcal{K}) = \mathbb{E} \sup_{\mathbf{u},\mathbf{v}\in\mathcal{K}} \langle \mathbf{u} - \mathbf{v}, \mathbf{g} \rangle \qquad \mathbf{g} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

- $\omega^2(\mathcal{K})$ measures intrinsic data dimension
- $\mathcal{K}$ is GMM with k Gaussians: $\omega^2(\mathcal{K}) = \mathcal{O}(k)$
- $\mathcal{K}$ is k-sparsely representable: $\omega^2(\mathcal{K}) = \mathcal{O}(k \log(n/k))$

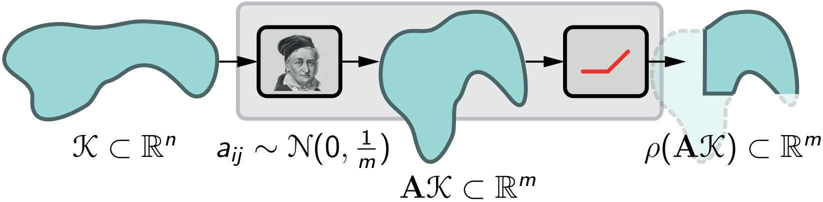
Plan & Vershynin, 2012

Low dimension in – low dimension out



Giryes, Sapiro, B, 2015

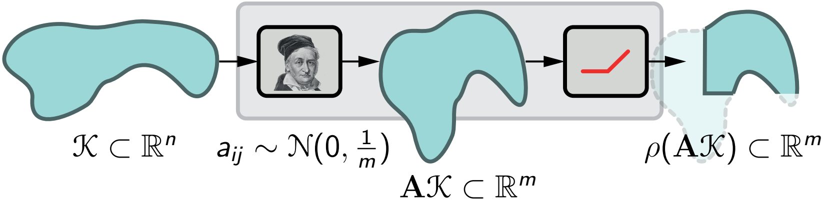
Low dimension in – low dimension out



Theorem: if $\omega^2(\mathcal{K}) \ll m$ **then** $\omega^2(\rho(\mathbf{A}\mathcal{K})) \approx \omega^2(\mathcal{K})$

Giryes, Sapiro, B, 2015

Low dimension in – low dimension out

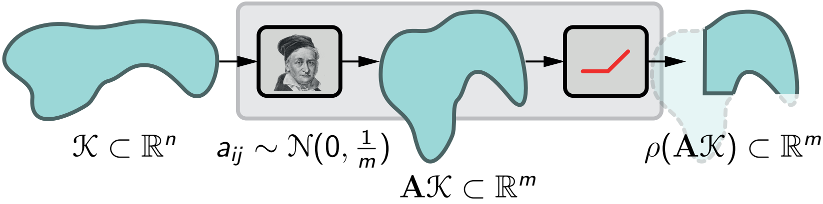


Theorem: **if** $\omega^2(\mathcal{K}) \ll m$ **then** $\omega^2(\rho(\mathbf{A}\mathcal{K})) \approx \omega^2(\mathcal{K})$

Proof: covering argument

Giryes, Sapiro, B, 2015

Low dimension in – low dimension out



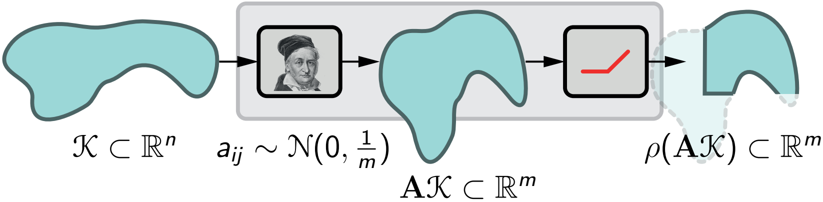
Theorem: if $\omega^2(\mathcal{K}) \ll m$ **then** $\omega^2(\rho(\mathbf{A}\mathcal{K})) \approx \omega^2(\mathcal{K})$

Proof: covering argument

- Intrinsic data dimension does not grow significantly through the network

Giryes, Sapiro, B, 2015

Low dimension in – low dimension out



Theorem: if $\omega^2(\mathcal{K}) \ll m$ **then** $\omega^2(\rho(\mathbf{A}\mathcal{K})) \approx \omega^2(\mathcal{K})$

Proof: covering argument

- Intrinsic data dimension does not grow significantly through the network
- It is sufficient to analyze a single layer in DNN

Giryes, Sapiro, B, 2015

Stable embedding

Theorem: the map $\mathbf{h} : (\mathcal{K} \subset \mathbb{S}^{n-1}, d_{\mathbb{S}^{n-1}}) \mapsto (\mathbf{h}(\mathcal{K}), d_{\mathbb{H}^m})$ defined by $\mathbf{h}(\mathbf{u}) = \mathrm{sign}(\rho(\mathbf{A}\mathbf{u}))$ is a δ-isometry with $\delta = c\, m^{-1/6}\, \omega^{1/3}(\mathcal{K})$

Plan & Vershynin, 2013; Giryes, Sapiro, B, 2015

Stable embedding

Theorem: the map $\mathbf{h} : (\mathcal{K} \subset \mathbb{S}^{n-1}, d_{\mathbb{S}^{n-1}}) \mapsto (\mathbf{h}(\mathcal{K}), d_{\mathbb{H}^m})$ **defined by** $\mathrm{h}(\mathbf{u}) = \mathrm{sign}(\rho(\mathbf{Au}))$ **is a** δ**-isometry, i.e.,**

$$|d_{\mathbb{S}^{n-1}}(\mathbf{u}, \mathbf{v}) - d_{\mathbb{H}^m}(\mathbf{h}(\mathbf{u}), \mathbf{h}(\mathbf{v}))| \leq \delta \qquad \forall \mathbf{u}, \mathbf{v} \in \mathcal{K}$$

and every $\mathbf{w} \in \mathrm{h}(\mathcal{K})$ **has some** $\mathbf{u} \in \mathcal{K}$ **such that** $d_{\mathbb{H}^m}(\mathbf{h}(\mathbf{u}, \mathbf{w}) \leq \delta$ **with** $\delta = c\, m^{-1/6}\, \omega^{1/3}(\mathcal{K})$

Plan & Vershynin, 2013; Giryes, Sapiro, B, 2015

Stable embedding

Theorem: the map $\mathbf{h} : (\mathcal{K} \subset \mathbb{S}^{n-1}, d_{\mathbb{S}^{n-1}}) \mapsto (\mathbf{h}(\mathcal{K}), d_{\mathbb{H}^m})$ **defined by** $\mathrm{h}(\mathbf{u}) = \mathrm{sign}(\rho(\mathbf{Au}))$ **is a** δ**-isometry, i.e.,**

$$|d_{\mathbb{S}^{n-1}}(\mathbf{u}, \mathbf{v}) - d_{\mathbb{H}^m}(\mathbf{h}(\mathbf{u}), \mathbf{h}(\mathbf{v}))| \leq \delta \qquad \forall \mathbf{u}, \mathbf{v} \in \mathcal{K}$$

and every $\mathbf{w} \in \mathrm{h}(\mathcal{K})$ **has some** $\mathbf{u} \in \mathcal{K}$ **such that** $d_{\mathbb{H}^m}(\mathbf{h}(\mathbf{u}, \mathbf{w}) \leq \delta$ **with** $\delta = c\, m^{-1/6}\, \omega^{1/3}(\mathcal{K})$

Proof: follows Plan & Vershynin, 2013

Plan & Vershynin, 2013; Giryes, Sapiro, B, 2015

Stable embedding

Theorem: **the map** $\mathbf{h} : (\mathcal{K} \subset \mathbb{S}^{n-1}, d_{\mathbb{S}^{n-1}}) \mapsto (\mathbf{h}(\mathcal{K}), d_{\mathbb{H}^m})$ **defined by** $\mathbf{h}(\mathbf{u}) = \mathrm{sign}(\rho(\mathbf{A}\mathbf{u}))$ **is a** δ**-isometry, i.e.,**

$$|d_{\mathbb{S}^{n-1}}(\mathbf{u}, \mathbf{v}) - d_{\mathbb{H}^m}(\mathbf{h}(\mathbf{u}), \mathbf{h}(\mathbf{v}))| \leq \delta \qquad \forall \mathbf{u}, \mathbf{v} \in \mathcal{K}$$

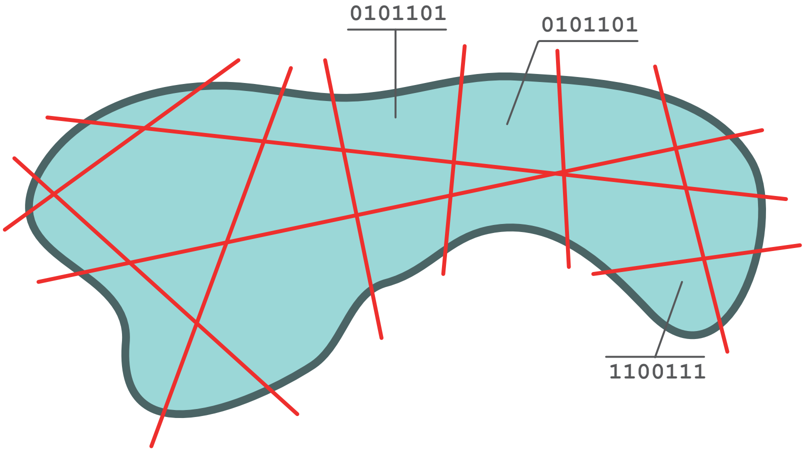
and every $\mathbf{w} \in \mathbf{h}(\mathcal{K})$ **has some** $\mathbf{u} \in \mathcal{K}$ **such that** $d_{\mathbb{H}^m}(\mathbf{h}(\mathbf{u}, \mathbf{w}) \leq \delta$ **with** $\delta = c\, m^{-1/6}\, \omega^{1/3}(\mathcal{K})$

Proof: follows Plan & Vershynin, 2013

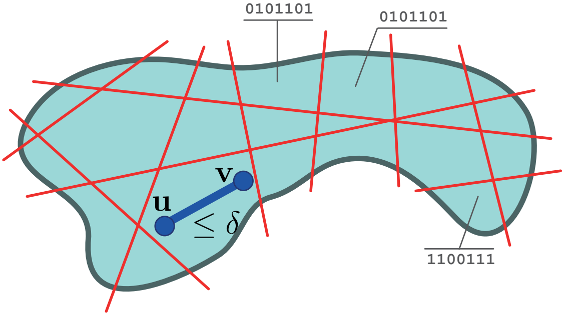
DNN layer performs stable embedding in the Gromov-Hausdorff sense

Plan & Vershynin, 2013; Giryes, Sapiro, B, 2015

Tessellation of the input space



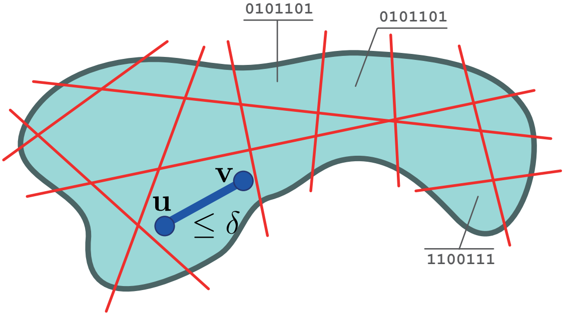
Tessellation of the input space



- Cell diameter $\leq \delta$

Giryes, Sapiro, B, 2015

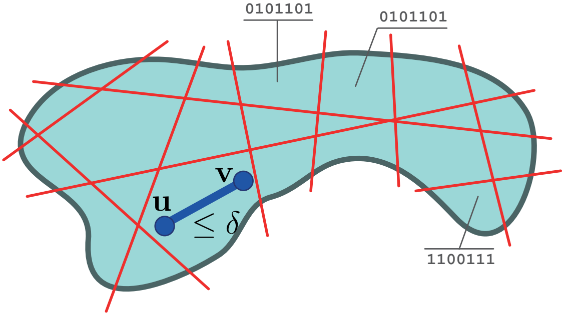
Tessellation of the input space



- Cell diameter $\leq \delta$
- If $\mathbf{h}(\mathbf{u}) = \mathbf{h}(\mathbf{v})$ then $d_{\mathbb{S}^{n-1}}(\mathbf{u}, \mathbf{v}) \leq \delta$

Giryes, Sapiro, B, 2015

Tessellation of the input space



- Cell diameter $\leq \delta$
- If $\mathbf{h}(\mathbf{u}) = \mathbf{h}(\mathbf{v})$ then $d_{\mathbb{S}^{n-1}}(\mathbf{u}, \mathbf{v}) \leq \delta$
- Input metric can be recovered up to a small distortion

Giryes, Sapiro, B, 2015

Stable embedding

Theorem: **there exists a procedure $\mathcal{P}$ such that**

$$\| \mathcal{K} - \mathcal{P}(\rho(\mathbf{A}\mathcal{K})) \| < \mathcal{O}\left(\frac{\omega(\mathcal{K})}{\sqrt{m}}\right) = \mathcal{O}(\delta^3)$$

Plan & Vershynin, 2013; Giryes, Sapiro, B, 2015

Stable embedding

Theorem: there exists a procedure $\mathcal{P}$ such that

$$\| \mathcal{K} - \mathcal{P}(\rho(\mathbf{A}\mathcal{K})) \| < \mathcal{O}\left(\frac{\omega(\mathcal{K})}{\sqrt{m}} \right) = \mathcal{O}(\delta^3)$$

Proof: follows Plan & Vershynin, 2013

Plan & Vershynin, 2013; Giryes, Sapiro, B, 2015

Stable embedding

Theorem: there exists a procedure $\mathcal{P}$ such that

$$\| \mathcal{K} - \mathcal{P}(\rho(\mathbf{A}\mathcal{K})) \| < \mathcal{O}\left(\frac{\omega(\mathcal{K})}{\sqrt{m}}\right) = \mathcal{O}(\delta^3)$$

Proof: follows Plan & Vershynin, 2013

- After N layers the error grows as $\mathcal{O}(N\delta^3)$

Plan & Vershynin, 2013; Giryes, Sapiro, B, 2015

Stable embedding

Theorem: there exists a procedure $\mathcal{P}$ such that

$$\| \mathcal{K} - \mathcal{P}(\rho(\mathbf{A}\mathcal{K})) \| < \mathcal{O}\left(\frac{\omega(\mathcal{K})}{\sqrt{m}}\right) = \mathcal{O}(\delta^3)$$

Proof: follows Plan & Vershynin, 2013

- After N layers the error grows as $\mathcal{O}(N\delta^3)$
- DNNs keep important information of the data

Plan & Vershynin, 2013; Giryes, Sapiro, B, 2015

Stable embedding

Theorem: there exists a procedure $\mathcal{P}$ such that

$$\| \mathcal{K} - \mathcal{P}(\rho(\mathbf{A}\mathcal{K})) \| < \mathcal{O}\left(\frac{\omega(\mathcal{K})}{\sqrt{m}}\right) = \mathcal{O}(\delta^3)$$

Proof: follows Plan & Vershynin, 2013

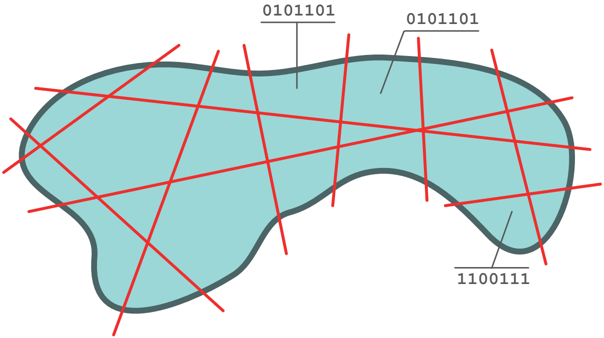
- After N layers the error grows as $\mathcal{O}(N\delta^3)$
- DNNs keep important information of the data
- Input can be recovered from output if output dimension m is big enough

Plan & Vershynin, 2013; Giryes, Sapiro, B, 2015

Inverting a CNN

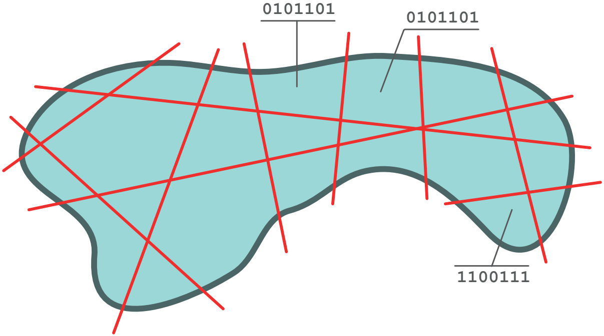
Mahendran & Vedaldi, 2015

Similarity-preserving hashing



Giryes, Sapiro, B, 2015

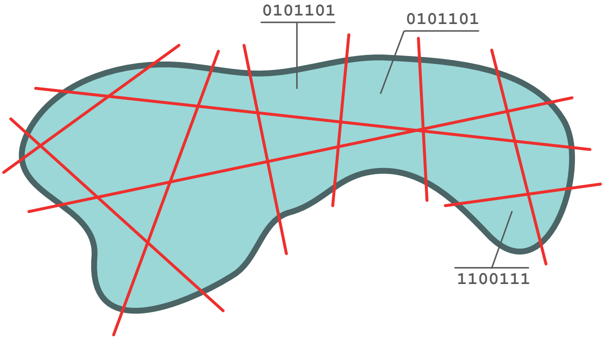
Similarity-preserving hashing



- Single layer = **locality sensitive hashing** **(LSH)**

Giryes, Sapiro, B, 2015

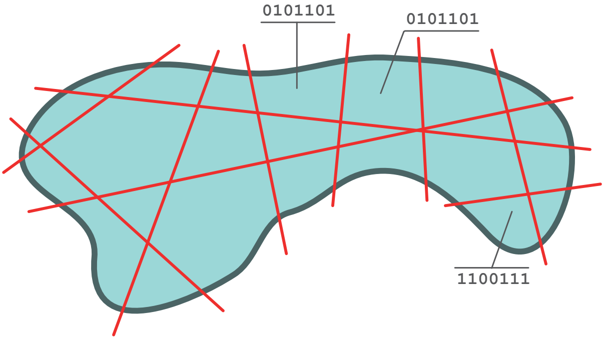
Similarity-preserving hashing



- Single layer = **locality sensitive hashing (LSH)**
- Random weights perform well universally

Giryes, Sapiro, B, 2015

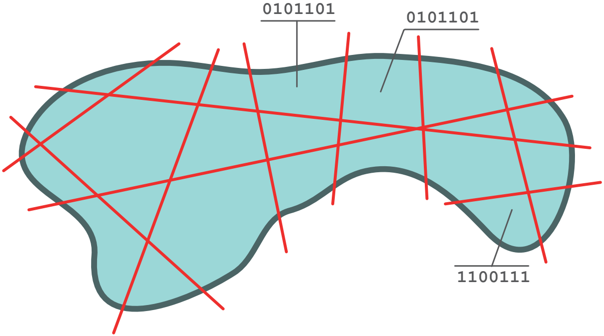
Similarity-preserving hashing



- Single layer = **locality sensitive hashing** **(LSH)**
- Random weights perform well universally
- Can be tuned to specific data by training

Giryes, Sapiro, B, 2015

Similarity-preserving hashing



For deep networks, number of cells in the tessellation is exponentially greater than the number of degrees of freedom

Montúfar *et al.*, 2014

Image retrieval with similarity-preserving hashing

60K images from 10 different classes taken from Tiny images
Represented using 384-dimensional GIST descriptor
Training: 200 images per class; Testing: 59K images

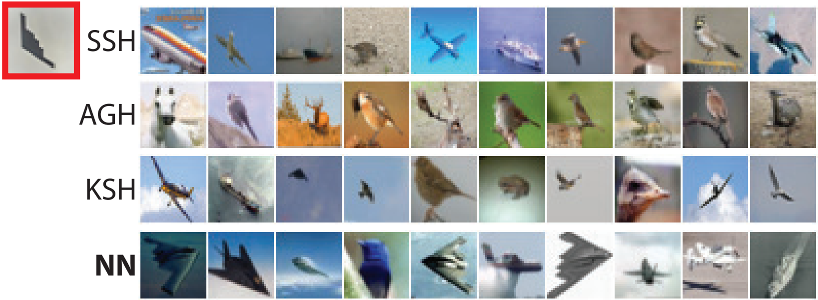
Method / *m*		**12**	**24**	**48**
	Raw		19.16	
	DiffHash	14.72	13.35	12.85
	SSH	15.42	16.75	17.06
	AGH	15.46	15.29	15.15
	KSH	25.79	29.01	30.84
NN	**1 layer**	31.48	35.41	36.79
	2 layer	**45.42**	**49.88**	**50.46**

Performance (mAP in %)

Data: Torralba et al. 2008, Krizhevsky 2009; Methods: Strecha et al. 2011 (diff-hash); Shakhnarovich 2005 (SSH); Liu et al. 2011 (AGH); Liu et al. 2012 (KSH); Masci, B^2, Schmidhuber 2012 (NN)

Image retrieval with similarity-preserving hashing

60K images from 10 different classes taken from Tiny images
Represented using 384-dimensional GIST descriptor
Training: 200 images per class; Testing: 59K images

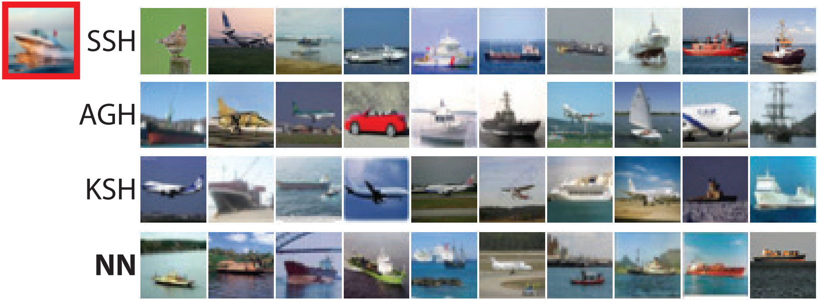


Ranking using 48-bit hashes

Data: Torralba et al. 2008, Krizhevsky 2009; Methods: Strecha et al. 2011 (diff-hash); Shakhnarovich 2005 (SSH); Liu et al. 2011 (AGH); Liu et al. 2012 (KSH); Masci, B^2, Schmidhuber 2012 (NN)

Image retrieval with similarity-preserving hashing

60K images from 10 different classes taken from Tiny images
Represented using 384-dimensional GIST descriptor
Training: 200 images per class; Testing: 59K images

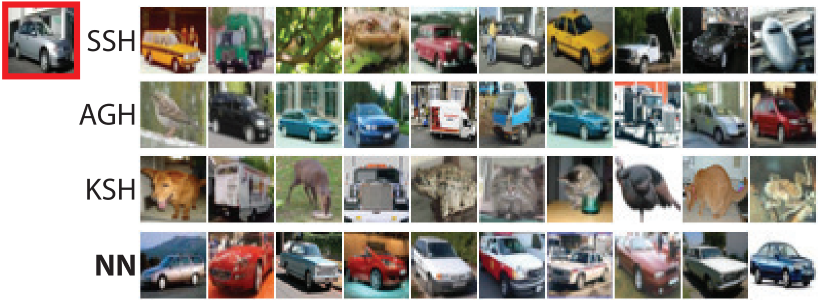


Ranking using 48-bit hashes

Data: Torralba et al. 2008, Krizhevsky 2009; Methods: Strecha et al. 2011 (diff-hash); Shakhnarovich 2005 (SSH); Liu et al. 2011 (AGH); Liu et al. 2012 (KSH); Masci, B^2, Schmidhuber 2012 (NN)

Image retrieval with similarity-preserving hashing

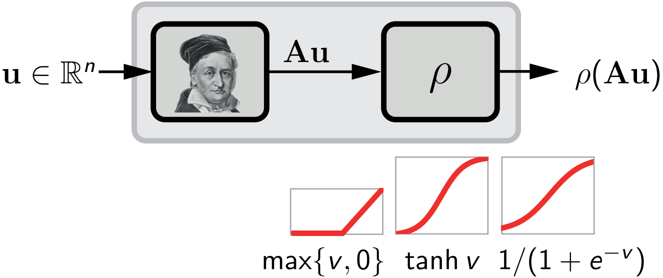
60K images from 10 different classes taken from Tiny images
Represented using 384-dimensional GIST descriptor
Training: 200 images per class; Testing: 59K images



Ranking using 48-bit hashes

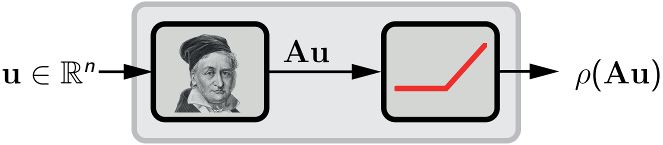
Data: Torralba et al. 2008, Krizhevsky 2009; Methods: Strecha et al. 2011 (diff-hash); Shakhnarovich 2005 (SSH); Liu et al. 2011 (AGH); Liu et al. 2012 (KSH); Masci, B^2, Schmidhuber 2012 (NN)

Assumptions



- Fully-connected linear layers with random Gaussian weights
- Element-wise approximately truncated linear activation
- No pooling
- Low dimensional input data

Assumptions



- Fully-connected linear layers with random Gaussian weights
- ReLU activation
- No pooling
- Low dimensional input data

Angle distortion

Theorem (concentration of output angle): **for** $\mathbf{u}, \mathbf{v} \in \mathcal{K}$

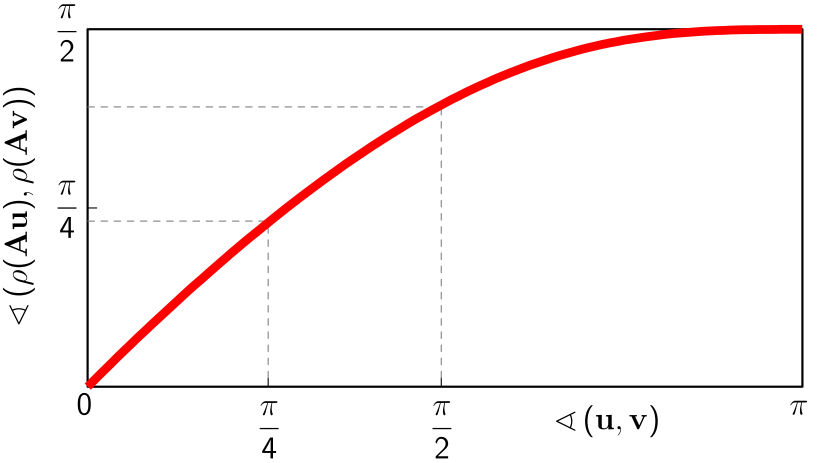
$$\cos \sphericalangle (\rho(\mathbf{A}\mathbf{u}), \rho(\mathbf{A}\mathbf{v})) \;\approx\; \cos \sphericalangle (\mathbf{u}, \mathbf{v}) + \psi(\sphericalangle (\mathbf{u}, \mathbf{v}))$$

where $\psi(\alpha) = \frac{1}{\pi}(\sin \alpha - \alpha \cos \alpha)$

Giryes, Sapiro, B, 2015

Angle distortion

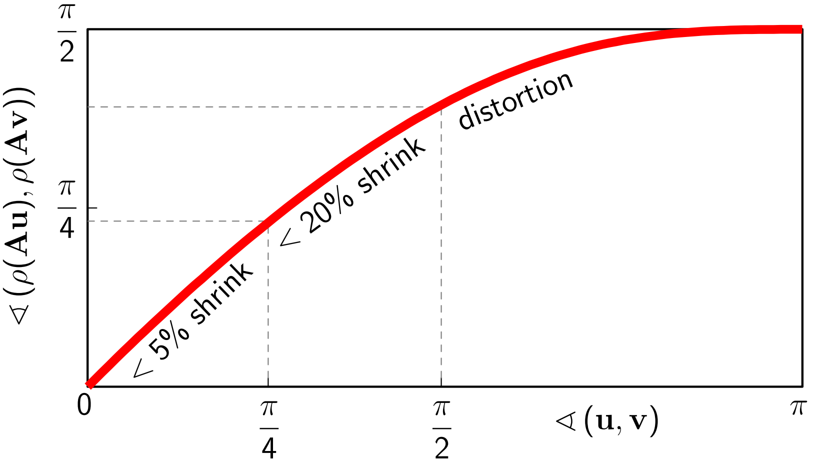
$$\cos\sphericalangle(\rho(\mathbf{Au}),\rho(\mathbf{Av})) \approx \cos\sphericalangle(\mathbf{u},\mathbf{v}) + \psi(\sphericalangle(\mathbf{u},\mathbf{v}))$$



Giryes, Sapiro, B, 2015

Angle distortion

$$\cos \sphericalangle (\rho(\mathbf{Au}), \rho(\mathbf{Av})) \approx \cos \sphericalangle (\mathbf{u}, \mathbf{v}) + \psi(\sphericalangle (\mathbf{u}, \mathbf{v}))$$



Giryes, Sapiro, B, 2015

Distance distortion

Theorem (concentration of output distance):

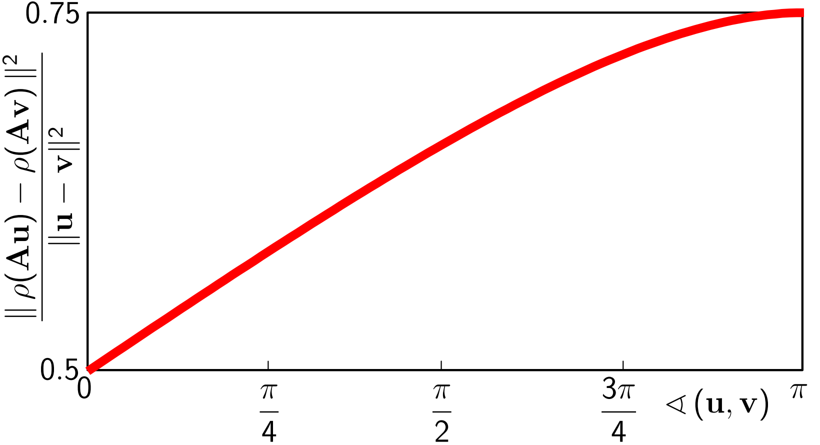
$$\| \rho(\mathbf{Au}) - \rho(\mathbf{Av}) \|^2 \approx \frac{1}{2}\|\mathbf{u} - \mathbf{v}\|^2 + \|\mathbf{u}\|\|\mathbf{v}\| \, \psi(\sphericalangle(\mathbf{u}, \mathbf{v}))$$

for $\mathbf{u}, \mathbf{v} \in \mathcal{K}$**, where** $\psi(\alpha) = \frac{1}{\pi}(\sin\alpha - \alpha\cos\alpha)$

Giryes, Sapiro, B, 2015

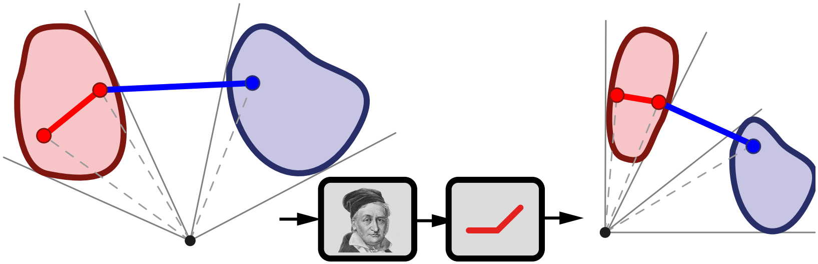
Distance distortion

$$\|\,\rho(\mathbf{Au})-\rho(\mathbf{Av})\,\|^2 \;\approx\; \frac{1}{2}\|\mathbf{u}-\mathbf{v}\|^2+\|\mathbf{u}\|\|\mathbf{v}\|\,\psi(\sphericalangle\,(\mathbf{u},\mathbf{v}))$$



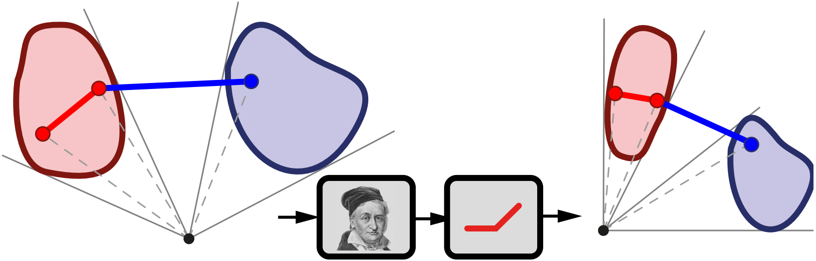
Giryes, Sapiro, B, 2015

Angle and distance distortion



Giryes, Sapiro, B, 2015

Angle and distance distortion

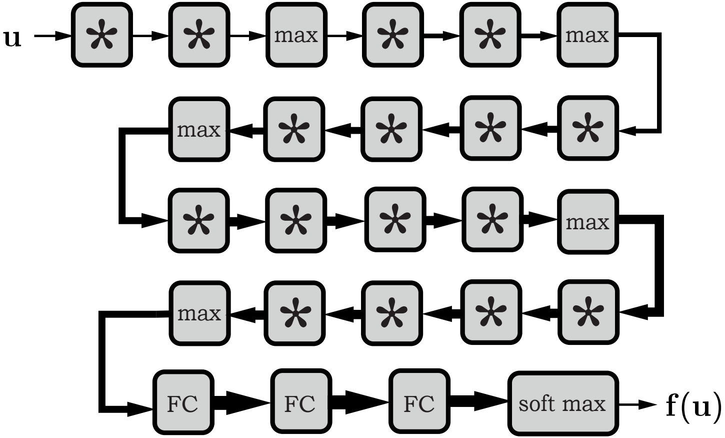


- Points with small angles between them become closer than points with large angles between them

Giryes, Sapiro, B, 2015

Inside a real network

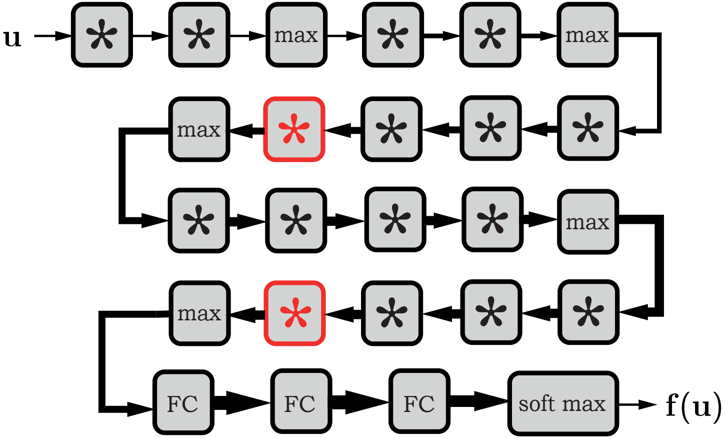
State-of-the-art 19-layer CNN trained on ImageNet



Simonyan & Zisserman, 2014

Inside a real network

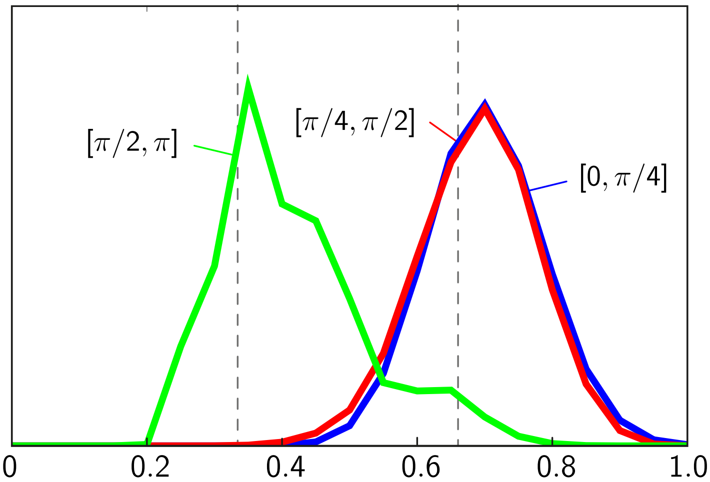
State-of-the-art 19-layer CNN trained on ImageNet



Simonyan & Zisserman, 2014

Angle distortion at 8-th layer

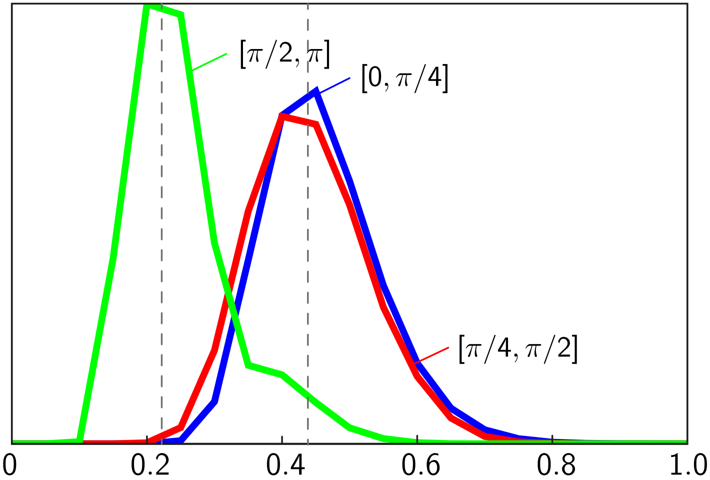
Distribution of $\sphericalangle\left(\mathbf{f}(\mathbf{u}), \mathbf{f}(\mathbf{v})\right) / \sphericalangle(\mathbf{u}, \mathbf{v})$



Giryes, Sapiro, B, 2015

Angle distortion at 16-th layer

Distribution of $\sphericalangle\left(\mathbf{f(u)},\mathbf{f(v)}\right)/\sphericalangle(\mathbf{u},\mathbf{v})$



Giryes, Sapiro, B, 2015

Training set size

- DNNs are stable: close points in the input are close in the output

Giryes, Sapiro, B, 2015

Training set size

- DNNs are stable: close points in the input are close in the output
- Network performing well on an ϵ-net $\mathcal{K}_\epsilon$ in $\mathcal{K}$ performs well on entire $\mathcal{K}$

Giryes, Sapiro, B, 2015

Training set size

- DNNs are stable: close points in the input are close in the output
- Network performing well on an ϵ-net $\mathcal{K}_\epsilon$ in $\mathcal{K}$ performs well on entire $\mathcal{K}$
- **Sudakov's minoration:**

$$\log |\mathcal{K}_\epsilon| \leq \frac{c\, \omega^2(\mathcal{K})}{\epsilon^2}$$

Giryes, Sapiro, B, 2015

Training set size

- DNNs are stable: close points in the input are close in the output
- Network performing well on an ϵ-net $\mathcal{K}_\epsilon$ in $\mathcal{K}$ performs well on entire $\mathcal{K}$
- **Sudakov's minoration:**

$$\log |\mathcal{K}_\epsilon| \leq \frac{c\,\omega^2(\mathcal{K})}{\epsilon^2}$$

- Not tight!

Giryes, Sapiro, B, 2015

Training set size

- DNNs are stable: close points in the input are close in the output
- Network performing well on an ϵ-net $\mathcal{K}_\epsilon$ in $\mathcal{K}$ performs well on entire $\mathcal{K}$
- **Sudakov's minoration:**

$$\log |\mathcal{K}_\epsilon| \leq \frac{c\,\omega^2(\mathcal{K})}{\epsilon^2}$$

- Not tight! ...but introduces Gaussian mean width $\omega(\mathcal{K})$ as the measure of data complexity

Giryes, Sapiro, B, 2015

Training set size

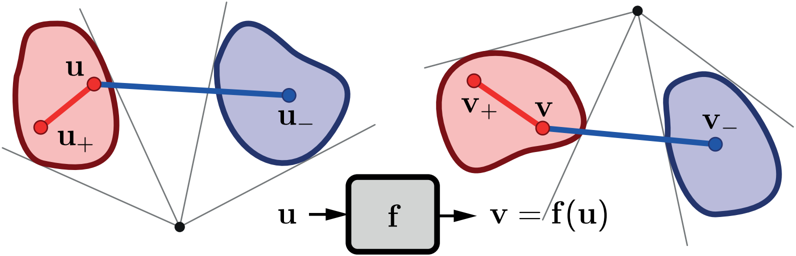
- DNNs are stable: close points in the input are close in the output
- Network performing well on an ϵ-net $\mathcal{K}_\epsilon$ in $\mathcal{K}$ performs well on entire $\mathcal{K}$
- **Sudakov's minoration:**

$$\log |\mathcal{K}_\epsilon| \leq \frac{c\,\omega^2(\mathcal{K})}{\epsilon^2}$$

- Not tight! ...but introduces Gaussian mean width $\omega(\mathcal{K})$ as the measure of data complexity
- Situation is much better in practice

Giryes, Sapiro, B, 2015

Random data points

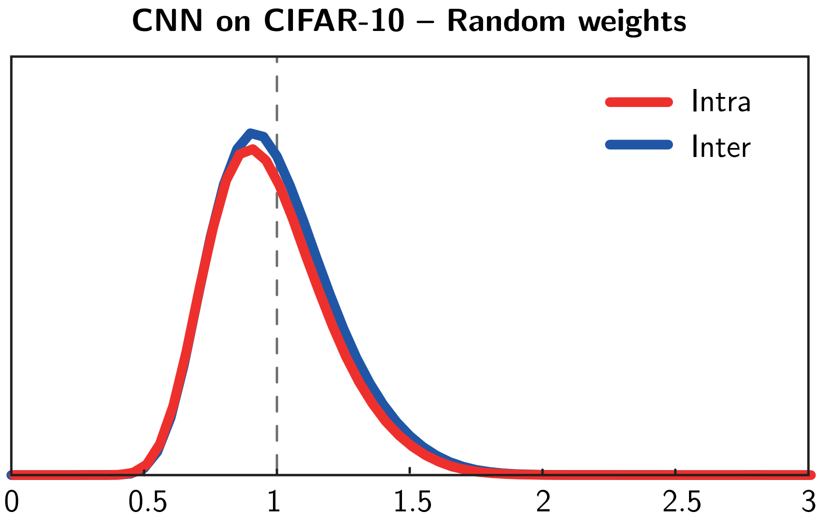


Distance ratios between random triplets $(\mathbf{u}, \mathbf{u}_+, \mathbf{u}_-)$

Intra-class $= \dfrac{\|\mathbf{v}_+ - \mathbf{v}\|}{\|\mathbf{u}_+ - \mathbf{u}\|}$ **Inter-class** $= \dfrac{\|\mathbf{v}_- - \mathbf{v}\|}{\|\mathbf{u}_- - \mathbf{u}\|}$

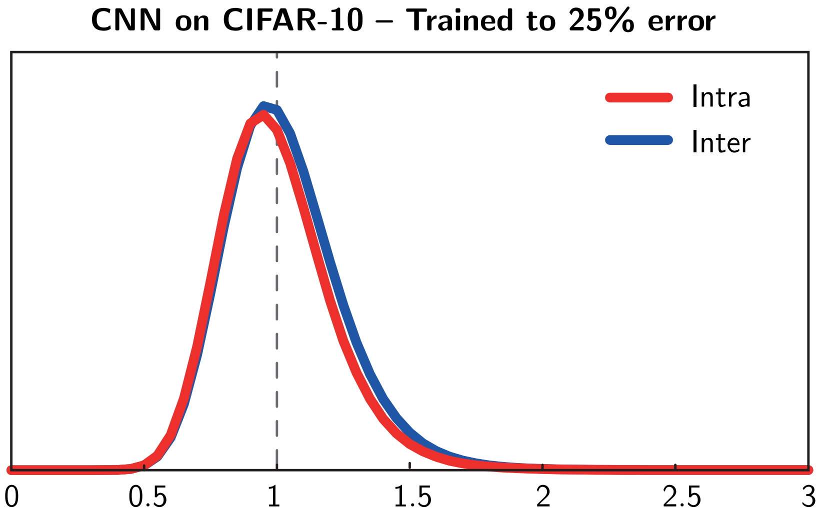
Giryes, Sapiro, B, 2015

Random data points



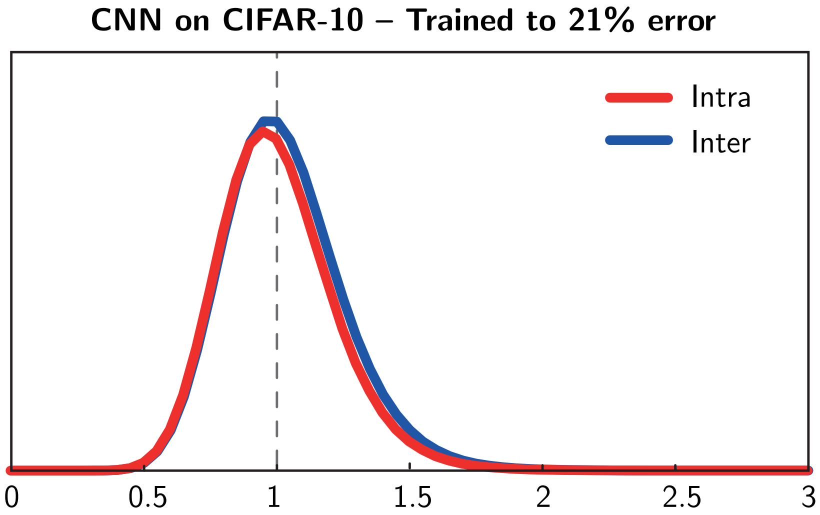
Giryes, Sapiro, B, 2015

Random data points



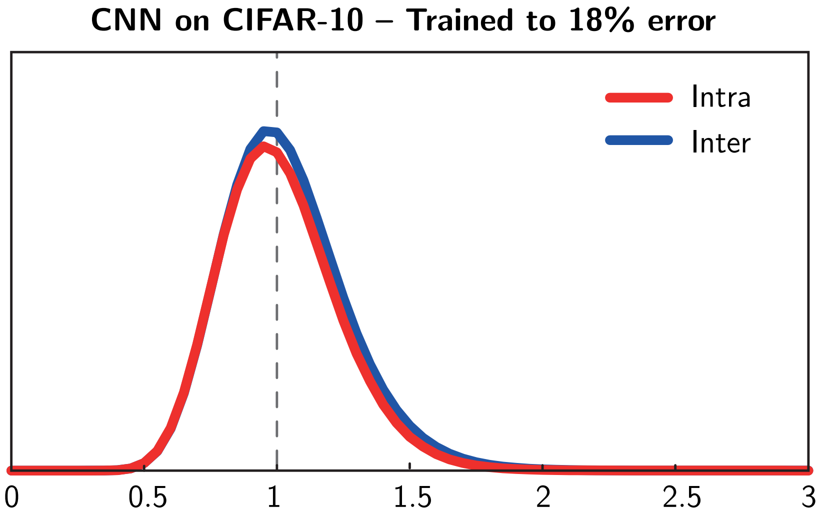
Giryes, Sapiro, B, 2015

Random data points



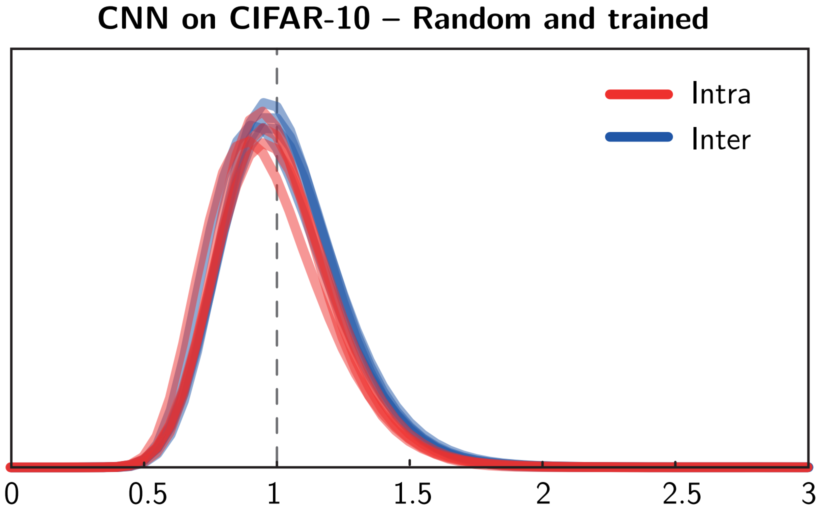
Giryes, Sapiro, B, 2015

Random data points



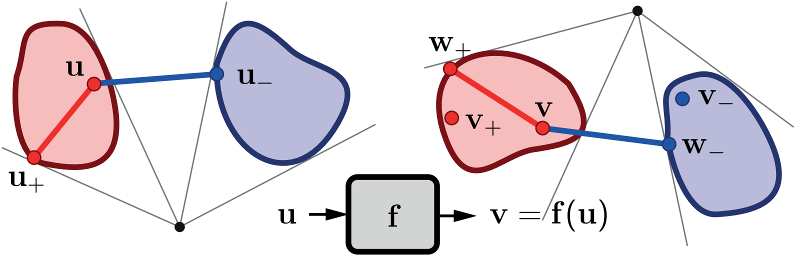
Giryes, Sapiro, B, 2015

Random data points



Giryes, Sapiro, B, 2015

Class boundary points

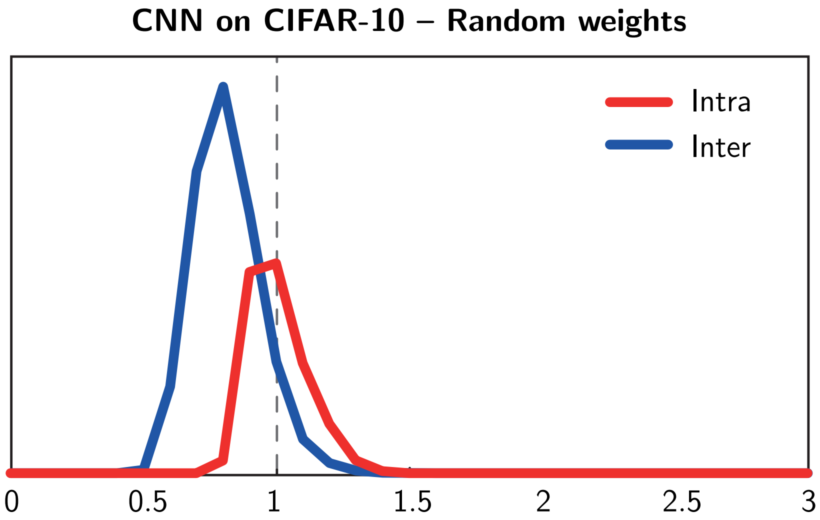


$\mathbf{u}$ random, $\mathbf{u}_+$ farthest in class, $\mathbf{u}_-$ closest not in class

$$\textbf{Intra-class} = \frac{\|\mathbf{w}_+ - \mathbf{v}\|}{\|\mathbf{u}_+ - \mathbf{u}\|} \qquad \textbf{Inter-class} = \frac{\|\mathbf{w}_- - \mathbf{v}\|}{\|\mathbf{u}_- - \mathbf{u}\|}$$

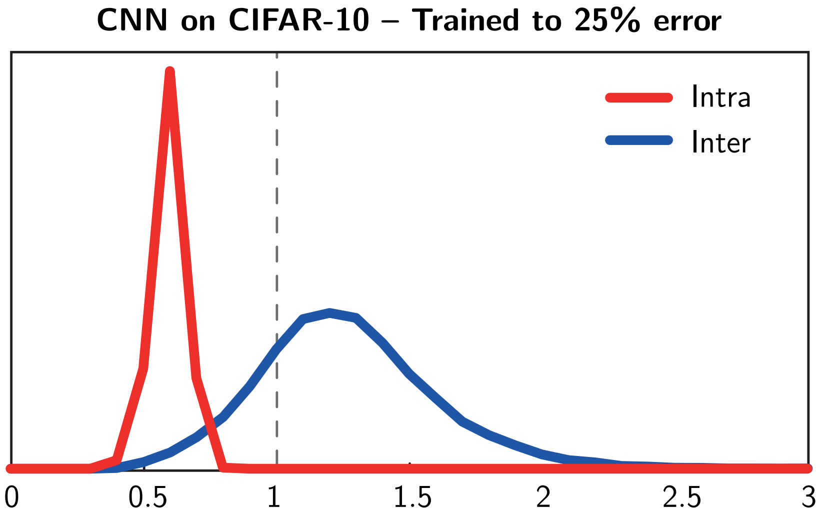
Giryes, Sapiro, B, 2015

Class boundary points



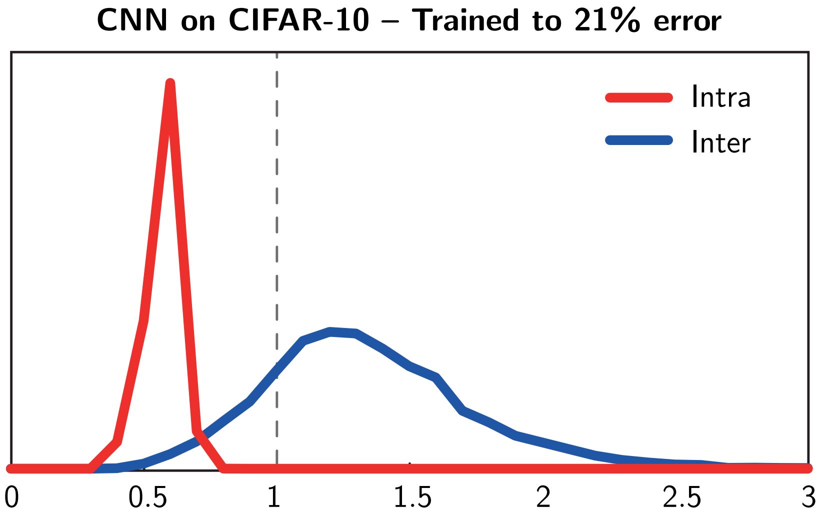
Giryes, Sapiro, B, 2015

Class boundary points



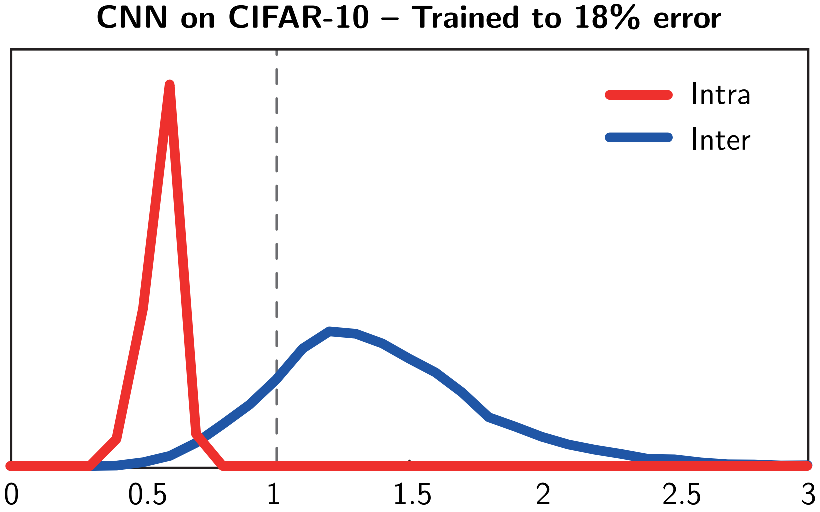
Giryes, Sapiro, B, 2015

Class boundary points



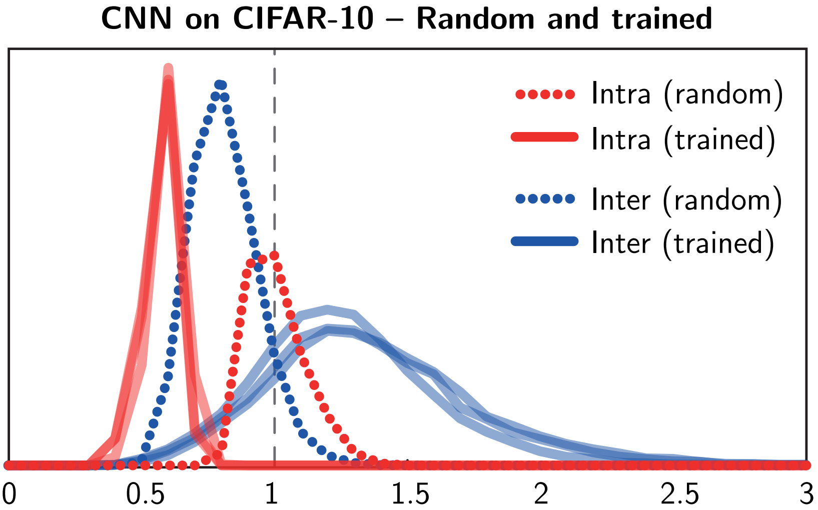
Giryes, Sapiro, B, 2015

Class boundary points



Giryes, Sapiro, B, 2015

Class boundary points



Giryes, Sapiro, B, 2015

Role of training

- Negligible effect on random data points

Giryes, Sapiro, B, 2015

Role of training

- Negligible effect on random data points
- Random weights perform well universally

Giryes, Sapiro, B, 2015

Role of training

- Negligible effect on random data points
- Random weights perform well universally
- Major effect on class boundary points

Giryes, Sapiro, B, 2015

Role of training

- Negligible effect on random data points
- Random weights perform well universally
- Major effect on class boundary points

  Intra-class distances shrink

Giryes, Sapiro, B, 2015

Role of training

- Negligible effect on random data points
- Random weights perform well universally
- Major effect on class boundary points

  Intra-class distances shrink

  Inter-class distances grow

Giryes, Sapiro, B, 2015

Role of training

- Negligible effect on random data points
- Random weights perform well universally
- Major effect on class boundary points

  Intra-class distances shrink

  Inter-class distances grow
- Only a small subset of $\mathcal{K}_\epsilon$ is required for training

Giryes, Sapiro, B, 2015

DNNs as metric learners

- Massive supervision required for DNN training

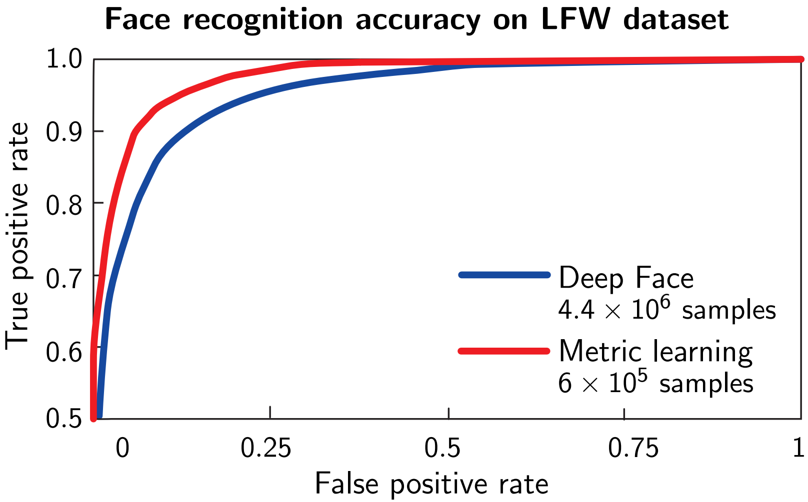
DNNs as metric learners

- Massive supervision required for DNN training
- Semi- and unsupervised training is a challenge

DNNs as metric learners

- Massive supervision required for DNN training
- Semi- and unsupervised training is a challenge
- Inject metric learning criterion into training objective to reduce the amount of labeled data

DNNs as metric learners



Huang *et al.*

Compressed discrimination

Compressed sensing: **reconstruct signal $\mathbf{x} \in \mathbb{R}^n$ given $m \ll n$ measurements**

Compressed discrimination

Compressed sensing: reconstruct signal $\mathbf{x} \in \mathbb{R}^n$ given $m \ll n$ measurements

Compressed discrimination: estimate parameter $\theta \in \mathbb{R}^k$ $(k \ll n)$ related to $\mathbf{x}$ given $m' < m$ measurements

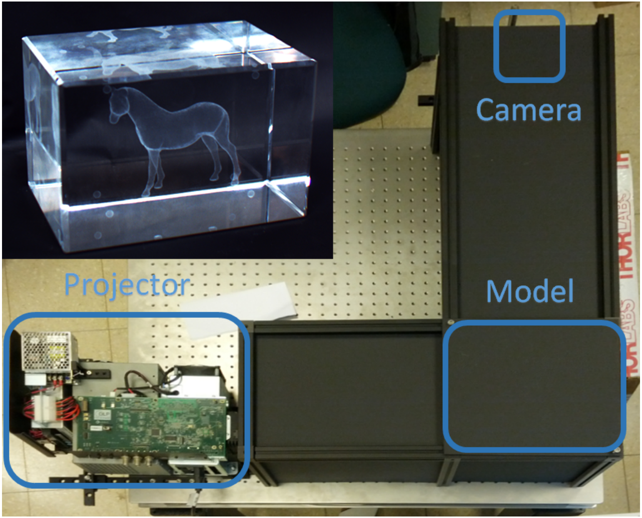
Compressed discrimination

Compressed sensing: reconstruct signal $\mathbf{x} \in \mathbb{R}^n$ given $m \ll n$ measurements

Compressed discrimination: estimate parameter $\boldsymbol{\theta} \in \mathbb{R}^k$ ($k \ll n$) related to $\mathbf{x}$ given $m' < m$ measurements

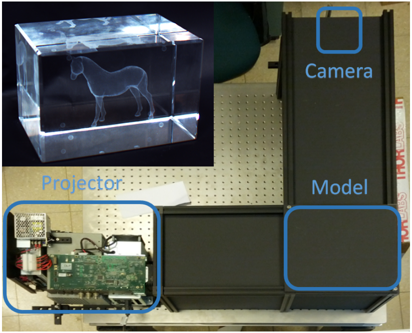
m' is insufficient to reconstruct the signal!

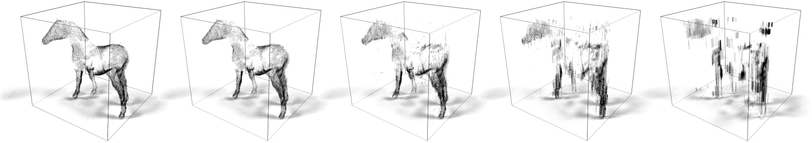
Compressed scattering tomography



Menashe & B, 2013

Compressed scattering tomography





$m : n = 1 : 2$ $\quad 1 : 4 \quad 1 : 8 \quad 1 : 16 \quad 1 : 32$

Menashe & B, 2013

Compressed discrimination

Compressed sensing: **reconstruct signal $\mathbf{x} \in \mathbb{R}^n$ given $m \ll n$ measurements**

Compressed discrimination: **estimate parameter $\theta \in \mathbb{R}^k$ ($k \ll n$) related to $\mathbf{x}$ given $m' < m$ measurements**

Compressed discrimination

Compressed sensing: reconstruct signal $\mathbf{x} \in \mathbb{R}^n$ given $m \ll n$ measurements

Random projection: global & linear

Compressed discrimination: estimate parameter $\boldsymbol{\theta} \in \mathbb{R}^k$ ($k \ll n$) related to $\mathbf{x}$ given $m' < m$ measurements

Compressed discrimination

Compressed sensing: reconstruct signal $\mathbf{x} \in \mathbb{R}^n$ given $m \ll n$ measurements

Random projection: global & linear

Compressed discrimination: estimate parameter $\theta \in \mathbb{R}^k$ $(k \ll n)$ related to $\mathbf{x}$ given $m' < m$ measurements

CNN: local & non-linear

Compressed discrimination

Compressed sensing: reconstruct signal $\mathbf{x} \in \mathbb{R}^n$ given $m \ll n$ measurements

Random projection: global & linear

Compressed discrimination: estimate parameter $\theta \in \mathbb{R}^k$ $(k \ll n)$ related to $\mathbf{x}$ given $m' < m$ measurements

CNN: local & non-linear

Compressed discrimination

Compressed sensing: reconstruct signal $\mathbf{x} \in \mathbb{R}^n$ given $m \ll n$ measurements

Random projection: global & linear

Compressed discrimination: estimate parameter $\theta \in \mathbb{R}^k$ $(k \ll n)$ related to $\mathbf{x}$ given $m' < m$ measurements

CNN: local & non-linear

Conclusion

- Gaussian mean width as a generic data complexity measure in DNN analysis

Conclusion

- Gaussian mean width as a generic data complexity measure in DNN analysis
- DNNs keep important information of the data

Conclusion

- Gaussian mean width as a generic data complexity measure in DNN analysis
- DNNs keep important information of the data
- Random Gaussian weights are good for classifying average data points

Conclusion

- Gaussian mean width as a generic data complexity measure in DNN analysis
- DNNs keep important information of the data
- Random Gaussian weights are good for classifying average data points
- Training improves performance at class boundaries

Conclusion

- Gaussian mean width as a generic data complexity measure in DNN analysis
- DNNs keep important information of the data
- Random Gaussian weights are good for classifying average data points
- Training improves performance at class boundaries
- Deep learning can be viewed as metric learning