Applied Differential Geometry and Harmonic Analysis in Deep Learning Regularization

Wei Zhu

Department of Mathematics
Duke University

Special Colloquium
Department of Mathematics
UC Santa Barbara

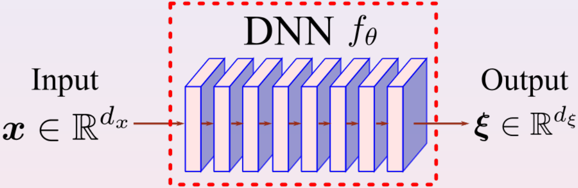
January 6, 2019

Numerous Success Stories in Deep Learning

Deep Neural Networks (DNNs) are extremely effective at learning from massive training data.

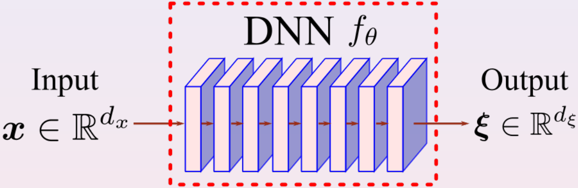
DNN Models

$f_{\boldsymbol{\theta}} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_\xi}$, and $\boldsymbol{\theta}$ is the collection of all trainable parameters.

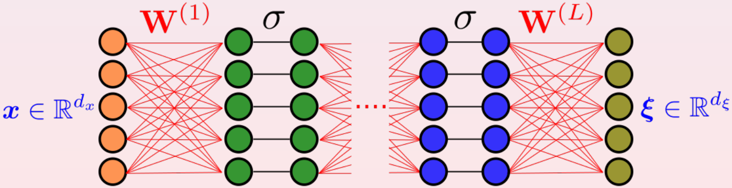


DNN Models

$f_{\boldsymbol{\theta}} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_\xi}$, and $\boldsymbol{\theta}$ is the collection of all trainable parameters.



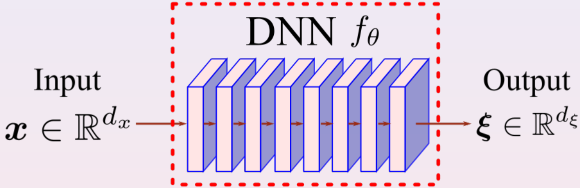
Fully-connected



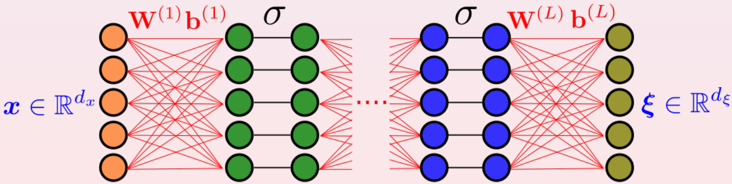
$$\boldsymbol{\xi} = f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathbf{W}^{(L)}\sigma\left(\cdots\sigma\left(\mathbf{W}^{(2)}\sigma\left(\mathbf{W}^{(1)}\boldsymbol{x}\right)\right)\cdots\right)$$

DNN Models

$f_{\boldsymbol{\theta}} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_\xi}$, and $\boldsymbol{\theta}$ is the collection of all trainable parameters.



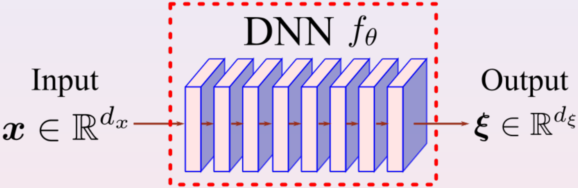
Fully-connected



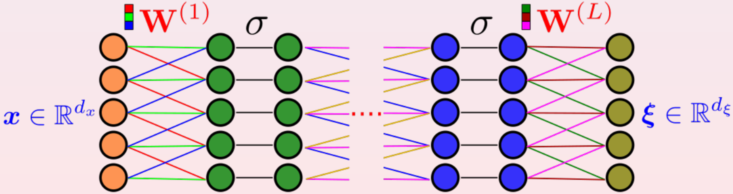
$$\boldsymbol{\xi} = f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathbf{W}^{(L)}\sigma\left(\cdots\sigma\left(\mathbf{W}^{(2)}\sigma\left(\mathbf{W}^{(1)}\boldsymbol{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)}\right)\cdots\right) + \mathbf{b}^{(L)}$$

DNN Models

$f_{\boldsymbol{\theta}} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_\xi}$, and $\boldsymbol{\theta}$ is the collection of all trainable parameters.

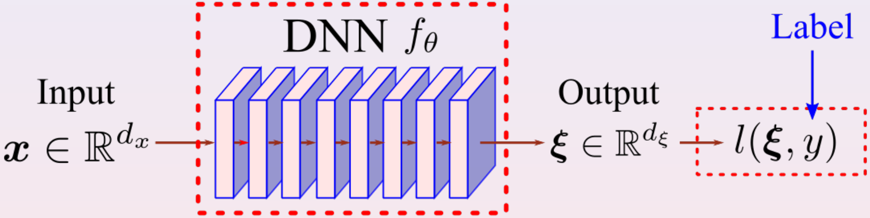


Convolutional Neural Network (CNN)



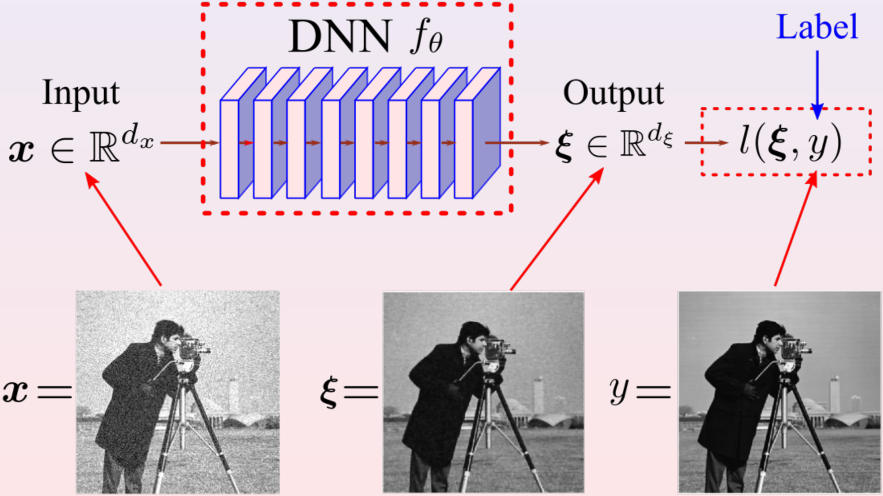
$$\boldsymbol{\xi} = f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathbf{W}^{(L)} * \sigma\left(\cdots \sigma\left(\mathbf{W}^{(2)} * \sigma\left(\mathbf{W}^{(1)} * \boldsymbol{x}\right)\right) \cdots\right)$$

DNN Models

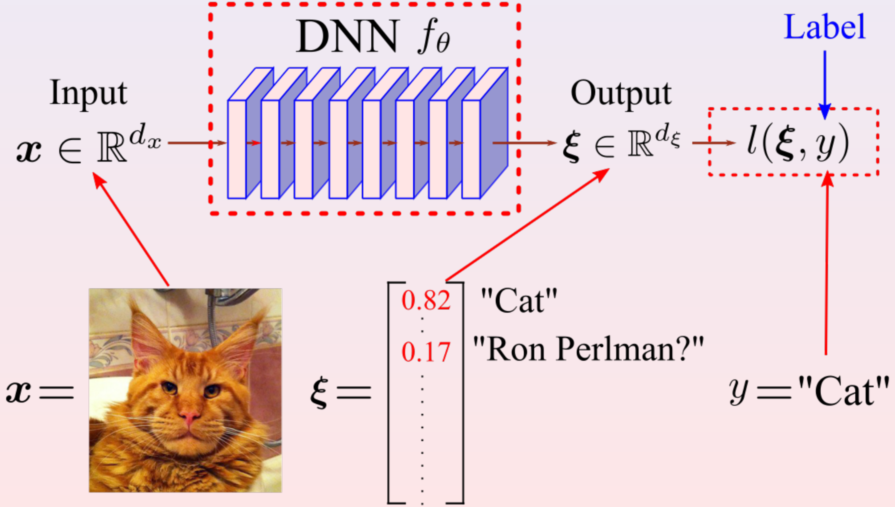


DNN f_θ
Input
$\boldsymbol{x} \in \mathbb{R}^{d_x}$
Output
$\boldsymbol{\xi} \in \mathbb{R}^{d_\xi}$
Label
$l(\boldsymbol{\xi}, y)$

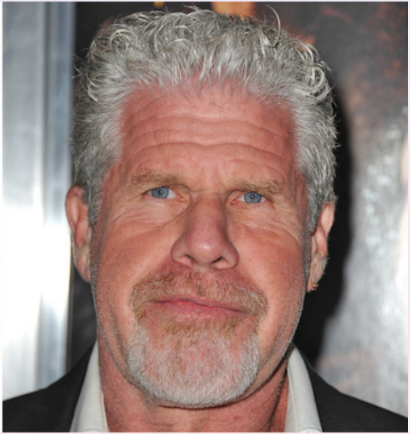
DNN Models


DNN f_θ
Input
$\boldsymbol{x} \in \mathbb{R}^{d_x}$
Output
$\boldsymbol{\xi} \in \mathbb{R}^{d_\xi}$
Label
$l(\boldsymbol{\xi}, y)$
$\boldsymbol{x} =$
$\boldsymbol{\xi} =$
$y =$

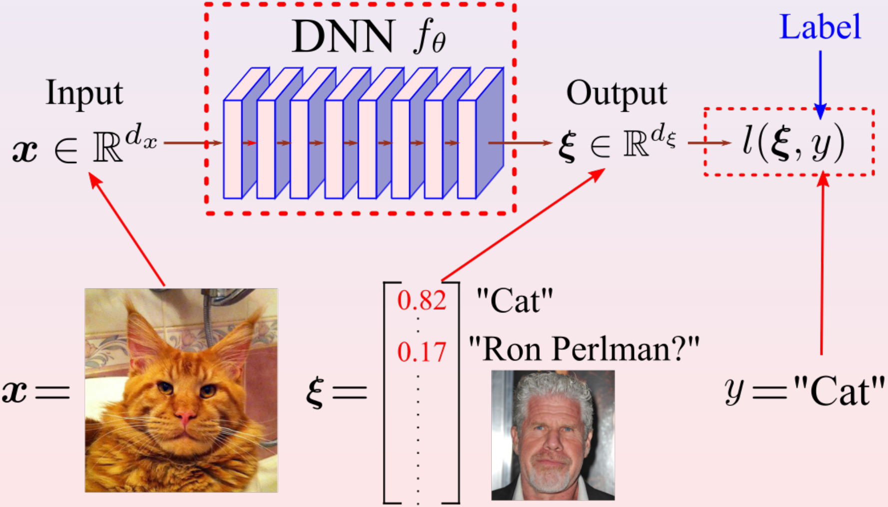
DNN Models


DNN f_θ
Label
Input
$\boldsymbol{x} \in \mathbb{R}^{d_x}$
Output
$\boldsymbol{\xi} \in \mathbb{R}^{d_\xi}$
$l(\boldsymbol{\xi}, y)$
$\boldsymbol{x} =$
$\boldsymbol{\xi} =$
0.82
"Cat"
0.17
"Ron Perlman?"
$y =$ "Cat"

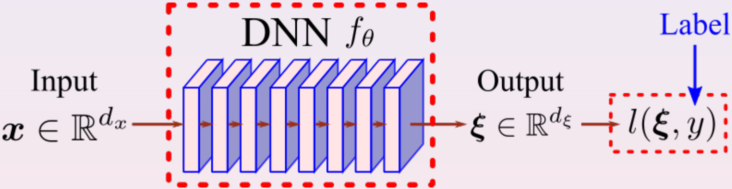
DNN Models

DNN Models


DNN f_θ
Label
Input
$\boldsymbol{x} \in \mathbb{R}^{d_x}$
Output
$\boldsymbol{\xi} \in \mathbb{R}^{d_\xi}$
$l(\boldsymbol{\xi}, y)$
$\boldsymbol{x} =$
$\boldsymbol{\xi} =$
0.82
0.17
"Cat"
"Ron Perlman?"
$y =$ "Cat"

DNN Models

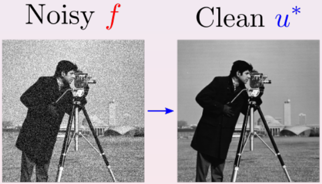


Training a DNN on the given labeled data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) := \frac{1}{N} \sum_{i=1}^{N} l(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) = \frac{1}{N} \sum_{i=1}^{N} l(\boldsymbol{\xi}_i, y_i)$$

From Model-Based to Data-Driven

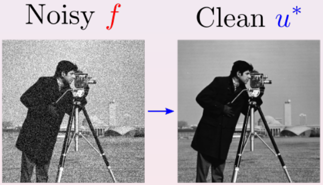
Traditional hand-crafted model-based methods are outperformed in many applications by data-driven end-to-end trained DNNs.



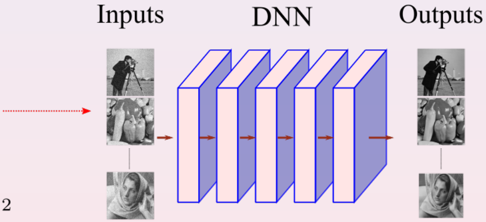
$$u^* = \arg\min_u \|\nabla u\|_1 + \lambda \|u - f\|^2$$

From Model-Based to Data-Driven

Traditional hand-crafted model-based methods are outperformed in many applications by data-driven end-to-end trained DNNs.

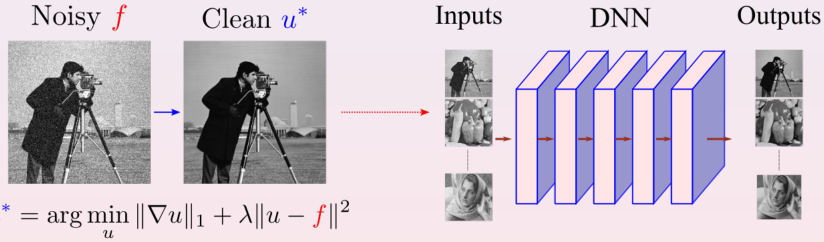


$$u^* = \arg\min_u \|\nabla u\|_1 + \lambda \|u - f\|^2$$



From Model-Based to Data-Driven

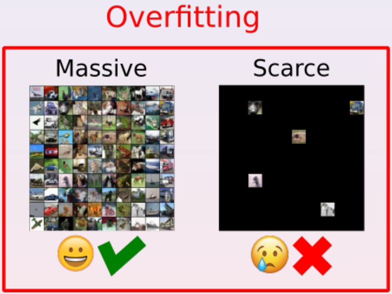
Traditional hand-crafted model-based methods are outperformed in many applications by data-driven end-to-end trained DNNs.



Nevertheless, model-based algorithms also have their own advantages:

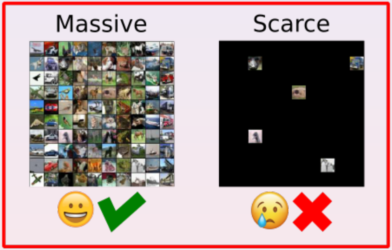
- Do not require a huge number of training data.
- More interpretable.
- More theoretical results.

Challenging Problems in Deep Learning



Challenging Problems in Deep Learning

Overfitting



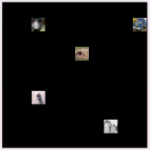
Interpretability

Challenging Problems in Deep Learning

Overfitting

Massive

Scarce



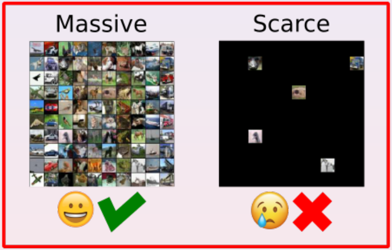
Interpretability

ξ

What's this?

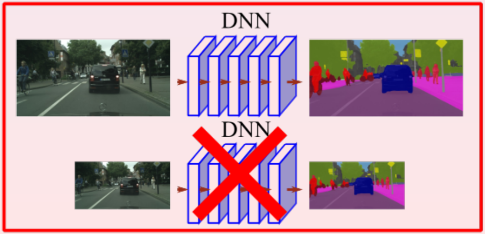
Challenging Problems in Deep Learning

Overfitting



Interpretability

Symmetry destroyed

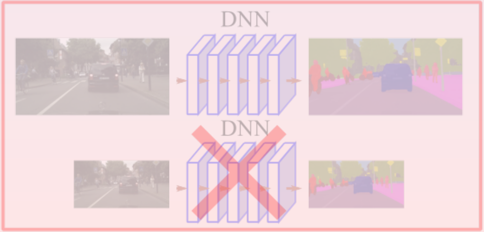


Research Objectives

Overfitting

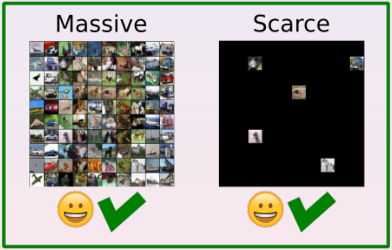
Interpretability

Symmetry destroyed



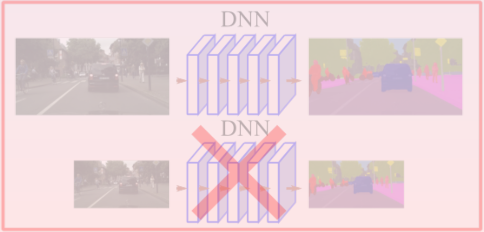
Research Objectives

Improved generalization



Interpretability

Symmetry destroyed

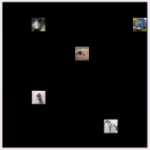


Research Objectives

Improved generalization

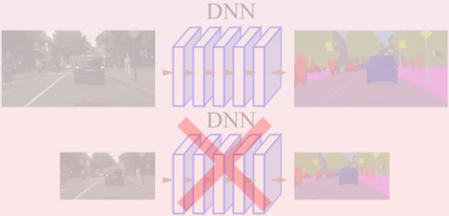
Massive

Scarce



Interpretability

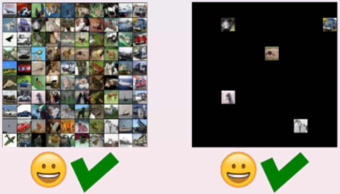
Symmetry destroyed



Research Objectives

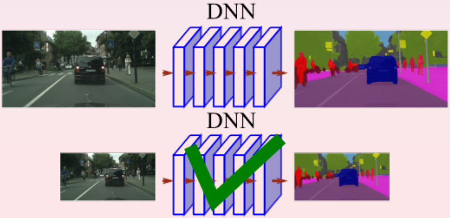
Improved generalization

Massive Scarce



Interpretability

Symmetry preserved



Contents

1. Applied differential geometry

    - **Low-Dimensional-Manifold-regularized neural Network (LDMNet)**

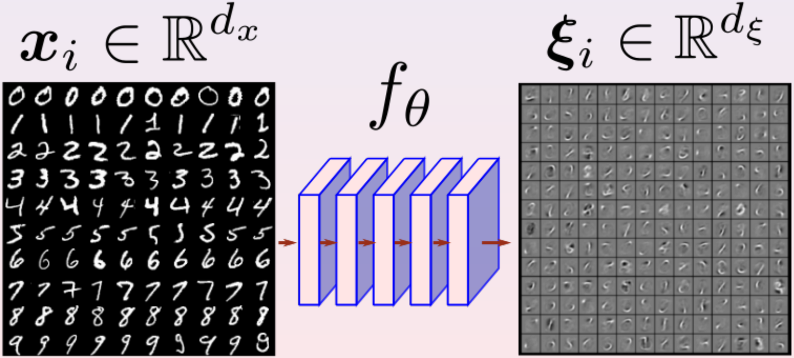
      [Z., Qiu, Huang, Calderbank, Sapiro, Daubechies 2018]

2. Applied harmonic analysis

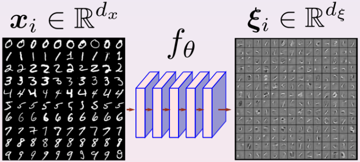
    - Scale-equivariant CNN with decomposed convolutional filters (ScDCFNet)

      [Z., Qiu, Calderbank, Sapiro, Cheng 2019]

Low Dimensional Manifold Regularization

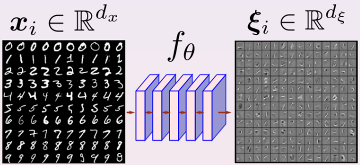


Low Dimensional Manifold Regularization

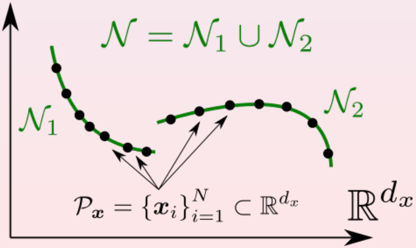


- $\mathcal{P}_{\boldsymbol{x}} = \{\boldsymbol{x}_i\}_{i=1}^N \subset \mathbb{R}^{d_x}$: data point cloud.
- $\{\boldsymbol{\xi}_i = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\}_{i=1}^N \subset \mathbb{R}^{d_\xi}$: output features.

Low Dimensional Manifold Regularization



- $\mathcal{P}_{\boldsymbol{x}} = \{\boldsymbol{x}_i\}_{i=1}^N \subset \mathbb{R}^{d_x}$: data point cloud.
- $\{\boldsymbol{\xi}_i = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\}_{i=1}^N \subset \mathbb{R}^{d_\xi}$: output features.

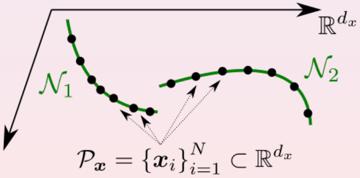


Geometric insight:

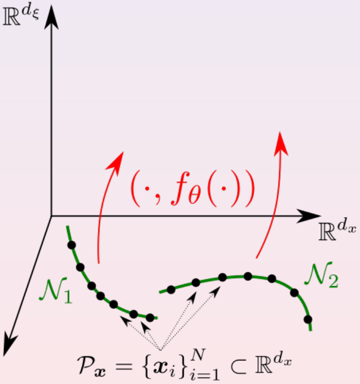
- $\mathcal{P}_{\boldsymbol{x}} \subset \mathcal{N} = \cup_{l=1}^L \mathcal{N}_l \subset \mathbb{R}^{d_x}$, and $\dim(\mathcal{N}_l) \ll d_x$.
- $f_{\boldsymbol{\theta}}|_{\mathcal{N}} : \mathcal{N} \to \mathbb{R}^{d_\xi}$ should be a smooth function over $\mathcal{N}$.

Low Dimensional Manifold Regularization

- $\mathcal{P}_{\boldsymbol{x}} \subset \mathcal{N} = \cup_{l=1}^{L} \mathcal{N}_l \subset \mathbb{R}^{d_x}$.
- $\dim(\mathcal{N}_l) \ll d_x$.

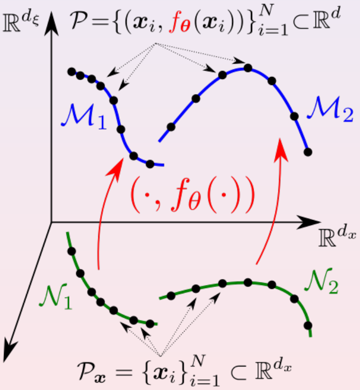


Low Dimensional Manifold Regularization



- $\mathcal{P}_{\boldsymbol{x}} \subset \mathcal{N} = \cup_{l=1}^{L} \mathcal{N}_l \subset \mathbb{R}^{d_x}$.
- $\dim(\mathcal{N}_l) \ll d_x$.
- $f_{\boldsymbol{\theta}} : \mathcal{N} \to \mathbb{R}^{d_\xi}$ is smooth.

Low Dimensional Manifold Regularization

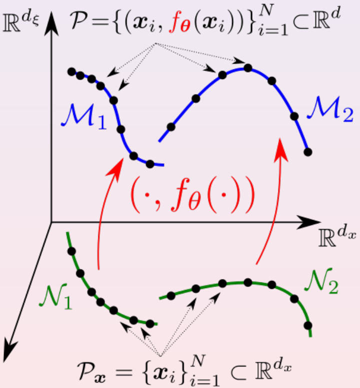


- $\mathcal{P}_{\boldsymbol{x}} \subset \mathcal{N} = \cup_{l=1}^L \mathcal{N}_l \subset \mathbb{R}^{d_x}$.
- $\dim(\mathcal{N}_l) \ll d_x$.
- $f_{\boldsymbol{\theta}} : \mathcal{N} \to \mathbb{R}^{d_\xi}$ is smooth.

Thus

- $\mathcal{M}_l = \{(\boldsymbol{x}, f_{\boldsymbol{\theta}}(\boldsymbol{x}))\}_{x \in \mathcal{N}_l} \subset \mathbb{R}^d$ is the graph of $f_{\boldsymbol{\theta}}$ over $\mathcal{N}_l$.
- $d = d_x + d_\xi$.
- $\dim(\mathcal{M}_l) \ll d$.

Low Dimensional Manifold Regularization



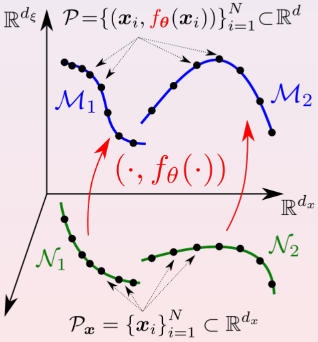
- $\mathcal{P}_{\boldsymbol{x}} \subset \mathcal{N} = \cup_{l=1}^L \mathcal{N}_l \subset \mathbb{R}^{d_x}$.
- $\dim(\mathcal{N}_l) \ll d_x$.
- $f_{\boldsymbol{\theta}} : \mathcal{N} \to \mathbb{R}^{d_\xi}$ is smooth.

Thus

- $\mathcal{M}_l = \{(\boldsymbol{x}, f_{\boldsymbol{\theta}}(\boldsymbol{x}))\}_{x \in \mathcal{N}_l} \subset \mathbb{R}^d$ is the graph of $f_{\boldsymbol{\theta}}$ over $\mathcal{N}_l$.
- $d = d_x + d_\xi$.
- $\dim(\mathcal{M}_l) \ll d$.

$\mathcal{P} = \{(\boldsymbol{x}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\}_{i=1}^N$ produced by a good feature extractor $f_{\boldsymbol{\theta}}$ should sample a collection of **low dimensional manifolds** $\mathcal{M} = \cup_{l=1}^L \mathcal{M}_l$.

Low Dimensional Manifold Regularized Neural Networks

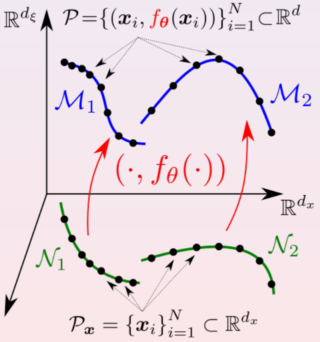


- **Overfitting** occurs when $\dim(\mathcal{M}_l)$ is too large after training.
- Use $\dim(\mathcal{M}_l)$ as a regularizer:

$$\min_{\boldsymbol{\theta}, \mathcal{M}=\cup_{l=1}^L \mathcal{M}_l} L(\boldsymbol{\theta}) + \lambda \sum_{l=1}^L |\mathcal{M}_l| \dim(\mathcal{M}_l)$$

s.t. $\mathcal{P} = \{(\boldsymbol{x}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\}_{i=1}^N \subset \mathcal{M}$.

Low Dimensional Manifold Regularized Neural Networks



- **Overfitting** occurs when $\dim(\mathcal{M}_l)$ is too large after training.
- Use $\dim(\mathcal{M}_l)$ as a regularizer:

$$\min_{\boldsymbol{\theta}, \mathcal{M}=\cup_{l=1}^L \mathcal{M}_l} L(\boldsymbol{\theta}) + \lambda \sum_{l=1}^L |\mathcal{M}_l| \dim(\mathcal{M}_l)$$

$$\text{s.t. } \mathcal{P} = \{(\boldsymbol{x}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\}_{i=1}^N \subset \mathcal{M}.$$

- **Question:** How to calculate $\dim(\mathcal{M}_l)$ in a tractable way?

Dimension of a Manifold

Proposition

Let $\mathcal{M}$ be a smooth submanifold isometrically embedded in $\mathbb{R}^d$. For any $\boldsymbol{p} = (p_j)_{j=1}^d \in \mathcal{M}$,

$$\dim(\mathcal{M}) = \sum_{j=1}^{d} |\nabla_{\mathcal{M}} \alpha_j(\boldsymbol{p})|^2 ,$$

where $\alpha_j(\boldsymbol{p}) = p_j$ is the (ambient space) coordinate function, and $\nabla_{\mathcal{M}}$ is the gradient operator on $\mathcal{M}$ (with the induced metric.)

Remark

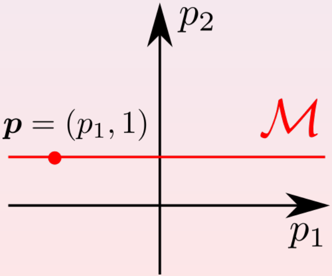
$\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_d) : \mathcal{M} \hookrightarrow \mathbb{R}^d$ *is the embedding of $\mathcal{M}$ in $\mathbb{R}^d$, i.e.,*

$$\boldsymbol{\alpha}(\boldsymbol{p}) = (\alpha_1(\boldsymbol{p}), \cdots, \alpha_d(\boldsymbol{p})) = (\boldsymbol{p}_1, \cdots, \boldsymbol{p}_d) = \boldsymbol{p}$$

Dimension of a Manifold

Sanity check: $\mathcal{M} = \{\boldsymbol{p} = (p_1, 1)\} \subset \mathbb{R}^2$, $\dim(\mathcal{M}) = 1$, $d = \dim(\mathbb{R}^2) = 2$.

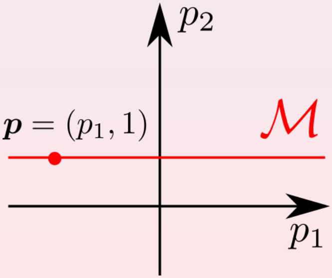
$$1 = \dim(\mathcal{M}) \stackrel{?}{=} \sum_{j=1}^{d} |\nabla_{\mathcal{M}} \alpha_j(\boldsymbol{p})|^2, \ \forall \boldsymbol{p} \in \mathcal{M}.$$



Dimension of a Manifold

Sanity check: $\mathcal{M} = \{\boldsymbol{p} = (p_1, 1)\} \subset \mathbb{R}^2$, $\dim(\mathcal{M}) = 1$, $d = \dim(\mathbb{R}^2) = 2$.

$$1 = \dim(\mathcal{M}) \stackrel{?}{=} \sum_{j=1}^{d} \left|\nabla_{\mathcal{M}} \alpha_j(\boldsymbol{p})\right|^2, \ \forall \boldsymbol{p} \in \mathcal{M}.$$



For any $\boldsymbol{p} = (p_1, 1) \in \mathcal{M}$:

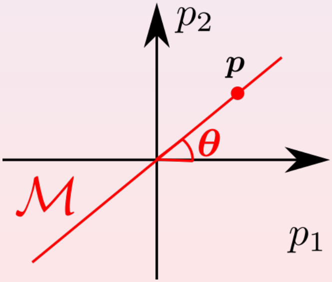
- $\alpha_1(\boldsymbol{p}) = p_1 \implies \nabla_{\mathcal{M}} \alpha_1(\boldsymbol{p}) = (1, 0)$.
- $\alpha_2(\boldsymbol{p}) \equiv 1 \implies \nabla_{\mathcal{M}} \alpha_2(\boldsymbol{p}) = (0, 0)$.
- Thus, for any $\boldsymbol{p} \in \mathcal{M}$,

$$\sum_{j=1}^{2} \left|\nabla_{\mathcal{M}} \alpha_j(\boldsymbol{p})\right|^2 = \left|(1,0)\right|^2 + \left|(0,0)\right|^2$$
$$= 1.$$

Dimension of a Manifold

Sanity check: $\mathcal{M} = \{(t\cos\theta, t\sin\theta), t \in \mathbb{R}\} \subset \mathbb{R}^2$, $\dim(\mathcal{M}) = 1$.

$$1 = \dim(\mathcal{M}) \stackrel{?}{=} \sum_{j=1}^{d} |\nabla_{\mathcal{M}} \alpha_j(\boldsymbol{p})|^2, \ \forall \boldsymbol{p} \in \mathcal{M}.$$



For any $\boldsymbol{p} = (t\cos\theta, t\sin\theta) \in \mathcal{M}$:

- $\nabla_{\mathcal{M}} \alpha_1(\boldsymbol{p}) = (\cos^2\theta, \cos\theta\sin\theta)$.
- $\nabla_{\mathcal{M}} \alpha_2(\boldsymbol{p}) = (\cos\theta\sin\theta, \sin^2\theta)$.
- Thus, for any $\boldsymbol{p} \in \mathcal{M}$,

$$\sum_{j=1}^{2} |\nabla_{\mathcal{M}} \alpha_j(\boldsymbol{p})|^2 = \cos^2\theta + \sin^2\theta$$
$$= 1.$$

Low Dimensional Manifold Regularized Neural Networks

$$\min_{\boldsymbol{\theta},\mathcal{M}} \; L(\boldsymbol{\theta}) + \lambda \sum_{l=1}^{L} |\mathcal{M}_l| \dim(\mathcal{M}_l) \quad \text{s.t. } \mathcal{P} = \{(\boldsymbol{x}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\}_{i=1}^{N} \subset \mathcal{M}.$$

- Using the proposition, we have

$$\sum_{l=1}^{L} |\mathcal{M}_l| \dim(\mathcal{M}_l) = \sum_{l=1}^{L} \int_{\mathcal{M}_l} \dim(\mathcal{M}_l) d\mu(\boldsymbol{p}) = \sum_{l=1}^{L} \int_{\mathcal{M}_l} \sum_{j=1}^{d} |\nabla_{\mathcal{M}_l} \alpha_j(\boldsymbol{p})|^2 \, d\mu(\boldsymbol{p})$$

$$= \sum_{j=1}^{d} \sum_{l=1}^{L} \|\nabla_{\mathcal{M}_l} \alpha_j\|^2_{L^2(\mathcal{M}_l)} =: \sum_{j=1}^{d} \|\nabla_{\mathcal{M}} \alpha_j\|^2_{L^2(\mathcal{M})}.$$

Low Dimensional Manifold Regularized Neural Networks

$$\min_{\boldsymbol{\theta},\mathcal{M}} \; L(\boldsymbol{\theta}) + \lambda \sum_{l=1}^{L} |\mathcal{M}_l| \dim(\mathcal{M}_l) \quad \text{s.t. } \mathcal{P} = \{(\boldsymbol{x}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\}_{i=1}^{N} \subset \mathcal{M}.$$

- Using the proposition, we have

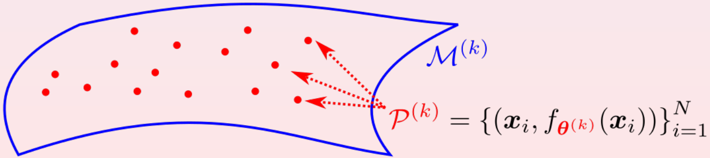
$$\begin{aligned}\sum_{l=1}^{L} |\mathcal{M}_l| \dim(\mathcal{M}_l) &= \sum_{l=1}^{L} \int_{\mathcal{M}_l} \dim(\mathcal{M}_l) d\mu(\boldsymbol{p}) = \sum_{l=1}^{L} \int_{\mathcal{M}_l} \sum_{j=1}^{d} |\nabla_{\mathcal{M}_l} \alpha_j(\boldsymbol{p})|^2 \, d\mu(\boldsymbol{p}) \\ &= \sum_{j=1}^{d} \sum_{l=1}^{L} \|\nabla_{\mathcal{M}_l} \alpha_j\|^2_{L^2(\mathcal{M}_l)} =: \sum_{j=1}^{d} \|\nabla_{\mathcal{M}} \alpha_j\|^2_{L^2(\mathcal{M})}.\end{aligned}$$

- Thus the original problem is equivalent to

$$\min_{\boldsymbol{\theta},\mathcal{M}} \; L(\boldsymbol{\theta}) + \lambda \sum_{j=1}^{d} \|\nabla_{\mathcal{M}} \alpha_j\|^2_{L^2(\mathcal{M})} \quad \text{s.t. } \mathcal{P} = \{(\boldsymbol{x}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\}_{i=1}^{N} \subset \mathcal{M}.$$

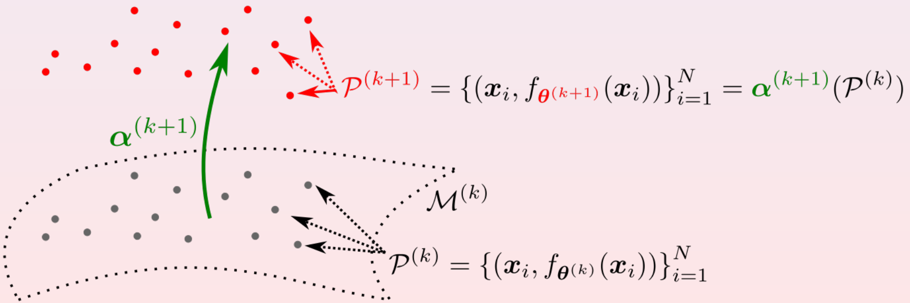
Alternate Direction of Minimization

$$\min_{\boldsymbol{\theta},\mathcal{M}} \; L(\boldsymbol{\theta}) + \lambda \sum_{j=1}^{d} \|\nabla_{\mathcal{M}} \alpha_j\|^2_{L^2(\mathcal{M})} \quad \text{s.t. } \mathcal{P} = \{(\boldsymbol{x}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\}_{i=1}^N \subset \mathcal{M}.$$



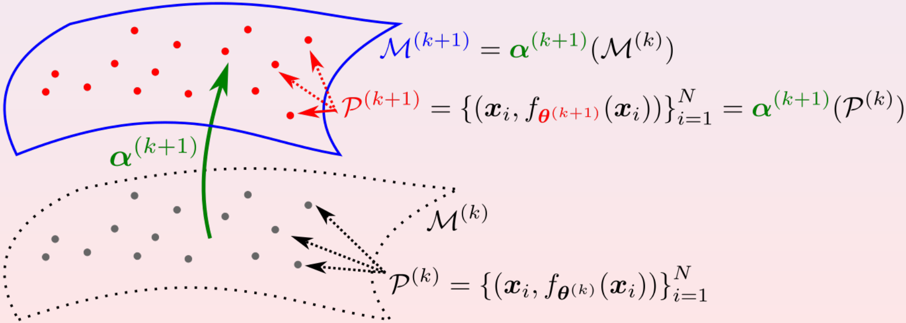
Alternate Direction of Minimization

$$\min_{\boldsymbol{\theta},\boldsymbol{\alpha}\in H^1(\mathcal{M}^{(k)})} L(\boldsymbol{\theta}) + \lambda \sum_{j=1}^{d} \|\nabla_{\mathcal{M}^{(k)}}\alpha_j\|_{L^2}^2, \quad \text{s.t. } \boldsymbol{\alpha}(\mathcal{P}^{(k)}) = \{(\boldsymbol{x}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\}_{i=1}^N$$



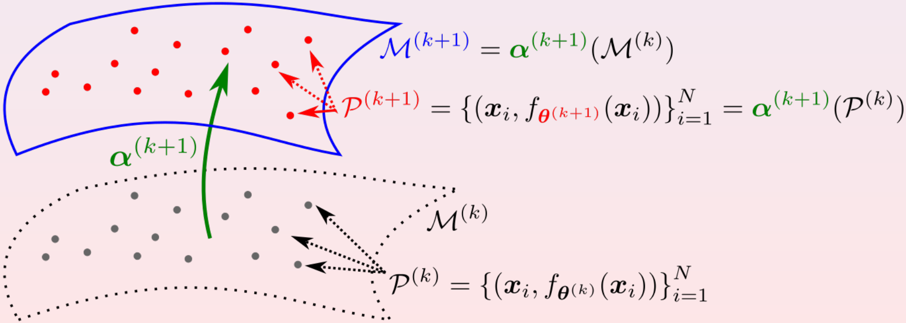
Alternate Direction of Minimization

$$\mathcal{M}^{(k+1)} := \boldsymbol{\alpha}^{(k+1)}(\mathcal{M}^{(k)}).$$



Alternate Direction of Minimization

$$\min_{\boldsymbol{\theta},\mathcal{M}} L(\boldsymbol{\theta}) + \lambda \sum_{j=1}^{d} \|\nabla_{\mathcal{M}} \alpha_j\|^2_{L^2(\mathcal{M})} \quad \text{s.t. } \mathcal{P} = \{(\boldsymbol{x}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\}_{i=1}^{N} \subset \mathcal{M}.$$



Solving the Perturbed Embedding Function $\boldsymbol{\alpha}$

Each α_j update can be cast into the following Euler-Lagrange equation:

$$\begin{cases} -\Delta_{\mathcal{M}} u(\boldsymbol{p}) + \gamma \sum_{\boldsymbol{q} \in P} \delta(\boldsymbol{p} - \boldsymbol{q})(u(\boldsymbol{q}) - v(\boldsymbol{q})) = 0, \ \boldsymbol{p} \in \mathcal{M} \\ \dfrac{\partial u}{\partial n} = 0, \ \boldsymbol{p} \in \partial \mathcal{M} \end{cases}$$

where $P \subset \mathcal{M}$ is a (given) point cloud sampling the manifold $\mathcal{M}$ (not explicitly parameterized), and v is a known function on P.

Solving the Perturbed Embedding Function $\boldsymbol{\alpha}$

Each α_j update can be cast into the following Euler-Lagrange equation:

$$\begin{cases} -\Delta_{\mathcal{M}} u(\boldsymbol{p}) + \gamma \sum_{\boldsymbol{q}\in P} \delta(\boldsymbol{p}-\boldsymbol{q})(u(\boldsymbol{q}) - v(\boldsymbol{q})) = 0, & \boldsymbol{p} \in \mathcal{M} \\ \dfrac{\partial u}{\partial n} = 0, & \boldsymbol{p} \in \partial\mathcal{M} \end{cases}$$

where $P \subset \mathcal{M}$ is a (given) point cloud sampling the manifold $\mathcal{M}$ (not explicitly parameterized), and v is a known function on P.

Difficulties:

- How to deal with $\delta(\boldsymbol{p}-\boldsymbol{q})$?
- How to approximate $\Delta_{\mathcal{M}} u$ on the manifold $\mathcal{M}$?

Point Integral Method (PIM)

Theorem ([Li, Shi, Sun 2016; Osher, Shi, Z. 2017])

Let $\mathcal{M}$ be a smooth manifold and $u \in C^3(\mathcal{M})$, then

$$\left\| -\frac{1}{t}\int_{\mathcal{M}} (u(\boldsymbol{x}) - u(\boldsymbol{y}))\, R_t(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{y} + 2\int_{\partial\mathcal{M}} \frac{\partial u}{\partial n}(\boldsymbol{y})R_t(\boldsymbol{x},\boldsymbol{y})d\tau_{\boldsymbol{y}} - \int_{\mathcal{M}} \Delta_{\mathcal{M}} u(\boldsymbol{y})R_t(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{y} \right\|_{L^2(\mathcal{M})} = O(t^{1/4}),$$

where R_t is the heat kernel:

$$R_t(\boldsymbol{x},\boldsymbol{y}) = C_t \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{4t}\right).$$

Point Integral Method (PIM)

Theorem ([Li, Shi, Sun 2016; Osher, Shi, Z. 2017])

Let $\mathcal{M}$ be a smooth manifold and $u \in C^3(\mathcal{M})$, then

$$\left\| -\frac{1}{t}\int_{\mathcal{M}} (u(\boldsymbol{x}) - u(\boldsymbol{y}))\, R_t(\boldsymbol{x},\boldsymbol{y}) d\boldsymbol{y} + \cancel{2\int_{\partial\mathcal{M}} \frac{\partial u}{\partial n}(\boldsymbol{y}) R_t(\boldsymbol{x},\boldsymbol{y}) d\tau_{\boldsymbol{y}}} - \int_{\mathcal{M}} \Delta_{\mathcal{M}} u(\boldsymbol{y}) R_t(\boldsymbol{x},\boldsymbol{y}) d\boldsymbol{y} \right\|_{L^2(\mathcal{M})} = O(t^{1/4}),$$

where R_t is the heat kernel:

$$R_t(\boldsymbol{x},\boldsymbol{y}) = C_t \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{4t}\right).$$

Solving the Perturbed Embedding Function $\boldsymbol{\alpha}$

$$\begin{cases} -\Delta_{\mathcal{M}} u(\boldsymbol{p}) + \gamma \sum_{\boldsymbol{q}\in P} \delta(\boldsymbol{p}-\boldsymbol{q})(u(\boldsymbol{q}) - v(\boldsymbol{q})) = 0, \ \boldsymbol{p} \in \mathcal{M} \\ \dfrac{\partial u}{\partial n} = 0, \ \boldsymbol{p} \in \partial\mathcal{M} \end{cases}$$

(A) Convolve with the heat kernel $R_t(\boldsymbol{p},\boldsymbol{q}) = C_t \exp\left(-\frac{|\boldsymbol{p}-\boldsymbol{q}|^2}{4t}\right)$

$$-t\int_{\mathcal{M}} \Delta_{\mathcal{M}} u(\boldsymbol{q}) R_t(\boldsymbol{p},\boldsymbol{q}) d\boldsymbol{q} + \gamma t \sum_{\boldsymbol{q}\in P} R_t(\boldsymbol{p},\boldsymbol{q})\left(u(\boldsymbol{q}) - v(\boldsymbol{q})\right) = 0.$$

(B) PIM: $-t\int_{\mathcal{M}} \Delta_{\mathcal{M}} u(\boldsymbol{y}) R_t(\boldsymbol{x},\boldsymbol{y}) d\boldsymbol{y} \approx \int_{\mathcal{M}} \left(u(\boldsymbol{x}) - u(\boldsymbol{y})\right) R_t(\boldsymbol{x},\boldsymbol{y}) d\boldsymbol{y}$

$$\int_{\mathcal{M}} \left(u(\boldsymbol{p}) - u(\boldsymbol{q})\right) R_t(\boldsymbol{p},\boldsymbol{q}) d\boldsymbol{q} + \gamma t \sum_{\boldsymbol{q}\in P} R_t(\boldsymbol{p},\boldsymbol{q})\left(u(\boldsymbol{q}) - v(\boldsymbol{q})\right) = 0$$

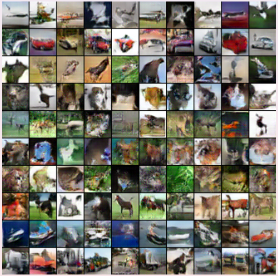
(C) Becomes a (sparse) linear system after discretization.

Experiments: Image Classification

MNIST

SVHN

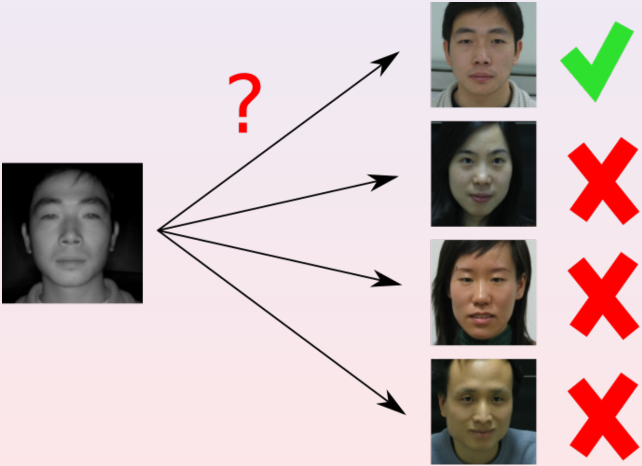
CIFAR-10



Classification Accuracy

	MNIST test accuracy (%)		
Training per class	Weight decay	DropOut	LDMNet
50	91.32 ± 0.23	92.31 ± 0.31	$\mathbf{95.57 \pm 0.28}$
100	93.38 ± 0.19	94.05 ± 0.17	$\mathbf{96.73 \pm 0.24}$
400	97.23 ± 0.21	97.95 ± 0.17	$\mathbf{98.41 \pm 0.15}$
700	97.67 ± 0.13	98.07 ± 0.11	$\mathbf{98.61 \pm 0.09}$
	SVHN test accuracy (%)		
50	71.46 ± 0.45	71.94 ± 0.37	$\mathbf{74.64 \pm 0.33}$
100	79.05 ± 0.28	79.94 ± 0.30	$\mathbf{81.36 \pm 0.24}$
400	87.38 ± 0.19	87.16 ± 0.41	$\mathbf{88.03 \pm 0.16}$
700	89.69 ± 0.26	89.83 ± 0.26	$\mathbf{90.07 \pm 0.12}$
	CIFAR-10 test accuracy (%)		
50	34.70 ± 0.80	35.94 ± 0.67	$\mathbf{41.55 \pm 0.71}$
100	42.45 ± 0.45	43.18 ± 0.32	$\mathbf{48.73 \pm 0.55}$
400	56.19 ± 0.34	56.79 ± 0.23	$\mathbf{60.08 \pm 0.24}$
700	61.84 ± 0.41	62.59 ± 0.28	$\mathbf{65.59 \pm 0.22}$
Full data	87.72 ± 0.10		$\mathbf{88.21 \pm 0.13}$

NIR-VIS Heterogeneous Face Recognition



The CASIA NIR-VIS 2.0 dataset.

NIR-VIS Heterogeneous Face Recognition

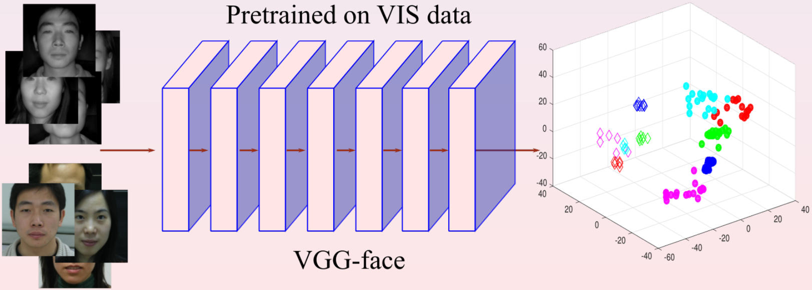
Difficulties in NIR-VIS face recognition:

- Limited NIR face images.

NIR-VIS Heterogeneous Face Recognition

Difficulties in NIR-VIS face recognition:

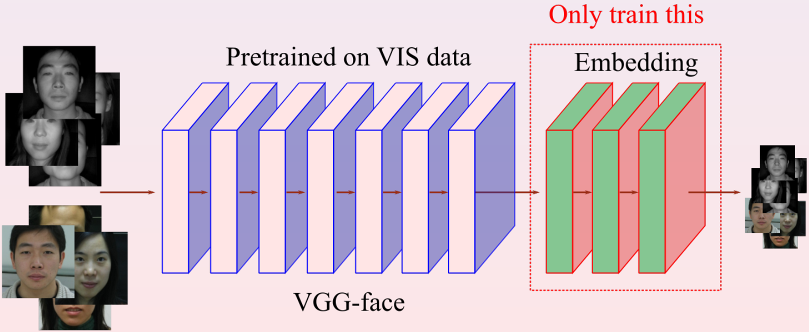
- Limited NIR face images.
- Cross-modality comparison.



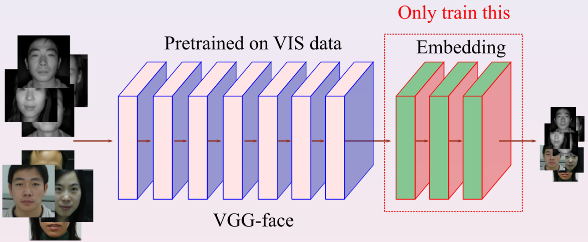
NIR-VIS Heterogeneous Face Recognition

Difficulties in NIR-VIS face recognition:

- Limited NIR face images.
- Cross-modality comparison.



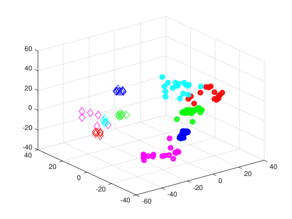
NIR-VIS Heterogeneous Face Recognition



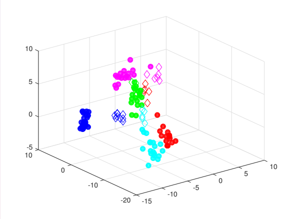
	Accuracy (%)
VGG-face	74.51 ± 1.28
VGG-face + triplet [Lezama et al., 2017]	75.96 ± 2.90
VGG-face + low-rank [Lezama et al., 2017]	80.69 ± 1.02
VGG-face Weight Decay	63.87 ± 1.33
VGG-face DropOut	66.97 ± 1.31
VGG-face LDMNet	85.02 ± 0.86

NIR-VIS Heterogeneous Face Recognition

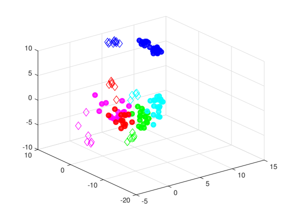
VGG-face



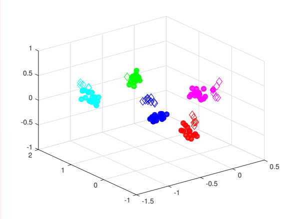
Weight decay



DropOut



LDMNet

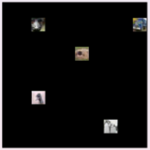


Research Objectives

Improved generalization

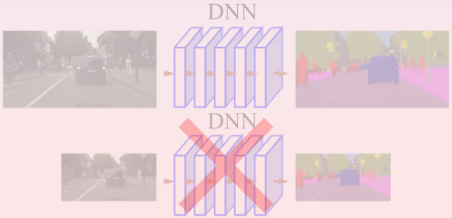
Massive

Scarce

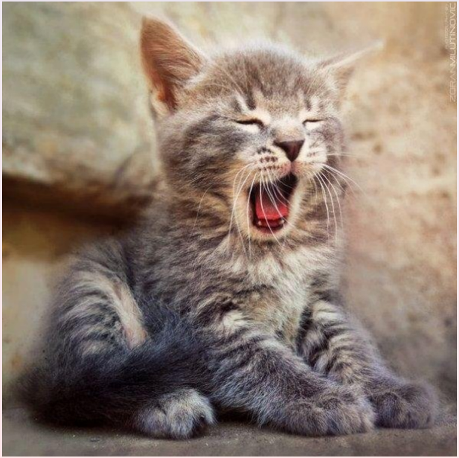


Interpretability

Symmetry destroyed



Wakey wakey!
Awesome pweeple!



Invariant/Equivariant Representation

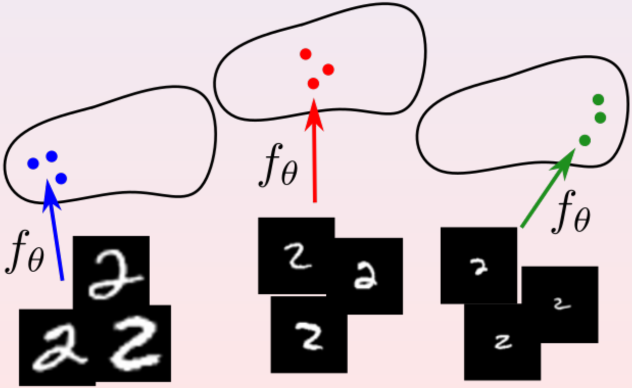
Geometric regularization improves the generalization of DNNs.

Question: How to better resolve the (low-dimensional) geometric structure using limited data.

Invariant/Equivariant Representation

Geometric regularization improves the generalization of DNNs.

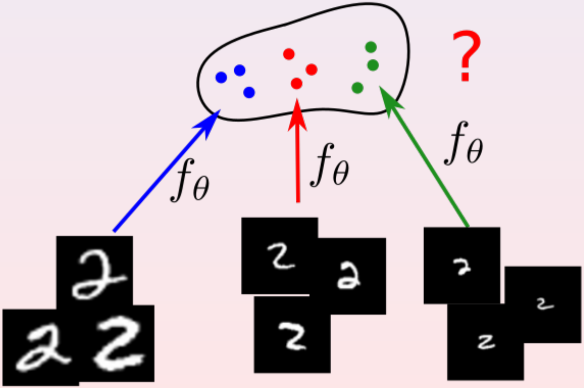
Question: How to better resolve the (low-dimensional) geometric structure using limited data.



Invariant/Equivariant Representation

Geometric regularization improves the generalization of DNNs.

Question: How to better resolve the (low-dimensional) geometric structure using limited data.



Contents

1. Applied differential geometry
   - Low-Dimensional-Manifold-regularized neural Network (LDMNet)

     [Z., Qiu, Huang, Calderbank, Sapiro, Daubechies 2018]

2. Applied harmonic analysis
   - **Scale-equivariant CNN with decomposed convolutional filters (ScDCFNet)**

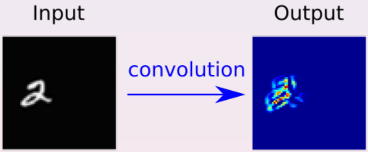
     [Z., Qiu, Calderbank, Sapiro, Cheng 2019]

CNNs Are Translation-Equivariant

- Input: $x : \mathbb{R}^2 \to \mathbb{R}$

Input

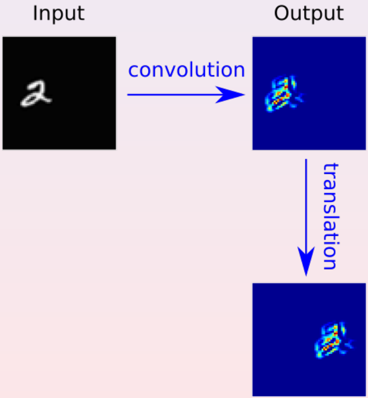
CNNs Are Translation-Equivariant



- Input: $x : \mathbb{R}^2 \to \mathbb{R}$
- Output: $y_w[x] : \mathbb{R}^2 \to \mathbb{R}$,

$$y_w[x](u) = \int_{\mathbb{R}^2} x(u + u')w(u')du'.$$

CNNs Are Translation-Equivariant

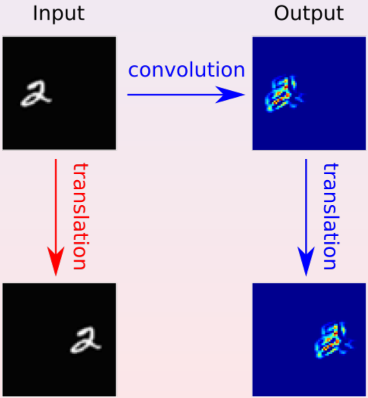


- Input: $x : \mathbb{R}^2 \to \mathbb{R}$
- Output: $y_w[x] : \mathbb{R}^2 \to \mathbb{R}$,

$$y_w[x](u) = \int_{\mathbb{R}^2} x(u+u')w(u')du'.$$

- Spatial translation: $D_v y(u) = y(u-v)$.

CNNs Are Translation-Equivariant

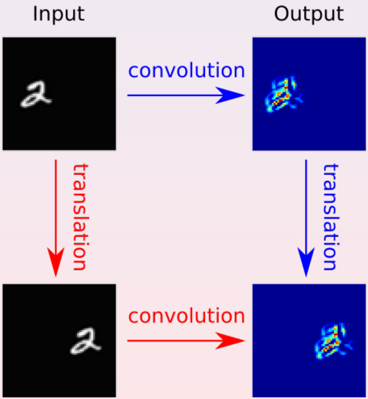


- Input: $x : \mathbb{R}^2 \to \mathbb{R}$
- Output: $y_w[x] : \mathbb{R}^2 \to \mathbb{R}$,

$$y_w[x](u) = \int_{\mathbb{R}^2} x(u + u')w(u')du'.$$

- Spatial translation: $D_v y(u) = y(u - v)$.

CNNs Are Translation-Equivariant



- Input: $x : \mathbb{R}^2 \to \mathbb{R}$
- Output: $y_w[x] : \mathbb{R}^2 \to \mathbb{R}$,

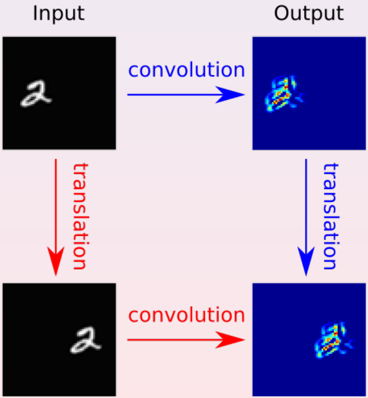
$$y_w[x](u) = \int_{\mathbb{R}^2} x(u + u')w(u')du'.$$

- Spatial translation: $D_v y(u) = y(u - v)$.
- Translation-equivariance:

$$\boldsymbol{y_w[D_v x]} = \boldsymbol{D_v y_w[x]},$$

i.e., the diagram is commutative.

CNNs Are Translation-Equivariant



- Input: $x : \mathbb{R}^2 \to \mathbb{R}$
- Output: $y_w[x] : \mathbb{R}^2 \to \mathbb{R}$,

$$y_w[x](u) = \int_{\mathbb{R}^2} x(u + u')w(u')du'.$$

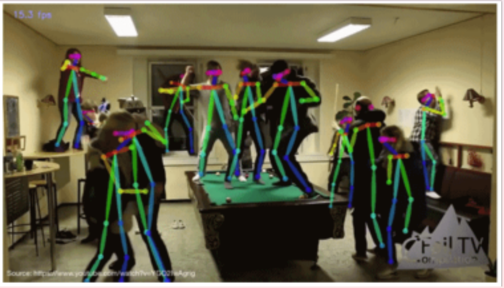
- Spatial translation: $D_v y(u) = y(u - v)$.
- Translation-equivariance:

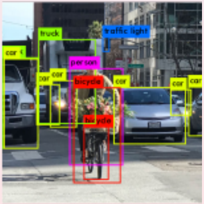
$$\boldsymbol{y_w[D_v x]} = \boldsymbol{D_v y_w[x]},$$

i.e., the diagram is commutative.

When the input is translated, the output is translated accordingly.

Tasks That Prefer Equivariant Models





Are CNNs Scale-Equivariant?

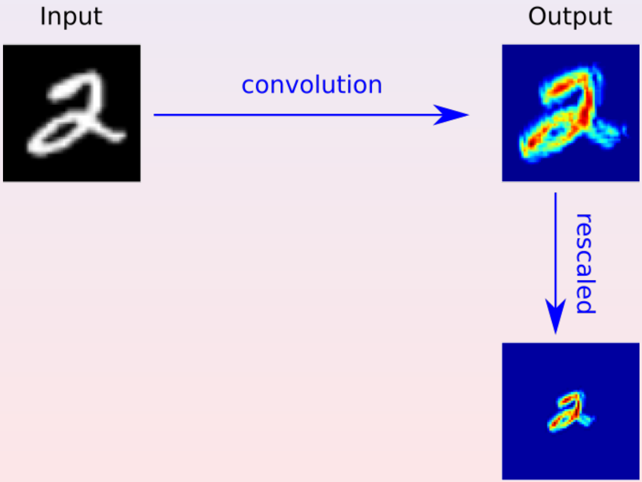
Are CNNs Scale-Equivariant? (Spoiler Alert: No!)

Are CNNs Scale-Equivariant? (Spoiler Alert: No!)

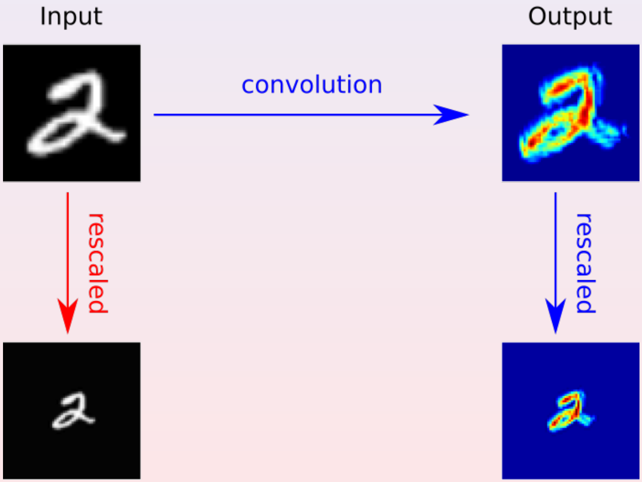
Input

Are CNNs Scale-Equivariant? (Spoiler Alert: No!)

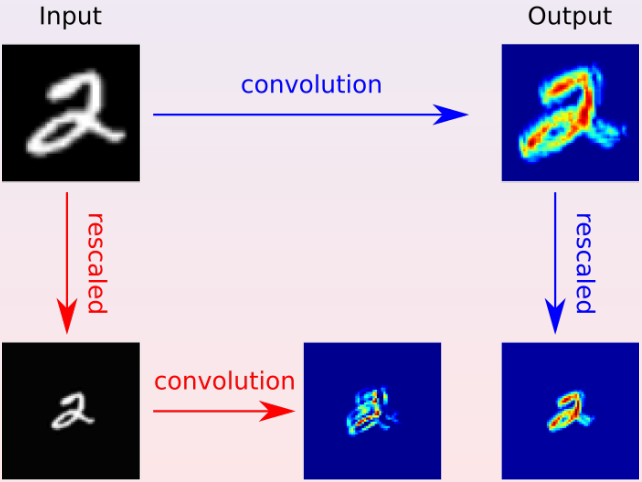
Are CNNs Scale-Equivariant? (Spoiler Alert: No!)



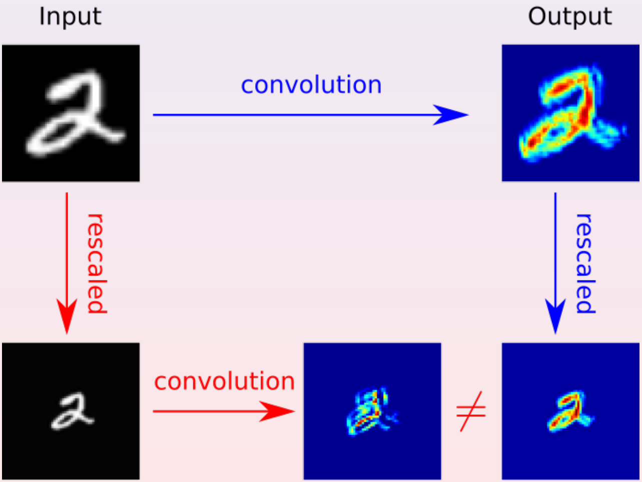
Are CNNs Scale-Equivariant? (Spoiler Alert: No!)



Are CNNs Scale-Equivariant? (Spoiler Alert: No!)



Are CNNs Scale-Equivariant? (Spoiler Alert: No!)



Previous Works on Group-Equivariant CNNs

- **Discrete symmetry groups**
  - [Cohen, Welling 2016].
- **2D rotation group**
  - [Marcos, Volpi, Komodakis, Tuia 2017].
  - [Worrall, Garbin, Turmukhambetov, Brostow 2017].
  - [Zhou, Ye, Qiu, Jiao 2017].
  - [Weiler, Hamprecht, Storath 2018].
  - [Cheng, Qiu, Calderbank, Sapiro 2019].
- **Scaling group**
  - [Kanazawa, Sharma, Jacobs 2014].
  - [Xu, Xiao, Zhang, Yang, Zhang 2014].
  - [Marcos, Kellenberger, Lobry, Tuia 2018].
  - [Ghosh, Gupta 2019].

What is lacking in the existing works for scale-equivariant CNNs?

- No general framework of imposing scale equivariance.
- No theory that guarantees the stability of the equivariant representation.

Group Equivariance

- $f : X \to Y$.

$$x \in X \xrightarrow{f} f(x) \in Y$$

Group Equivariance

$$
\begin{tikzcd}
x \in X \arrow[r, "f"] \arrow[d, "D_g"] & f(x) \in Y \arrow[d, "T_g"] \\
D_g x & T_g(f(x))
\end{tikzcd}
$$

- $f : X \to Y$.
- G is a group. $D_g : X \to X$ and $T_g : Y \to Y$ are group actions on X and Y.

Group Equivariance

$$\begin{tikzcd}
x \in X \arrow[r, "f"] \arrow[d, "D_g"] & f(x) \in Y \arrow[d, "T_g"] \\
D_g x \arrow[r, "f"] & f(D_g x) = T_g(f(x))
\end{tikzcd}$$

- $f : X \to Y$.
- G is a group. $D_g : X \to X$ and $T_g : Y \to Y$ are group actions on X and Y.
- The mapping f is said to be G-**equivariant** if

$$f(D_g x) = T_g(f(x)), \quad \forall x, g.$$

Group Equivariance

$$\begin{tikzcd}
x \in L^2(\mathbb{R}^2) \arrow[r, "\text{Convolution}"] \arrow[d, "\text{Translation}"] & f(x) \in L^2(\mathbb{R}^2) \arrow[d, "\text{Translation}"] \\
D_g x \arrow[r, "\text{Convolution}"] & f(D_g x) = T_g(f(x))
\end{tikzcd}$$

- $f : X \to Y$.
- G is a group. $D_g : X \to X$ and $T_g : Y \to Y$ are group actions on X and Y.
- The mapping f is said to be G-**equivariant** if

$$f(D_g x) = T_g(f(x)), \quad \forall x, g.$$

Group Equivariance

$$
\begin{tikzcd}
x \in X \arrow[r, "f"] \arrow[d, "D_g"] & f(x) \in Y \arrow[d, "\mathrm{Id}_Y"] \\
D_g x \arrow[r, "f"] & f(D_g x) = f(x)
\end{tikzcd}
$$

- $f : X \to Y$.
- G is a group. $D_g : X \to X$ and $T_g : Y \to Y$ are group actions on X and Y.
- The mapping f is said to be G-**equivariant** if

$$f(D_g x) = T_g(f(x)), \quad \forall x, g.$$

- When $T_g = \mathrm{Id}_Y$,

$$f(D_g x) = f(x), \quad \forall x, g,$$

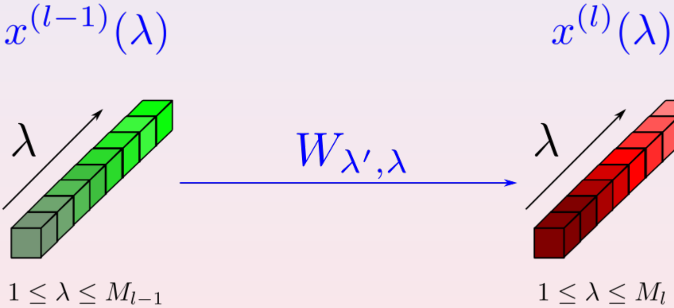
i.e., f is G-**invariant**.

Intuition on Constructing Scale-Equivariant CNNs

How to construct scale-equivariant CNNs, i.e., CNN models that are equivariant to the scaling-translation group $\mathcal{ST} = \mathcal{S} \ltimes \mathbb{R}^2 \cong \mathbb{R} \times \mathbb{R}^2$?

Intuition on Constructing Scale-Equivariant CNNs

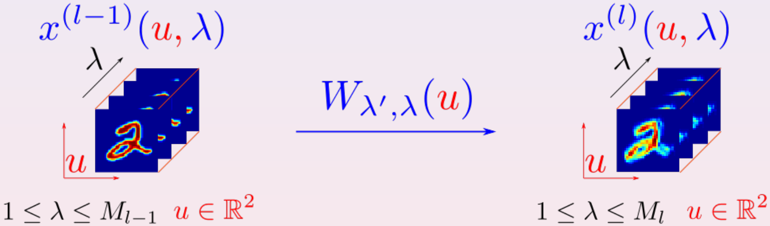
How to construct scale-equivariant CNNs, i.e., CNN models that are equivariant to the scaling-translation group $\mathcal{ST} = \mathcal{S} \ltimes \mathbb{R}^2 \cong \mathbb{R} \times \mathbb{R}^2$?



$$\boxed{x^{(l)}(\lambda) = \sum_{\lambda'=1}^{M_{l-1}} x^{(l-1)}(\lambda') W_{\lambda',\lambda}}$$

Intuition on Constructing Scale-Equivariant CNNs

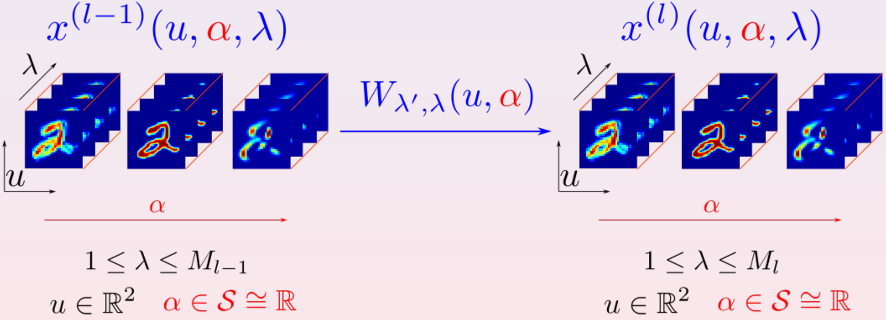
How to construct scale-equivariant CNNs, i.e., CNN models that are equivariant to the scaling-translation group $\mathcal{ST} = \mathcal{S} \ltimes \mathbb{R}^2 \cong \mathbb{R} \times \mathbb{R}^2$?



$$
\boxed{
\begin{aligned}
x^{(l)}(u, \lambda) &= \sum_{\lambda'=1}^{M_{l-1}} \int_{\mathbb{R}^2} x^{(l-1)}(u + u', \lambda') W_{\lambda',\lambda}(u') du' \\
&= \sum_{\lambda'=1}^{M_{l-1}} \left(x^{(l-1)}(\cdot, \lambda') * W_{\lambda',\lambda}(\cdot) \right)(u)
\end{aligned}
}
$$

Intuition on Constructing Scale-Equivariant CNNs

How to construct scale-equivariant CNNs, i.e., CNN models that are equivariant to the scaling-translation group $\mathcal{ST} = \mathcal{S} \ltimes \mathbb{R}^2 \cong \mathbb{R} \times \mathbb{R}^2$?

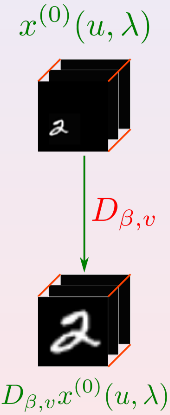


$$\boxed{x^{(l)}(u, \alpha, \lambda) = \sum_{\lambda'=1}^{M_{l-1}} \left(x^{(l-1)}(\cdot, \cdot, \lambda') \overset{?}{*} W_{\lambda', \lambda}(\cdot, \cdot) \right)(u, \alpha)}$$

Scale-Equivariant CNN

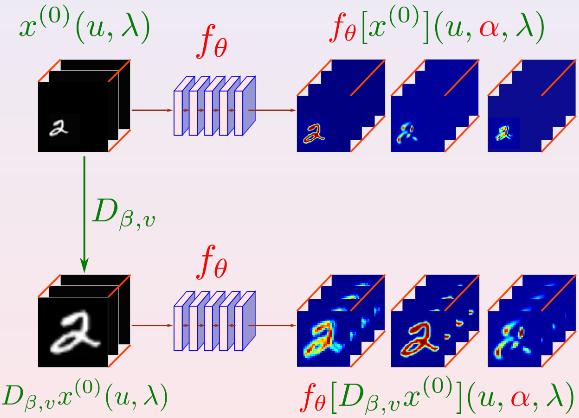
$x^{(0)}(u, \lambda)$

Scale-Equivariant CNN



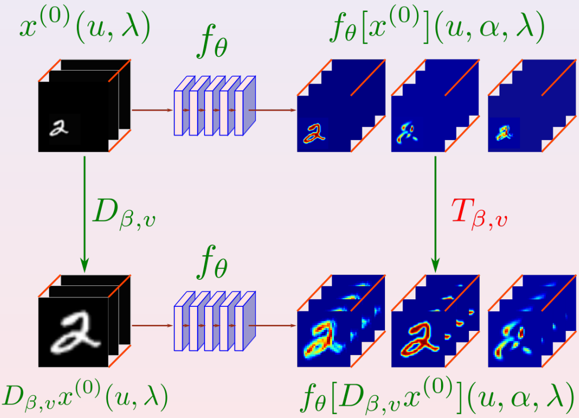
- $D_{\beta,v}x^{(0)}(u,\lambda) \coloneqq x^{(0)}\left(2^{-\beta}(u-v),\lambda\right),\ \forall(\beta,v)\in\mathcal{ST}\cong\mathbb{R}\times\mathbb{R}^2.$

Scale-Equivariant CNN



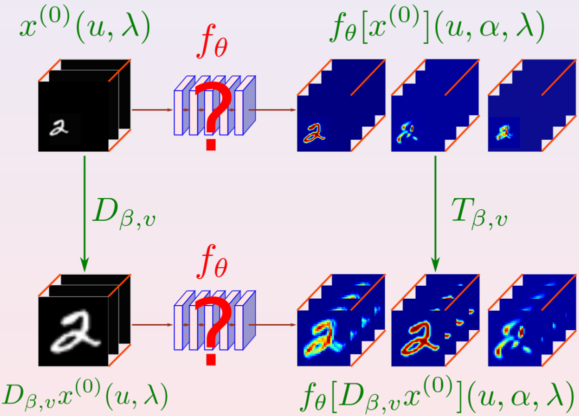
- $D_{\beta,v}x^{(0)}(u,\lambda) \coloneqq x^{(0)}\left(2^{-\beta}(u-v),\lambda\right),\ \forall(\beta,v)\in\mathcal{ST}\cong\mathbb{R}\times\mathbb{R}^2.$

Scale-Equivariant CNN



- $D_{\beta,v}x^{(0)}(u,\lambda) := x^{(0)}\left(2^{-\beta}(u-v),\lambda\right), \ \forall(\beta,v)\in\mathcal{ST}\cong\mathbb{R}\times\mathbb{R}^2.$
- $T_{\beta,v}x^{(l)}(u,\alpha,\lambda) := x^{(l)}\left(2^{-\beta}(u-v),\alpha-\beta,\lambda\right), \ \forall(\beta,v)\in\mathcal{ST}.$

Scale-Equivariant CNN



- $D_{\beta,v}x^{(0)}(u,\lambda) \coloneqq x^{(0)}\left(2^{-\beta}(u-v),\lambda\right),\ \forall(\beta,v)\in\mathcal{ST}\cong\mathbb{R}\times\mathbb{R}^2.$
- $T_{\beta,v}x^{(l)}(u,\alpha,\lambda) \coloneqq x^{(l)}\left(2^{-\beta}(u-v),\alpha-\beta,\lambda\right),\ \forall(\beta,v)\in\mathcal{ST}.$

Scale-Equivariant CNNs (Joint Convolution over $\mathbb{R}^2 \times \mathcal{S}$)

Theorem (Z., Qiu, Calderbank, Sapiro, Cheng 2019)

A feedforward neural network with an extra index $\alpha \in \mathcal{S}$ is scale-equivariant if and only if the layerwise operations are:

$$x^{(1)}[x^{(0)}](u,\alpha,\lambda) = \sigma\left(\sum_{\lambda'}\int_{\mathbb{R}^2} x^{(0)}(u+u',\lambda')W^{(1)}_{\lambda',\lambda}\left(2^{-\alpha}u'\right)2^{-2\alpha}du' + b^{(1)}(\lambda)\right)$$

$$x^{(l)}[x^{(l-1)}](u,\alpha,\lambda) = \sigma\left(\sum_{\lambda'}\int_{\mathbb{R}^2}\int_{\mathbb{R}} x^{(l-1)}(u+u',\alpha+\alpha',\lambda')W^{(l)}_{\lambda',\lambda}\left(2^{-\alpha}u',\alpha'\right)\cdot 2^{-2\alpha}d\alpha' du' + b^{(l)}(\lambda)\right), \quad \forall l > 1,$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a pointwise nonlinear activation, e.g., ReLU, and $W^{(1)}_{\lambda',\lambda}(u'), W^{(l)}_{\lambda',\lambda}(u',\alpha')$ are the (trainable) convolutional filters.

Scale-Equivariant CNNs (Joint Convolution over $\mathbb{R}^2 \times \mathcal{S}$)

Theorem (Z., Qiu, Calderbank, Sapiro, Cheng 2019)

A feedforward neural network with an extra index $\alpha \in \mathcal{S}$ is scale-equivariant if and only if the layerwise operations are:

$$x^{(1)}[x^{(0)}](u,\alpha,\lambda) = \sigma\left(\sum_{\lambda'}\int_{\mathbb{R}^2} x^{(0)}(u+u',\lambda')W^{(1)}_{\lambda',\lambda}\left(2^{-\alpha}u'\right)2^{-2\alpha}du' + b^{(1)}(\lambda)\right)$$

$$x^{(l)}[x^{(l-1)}](u,\alpha,\lambda) = \sigma\left(\sum_{\lambda'}\int_{\mathbb{R}^2}\int_{\mathbb{R}} x^{(l-1)}(u+u',\alpha+\alpha',\lambda')W^{(l)}_{\lambda',\lambda}\left(2^{-\alpha}u',\alpha'\right)\cdot 2^{-2\alpha}d\alpha' du' + b^{(l)}(\lambda)\right), \quad \forall l > 1,$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a pointwise nonlinear activation, e.g., ReLU, and $W^{(1)}_{\lambda',\lambda}(u'), W^{(l)}_{\lambda',\lambda}(u',\alpha')$ are the (trainable) convolutional filters.

Scale-Equivariant CNNs (Joint Convolution over $\mathbb{R}^2 \times \mathcal{S}$)

Theorem (Z., Qiu, Calderbank, Sapiro, Cheng 2019)

A feedforward neural network with an extra index $\alpha \in \mathcal{S}$ is scale-equivariant if and only if the layerwise operations are:

$$x^{(1)}[x^{(0)}](u,\alpha,\lambda) = \sigma\left(\sum_{\lambda'}\int_{\mathbb{R}^2} x^{(0)}(u+u',\lambda')W^{(1)}_{\lambda',\lambda}\left(2^{-\alpha}u'\right)2^{-2\alpha}du' + b^{(1)}(\lambda)\right)$$

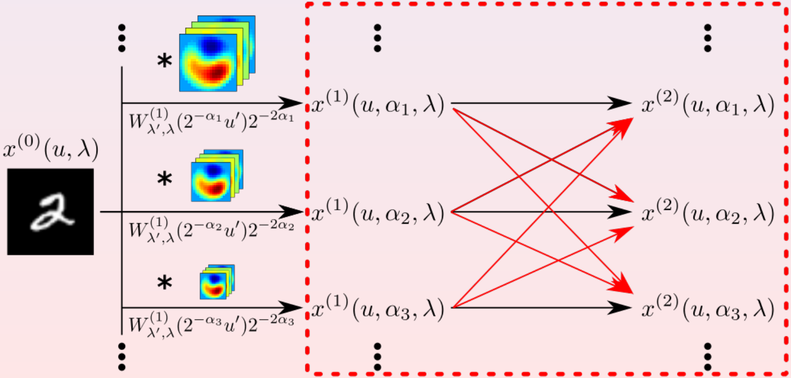
$$x^{(l)}[x^{(l-1)}](u,\alpha,\lambda) = \sigma\left(\sum_{\lambda'}\int_{\mathbb{R}^2}\int_{\mathbb{R}} x^{(l-1)}(u+u',\alpha+\alpha',\lambda')W^{(l)}_{\lambda',\lambda}\left(2^{-\alpha}u',\alpha'\right)\cdot 2^{-2\alpha}d\alpha' du' + b^{(l)}(\lambda)\right), \quad \forall l > 1,$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a pointwise nonlinear activation, e.g., ReLU, and $W^{(1)}_{\lambda',\lambda}(u')$, $W^{(l)}_{\lambda',\lambda}(u',\alpha')$ are the (trainable) convolutional filters.

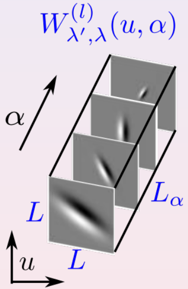
Scale-Equivariant CNNs

$$x^{(1)}(u,\alpha,\lambda) = \sum_{\lambda'} \int_{\mathbb{R}^2} x^{(0)}(u+u',\lambda') W^{(1)}_{\lambda',\lambda}\left(2^{-\alpha}u'\right) 2^{-2\alpha} du',$$

$$x^{(l)}(u,\alpha,\lambda) = \sum_{\lambda'} \int_{\mathbb{R}^2} \int_{\mathbb{R}} x^{(l-1)}(u+u',\alpha+\alpha',\lambda') W^{(l)}_{\lambda',\lambda}\left(2^{-\alpha}u',\alpha'\right) 2^{-2\alpha} d\alpha' du'.$$

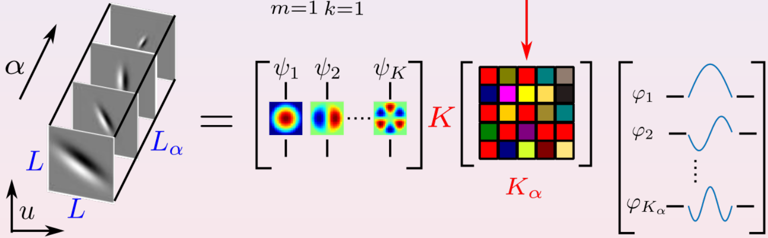


Separable Basis Decomposition



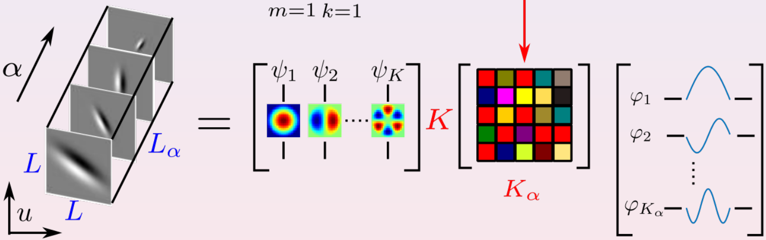
Separable Basis Decomposition

$$W^{(l)}_{\lambda',\lambda}(u,\alpha) = \sum_{m=1}^{K_\alpha} \sum_{k=1}^{K} \psi_k(u)\; a^{(l)}_{\lambda',\lambda}(k,m)\, \varphi_m(\alpha)$$



Separable Basis Decomposition

$$W^{(l)}_{\lambda',\lambda}(u,\alpha) = \sum_{m=1}^{K_\alpha} \sum_{k=1}^{K} \psi_k(u)\; a^{(l)}_{\lambda',\lambda}(k,m)\, \varphi_m(\alpha)$$

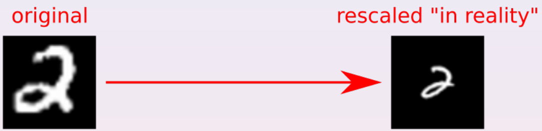


Theorem (Z., Qiu, Calderbank, Sapiro, Cheng 2019)

Both the training parameters and computational burden are reduced to a factor of KK_α/L^2L_α *after truncated basis decomposition.*

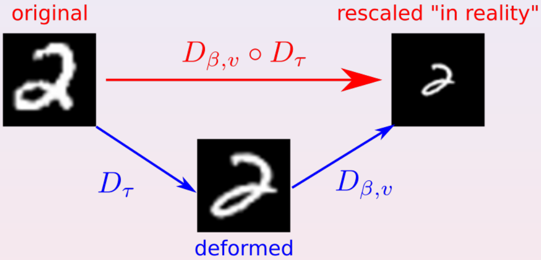
In particular, $L = L_\alpha = 5,\ K = 8,\ K_\alpha = 3 \implies KK_\alpha/L^2L_\alpha = 19.2\%.$

Stability of the Equivariant Representation to Input Deformation



- The scaling effect in reality is never exact, e.g., changing view angles.

Stability of the Equivariant Representation to Input Deformation

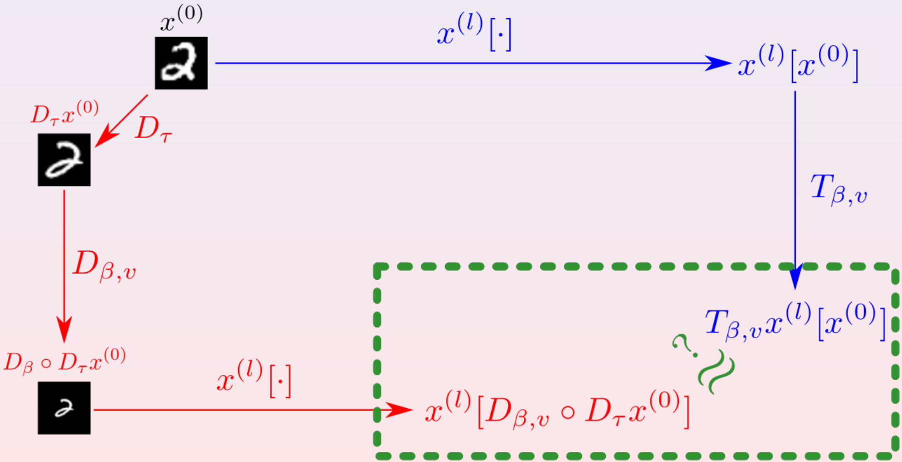


- The scaling effect in reality is never exact, e.g., changing view angles.
- A **"perfect" scaling** $D_{\beta,v}$ and a **local deformation** D_τ:

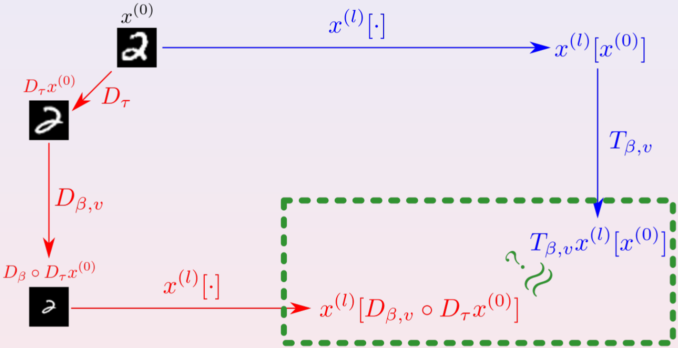
$$x^{(0)} \mapsto D_{\beta,v} \circ D_\tau x^{(0)},$$

where $D_\tau x^{(0)}(u,\lambda) = x^{(0)}(u - \tau(u), \lambda)$, and $\tau \in C^2(\mathbb{R}^2 \to \mathbb{R}^2)$ is a small local deformation.

Stability of the Equivariant Representation to Input Deformation



Stability of the Equivariant Representation to Input Deformation

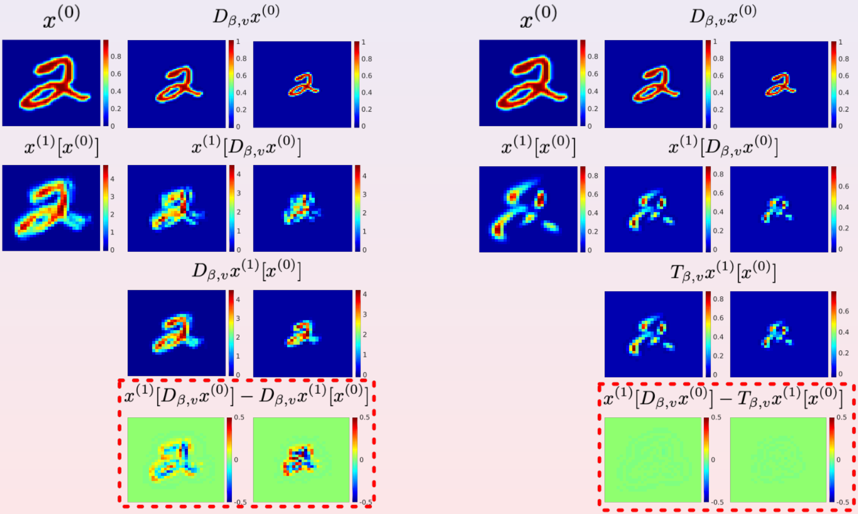


Theorem (Z., Qiu, Calderbank, Sapiro, Cheng 2019)

In an ScDCFNet with bounded expansion coefficients $a^{(l)}_{\lambda',\lambda}$ under the Fourier-Bessel norm (which is facilitated by truncated basis decomposition), we have, for any L,

$$\left\| x^{(L)}[D_{\beta,v} \circ D_\tau x^{(0)}] - T_{\beta,v} x^{(L)}[x^{(0)}] \right\| \le 2^{\beta+1} \left(4L|\nabla\tau|_\infty + 2^{-j_L}|\tau|_\infty\right) \|x^{(0)}\|.$$

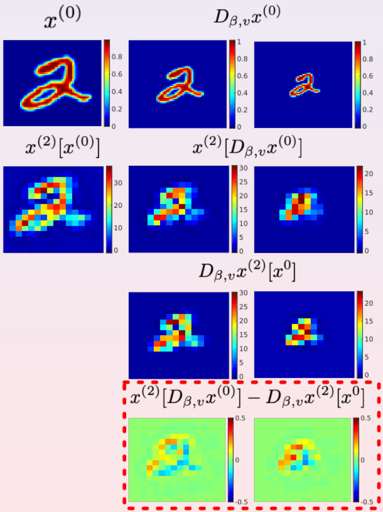
Verification of Scale Equivariance (First-Layer Feature Maps)



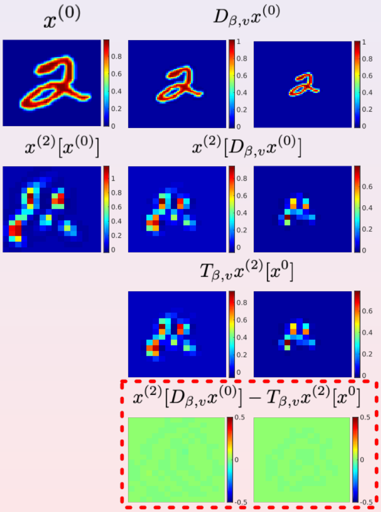
(a) Regular CNN.

(b) ScDCFNet.

Verification of Scale Equivariance (Second-Layer Feature Maps)



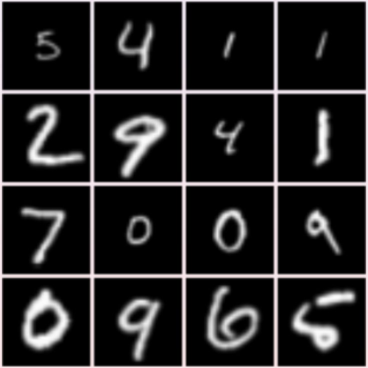
(a) Regular CNN.



(b) ScDCFNet.

Multiscale Image Classification

SMNIST

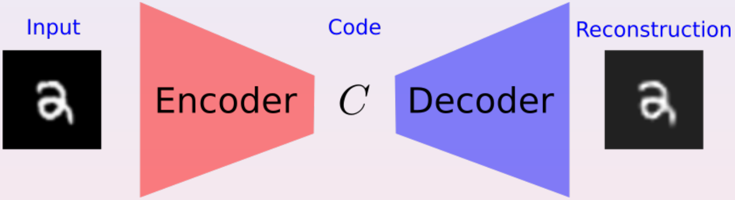


SFashion

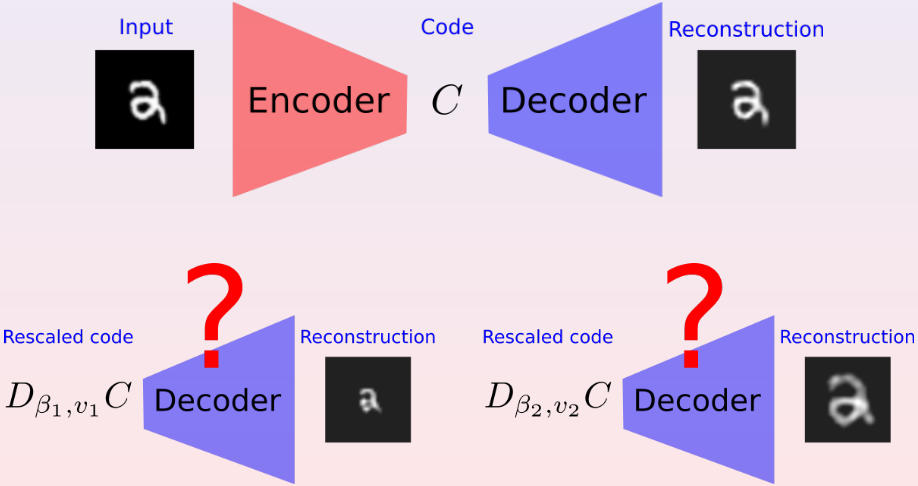
Multiscale Image Classification

Architectures	Ratio	SMNIST test accuracy (%)		SFashion test accuracy (%)	
		$N_{\text{tr}} = 2000$	$N_{\text{tr}} = 5000$	$N_{\text{tr}} = 2000$	$N_{\text{tr}} = 5000$
CNN, $M = 32$	1.00	92.60 ± 0.17	94.86 ± 0.25	77.74 ± 0.28	82.57 ± 0.38
ScDCF, $M = 16$					
$K = 10, K_\alpha = 3$	0.84	93.75 ± 0.02	95.70 ± 0.09	78.95 ± 0.31	**83.51 ± 0.71**
$K = 8, K_\alpha = 3$	**0.67**	**93.91 ± 0.30**	**95.71 ± 0.10**	79.22 ± 0.50	83.06 ± 0.32
$K = 5, K_\alpha = 3$	**0.42**	93.52 ± 0.29	95.19 ± 0.13	**79.74 ± 0.44**	83.46 ± 0.69
$K = 5, K_\alpha = 2$	0.28	93.51 ± 0.30	95.35 ± 0.21	78.57 ± 0.53	82.95 ± 0.46
ScDCF, $M = 8$					
$K = 10, K_\alpha = 2$	0.14	93.68 ± 0.17	95.21 ± 0.12	79.11 ± 0.76	82.92 ± 0.68
$K = 8, K_\alpha = 2$	0.11	93.39 ± 0.25	95.25 ± 0.47	78.43 ± 0.76	83.05 ± 0.58
$K = 5, K_\alpha = 2$	0.09	93.21 ± 0.20	94.99 ± 0.12	77.97 ± 0.37	82.21 ± 0.67

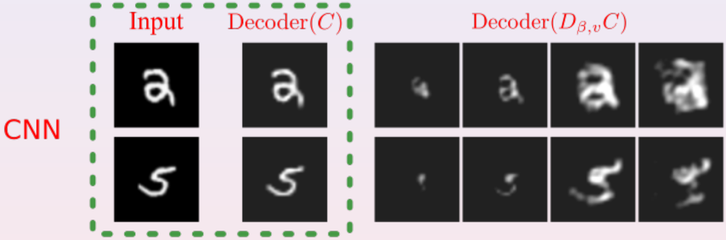
Autoencoder



Autoencoder

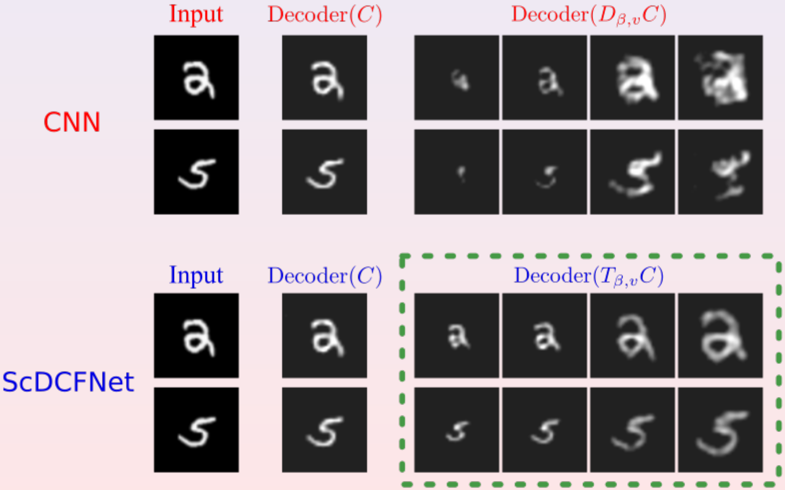


Autoencoder



Autoencoder

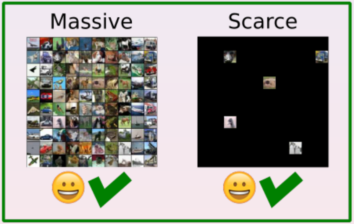
Autoencoder



Summary

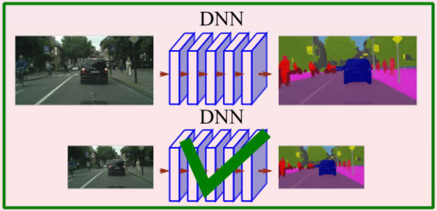
By “injecting” the modeling flavor back into deep learning, we achieved

Improved generalization

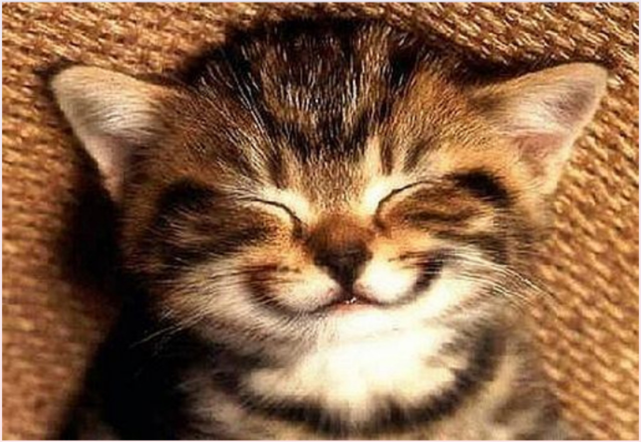


Interpretability

Symmetry preserved

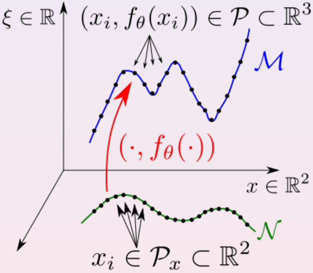


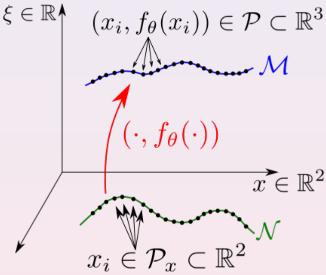
Thank you!!!



Directions for Further Development

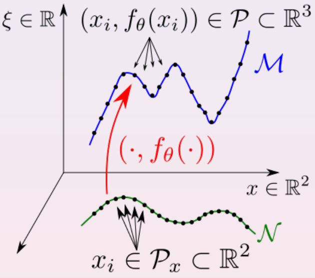
Dimension minimization vs **curvature** minimization.





Directions for Further Development

Dimension minimization vs **curvature** minimization.



Proposition

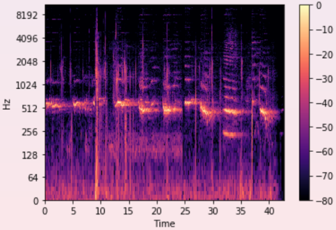
Let $\boldsymbol{\alpha} : \mathcal{M}^k \to \mathbb{R}^d$ *be the isometric embedding of* $\mathcal{M}$ *in* $\mathbb{R}^d$. *The mean curvature vector* $H(\boldsymbol{p})$ *at any* $\boldsymbol{p} \in \mathcal{M}$ *can be obtained via the following*

$$\Delta_{\mathcal{M}} \boldsymbol{\alpha}(\boldsymbol{p}) = (\Delta_{\mathcal{M}} \alpha_1(\boldsymbol{p}), \cdots, \Delta_{\mathcal{M}} \alpha_d(\boldsymbol{p})) = kH(\boldsymbol{p}).$$

Directions for Further Development

Symmetry-preserving DNNs on **complex data sources**:

Spectrogram



Multi-view data

DNN Regularizations

Most widely-used DNN regularizations typically do not take into account the geometry of the data.

- L^p weight decay, i.e., $\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_p^p$.
- DropOut.
- Data augmentation.
- ......

DNN Regularizations

Most widely-used DNN regularizations typically do not take into account the geometry of the data.

- L^p weight decay, i.e., $\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_p^p$.
- DropOut.
- Data augmentation.
- ......

Data-dependent regularizations are mostly motivated by the empirical observation that data of interest typically lie close to manifolds.

- Tangent distance algorithm
- Tangent prop algorithm
- Manifold tangent classifier
- ......

Scale-Equivariant CNNs (Joint Convolution over $\mathbb{R}^2 \times \mathcal{S}$)

Theorem (Z., Qiu, Calderbank, Sapiro, Cheng 2019)

$$x^{(1)}[x^{(0)}](u,\alpha,\lambda) = \sigma\left(\sum_{\lambda'}\int_{\mathbb{R}^2} x^{(0)}(u+u',\lambda')W^{(1)}_{\lambda',\lambda}\left(2^{-\alpha}u'\right)2^{-2\alpha}du' + b^{(1)}(\lambda)\right)$$

$$x^{(l)}[x^{(l-1)}](u,\alpha,\lambda) = \sigma\left(\sum_{\lambda'}\int_{\mathbb{R}^2}\int_{\mathbb{R}} x^{(l-1)}(u+u',\alpha+\alpha',\lambda')W^{(l)}_{\lambda',\lambda}\left(2^{-\alpha}u',\alpha'\right)\cdot 2^{-2\alpha}d\alpha' du' + b^{(l)}(\lambda)\right), \quad \forall l > 1,$$

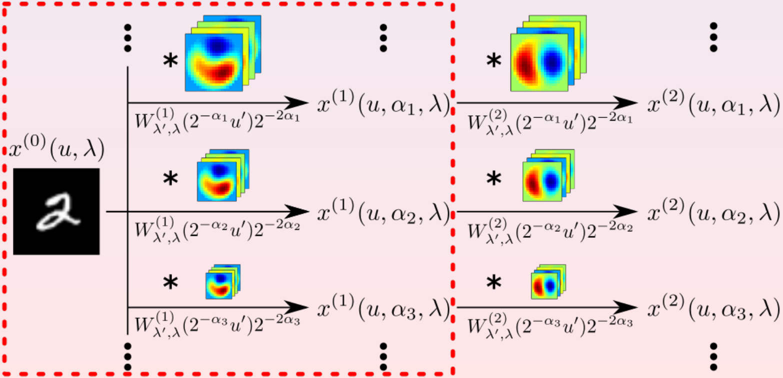
Special case: If $W^{(l)}_{\lambda',\lambda}(u,\alpha) = W^{(l)}_{\lambda',\lambda}(u)\cdot\delta(\alpha)$, then the joint convolutions reduce to only (multiscale) spatial convolutions

$$\sum_{\lambda'}\int_{\mathbb{R}^2} x^{(l-1)}(u+u',\alpha,\lambda')W^{(l)}_{\lambda',\lambda}\left(2^{-\alpha}u'\right)2^{-2\alpha}du'.$$

Scale-Equivariant CNNs (a Special Case)

$$x^{(1)}(u,\alpha,\lambda) = \sum_{\lambda'} \int_{\mathbb{R}^2} x^{(0)}(u+u',\lambda') W^{(1)}_{\lambda',\lambda}\left(2^{-\alpha}u'\right) 2^{-2\alpha} du',$$

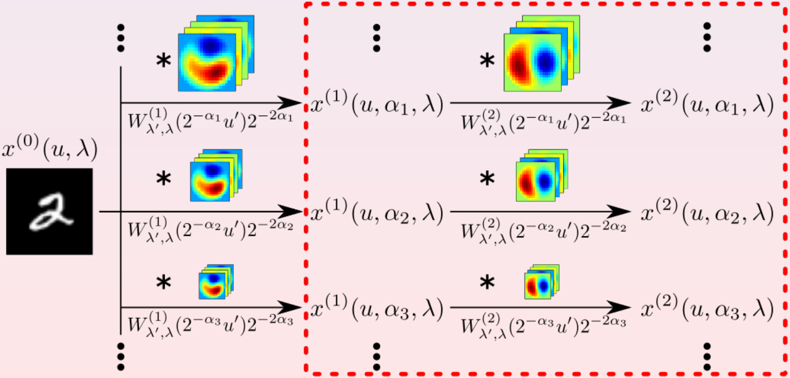
$$x^{(l)}(u,\alpha,\lambda) = \sum_{\lambda'} \int_{\mathbb{R}^2} x^{(l-1)}(u+u',\alpha,\lambda') W^{(l)}_{\lambda',\lambda}\left(2^{-\alpha}u'\right) 2^{-2\alpha} du', \ \forall l > 1.$$



Scale-Equivariant CNNs (a Special Case)

$$x^{(1)}(u,\alpha,\lambda) = \sum_{\lambda'} \int_{\mathbb{R}^2} x^{(0)}(u+u',\lambda') W^{(1)}_{\lambda',\lambda}\left(2^{-\alpha}u'\right) 2^{-2\alpha} du',$$

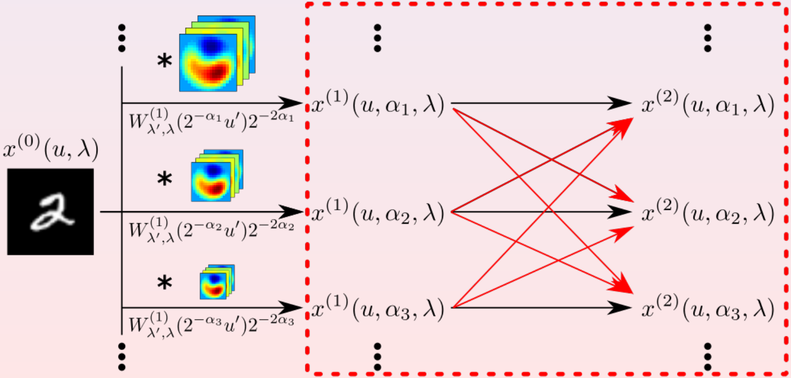
$$x^{(l)}(u,\alpha,\lambda) = \sum_{\lambda'} \int_{\mathbb{R}^2} x^{(l-1)}(u+u',\alpha,\lambda') W^{(l)}_{\lambda',\lambda}\left(2^{-\alpha}u'\right) 2^{-2\alpha} du', \ \forall l > 1.$$



Scale-Equivariant CNNs (the General Case)

$$x^{(1)}(u,\alpha,\lambda) = \sum_{\lambda'} \int_{\mathbb{R}^2} x^{(0)}(u+u',\lambda') W^{(1)}_{\lambda',\lambda}\left(2^{-\alpha}u'\right) 2^{-2\alpha} du',$$

$$x^{(l)}(u,\alpha,\lambda) = \sum_{\lambda'} \int_{\mathbb{R}^2} \int_{\mathbb{R}} x^{(l-1)}(u+u',\alpha+\alpha',\lambda') W^{(l)}_{\lambda',\lambda}\left(2^{-\alpha}u',\alpha'\right) 2^{-2\alpha} d\alpha' du'.$$

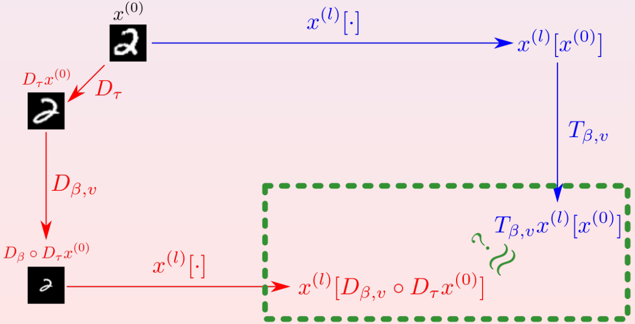


Stability of the Equivariant Representation to Input Deformation

Theorem (Z., Qiu, Calderbank, Sapiro, Cheng 2019)

In an ScDCFNet with bounded expansion coefficients $a^{(l)}_{\lambda',\lambda}$ under the Fourier-Bessel norm (which is facilitated by truncated basis decomposition), we have, for any L,

$$\left\| x^{(L)}[D_{\beta,v} \circ D_\tau x^{(0)}] - T_{\beta,v} x^{(L)}[x^{(0)}] \right\| \le 2^{\beta+1} \left(4L|\nabla\tau|_\infty + 2^{-j_L}|\tau|_\infty \right) \|x^{(0)}\|.$$



Sketch of the Proof

- If $F(u) = \sum_k a(k)\psi_{j,k}(u)$ is a smooth function on $2^j\overline{B(0,1)}$, then
$$\int |F(u)|\,du,\ \int |u|\,|\nabla F(u)|\,du,\ 2^j \int |\nabla F(u)|\,du \le \pi \|a\|_{\mathsf{FB}}.$$
- Layerwise non-expansiveness: $\|x^{(l)}[x_1] - x^{(l)}[x_2]\| \le \|x_1 - x_2\|,\ \forall x_1, x_2, l \ge 1.$
- $\left\| x^{(l)}[D_\tau x^{(l-1)}] - D_\tau x^{(l)}[x^{(l-1)}] \right\| \le 8|\nabla\tau|_\infty \|x^{(0)}\|,\ \forall l \ge 1.$
- $\left\| T_{\beta,v} x^{(l)}[D_\tau x^{(l-1)}] - T_{\beta,v} D_\tau x^{(l)}[x^{(l-1)}] \right\| \le 2^{\beta+3}|\nabla\tau|_\infty \|x^{(0)}\|,\ \forall l \ge 1.$
- $\left\| x^{(l)}[T_{\beta,v} \circ D_\tau x^{(l-1)}] - T_{\beta,v} D_\tau x^{(l)}[x^{(l-1)}] \right\| \le 2^{\beta+3}|\nabla\tau|_\infty \|x^{(0)}\|,\ \forall l \ge 1.$
- $\left\| \underline{x^{(L)}[D_{\beta,v} \circ D_\tau x^{(0)}]} - T_{\beta,v} D_\tau x^{(L)}[x^{(0)}] \right\| \le 2^{\beta+3} L|\nabla\tau|_\infty \|x^{(0)}\|.$
- $\left\| T_{\beta,v} D_\tau x^{(L)}[x^{(0)}] - \underline{T_{\beta,v} x^{(L)}[x^{(0)}]} \right\| \le 2^{\beta+1-j_L}|\tau|_\infty \|x^{(0)}\|.$
- $\left\| \underline{x^{(L)}[D_{\beta,v} \circ D_\tau x^{(0)}]} - \underline{T_{\beta,v} x^{(L)}[x^{(0)}]} \right\| \le 2^{\beta+1}\left(4L|\nabla\tau|_\infty + 2^{-j_L}|\tau|_\infty\right) \|x^{(0)}\|.$

Multiscale Image Classification

		SMNIST test accuracy (%)		SFashion test accuracy (%)	
Architectures	Ratio	$N_{\text{tr}} = 2000$	$N_{\text{tr}} = 5000$	$N_{\text{tr}} = 2000$	$N_{\text{tr}} = 5000$
CNN, $M = 32$	1.00	92.60 ± 0.17	94.86 ± 0.25	77.74 ± 0.28	82.57 ± 0.38
CNN (augment)	1.00	93.85 ± 0.15	95.51 ± 0.21	79.41 ± 0.22	83.33 ± 0.38
ScDCF, $M = 16$					
$K = 10, K_\alpha = 3$	0.84	93.75 ± 0.02	95.70 ± 0.09	78.95 ± 0.31	$\mathbf{83.51 \pm 0.71}$
$K = 8, K_\alpha = 3$	$\mathbf{0.67}$	$\mathbf{93.91 \pm 0.30}$	$\mathbf{95.71 \pm 0.10}$	79.22 ± 0.50	83.06 ± 0.32
$K = 5, K_\alpha = 3$	$\mathbf{0.42}$	93.52 ± 0.29	95.19 ± 0.13	$\mathbf{79.74 \pm 0.44}$	83.46 ± 0.69
$K = 5, K_\alpha = 2$	0.28	93.51 ± 0.30	95.35 ± 0.21	78.57 ± 0.53	82.95 ± 0.46
ScDCF, $M = 8$					
$K = 10, K_\alpha = 2$	0.14	93.68 ± 0.17	95.21 ± 0.12	79.11 ± 0.76	82.92 ± 0.68
$K = 8, K_\alpha = 2$	0.11	93.39 ± 0.25	95.25 ± 0.47	78.43 ± 0.76	83.05 ± 0.58
$K = 5, K_\alpha = 2$	0.09	93.21 ± 0.20	94.99 ± 0.12	77.97 ± 0.37	82.21 ± 0.67

Multiscale Image Classification

Architectures	Ratio	SMNIST test accuracy (%)		SFashion test accuracy (%)	
		$N_{\mathrm{tr}} = 2000$	$N_{\mathrm{tr}} = 5000$	$N_{\mathrm{tr}} = 2000$	$N_{\mathrm{tr}} = 5000$
CNN, $M = 32$	1.00	92.60 ± 0.17	94.86 ± 0.25	77.74 ± 0.28	82.57 ± 0.38
CNN (augment)	1.00	93.85 ± 0.15	95.51 ± 0.21	79.41 ± 0.22	83.33 ± 0.38
ScDCF, $M = 16$					
$K = 10, K_\alpha = 3$	0.84	93.75 ± 0.02	95.70 ± 0.09	78.95 ± 0.31	$\mathbf{83.51 \pm 0.71}$
$K = 8, K_\alpha = 3$	**0.67**	$\mathbf{93.91 \pm 0.30}$	$\mathbf{95.71 \pm 0.10}$	79.22 ± 0.50	83.06 ± 0.32
$K = 5, K_\alpha = 3$	**0.42**	93.52 ± 0.29	95.19 ± 0.13	$\mathbf{79.74 \pm 0.44}$	83.46 ± 0.69
$K = 5, K_\alpha = 2$	0.28	93.51 ± 0.30	95.35 ± 0.21	78.57 ± 0.53	82.95 ± 0.46
ScDCF, $M = 8$					
$K = 10, K_\alpha = 2$	0.14	93.68 ± 0.17	95.21 ± 0.12	79.11 ± 0.76	82.92 ± 0.68
$K = 8, K_\alpha = 2$	0.11	93.39 ± 0.25	95.25 ± 0.47	78.43 ± 0.76	83.05 ± 0.58
$K = 5, K_\alpha = 2$	0.09	93.21 ± 0.20	94.99 ± 0.12	77.97 ± 0.37	82.21 ± 0.67
ScDCF (augment)	0.67	$\mathbf{94.30 \pm 0.17}$	$\mathbf{96.01 \pm 0.23}$	$\mathbf{80.62 \pm 0.25}$	$\mathbf{83.94 \pm 0.31}$

Thank you!!!