Geometric Deep Learning

On the interplay between mathematics and deep neural networks

Jan E. Gerken

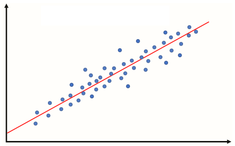
CHALMERS
UNIVERSITY OF TECHNOLOGY

Colloquium
Department of Mathematical Sciences
24 January 2021

Based on joint work with
Jimmy Aronsson, Oscar Carlsson, Hampus Linander,
Fredrik Ohlsson, Daniel Persson, Christoffer Petersson

Introduction to Deep Learning

Machine Learning



- Objective: Approximate function f from noisy observations

$$x_i \quad \text{and} \quad y_i = f(x_i) + \epsilon \qquad \epsilon \sim \mathcal{N}(0, \sigma) \quad i = 1, \ldots, N$$

- For parametric models: Look for best approximation in some function space

$$\{f_\theta : \theta \in \Theta\}$$

- Define ***loss*** $\mathcal{L}$ which measures distance between f_θ and f on training data, e.g. squared error loss

$$\mathcal{L}(f, f_\theta) = \frac{1}{N} \sum_{i=1}^{N} (f_\theta(x_i) - y_i)^2$$

- Find best approximation to f by minimizing $\mathcal{L}$ with respect to θ
- Commonly use (variant of) gradient descent:

$$\theta_{n+1} = \theta_n - \overset{\text{learning rate}}{\eta} \nabla_\theta \mathcal{L}(f, f_{\theta_n})$$

Neural networks

- A neural network $\mathcal{N}$ is a certain way to define a family of functions f_θ
- For $x \in \mathbb{R}^n$, a ***fully connected*** layer computes

$$z_j(x) = \sigma(W_j \cdot x + b_j) \qquad \text{where} \quad \overset{\textit{weights}}{W_j \in \mathbb{R}^{m \times n}}, \quad \overset{\textit{biases}}{b_j \in \mathbb{R}^m}, \quad \sigma : \mathbb{R}^m \to \mathbb{R}^m \tag{$\star$}$$

- σ is a (simple) non-linear function that acts component-wise, often ***ReLU***

$$\mathrm{ReLU}(x) = \max(x, 0)$$

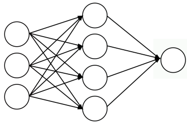
- The neural network iterates z

$$\mathcal{N}_\theta(x) = z_L \circ z_{L-1} \circ \cdots \circ z_1(x) \tag{$\star\star$}$$

  with the parameters

$$\theta = \{W_1, \ldots, W_L, b_1, \ldots, b_L\}$$

- Choose output dimension to match task, e.g. $\mathcal{N}_\theta(x) \in \mathbb{R}$ for binary classifier
- Compute gradient of loss w.r.t. W_j, b_j using ***backpropagation***
- Name stems from similarity of ($\star$) to neurons and visualization of ($\star\star$) as graph



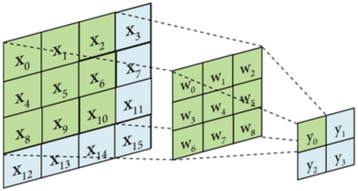
Convolutional neural networks (CNNs)

[Fukushima 1980]

- Used mostly for input images: RGB pixel values on a grid $\mathcal{G} = ([0, w] \times [0, h]) \cap \mathbb{Z}^2$
- Interpret input as function $f : \mathcal{G} \to \mathbb{R}^3$
- Convolutional layer

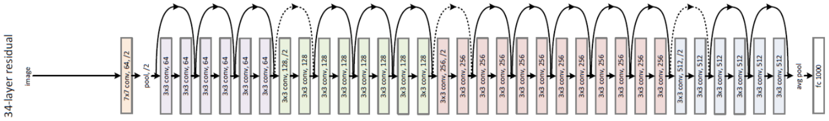
$$z_j(x) = \sigma\big((\kappa_j * f)(x) + b_j\big) = \sigma\Big(\sum_{y \in \mathcal{G}} \overset{\textit{kernel}}{\kappa_j}(y - x) \cdot f(y) + b_j\Big), \quad \text{where} \quad \kappa_j : \mathcal{G} \to \mathbb{R}^{m \times n}$$

- Kernel has finite support, e.g. 3×3 or 5×5
- Interpret convolution as sliding filter over image



- Effectively: Fully connected layer with constraints on W
- CNNs stack many convolutional layers
- Higher-dimensional versions and versions on graphs exist

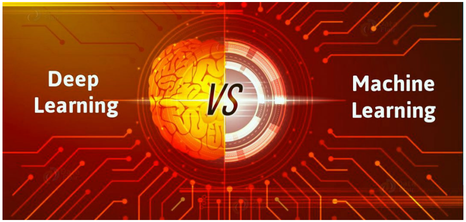
Deep learning



- Stack many (usually ~ 5 – 100) NN layers to build ***deep*** neural networks
- Typically many parameters (largest model has more than 1 trillion parameters)
- Typically trained on large datasets (e.g. TBs of text data, hundreds of millions of images)
- Need various tricks to make training stable and efficient
- Use graphical processing units (GPUs) and computing clusters to run training
- Hugely successful, at the heart of the AI boom:
  - Computer Vision
  - Natural Language Processing
  - Reinforcement Learning
  - Protein Folding

  ⋮

Characteristics of deep learning



- Networks are highly overparametrized
  $\Rightarrow$ Overfitting is often a problem
- New in DL: Learn features instead of craft them by hand
  $\Rightarrow$ This made DL so successful!
- Have to solve non-convex optimization problem
  $\Rightarrow$ Optimization can get stuck in local minima
- DL requires huge (usually labeled) datasets
  $\Rightarrow$ Try to learn from unlabeled datasets (un- or self-supervised learning)
- Training large networks requires big computational resources
  $\Rightarrow$ Develop dedicated hardware to make training more efficient
- Deep neural networks are not interpretable
  $\Rightarrow$ Develop attribution methods (***explainable AI***)

Mathematics of deep learning

Many open questions in the mathematics of deep learning:

- Which functions can be approximated by deep NN?

  Universal approximation theorem: [Cybenko, 1989] [Hornik, 1991]

  Any continuous function can be approximated arbitrarily well by a two-layer, infinitely wide neural network
- Rigorous results exist only for very special edge cases
- Finite-width networks poorly understood
- Training procedure poorly understood
- Why does overparametrization and non-convex loss not hurt performance more?

Connection to renormalization group flow in quantum field theory (QFT): [Roberts et. al. 2021]

- Use QFT techniques to compute probability distributions of features in NN layers
- Perturbatively expand around infinite-width limit
- Interpret evolution of features through network as renormalization group flow
- Compute fixed points of flow for different nonlinearities

Another interesting direction: Consider low-dimensional ***data manifold*** geometrically

Geometric Deep Learning

Introduction to geometric deep learning

[Bronstein et. al. 2017]

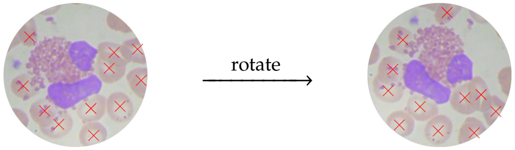
- Suppose the data has some symmetry,

$$x \in \mathcal{M}_{\text{data}} \qquad \Longrightarrow \qquad T_g(x) \in \mathcal{M}_{\text{data}}, \quad g \in G$$

  and the labels satisfy

$$y(T_g(x)) = T'_g(y(x)) \tag{$\star$}$$

- Example: Pictures of cells



- According to ($\star$), the neural network has to learn an ***equivariant*** function

Geometric deep learning

Build symmetry properties of the data into the neural network architecture, i.e. use an architecture which satisfies

$$\mathcal{N}(T_g(x)) = T'_g(\mathcal{N}(x))$$

by construction.

Introduction to geometric deep learning

[Bronstein et. al. 2017]

Advantages of geometric deep learning

- Gives systematic way to construct NN architectures (inspired by physics)
- Symmetries of the problem are respected exactly
- Higher performance on symmetric problems
- Less training data required
- Less parameters required (***weight sharing***)

But

- Network architectures usually more complex
- Many competing equivariant architectures available

Convolutional neural networks are equivariant

CNNs are the most prominent example of equivariant architectures:

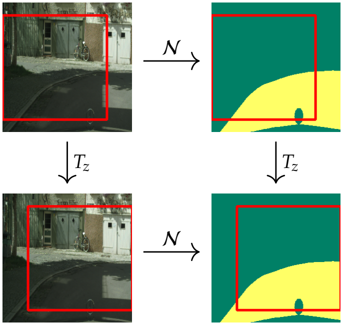
$$(\kappa * f)(x) = \sum_{y \in \mathbb{Z}^2} \kappa(y - x) \cdot f(y)$$

is equivariant with respect to translations

$$(\kappa * T_z(f)) = T_z(\kappa * f) \qquad \text{where} \quad (T_z f)(x) = f(x + z)$$

Proven to be extremely beneficial in computer vision

- Intuitively: Can build same features, no matter where object is located
- E.g. Semantic segmentation and object detection are inherently equivariant



Group convolutions

[Cohen et. al. 2018]

- Ordinary convolutional layers can easily be generalized to arbitrary symmetry groups

$$(\kappa * f)(x) = \sum_{y \in \mathbb{Z}^2} \kappa(y - x) f(y)$$

suggests to replace x by a group element $g \in G$

$$(\kappa * f)(g) = \sum_{y \in \mathbb{Z}^2} \kappa(T_{g^{-1}}(y)) f(y) \qquad (\star)$$

for the next layer, we have a feature map on G, so

$$(\kappa * f)(g) = \sum_{h \in G} \kappa(g^{-1}h) f(h)$$

- This architecture is equivariant. By summing over G in the last layer, reach invariance.
- Note: Since nonlinearities act pointwise, transformation properties are preserved.
- For continuous groups, replace sum over G by integral with respect to Haar measure
- For the layer $(\star)$, we can replace $\mathbb{Z}^2$ by any homogeneous space G/K.
- Implementations exist for small finite groups like C_4 or D_4

Non-trivial feature representations

- Let f and $f' = \kappa * f$ transform in a non-trivial representation of G

$$[\pi_1(g)f](x) = \rho_1(g)f(\sigma_1^{-1}(g)x)$$
$$[\pi_2(g)f'](x) = \rho_2(g)f'(\sigma_2^{-1}(g)x)$$

- If G acts regularly on $x \in \mathcal{X}$,

$$(\kappa * f)(x) = \int_G \mathrm{d}g\, \rho_2(g)\, \kappa\left(\sigma_2^{-1}(g)x\right)\, \rho_1^{-1}(g)\, f(\sigma_1(g)\underset{\text{reference point}}{x_0})$$

  is an equivariant convolution
- For non-regular group actions, have non-trivial constraint on κ in terms of $\rho_{1,2}$
- For instance, if $G = N \rtimes H$ the convolution becomes

$$[\kappa * f](x) = \int_N \mathrm{d}n\, \rho_2(n)\, \kappa\left(\sigma_2^{-1}(n)x\right)\, \rho_1^{-1}(n)\, f(\sigma_1(n)x_0)$$

  and the kernel satisfies the constraint

$$\kappa(\sigma_2(h)x) = \rho_2(h)\kappa(x)\rho_1^{-1}(h), \qquad \forall\, h \in H$$

  This case is known as ***steerable*** GCNNs [Weiler et. al. 2018]

Spherical Convolutions

[Cohen et. al. 2018]

- For $G = \mathrm{SO}(3)$, $H = \mathrm{SO}(2)$, so $G/H = S^2$, explicit implementations are available
- Consider data $f : S^2 \to \mathbb{R}^n$ on the sphere, e.g. pictures from fisheye cameras
- In the first layer, the group convolution becomes

$$(\kappa * f)(R) = \int_{S^2} \mathrm{d}x\, \kappa(R^{-1}x) f(x)$$

- In subsequent layers, the group convolution is

$$(\psi * g)(S) = \int_{\mathrm{SO}(3)} \mathrm{d}R\, \psi(R^{-1}S) g(R)$$

- Both convolutions are in practice computed in the Fourier domain: On S^2 use spherical harmonics $Y^l_m(x)$, on SO(3) use Wigner matrices $\mathcal{D}^l_{mn}(R)$, leverage FFT
- The convolutions become pointwise multiplications

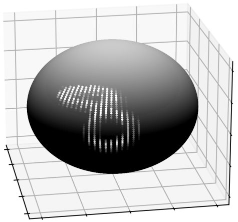
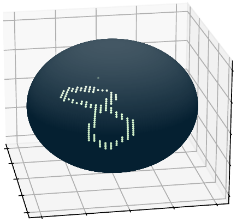
$$(\kappa * f)^l_{mn} = \kappa^l_m f^l_n \qquad (\psi * g)^l_{mn} = \sum_{k=-l}^{l} f^l_{mk} \psi^l_{kn}$$

- After final convolution integrate over SO(3) for invariance (output in $\mathbb{R}^c$) and over SO(2) for equivariance (output on S^2)

Equivariant networks vs. data augmentation

[Gerken et. al. to appear]

- Traditional way to learn symmetries in data: ***data augmentation***
- Add symmetry orbit of samples to dataset ⇒ increases amount of training data
- For instance, consider semantic segmentation of MNIST projected onto the sphere and randomly rotated:

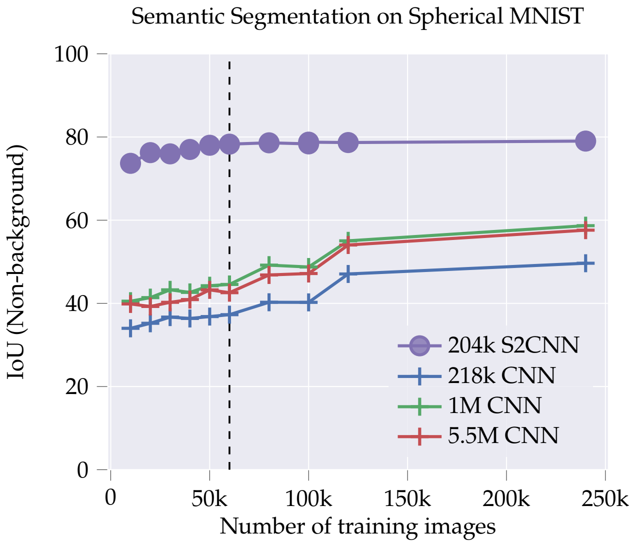



- For data augmentation, add rotated versions of projected MNIST digits

Equivariant networks vs. data augmentation

[Gerken et. al. to appear]

- In this case, performance of equivariant networks cannot be reached by data augmentation



Applications of equivariant architectures

Equivariant architectures have been applied successfully in a wide range of applications, e.g.

- 3D shape recognition [Weiler et. al. 2018] [Thomas et. al. 2018]
- Cosmology (CMB, galaxy counting, gravitational lensing) [Perraudin et. al. 2018]
- Segmentation of satellite images of deforestation [Irvin et. al. 2020]
- Protein structure classification [Weiler et. al. 2018]
- Prediction of molecular properties [Kondor et. al. 2018] [Unke et. al. 2021]
- Medical image segmentation [Worall et. al. 2018] [Winkels et. al. 2018] [Müller et. al. 2021] [Thomas et. al. 2021]

(a highly biased and incomplete list)

Gauge networks

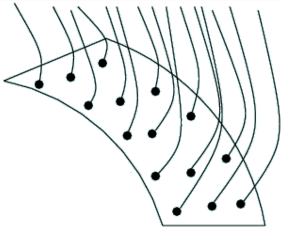
[Cheng et. al. 2019]

- In physics: Gauge theory = theory with local symmetry, i.e. $g = g(x)$, $g \in G$
- Basis for e.g. standard model of particle physics
- General relativity is diffeomorphism invariant i.e. can choose coordinates at every point
- Gauge networks: neural networks which are invariant with respect to local changes of coordinates

Can formulate gauge networks in terms of fiber bundles

Fiber bundles (reminder)

- A bundle consists of total space E, base $\mathcal{M}$ and projection $\pi : E \to \mathcal{M}$
- Fibers are given by $E_x = \pi^{-1}(x)$
- A section of E is a map $\sigma : \mathcal{M} \to E$ such that $\pi \circ \sigma = \mathrm{id}$



Gauge networks

[Cheng et. al. 2019]
[Gerken et. al. 2021]

Fiber bundles (reminder)

- A bundle consists of total space E, base $\mathcal{M}$ and projection $\pi : E \to \mathcal{M}$
- Fibers are given by $E_x = \pi^{-1}(x)$
- A section of E is a map $\sigma : \mathcal{M} \to E$ such that $\pi \circ \sigma = \mathrm{id}$

- Formulate gauge symmetry in terms of principle G-bundle P over input manifold $\mathcal{M}$: Bundle P with regular right action of G on P
$$\triangleleft : P \times G \to P\,, \quad \text{satisfying} \quad \pi_P(p \triangleleft g) = \pi_P(p)$$
- A ***gauge*** is a section of P
- In general relativity: P is frame bundle, section of P is choice of basis in $\mathcal{T}\mathcal{M}$
- Let V be a vector space on which G acts from the left via representation ρ
$$g \triangleright v = \rho(g)v$$
- Define equivalence relation on $P \times V$ by
$$(p, v) \sim_\rho (p \triangleleft g, g^{-1} \triangleright v)\,, \quad g \in G$$
- Feature maps are sections of associated bundle $P \times_\rho V = P \times V / \sim_\rho$ with projection
$$\pi_\rho([p, v]) = \pi_P(p)$$

Gauge networks

[Cheng et. al. 2019]
[Gerken et. al. 2021]

Can construct explicit coordinate independent convolution by using

- Exponential map
- Parallel transport by means of connection on P

For a section s of $P \times_\rho V$, the convolution is given by

$$(\Phi s)(x) = \int_{B_R} \mathrm{d}X \, \kappa(x, X) s|_{\exp_x X}(x) \sqrt{\det(g_{\mathcal{M}})}$$

with a kernel

$$\kappa : \mathcal{M} \times \mathcal{T}\mathcal{M} \to \mathrm{Hom}(E_\rho, E_\eta)\,,$$

satisfiying

$$\kappa(x, X') = \eta(k^{-1})\kappa(x, X)\rho(k) \qquad \text{where} \quad X' = k \triangleright X$$

- Few concrete implementations of these concepts yet [Cohen et. al. 2019] [de Haan et. al. 2020]
- Gauge equivariant convolutions also exist for *internal* gauge symmetries, used for lattice field theory computations [Favoni et. al. 2020] [Luo et. al. 2020]

Gauge networks

[Cheng et. al. 2019]
[Gerken et. al. 2021]

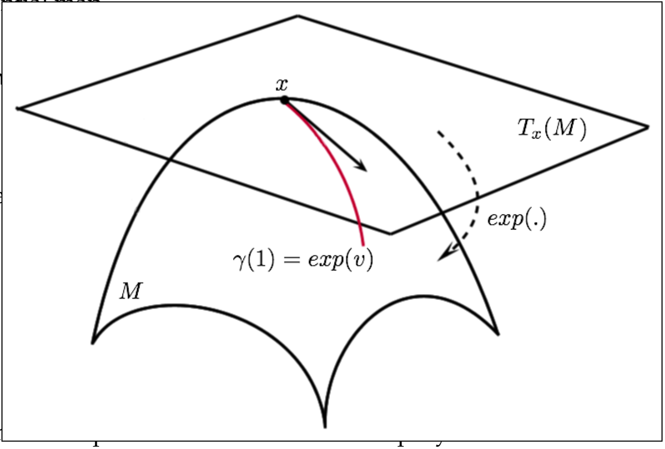
Can construct explicit coordinate independent convolution by using

- Exponential map
- Parallel

For a section

with a kerne

satisfiying



- Few co

[Cohen et. al. 2019]
[de Haan et. al. 2020]

- Gauge equivariant convolutions also exist for *internal* gauge symmetries, used for lattice field theory computations

[Favoni et. al. 2020]
[Luo et. al. 2020]

Gauge networks

[Cheng et. al. 2019]
[Gerken et. al. 2021]

Can construct explicit coordinate independent convolution by using

- Exponential map
- Parallel transport by means of connection on P

For a section s of $P \times_\rho V$, the convolution is given by

$$(\Phi s)(x) = \int_{B_R} \mathrm{d}X \, \kappa(x, X) s|_{\exp_x X}(x) \sqrt{\det(g_{\mathcal{M}})}$$

with a kernel

$$\kappa : \mathcal{M} \times \mathcal{T}\mathcal{M} \to \mathrm{Hom}(E_\rho, E_\eta)\,,$$

satisfiying

$$\kappa(x, X') = \eta(k^{-1})\kappa(x, X)\rho(k) \qquad \text{where} \quad X' = k \triangleright X$$

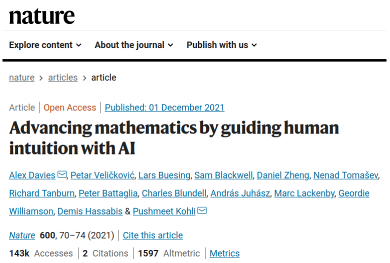
- Few concrete implementations of these concepts yet [Cohen et. al. 2019] [de Haan et. al. 2020]
- Gauge equivariant convolutions also exist for *internal* gauge symmetries, used for lattice field theory computations [Favoni et. al. 2020] [Luo et. al. 2020]

Mathematics from Deep Learning

New mathematics from deep learning

Deep learning starts to influence (pure) mathematics

- Long history in mathematics of using data to guide intuition
- In math, deep learning can be useful to
  - Find counterexamples
  - Accelerate computation
  - Generate symbolic solutions
  - Detect structure in mathematical objects
- Last year, nature paper by Deep Mind got a lot of attention:



nature

Explore content ⌄ About the journal ⌄ Publish with us ⌄

nature › articles › article

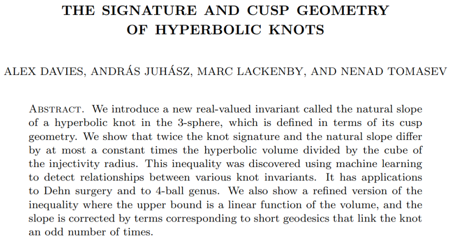
Article | Open Access | Published: 01 December 2021

Advancing mathematics by guiding human intuition with AI

Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, Marc Lackenby, Geordie Williamson, Demis Hassabis & Pushmeet Kohli

Nature **600**, 70–74 (2021) | Cite this article

143k Accesses | **2** Citations | **1597** Altmetric | Metrics



THE SIGNATURE AND CUSP GEOMETRY OF HYPERBOLIC KNOTS

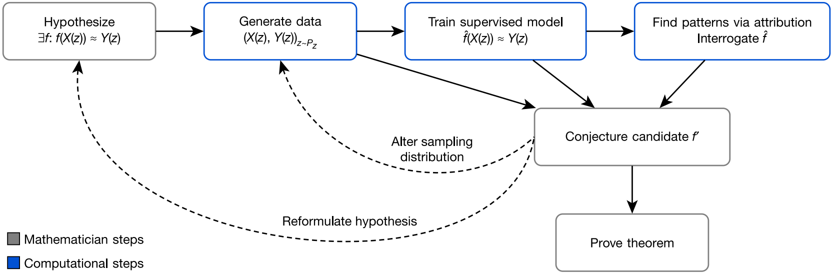
ALEX DAVIES, ANDRÁS JUHÁSZ, MARC LACKENBY, AND NENAD TOMASEV

Abstract. We introduce a new real-valued invariant called the natural slope of a hyperbolic knot in the 3-sphere, which is defined in terms of its cusp geometry. We show that twice the knot signature and the natural slope differ by at most a constant times the hyperbolic volume divided by the cube of the injectivity radius. This inequality was discovered using machine learning to detect relationships between various knot invariants. It has applications to Dehn surgery and to 4-ball genus. We also show a refined version of the inequality where the upper bound is a linear function of the volume, and the slope is corrected by terms corresponding to short geodesics that link the knot an odd number of times.

New mathematics from deep learning

[Davies et. al. 2021]

Process proposed in Davies et. al. 2021:



Hypothesize
$\exists f$: $f(X(z)) \approx Y(z)$
Generate data
$(X(z), Y(z))_{z \sim P_Z}$
Train supervised model
$\hat{f}(X(z)) \approx Y(z)$
Find patterns via attribution
Interrogate $\hat{f}$
Conjecture candidate f'
Alter sampling distribution
Reformulate hypothesis
Prove theorem
Mathematician steps
Computational steps

Example:

- Let z be a complex polyhedron
- Let $X(z) \in \mathbb{Z}^2 \times \mathbb{R}^2$ be the number of edges and vertices of z as well as volume and surface area
- Let $Y(z)$ be the number of faces of z
- Euler's formula $\Rightarrow X(z) \cdot (-1, 1, 0, 0) + 2 = Y(z)$
- Note: Deep learning approach also works for highly non-linear functions f

Conclusion

Summary

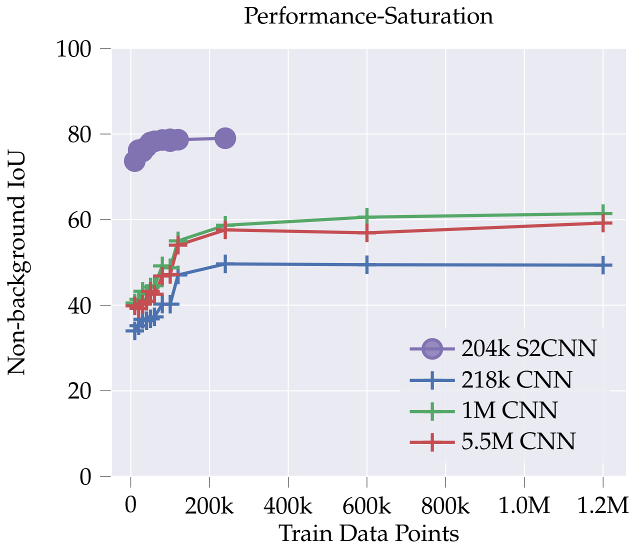
- Deep neural networks have led to remarkable results in many application domains but are poorly understood theoretically
- Geometric deep learning provides a systematic framework to construct neural network architectures
- Group equivariant convolutions show promising results in comparison to traditional methods
- New areas of mathematics enter the study of neural networks with this field
- Deep learning starts to be an interesting tool also for (pure) mathematicians

Future directions

- Make geometric deep learning into a standard tool for deep learning
- Understand mathematical structure of geometric deep learning better
- Search for an effective description of neural networks
- Incorporate deep learning in traditional mathematical research

Appendix

Performance saturation for non-equivariant runs


Performance-Saturation
Non-background IoU
Train Data Points
200k
400k
600k
800k
1.0M
1.2M
204k S2CNN
218k CNN
1M CNN
5.5M CNN