On the Inductive Bias of Neural Tangent Kernels

Alberto Bietti Julien Mairal

Inria Grenoble

GIPSA-lab. November 7, 2019.

Generalization and Inductive Bias

Goal of supervised learning:
Given $(x_1, y_1), \ldots, (x_N, y_N) \sim \mathcal{D}$, find $f \in \mathcal{F}$ with small *expected loss*:

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(y_i, f(x_i))]$$

Generalization and Inductive Bias

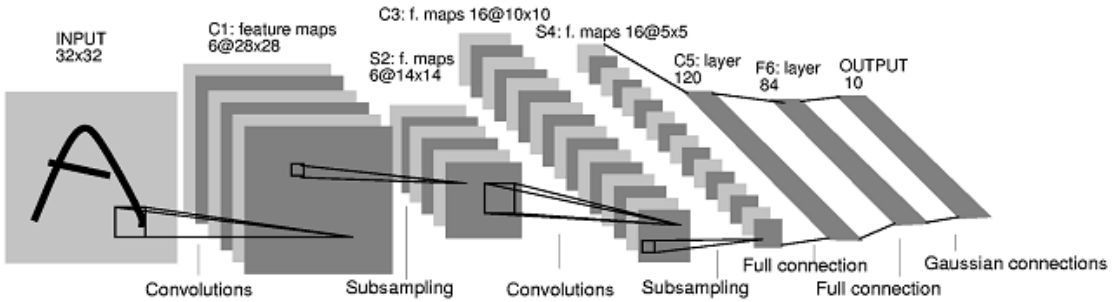
Goal of supervised learning:
Given $(x_1, y_1), \ldots, (x_N, y_N) \sim \mathcal{D}$, find $f \in \mathcal{F}$ with small *expected loss*:

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(y_i, f(x_i))]$$

Inductive bias:

- Cannot generalize if $\mathcal{F}$ too large
  - all functions: "no free lunch theorem"
  - Lipschitz functions: "curse of dimensionality" (need $O(1/\epsilon^p)$ samples)
- Need for prior knowledge or *inductive bias* in the choice of $\mathcal{F}$

Generalization in deep learning



$$f_\theta(x) = W^n \sigma(W^{n-1} \cdots \sigma(W^1 x) \cdots)$$

- Heavily over-parameterized (millions/billions of parameters)
- Trained to fit the training data (almost) perfectly
- Often specific choices of architectures (convolutions, recurrent, attention)

Generalization in deep learning: complexity

What controls the "complexity" of the hypothesis class $\mathcal{F}$?

- Number of parameters is a poor measure of complexity!

For valid generalization, the size of the weights is more important than the size of the network

Peter L. Bartlett
Department of Systems Engineering
Research School of Information Sciences and Engineering
Australian National University
Canberra, 0200 Australia
Peter.Bartlett@anu.edu.au

UNDERSTANDING DEEP LEARNING REQUIRES RETHINKING GENERALIZATION

Chiyuan Zhang*
Massachusetts Institute of Technology
chiyuan@mit.edu

Samy Bengio
Google Brain
bengio@google.com

Moritz Hardt
Google Brain
mrtz@google.com

Benjamin Recht†
University of California, Berkeley
brecht@berkeley.edu

Oriol Vinyals
Google DeepMind
vinyals@google.com

Generalization in deep learning: complexity

What controls the "complexity" of the hypothesis class $\mathcal{F}$?

- Number of parameters is a poor measure of complexity!
- Some kind of norm of the weights? (*e.g.*, spectral norms)
  - often too large in practice

From Bartlett et al. (2017):

$$R_{\mathcal{A}} := \left(\prod_{i=1}^{L} \rho_i \|A_i\|_\sigma \right) \left(\sum_{i=1}^{L} \frac{\|A_i - M_i\|_1^{2/3}}{\|A_i\|_\sigma^{2/3}} \right)^{3/2}$$

Generalization in deep learning: complexity

What controls the "complexity" of the hypothesis class $\mathcal{F}$?

- Number of parameters is a poor measure of complexity!
- Some kind of norm of the weights? (*e.g.*, spectral norms)
  - often too large in practice
- Norm of the corresponding function in a functional space? $\|f_\theta\|_{\mathcal{H}}$ instead of $\|\theta\|$
  - Question: what is an appropriate functional space?

Generalization in deep learning: role of optimization

- Optimization of deep models often framed as the (non-convex) problem:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f_\theta(x_i)) \tag{1}$$

Generalization in deep learning: role of optimization

- Optimization of deep models often framed as the (non-convex) problem:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f_\theta(x_i)) \tag{1}$$

- **But**: for over-parameterized model/deep networks, many possible solutions with ≈ 0 loss! (interpolating solution $\forall i, f_\theta(x_i) = y_i$)

Generalization in deep learning: role of optimization

- Optimization of deep models often framed as the (non-convex) problem:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f_\theta(x_i)) \tag{1}$$

- **But**: for over-parameterized model/deep networks, many possible solutions with ≈ 0 loss! (interpolating solution $\forall i, f_\theta(x_i) = y_i$)
- $\implies$ **optimization algorithm** selects the better, "simpler" models

Generalization in deep learning: role of optimization

- Optimization of deep models often framed as the (non-convex) problem:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f_\theta(x_i)) \tag{1}$$

- **But**: for over-parameterized model/deep networks, many possible solutions with ≈ 0 loss! (interpolating solution $\forall i, f_\theta(x_i) = y_i$)
- $\implies$ **optimization algorithm** selects the better, "simpler" models
- **Implicit bias** of (stochastic) gradient descent: for which norm $\|\cdot\|$ does (S)GD lead to the solution of

$$\begin{aligned} \min \quad & \|\theta\| \quad \text{or} \quad \|f_\theta\| \\ s.t. \quad & f_\theta(x_i) = y_i, \quad i = 1, \ldots, N \end{aligned}$$

Kernels for deep learning

Kernels?

- Map data x to high-dimensional space, $\Phi(x) \in \mathcal{H}$ ($\mathcal{H}$: "RKHS")
- Non-linear $f \in \mathcal{H}$ takes linear form: $f(x) = \langle f, \Phi(x) \rangle$
- Learning with a positive definite kernel $K(x, x') = \langle \Phi(x), \Phi(x') \rangle$
- Convex optimization problem (tractable)
- Statistical and approximation properties well understood
- Costly (kernel matrix of size N^2) but approximations are possible

Kernels for deep learning

Kernels from deep architectures

- Can construct *hierarchical kernels* following a given architecture
  - *e.g.*, convolutional kernels (Mairal, 2016)
- Arise naturally from infinite-width networks at initialization
- Recent work shows that similar kernels arise for *optimization* of over-parameterized networks: **neural tangent kernels** (NTKs, Jacot et al., 2018)
- Leads to **tractable** functional spaces to study deep networks

Kernels for deep learning

Kernels from deep architectures

- Can construct *hierarchical kernels* following a given architecture
  - *e.g.*, convolutional kernels (Mairal, 2016)
- Arise naturally from infinite-width networks at initialization
- Recent work shows that similar kernels arise for *optimization* of over-parameterized networks: **neural tangent kernels** (NTKs, Jacot et al., 2018)
- Leads to **tractable** functional spaces to study deep networks

This work:

- What are properties of the functional space for NTKs? (approximation, smoothness, stability to deformations for CNNs)
- How do they compare to previous deep kernels?

Outline

Warmup: random features

Two-layer ReLU network, with fixed first layer (a.k.a. random features):

$$f_\theta(x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \sigma(w_i^\top x),$$

- $w_i \sim N(0, I)$ fixed
- $\theta = v \in \mathbb{R}^m$ optimized ($1/\sqrt{m}$ scaling is standard in initializations of v_i)
- $\sigma(u) = \max(0, u)$ (ReLU)
- Optimize loss with gradient descent on θ (convex problem here)

Warmup: random features

- Same as linear model $f_\theta(x) = \langle \theta, \phi(x) \rangle$ with random features

$$\phi(x) = \frac{1}{\sqrt{m}}(\sigma(w_1^\top x), \ldots, \sigma(w_m^\top x))$$

- Limiting kernel:

$$\begin{aligned} K_1(x, x') &= \lim_{m \to \infty} \langle \phi(x), \phi(x') \rangle \\ &= \mathbb{E}_{w \sim N(0,I)}[\sigma(w^\top x)\sigma(w^\top x')] \\ &= \|x\|\|x'\|\kappa_1\left(\frac{x^\top x'}{\|x\|\|x'\|}\right), \end{aligned}$$

with $\kappa_1(u) = \frac{1}{\pi}\left(u \cdot (\pi - \arccos(u)) + \sqrt{1-u^2}\right)$.

Warmup: random features

- Gradient descent implicitly acts as an ℓ^2 regularizer on θ
- For $m \to \infty$ and square loss, GD is implicitly solving a **regularized** version of the problem

$$\min_{f \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i)),$$

  - $t \to \infty$ leads to **minimum-norm** interpolating solution
  - smaller t (**early stopping**) has a stronger regularization effect
- For finite but large enough m, similar generalization properties as full kernel (Bach, 2017b; Rudi and Rosasco, 2017).

Training both layers: lazy training

$$f_\theta(x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \sigma(w_i^\top x),$$

What happens when training both layers? (or even multiple layers)

- $\theta = (v, w)$, initialization θ_0, i.i.d. $N(0, 1)$ params
- If m is (very) large, many results from past year essentially show that model stays close to **linearization** throughout training:

$$f_\theta(x) \approx f_{\theta_0}(x) + \langle \theta - \theta_0, \nabla_\theta f_\theta(x)|_{\theta=\theta_0} \rangle.$$

- weights move very little ("lazy training", Chizat et al., 2019)
  - tldr: 2nd order Taylor term remains negligible due to $1/\sqrt{m}$ scaling
  - not really what happens in practice... (for later)

Training both layers: neural tangent kernel (NTK)

$$f_\theta(x) \approx f_{\theta_0}(x) + \langle \theta - \theta_0, \nabla_\theta f_\theta(x)|_{\theta=\theta_0} \rangle.$$

- When $m \to \infty$, $f_{\theta_0}(x) \approx 0$
- Again, linear model on random features $\psi(x) = \nabla_\theta f_\theta(x)|_{\theta=\theta_0}$
- Limiting kernel:

$$\begin{aligned} K_{NTK}(x, x') &= \lim_{m\to\infty} \langle \psi(x), \psi(x') \rangle \\ &= (x^\top x')\, \mathbb{E}_{w\sim\mathcal{N}(0,I)}[1\{w^\top x \geq 0\} 1\{w^\top x' \geq 0\}] \\ &\quad + \mathbb{E}_{w\sim\mathcal{N}(0,I)}[(w^\top x)_+ (w^\top x')_+] \\ &= \|x\|\|x'\| \kappa_{NTK}\left(\frac{x^\top x'}{\|x\|\|x'\|}\right), \end{aligned}$$

with $\kappa_{NTK}(u) = u\kappa_0(u) + \kappa_1(u)$ ($\kappa_{1/0}$ are arc-cosine kernels for different activations, ReLU and threshold)

Outline

K_1 vs K_{NTK}

Training top layer vs both layers?

- Both kernels are *homogeneous dot-product kernels* of the form

$$K(x, x') = \|x\|\|x'\|\kappa\left(\frac{x^\top x'}{\|x\|\|x'\|}\right).$$

- For RF, $\kappa(u) = \kappa_1(u)$
- For NTK, $\kappa(u) = u\kappa_0(u) + \kappa_1(u)$
- How do the RKHSs differ?
- $\implies$ spectral description through Mercer decomposition

Mercer's theorem (*e.g.*, Schölkopf and Smola, 2001)

Define integral operator: $T_K f(x) = \int K(x, y) d\nu(y)$ for some $d\nu$

Mercer's theorem (*e.g.*, Schölkopf and Smola, 2001)

Define integral operator: $T_K f(x) = \int K(x, y) d\nu(y)$ for some $d\nu$

Theorem (Mercer)

If $\mathcal{X}$ is compact and K is continuous, then there exists an orthonormal basis $(\phi_i)_{i \in \mathbb{N}}$ of $L^2(\mathcal{X}, d\nu)$, and non-negative eigenvalues $(\mu_i)_{i \in \mathbb{N}}$ such that

$$T_K \phi_i = \mu_i \phi_i, \quad i = 1, 2, \ldots$$

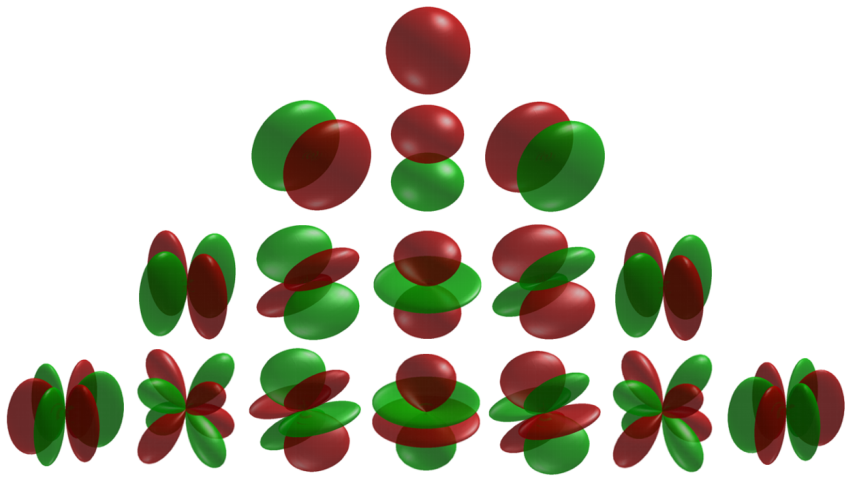
$$K(x, y) = \sum_{i=1}^{\infty} \mu_i \phi_i(x) \phi_i(y)$$

$$\mathcal{H} = \left\{ f = \sum_{i : \mu_i \neq 0} a_i \phi_i : a_i \in \mathbb{R}, \sum_{i : \mu_i \neq 0} \frac{a_i^2}{\mu_i} < \infty \right\},$$

where the norm of $f = \sum_{i:\mu_i \neq 0} a_i \phi_i$ is given by $\|f\|_{\mathcal{H}}^2 = \sum_{i:\mu_i \neq 0} \frac{a_i^2}{\mu_i}$.

Dot-product kernels on the sphere

- When $x, s' \in \mathbb{S}^{p-1}$, K_1 and K_{NTK} are dot-product kernels $K(x, x') = \kappa(x^\top x')$
- Such kernels are invariant to rotations (orthogonal matrices)
- Integral operator is diagonalized in the basis of **spherical harmonics**



Dot-product kernels on the sphere

We have

$$\kappa(x^\top y) = \sum_{k=0}^{\infty} \mu_k \sum_{j=1}^{N(p,k)} Y_{k,j}(x) Y_{k,j}(y) = \sum_{k=0}^{\infty} \mu_k N(p,k) P_k(x^\top y)$$

- $Y_{k,j}$, $j = 1, \ldots, N(p,k)$ are spherical harmonic polynomials of degree k
- P_k is the corresponding Legendre polynomials of degree k
- μ_k eigenvalue associated to spherical harmonics of degree k, given by

$$\mu_k = \frac{\omega_{p-2}}{\omega_{p-1}} \int_{-1}^{1} \kappa(t) P_k(t) (1-t^2)^{(p-3)/2} dt.$$

Decay of μ_k

- "Size" of the RKHS governed by the eigenvalue decay
- For κ_0 and κ_1, the μ_k follow from decompositions of positively homogeneous activations by Bach (2017a); slower decay for κ_0
- For κ_{NTK}, can use recurrence on Legendre polynomials: slower decay similar to $\kappa_0 \implies$ **larger** RKHS compared to κ_1

Decay of μ_k

- "Size" of the RKHS governed by the eigenvalue decay
- For κ_0 and κ_1, the μ_k follow from decompositions of positively homogeneous activations by Bach (2017a); slower decay for κ_0
- For κ_{NTK}, can use recurrence on Legendre polynomials: slower decay similar to $\kappa_0 \implies$ **larger** RKHS compared to κ_1
- We have (for fixed dimension p)

	κ_1	κ_{NTK}
μ_k	$O(k^{-p-2})$	$O(k^{-p})$

Approximation properties

- As in Fourier, decay of coefficients $\leftrightarrow$ regularity
- Leads to approximation properties similar to Sobolev spaces, following Bach (2017a)

Approximation properties

- As in Fourier, decay of coefficients $\leftrightarrow$ regularity
- Leads to approximation properties similar to Sobolev spaces, following Bach (2017a)
- For κ_1, $p/2$ is replaced by $p/2+1$ (need more regularity)

Corollary (Sufficient condition for $f \in \mathcal{H}_{NTK}$)

Let $f : \mathbb{S}^{p-1} \to \mathbb{R}$ be an even function such that all i-th order derivatives exist and are bounded by η for $0 \leq i \leq s$, with $s \geq p/2$. Then $f \in \mathcal{H}$ with $\|f\|_{\mathcal{H}} \leq C(p)\eta$, where $C(p)$ is a constant that only depends on p.

Approximation properties

- As in Fourier, decay of coefficients $\leftrightarrow$ regularity
- Leads to approximation properties similar to Sobolev spaces, following Bach (2017a)
- For κ_1, $p/2$ is replaced by $p/2+1$ (need more regularity)

Corollary (Approximation of Lipschitz functions)

Let $f : \mathbb{S}^{p-1} \to \mathbb{R}$ be an even function such that $f(x) \leq \eta$ and $|f(x) - f(y)| \leq \eta \|x - y\|$, for all $x, y \in \mathbb{S}^{p-1}$. There is a function $g \in \mathcal{H}$ with $\|g\|_{\mathcal{H}} \leq \delta$, where δ is larger than a constant depending only on p, such that

$$\sup_{x \in \mathbb{S}^{p-1}} |f(x) - g(x)| \leq C(p)\eta \left(\frac{\delta}{\eta}\right)^{-1/(p/2-1)} \log\left(\frac{\delta}{\eta}\right).$$

Outline

Interlude: infinite width beyond two-layers

- **Deep network**: define "pre-activations" $\tilde{a}^1 = W^1 x$ and

$$\tilde{a}^k = \frac{1}{\sqrt{m_{k-1}}} W^k \sigma(\tilde{a}^{k-1})$$

- For inputs x, x' on the sphere and infinite width, $\tilde{a}_i^k, \tilde{a}_i'^k$ are Gaussian
- Covariance follows recurrence $\Sigma_1(x, x') = x^\top x'$

$$B_k = \begin{pmatrix} 1 & \Sigma_{k-1}(x, x') \\ \Sigma_{k-1}(x, x') & 1 \end{pmatrix}$$

$$\begin{aligned} \Sigma_k(x, x') &= \mathbb{E}_{(u,v)\sim N(0, B_k)}[\sigma(u)\sigma(v)] \\ &= \kappa_1(\Sigma_{k-1}(x, x')). \end{aligned}$$

Interlude: infinite width beyond two-layers

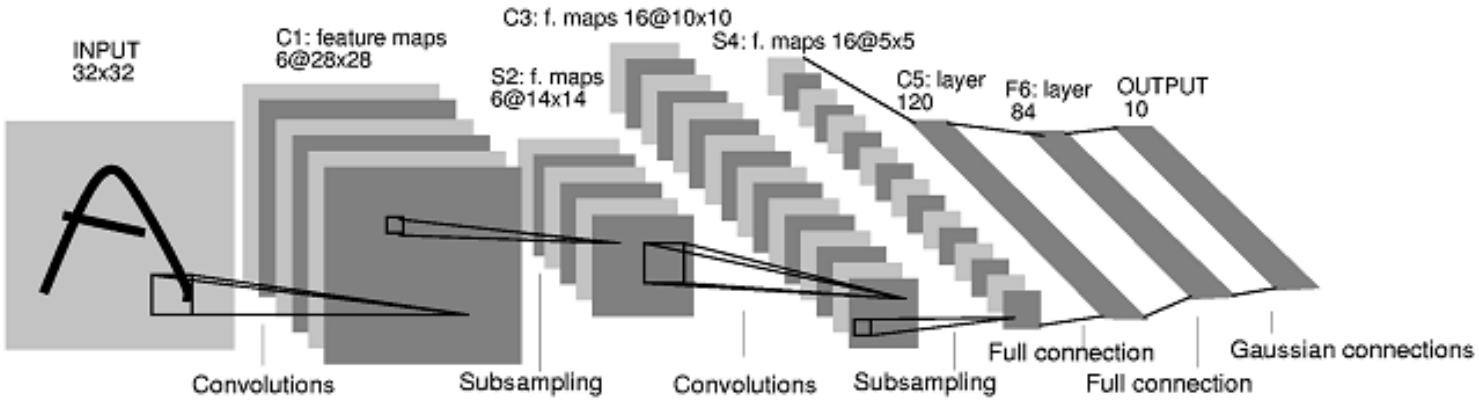
$$\Sigma_k(x, x') = \kappa_1(\Sigma_{k-1}(x, x'))$$

- Defines **hierarchical kernels**
- If $\kappa_1(x^\top x') = \langle \varphi_1(x), \varphi_1(x') \rangle$, then Σ_k has feature map

$$\Phi_k(x) = \underbrace{\varphi_1 \circ \cdots \circ \varphi_1}_{k \text{ times}}(x)$$

- Corresponds to training only last layer
- Similar for the NTK, but a sum of such kernels, one for each layer

Kernels for deep convolutional networks (CNNs)



Convolutional Neural Networks (CNNs):

- Capture **multi-scale** and **compositional** structure in natural signals
- Provide some **invariance**
- Model **local stationarity**
- **State-of-the-art** in many applications

Smoothness and stability with kernels

Study geometry of the kernel mapping $f(x) = \langle f, \Phi(x) \rangle$

$$|f(x) - f(x')| \leq \|f\|_{\mathcal{H}} \cdot \|\Phi(x) - \Phi(x')\|_{\mathcal{H}}$$

- $\|f\|_{\mathcal{H}}$ controls **smoothness** and **complexity** of the model
- $\Phi(x)$ encodes CNN **architecture** independently of the model (smoothness, invariance, stability to deformations)

Smoothness and stability with kernels

Study geometry of the kernel mapping $f(x) = \langle f, \Phi(x) \rangle$

$$|f(x) - f(x')| \leq \|f\|_{\mathcal{H}} \cdot \|\Phi(x) - \Phi(x')\|_{\mathcal{H}}$$

- $\|f\|_{\mathcal{H}}$ controls **smoothness** and **complexity** of the model
- $\Phi(x)$ encodes CNN **architecture** independently of the model (smoothness, invariance, stability to deformations)
- Training last layer of infinite-width CNN leads to a special case of **convolutional kernel networks** (CKNs, Mairal, 2016; Bietti and Mairal, 2019a)
- NTK (all layers) similar but more involved

Convolutional kernel network (CKN)

- $x_0 : \Omega \to \mathcal{H}_0$: initial (**continuous**) signal
  - $u \in \Omega = \mathbb{R}^d$: location ($d = 2$ for images)
  - $x_0(u) \in \mathcal{H}_0$: value ($\mathcal{H}_0 = \mathbb{R}^3$ for RGB images)

Convolutional kernel network (CKN)

- $x_0 : \Omega \to \mathcal{H}_0$: initial (**continuous**) signal
  - $u \in \Omega = \mathbb{R}^d$: location ($d = 2$ for images)
  - $x_0(u) \in \mathcal{H}_0$: value ($\mathcal{H}_0 = \mathbb{R}^3$ for RGB images)
- $x_k : \Omega \to \mathcal{H}_k$: *feature map* at layer k

$$P_k x_{k-1}$$

  - P_k: **patch extraction** operator, extract small patch of feature map x_{k-1} around each point u

Convolutional kernel network (CKN)

- $x_0 : \Omega \to \mathcal{H}_0$: initial (**continuous**) signal
  - $u \in \Omega = \mathbb{R}^d$: location ($d = 2$ for images)
  - $x_0(u) \in \mathcal{H}_0$: value ($\mathcal{H}_0 = \mathbb{R}^3$ for RGB images)
- $x_k : \Omega \to \mathcal{H}_k$: *feature map* at layer k

$$M_k P_k x_{k-1}$$

  - P_k: **patch extraction** operator, extract small patch of feature map x_{k-1} around each point u
  - M_k: **non-linear mapping** operator, maps each patch to a new point with a **pointwise** non-linear function $\varphi(\cdot)$ (kernel mapping)

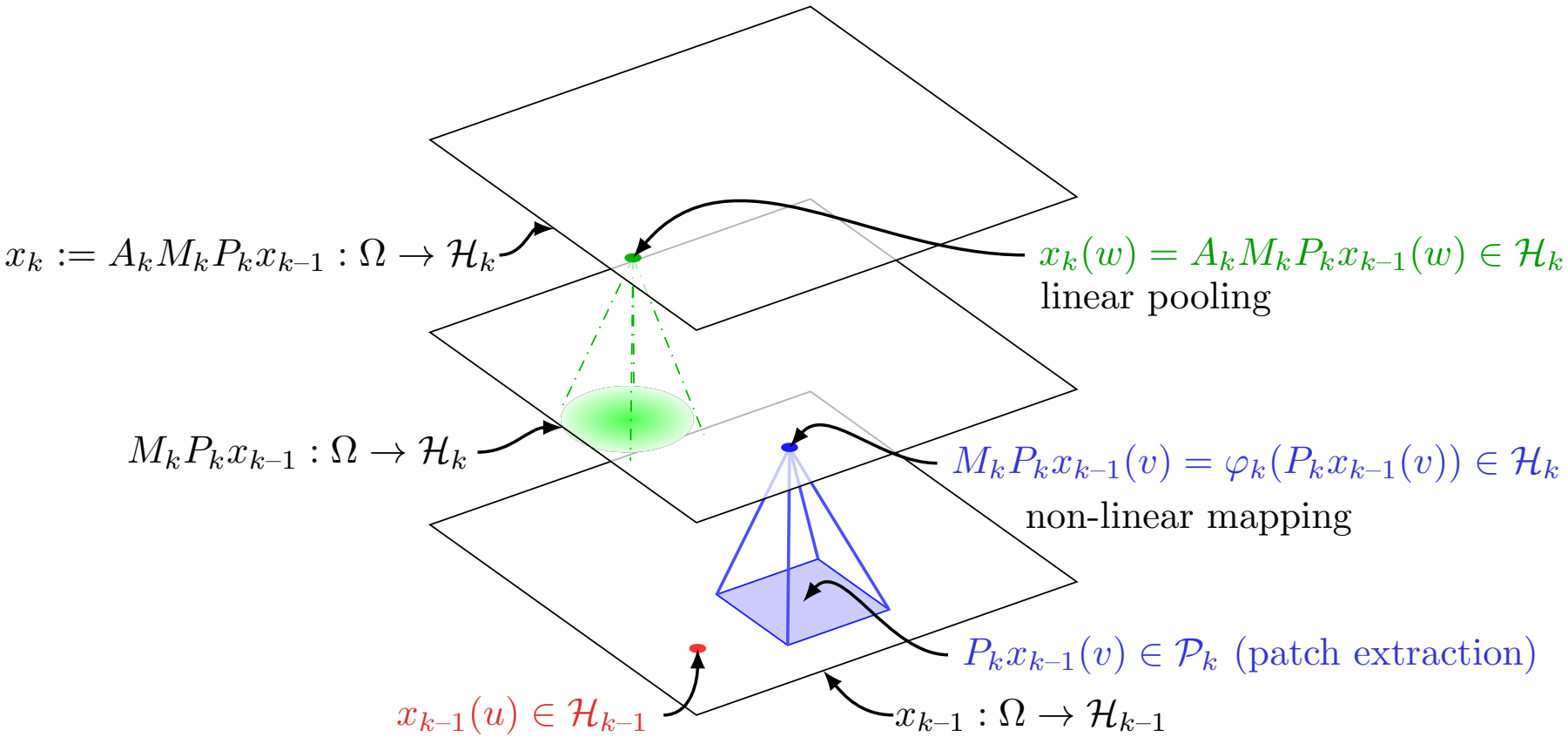
Convolutional kernel network (CKN)

- $x_0 : \Omega \to \mathcal{H}_0$: initial (**continuous**) signal
  - $u \in \Omega = \mathbb{R}^d$: location ($d = 2$ for images)
  - $x_0(u) \in \mathcal{H}_0$: value ($\mathcal{H}_0 = \mathbb{R}^3$ for RGB images)
- $x_k : \Omega \to \mathcal{H}_k$: *feature map* at layer k

$$x_k = A_k M_k P_k x_{k-1}$$

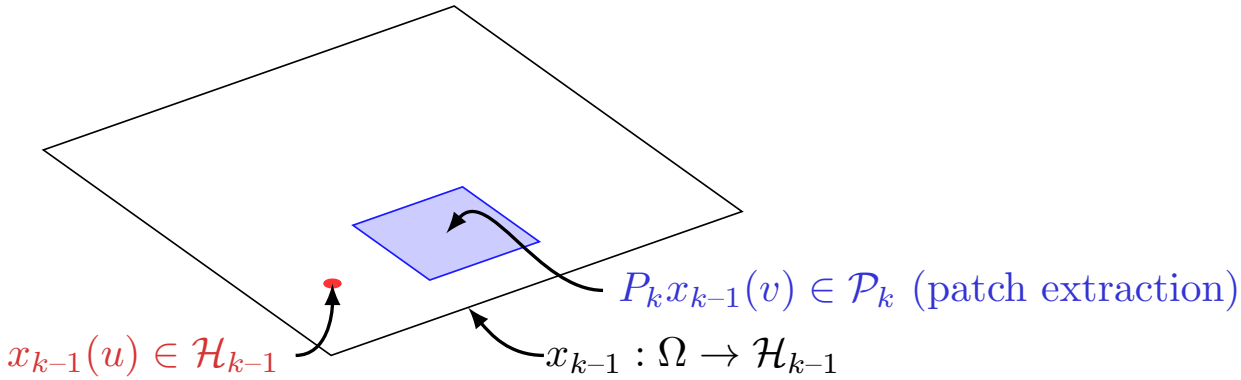
  - P_k: **patch extraction** operator, extract small patch of feature map x_{k-1} around each point u
  - M_k: **non-linear mapping** operator, maps each patch to a new point with a **pointwise** non-linear function $\varphi(\cdot)$ (kernel mapping)
  - A_k: (linear, Gaussian) **pooling** operator at scale σ_k

CKN construction



Patch extraction operator P_k

$$P_k x_{k-1}(u) := (v \in S_k \mapsto x_{k-1}(u+v)) \in \mathcal{P}_k = \mathcal{H}_{k-1}^{S_k}$$



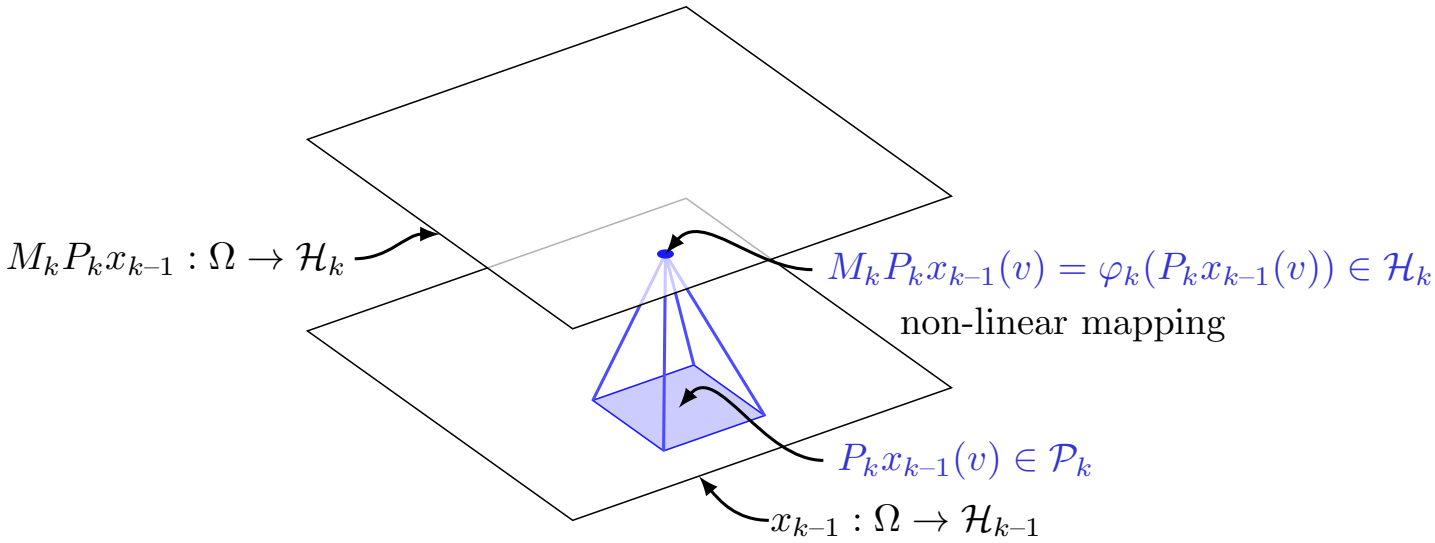
Patch extraction operator P_k

$$P_k x_{k-1}(u) := (v \in S_k \mapsto x_{k-1}(u+v)) \in \mathcal{P}_k = \mathcal{H}_{k-1}^{S_k}$$

- S_k: patch shape, e.g. box
- P_k is **linear**, and **preserves the L^2 norm**: $\|P_k x_{k-1}\| = \|x_{k-1}\|$

Non-linear mapping operator M_k

$$M_k P_k x_{k-1}(u) := \varphi(P_k x_{k-1}(u)) \in \mathcal{H}_k$$



Non-linear mapping operator M_k

$$M_k P_k x_{k-1}(u) := \varphi(P_k x_{k-1}(u)) \in \mathcal{H}_k$$

- $\varphi : \mathcal{P}_k \to \mathcal{H}_k$ pointwise non-linearity on patches (kernel map)
- For ReLU, we use φ_1 from arc-cosine kernel, but other kernels are possible
- We assume **non-expansivity**: for $z, z' \in \mathcal{P}_k$

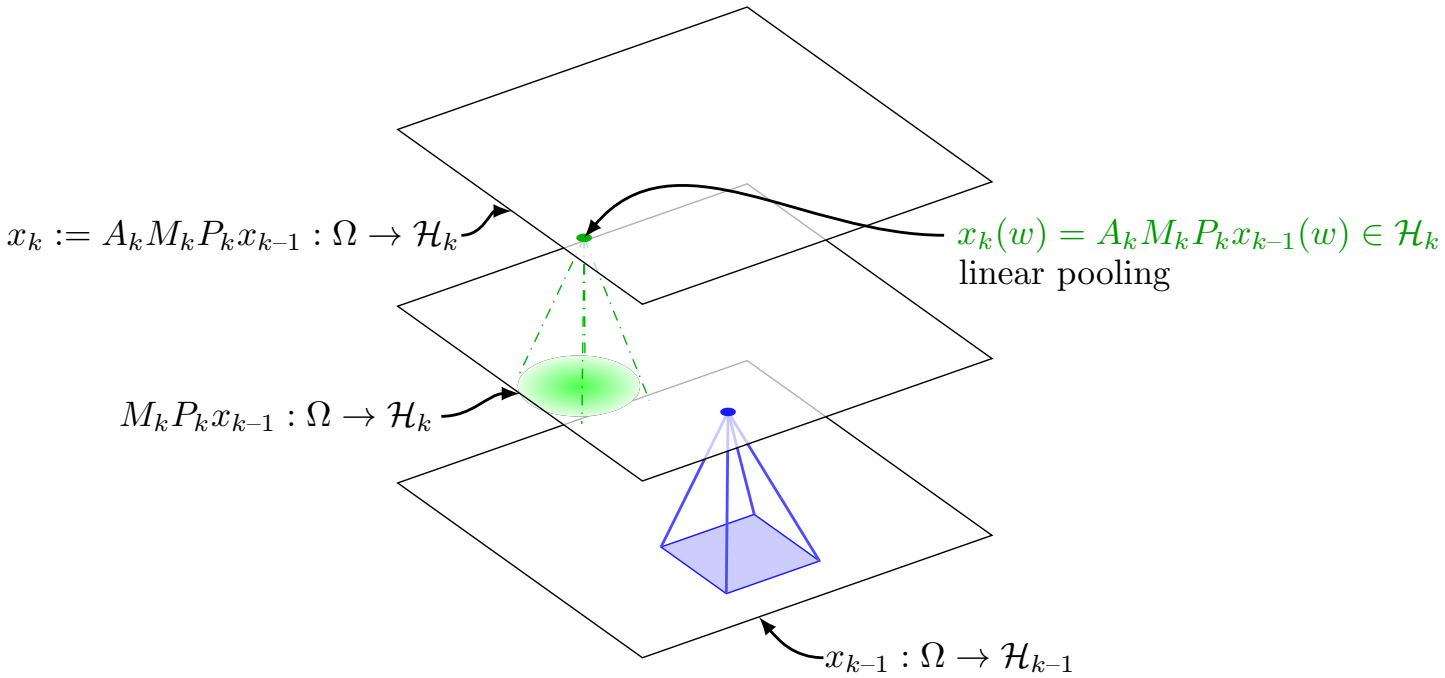
$$\|\varphi(z)\| \leq \|z\| \quad \text{and} \quad \|\varphi(z) - \varphi(z')\| \leq \|z - z'\|$$

- M_k then satisfies, for $x, x' \in L^2(\Omega, \mathcal{P}_k)$

$$\|M_k x\| \leq \|x\| \quad \text{and} \quad \|M_k x - M_k x'\| \leq \|x - x'\|$$

Pooling operator A_k

$$x_k(u) = A_k M_k P_k x_{k-1}(u) = \int_{\mathbb{R}^d} h_{\sigma_k}(u-v) M_k P_k x_{k-1}(v) dv \in \mathcal{H}_k$$

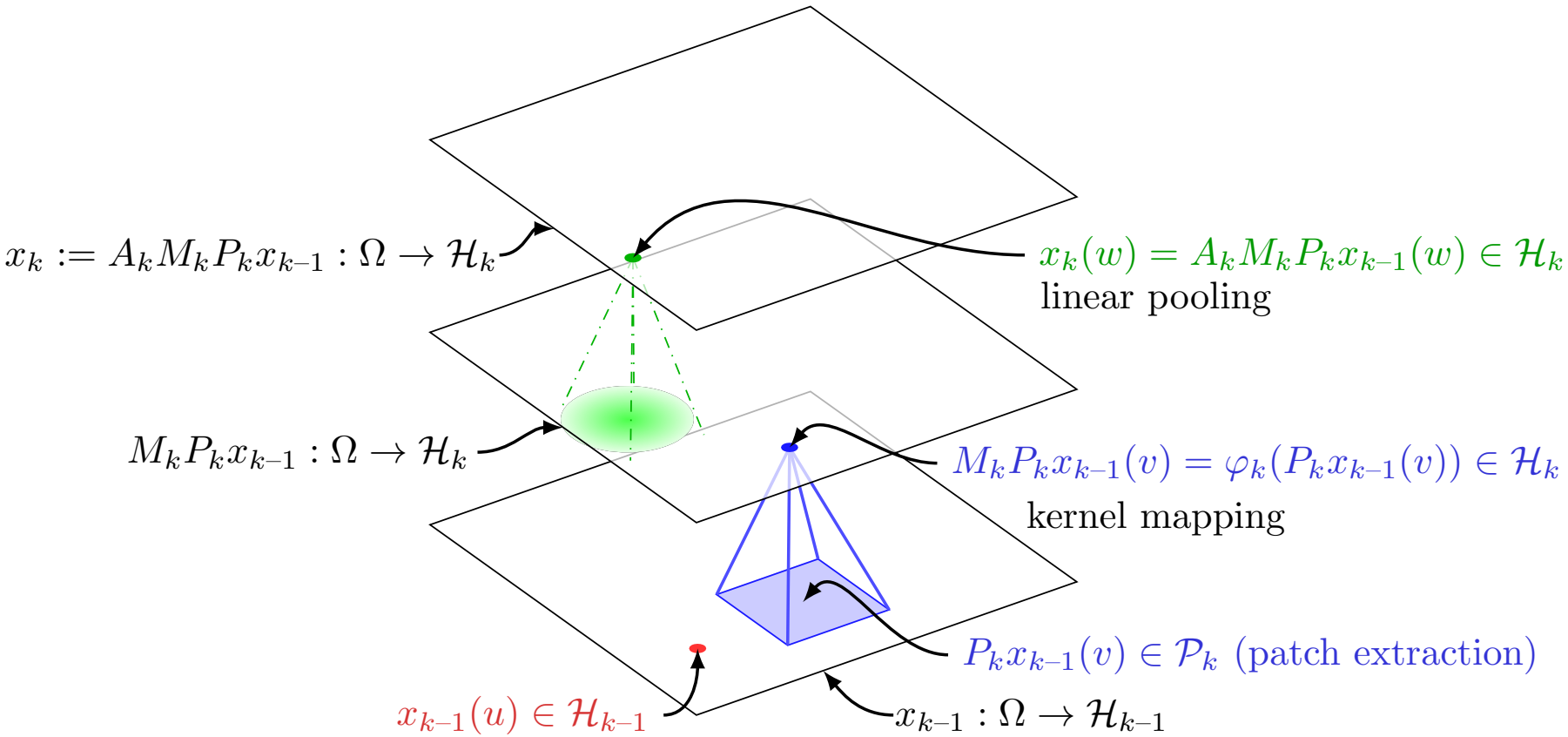


Pooling operator A_k

$$x_k(u) = A_k M_k P_k x_{k-1}(u) = \int_{\mathbb{R}^d} h_{\sigma_k}(u-v) M_k P_k x_{k-1}(v) dv \in \mathcal{H}_k$$

- h_{σ_k}: pooling filter at scale σ_k
- $h_{\sigma_k}(u) := \sigma_k^{-d} h(u/\sigma_k)$ with $h(u)$ **Gaussian**
- **linear, non-expansive operator**: $\|A_k\| \leq 1$

Recap: P_k, M_k, A_k



Final CKN kernel mapping

Assumption on x_0

- x_0 is typically a **discrete** signal aquired with physical device.
- Natural assumption: $x_0 = A_0 x$, with x the original continuous signal, A_0 local integrator with scale σ_0 (**anti-aliasing**).

Final CKN kernel mapping

Assumption on x_0

- x_0 is typically a **discrete** signal aquired with physical device.
- Natural assumption: $x_0 = A_0 x$, with x the original continuous signal, A_0 local integrator with scale σ_0 (**anti-aliasing**).

Multilayer representation

$$\Phi(x) = A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x_0 \ \in \ L^2(\Omega, \mathcal{H}_n).$$

- S_k, σ_k grow exponentially in practice (i.e., fixed with subsampling).

Final CKN kernel mapping

Assumption on x_0

- x_0 is typically a **discrete** signal aquired with physical device.
- Natural assumption: $x_0 = A_0 x$, with x the original continuous signal, A_0 local integrator with scale σ_0 (**anti-aliasing**).

Multilayer representation

$$\Phi(x) = A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \cdots A_1 M_1 P_1 x_0 \ \in \ L^2(\Omega, \mathcal{H}_n).$$

- S_k, σ_k grow exponentially in practice (i.e., fixed with subsampling).

Final kernel

$$K_{CKN}(x, x') = \langle \Phi_n(x), \Phi_n(x') \rangle_{L^2(\Omega)} = \int_\Omega \langle x_n(u), x'_n(u) \rangle du$$

Convolutional NTK kernel mapping

Define

$$M(x, y)(u) = \begin{pmatrix} \varphi_0(x(u)) \otimes y(u) \\ \varphi_1(x(u)) \end{pmatrix}$$

Theorem (NTK feature map for CNN)

$$K_{NTK}(x, x') = \langle \Phi(x), \Phi(x') \rangle_{L^2(\Omega)},$$

with $\Phi(x)(u) = A_n M(x_n, y_n)(u)$, *where* $y_1(u) = x_1(u) = P_1 x(u)$ *and*

$$x_k(u) = P_k A_{k-1} \varphi_1(x_{k-1})(u)$$
$$y_k(u) = P_k A_{k-1} M(x_{k-1}, y_{k-1})(u).$$

Smoothness: CKN vs NTK

Weaker smoothness for NTK!

CKN: Lipschitz

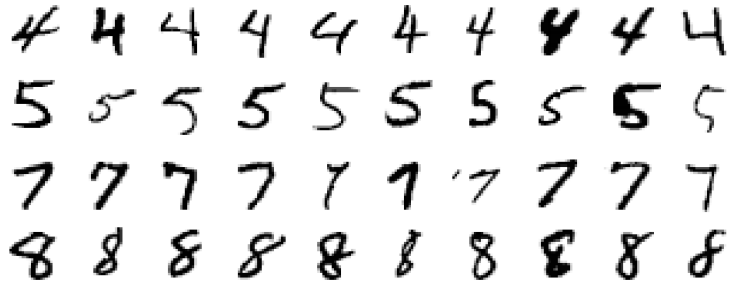
$$\|\Phi(x) - \Phi(x')\| \leq \|x - x'\|$$

NTK: $\approx$ Hölder

$$\|\Phi(x) - \Phi(x')\| \leq (n+1)\|x - x'\| + O(n^{5/4})\sqrt{\|\mathbf{x}\|\|\mathbf{x} - \mathbf{x}'\|}.$$

Stability to deformations: definitions

- $\tau : \Omega \to \Omega$: C^1-diffeomorphism
- $L_\tau x(u) = x(u - \tau(u))$: action operator
- Much richer group of transformations than translations



- Studied for wavelet-based scattering transform (Mallat, 2012; Bruna and Mallat, 2013)

Stability to deformations: definitions

- Representation $\Phi(\cdot)$ is **stable** (Mallat, 2012) if:

$$\|\Phi(L_\tau x) - \Phi(x)\| \leq (C_1 \|\nabla \tau\|_\infty + C_2 \|\tau\|_\infty) \|x\|$$

- $\|\nabla \tau\|_\infty = \sup_u \|\nabla \tau(u)\|$ controls deformation
- $\|\tau\|_\infty = \sup_u |\tau(u)|$ controls translation
- $C_2 \to 0$: translation invariance

Stability to deformations: CKN (Bietti and Mairal, 2019a)

Theorem

Let $\Phi_n(x) = \Phi(A_0 x)$ *and assume* $\|\nabla\tau\|_\infty \leq 1/2$,

$$\|\Phi_n(L_\tau x) - \Phi_n(x)\| \leq \left(C_\beta \, (n+1) \, \|\nabla\tau\|_\infty + \frac{C}{\sigma_n} \|\tau\|_\infty \right) \|x\|$$

- translation invariance: large σ_n
- stability: small patch sizes ($\beta \approx$ patch size, $C_\beta = O(\beta^3)$ for images)
- signal preservation: subsampling factor $\approx$ patch size
- $\Longrightarrow$ **needs several layers**

Stability to deformations: NTK (Bietti and Mairal, 2019b)

Theorem (Stability of NTK)

Let $\Phi_n(x) = \Phi(A_0 x)$, *and assume* $\|\nabla\tau\|_\infty \leq 1/2$

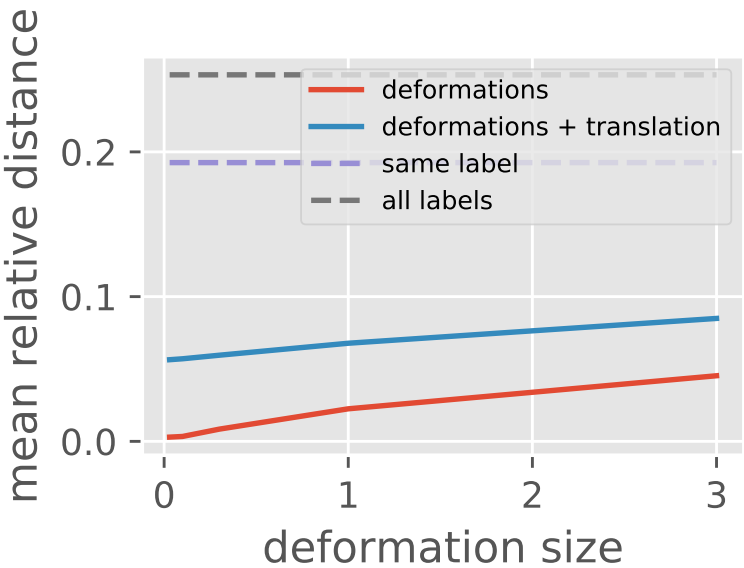
$$\|\Phi_n(L_\tau x) - \Phi_n(x)\| \leq \left(C_\beta n^{7/4} \|\nabla\tau\|_\infty^{1/2} + C'_\beta n^2 \|\nabla\tau\|_\infty + \sqrt{n+1}\frac{C}{\sigma_n}\|\tau\|_\infty \right) \|x\|,$$

Stability to deformations: NTK (Bietti and Mairal, 2019b)

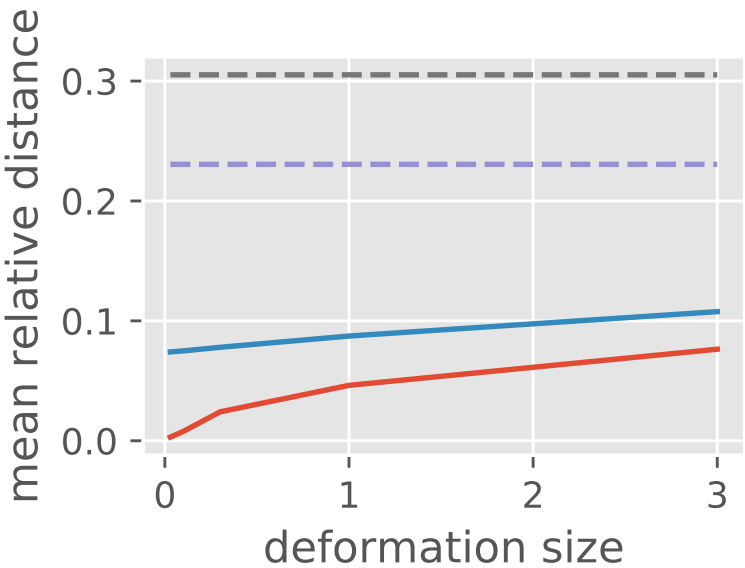
Theorem (Stability of NTK)

Let $\Phi_n(x) = \Phi(A_0 x)$, *and assume* $\|\nabla\tau\|_\infty \leq 1/2$

$$\|\Phi_n(L_\tau x) - \Phi_n(x)\|$$
$$\leq \left(C_\beta n^{7/4} \|\nabla\tau\|_\infty^{1/2} + C'_\beta n^2 \|\nabla\tau\|_\infty + \sqrt{n+1}\frac{C}{\sigma_n}\|\tau\|_\infty \right) \|x\|,$$



CKN



NTK

Conclusion

CKN vs NTK, last layer vs all layers

- NTK leads to a "larger" RKHS, better approximation
- NTK functions for CNNs are less smooth/stable

Conclusion

CKN vs NTK, last layer vs all layers

- NTK leads to a "larger" RKHS, better approximation
- NTK functions for CNNs are less smooth/stable

Limitations: things are different for finite neurons!

- need huge number of neurons for NTK regime
- functions with finite neurons are not smooth! (*e.g.*, adversarial examples?)
- lazy training is limited (*e.g.*, cannot explain learning Gabor filters)

Conclusion

CKN vs NTK, last layer vs all layers

- NTK leads to a "larger" RKHS, better approximation
- NTK functions for CNNs are less smooth/stable

Limitations: things are different for finite neurons!

- need huge number of neurons for NTK regime
- functions with finite neurons are not smooth! (*e.g.*, adversarial examples?)
- lazy training is limited (*e.g.*, cannot explain learning Gabor filters)

Future directions

- approximation properties for deep networks / CNNs
- a "less lazy" regime? (*e.g.*, only first layers move)

References I

F. Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research (JMLR)*, 18(19):1–53, 2017a.

F. Bach. On the equivalence between kernel quadrature rules and random feature expansions. *Journal of Machine Learning Research (JMLR)*, 18(21):1–38, 2017b.

P. L. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

A. Bietti and J. Mairal. Invariance and stability of deep convolutional representations. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

A. Bietti and J. Mairal. Group invariance, stability to deformations, and complexity of deep convolutional representations. *Journal of Machine Learning Research (JMLR)*, 20(25):1–49, 2019a.

A. Bietti and J. Mairal. On the inductive bias of neural tangent kernels. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019b.

References II

J. Bruna and S. Mallat. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(8): 1872–1886, 2013.

L. Chizat, E. Oyallon, and F. Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

J. Mairal. End-to-End Kernel Learning with Supervised Convolutional Kernel Networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

S. Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

A. Rudi and L. Rosasco. Generalization properties of learning with random features. In *Advances in Neural Information Processing Systems*, pages 3215–3225, 2017.

B. Schölkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. 2001.