| | |
|---|---|
| **Document Title:** | **Theory of Invertible and Injective Deep Neural Networks for Likelihood Estimation and Uncertainty Quantification** |
| **Author(s):** | **Michael Puthawala, Matti Lassas, Ivan Dokmanic, Pekka Pankka, Maarten de Hoop** |
| **Document Number:** | **308222** |
| **Date Received:** | **December 2023** |
| **Award Number:** | **N/A** |

# Theory of Invertible and Injective Deep Neural Networks for Likelihood Estimation and Uncertainty Quantification


Michael Puthawala [1]

Matti Lassas[2], Ivan Dokmanić[3], Pekka Pankka[2], Maarten de Hoop[4]

[1]South Dakota State University, [2]University of Helsinki, [3]University of Basel, [4]Rice University


June 14, 2023

# Acknowledgements

Partial research support was provided by CAPITAL Services of Sioux Falls, SD and the Simons Foundation.
My travel to ICFIS 2023 was supported by the Office of Justice Programs, of the US Department of Justice under award number GS-00F-439GA.
The content is solely the responsibility of the authors and does not necessarily represent the official views of the Department of Justice.

# Disclaimer and Goal of Talk

I am an applied mathematician who recently started at South Dakota State University. My research is in the mathematical foundations of deep learning, especially at the intersection of geometry, topology and universality.

# Disclaimer and Goal of Talk

I am an applied mathematician who recently started at South Dakota State University.
My research is in the mathematical foundations of deep learning, especially at the intersection of geometry, topology and universality.
A bit about my background, my interested in forensic statistics started Monday at approximately 9:30am.

# Disclaimer and Goal of Talk

I am an applied mathematician who recently started at South Dakota State University.
My research is in the mathematical foundations of deep learning, especially at the intersection of geometry, topology and universality.
A bit about my background, my interested in forensic statistics started Monday at approximately 9:30am.
I'd like to introduce you to some of the mathematical theory behind deep learning.

# Disclaimer and Goal of Talk

I am an applied mathematician who recently started at South Dakota State University.
My research is in the mathematical foundations of deep learning, especially at the intersection of geometry, topology and universality.
A bit about my background, my interested in forensic statistics started Monday at approximately 9:30am.
I'd like to introduce you to some of the mathematical theory behind deep learning.
My goal isn't to give a complete description of the theory. Rather I want to give you a taste of how deep learning can be formalized and how these formalisms yield interesting mathematics.

# Machine Learning, Deep Learning

Machine learning tries to let machines 'learn' patterns in data.

# Machine Learning, Deep Learning

Machine learning tries to let machines ‘learn’ patterns in data.

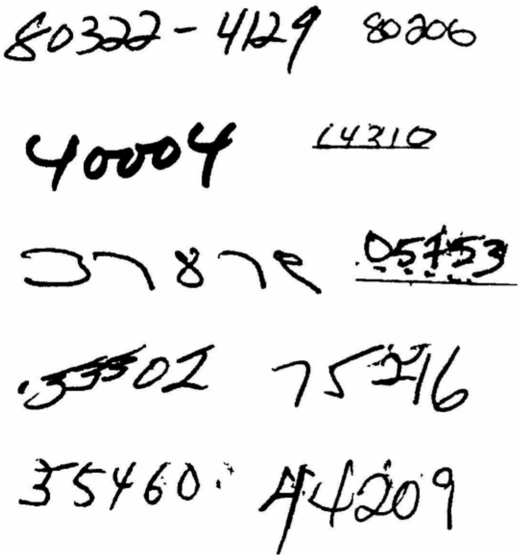
Deep learning is a subset of machine learning, and is best defined through example.

# Humble Beginnings

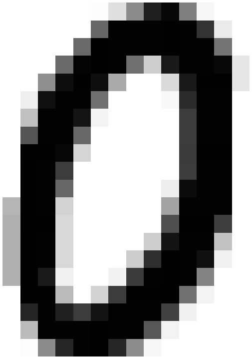
Modern deep learning can be traced to Yann LeCun's work[1] at Bell Labs in 1989.

[1] LeCun et al., "Backpropagation applied to handwritten zip code recognition".

# Handwritten Zip Codes



# Handwritten Zip Codes

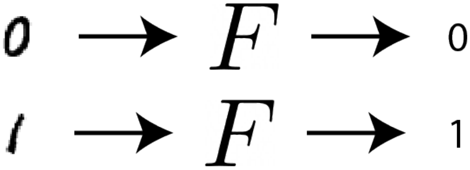


# Function that Recognizes Digits

 $\longrightarrow F \longrightarrow$ 0

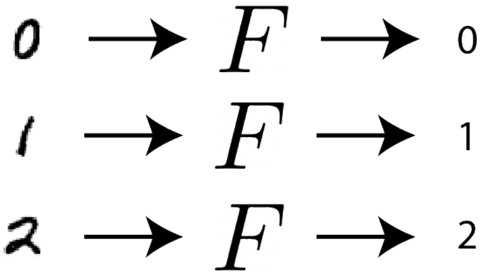
The goal is to find a function $F : \mathbb{R}^{16\times 16} \to \{0, 1, \ldots, 9\}$ so that

# Function that Recognizes Digits



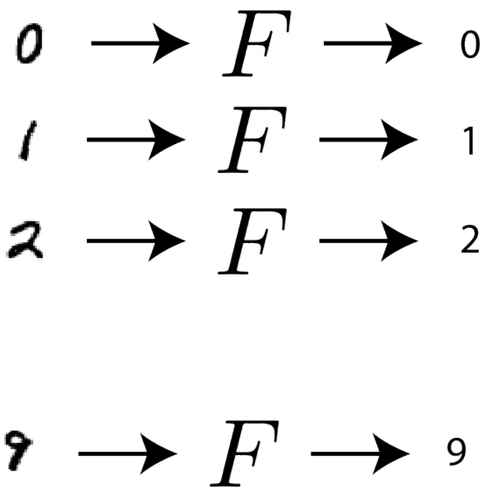

The goal is to find a function
$F : \mathbb{R}^{16\times 16} \to \{0, 1, \ldots, 9\}$ so that

# Function that Recognizes Digits



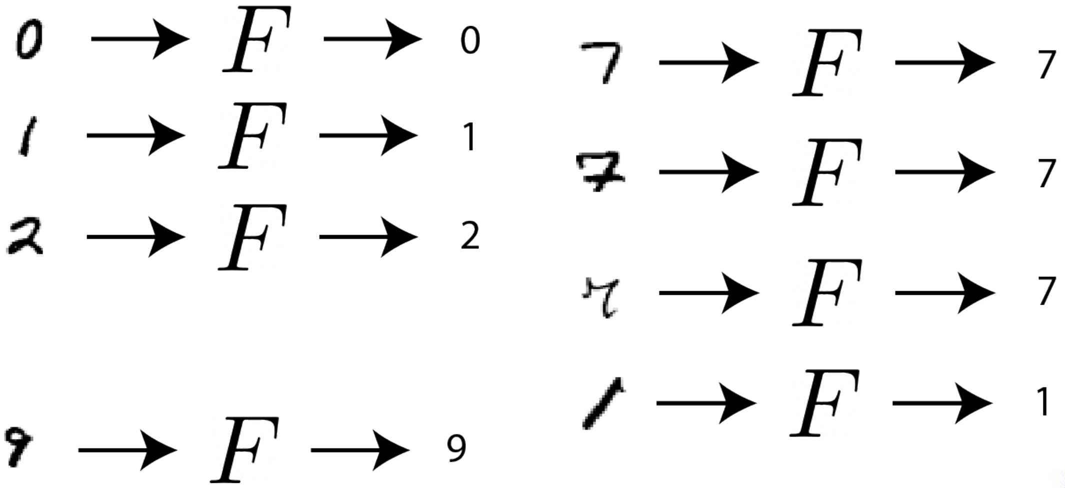

The goal is to find a function
$F : \mathbb{R}^{16\times 16} \to \{0, 1, \ldots, 9\}$ so that

# Function that Recognizes Digits




The goal is to find a function $F : \mathbb{R}^{16\times 16} \to \{0, 1, \ldots, 9\}$ so that

# Function that Recognizes Digits



# Four-step approach

1. Collect 60,000 examples of hand-written digits, and form pairs $\{(x_i, z_i)\}_{i=1}^{60,000}$ where $x_i \in \mathbb{R}^{16^2}$ are vectorized 16$x$16 images and $z_i \in \{0, \ldots, 9\}$ are the true labels.

## Four-step approach

1. Collect 60,000 examples of hand-written digits, and form pairs $\{(x_i, z_i)\}_{i=1}^{60,000}$ where $x_i \in \mathbb{R}^{16^2}$ are vectorized 16x16 images and $z_i \in \{0, \ldots, 9\}$ are the true labels.
2. Let $F \colon \mathbb{R}^{16^2} \times \Theta \to \{0, \ldots, 9\}$ be of the form

$$F(x; \theta) = \phi \circ W_{3,\theta} \circ \phi \circ W_{2,\theta} \circ \phi \circ W_{1,\theta}(x)$$

where for $y \in \mathbb{R}^m$, and for $\ell = 1, \ldots, 3$,

$$\phi(y) = \begin{bmatrix} \max(y_1, 0) \\ \max(y_2, 0) \\ \vdots \\ \max(y_m, 0) \end{bmatrix}, \quad W_{\ell,\theta}(y) := \begin{bmatrix} \theta_{\ell,1,1} & \cdots & \theta_{\ell,1,m} \\ \vdots & \ddots & \vdots \\ \theta_{\ell,m,1} & \cdots & \theta_{\ell,m,m} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}.$$

Note that functions of this form don't have *anything* to do with handwriting recognition.

# Four-step approach

1. Collect 60,000 examples of hand-written digits, and form pairs $\{(x_i, z_i)\}_{i=1}^{60{,}000}$ where $x_i \in \mathbb{R}^{16^2}$ are vectorized 16x16 images and $z_i \in \{0, \ldots, 9\}$ are the true labels.
2. Let $F : \mathbb{R}^{16^2} \times \Theta \to \{0, \ldots, 9\}$ be of the form

$$F(x; \theta) = \phi \circ W_{3,\theta} \circ \phi \circ W_{2,\theta} \circ \phi \circ W_{1,\theta}(x)$$

3. 'Train' $\theta$ by looking for a minimizer of

$$\min_{\theta \in \Theta} \sum_{i=1}^{50{,}000} |F(x_i; \theta) - z_i|$$

   using stochastic gradient descent.

# Four-step approach

1. Collect 60,000 examples of hand-written digits, and form pairs $\{(x_i, z_i)\}_{i=1}^{60{,}000}$ where $x_i \in \mathbb{R}^{16^2}$ are vectorized 16x16 images and $z_i \in \{0, \ldots, 9\}$ are the true labels.
2. Let $F : \mathbb{R}^{16^2} \times \Theta \to \{0, \ldots, 9\}$ be of the form

$$F(x; \theta) = \phi \circ W_{3,\theta} \circ \phi \circ W_{2,\theta} \circ \phi \circ W_{1,\theta}(x)$$

3. 'Train' $\theta$ by looking for a minimizer of $\min_{\theta \in \Theta} \sum_{i=1}^{50{,}000} |F(x_i; \theta) - z_i|$ using stochastic gradient descent.
4. Evaluate how close $F(\tilde{x})$ and $\tilde{z}$ are for $(\tilde{x}, \tilde{z})$ using $\{(x_i, z_i)\}_{i=50{,}001}^{60{,}000}$ that your model hasn't seen.

# Deep Learning

Deep learning composes simple functions to build more complex ones.

# The Four Step Program

These four steps still guide deep learning to this day

1. Collect and clean data.

# The Four Step Program

These four steps still guide deep learning to this day

1. Collect and clean data.
2. Choose an architecture that depends on a parameter $\theta$.

# The Four Step Program

These four steps still guide deep learning to this day

1. Collect and clean data.
2. Choose an architecture that depends on a parameter $\theta$.
3. Train the network to find the 'right' $\theta \in \Theta$.

# The Four Step Program

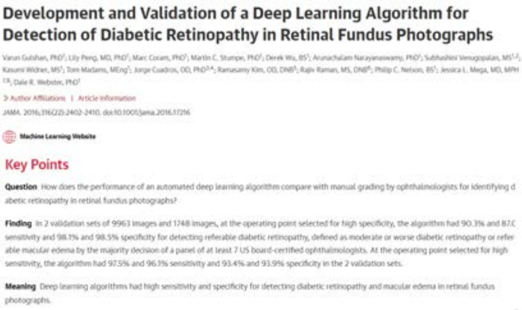
These four steps still guide deep learning to this day

1. Collect and clean data.
2. Choose an architecture that depends on a parameter $\theta$.
3. Train the network to find the 'right' $\theta \in \Theta$.
4. Measure how well $F$ works on data that it wasn't trained on.

# Deep Learning has been busy since 1989

1. Diagnosing Diabetic Retinopathy[a].

[a] Gulshan et al., "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs".



## Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs

Varun Gulshan, PhD[1]; Lily Peng, MD, PhD[1]; Marc Coram, PhD[1]; Martin C. Stumpe, PhD[1]; Derek Wu, BS[1]; Arunachalam Narayanaswamy, PhD[1]; Subhashini Venugopalan, MS[1,2]; Kasumi Widner, MS[1]; Tom Madams, MEng[1]; Jorge Cuadros, OD, PhD[3,4]; Ramasamy Kim, OD, DNB[5]; Rajiv Raman, MS, DNB[6]; Philip C. Nelson, BS[1]; Jessica L. Mega, MD, MPH[7,8]; Dale R. Webster, PhD[1]

> Author Affiliations | Article Information

JAMA. 2016;316(22):2402-2410. doi:10.1001/jama.2016.17216

Machine Learning Website

### Key Points

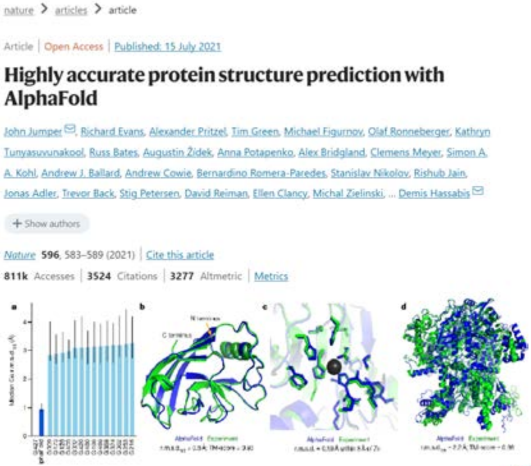
**Question** How does the performance of an automated deep learning algorithm compare with manual grading by ophthalmologists for identifying diabetic retinopathy in retinal fundus photographs?

**Finding** In 2 validation sets of 9963 images and 1748 images, at the operating point selected for high specificity, the algorithm had 90.3% and 87.0% sensitivity and 98.1% and 98.5% specificity for detecting referable diabetic retinopathy, defined as moderate or worse diabetic retinopathy or referable macular edema by the majority decision of a panel of at least 7 US board-certified ophthalmologists. At the operating point selected for high sensitivity, the algorithm had 97.5% and 96.1% sensitivity and 93.4% and 93.9% specificity in the 2 validation sets.

**Meaning** Deep learning algorithms had high sensitivity and specificity for detecting diabetic retinopathy and macular edema in retinal fundus photographs.

# Deep Learning has been busy since 1989

1. Diagnosing Diabetic Retinopathy[a].
2. Predicting Protean Folding[b].


nature > articles > article

Article | Open Access | Published: 15 July 2021

**Highly accurate protein structure prediction with AlphaFold**

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, ... Demis Hassabis

+ Show authors

*Nature* **596**, 583–589 (2021) | Cite this article

**811k** Accesses | **3524** Citations | **3277** Altmetric | Metrics

[a] Gulshan et al., "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs".

[b] Jumper et al., "Highly accurate protein structure prediction with AlphaFold".

# Deep Learning has been busy since 1989

1. Diagnosing Diabetic Retinopathy[a].
2. Predicting Protean Folding[b].
3. Speeding Up Matrix Multiplication[c].

[a] Gulshan et al., "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs".

[b] Jumper et al., "Highly accurate protein structure prediction with AlphaFold".

[c] Fawzi et al., "Discovering faster matrix multiplication algorithms with reinforcement learning".

nature > articles > article

Article | Open Access | Published: 05 October 2022

**Discovering faster matrix multiplication algorithms with reinforcement learning**

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis & Pushmeet Kohli

*Nature* **610**, 47–53 (2022) | Cite this article

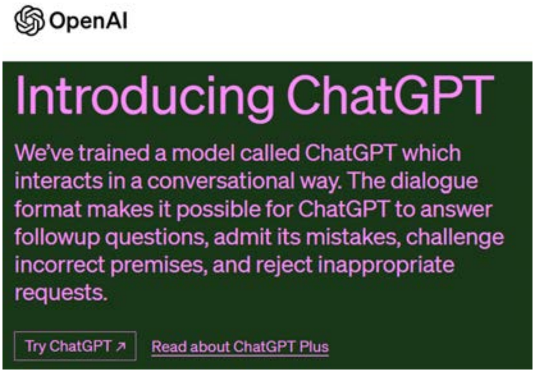
**1971** Altmetric | Metrics

## Abstract

Improving the efficiency of algorithms for fundamental computations can have a widespread impact, as it can affect the overall speed of a large amount of computations. Matrix multiplication is one such primitive task, occurring in many systems—from neural networks to scientific computing routines. The automatic discovery of algorithms using machine learning offers the prospect of reaching beyond human intuition and outperforming the current best human-designed algorithms. However, automating the algorithm discovery procedure is intricate, as the space of possible algorithms is enormous. Here we report a deep

# Deep Learning has been busy since 1989

1. Diagnosing Diabetic Retinopathy[a].
2. Predicting Protean Folding[b].
3. Speeding Up Matrix Multiplication[c].
4. Helping students cheat on homework.




---

[a] Gulshan et al., "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs".

[b] Jumper et al., "Highly accurate protein structure prediction with AlphaFold".

[c] Fawzi et al., "Discovering faster matrix multiplication algorithms with reinforcement learning".

# Why can ML models solve problems that are so general?

Figure: From `https://thispersondoesnotexist.com/` and paper[2].

[2] Karras et al., "Analyzing and improving the image quality of stylegan".

# How is this possible?

How does deep learning generate human looking faces?

# How is this possible?

How does deep learning generate human looking faces?

What does it even mean, mathematically, to generate faces?

# How is this possible?

How does deep learning generate human looking faces?

What does it even mean, mathematically, to generate faces?

Statistical Learning Theory (SLT) give us a useful formalism for understanding these questions.

# SLT, The Training Data

Figure: Training data taken from flickr and paper[3].

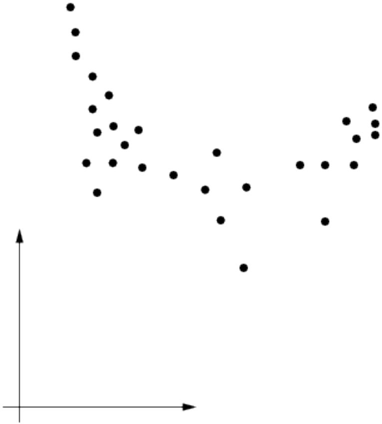
[3] Karras, Laine, and Aila, "A style-based generator architecture for generative adversarial networks".

# SLT, Data Distribution Ansatz

# SLT, Data Distribution Ansatz

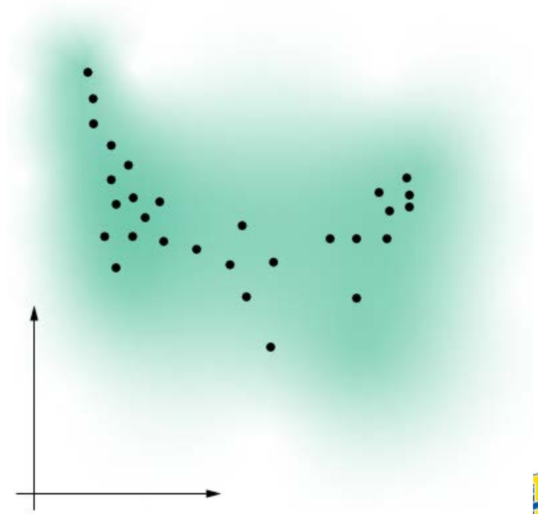






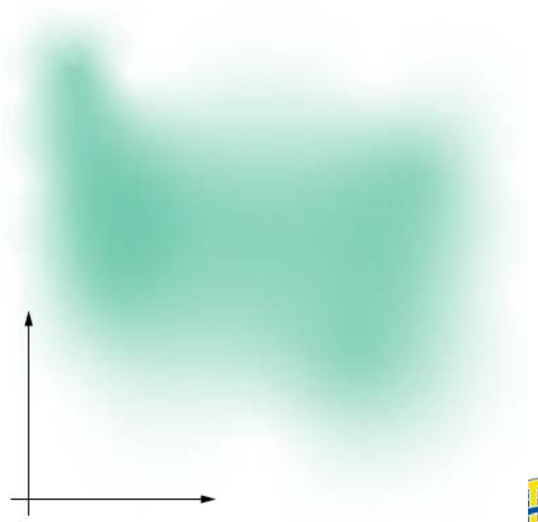
# SLT, Data Distribution Ansatz

The SLT ansatz is that there is a random variable $\mathrm{Y}_{\mathrm{faces}}$ with distribution $\rho$, and data points are samples from $\mathrm{Y}_{\mathrm{faces}}$.



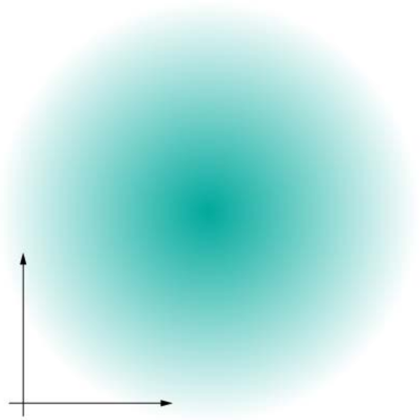
# SLT, Data Distribution Ansatz

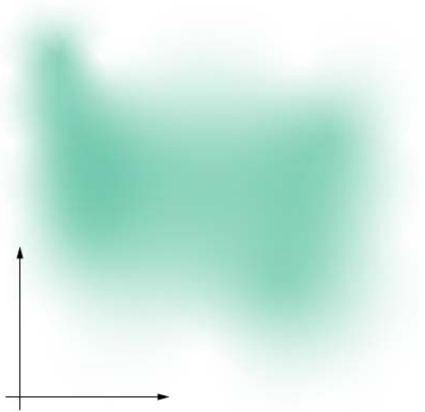
Generating faces means to sample points from $\mathrm{Y}_{\text{faces}}$.



# SLT Objective

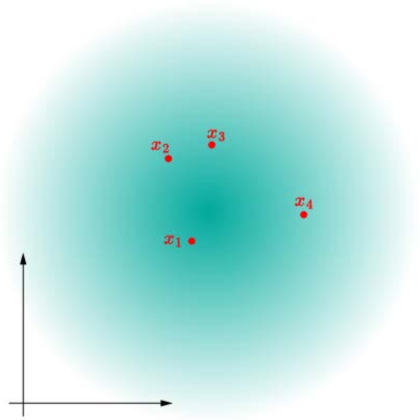
By choosing a starting distribution $\mathrm{X} \sim \mu$ and $f$ so that $\mathrm{Y}_{\text{faces}} = f(\mathrm{X})$ or $\rho = f_{\#}\mu$, we can sample points from $\mathrm{Y}_{\text{faces}}$.

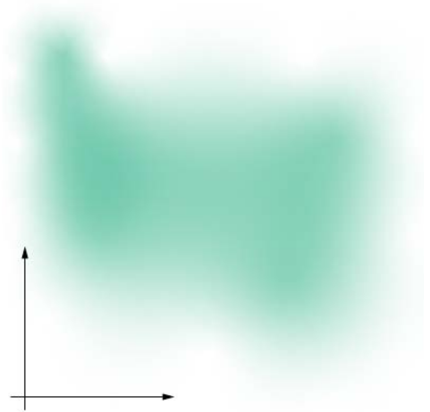




# SLT Objective

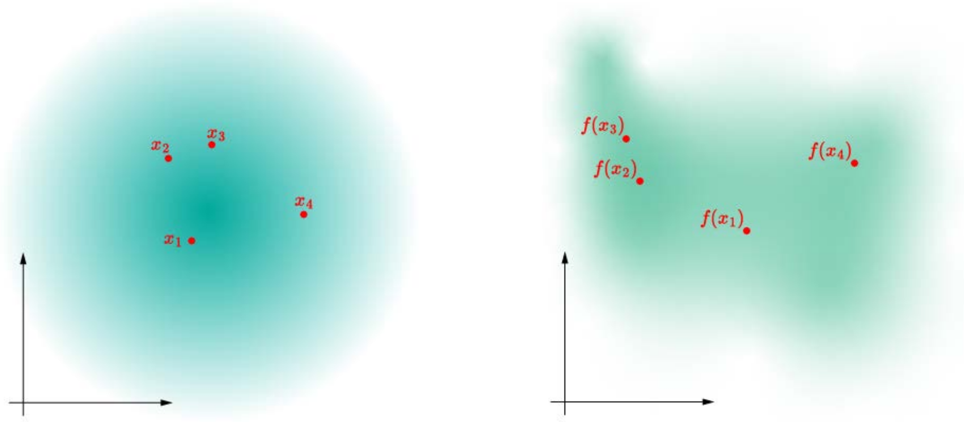
By choosing a starting distribution $\mathrm{X} \sim \mu$ and $f$ so that $\mathrm{Y}_{\text{faces}} = f(\mathrm{X})$ or $\rho = f_{\#}\mu$, we can sample points from $\mathrm{Y}_{\text{faces}}$.



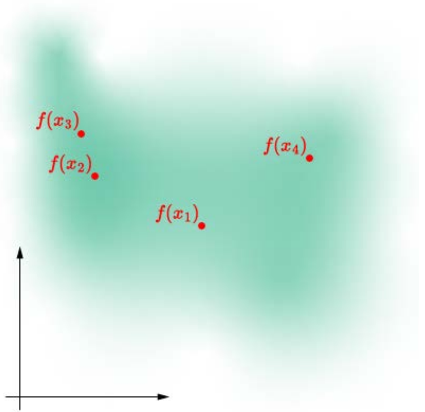
# SLT Objective

By choosing a starting distribution $\mathrm{X} \sim \mu$ and $f$ so that $\mathrm{Y}_{\text{faces}} = f(\mathrm{X})$ or $\rho = f_{\#}\mu$, we can sample points from $\mathrm{Y}_{\text{faces}}$.



# SLT Objective

By choosing a starting distribution $\mathrm{X} \sim \mu$ and $f$ so that $\mathrm{Y}_{\text{faces}} = f(\mathrm{X})$ or $\rho = f_{\#}\mu$, we can sample points from $\mathrm{Y}_{\text{faces}}$.



# How to generate faces

The question

- how does deep learning generate human looking faces?

# How to generate faces

The question

- how does deep learning generate human looking faces?

Via the SLT ansatz becomes

- are ML models able to learn a function $f$ so that $f(\mathrm{X}) = \mathrm{Y}$ where $\mathrm{X} \sim \mathcal{N}(0, I)$?

# Network Form

Recall that deep networks are functions of the form

$$f(\cdot;\theta)\colon X \to Y$$

for some parameter $\theta \in \Theta$. Let us notate $\mathcal{F} := \{f_\theta\}_{\theta\in\Theta}$. How 'large' is $\mathcal{F}$?

# Universal Approximation

Suppose that $\mathcal{F}$ and $\mathcal{G}$ are two families of functions from $X \to Y$.

**Definition (Universal Approximator)**

We say that $\mathcal{F}$ is a uniform approximator of $\mathcal{G}$ if for every compact $K \subset X$ and $g \in \mathcal{G}$,

$$\inf_{f \in \mathcal{F}} \sup_{x \in K} \| f(x) - g(x) \| = 0.$$

The Stone-Weirstrauss Approximation theorem from analysis says that polynomials on $[a, b]$ are universal approximators for $C^0([a, b])$.

# Uniform Approximation Picture

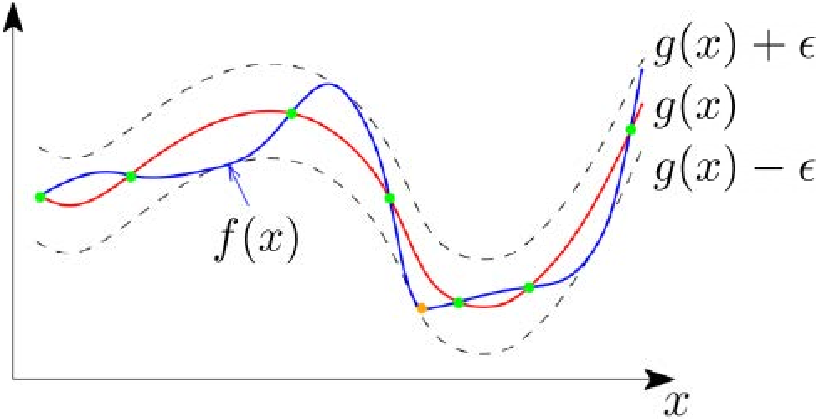



Figure adopted from Francis Bach's Blog

# Universality as a Measure of 'Strength' of a Network

If $\mathcal{F}$ contains networks of the form

$$\phi \circ W_{L,\theta} \circ \ldots \phi \circ W_{1,\theta},$$

if $\mathcal{F}$ a universal approximator with respect to some 'interesting' class of functions $\mathcal{G}$?

# Answer

Yes.

# Old Results

There are works that show that neural networks are universal w.r.t. e.g. continuous maps under very general conditions[4][5][6][7].

[4] Cybenko, "Approximation by superpositions of a sigmoidal function".
[5] Yarotsky, "Error bounds for approximations with deep ReLU networks".
[6] Kratsios and Bilokopytov, "Non-euclidean universal approximation".
[7] Kovachki et al., "Neural operator: Learning maps between function spaces".

# Revisiting Faces

Networks are able to learn to approximate human faces, because *any* continuous pushforward function can be uniformly approximated.

# Regression, Classification, Supervised Learning, etc.

The faces example was an unsupervised generative problem. What about supervised or semi-supervised problems? What about regression or classification problems?

# Regression, Classification, Supervised Learning, etc.

The faces example was an unsupervised generative problem. What about supervised or semi-supervised problems? What about regression or classification problems?
The details are different, but with the SLT framework, "most" problems can be reduced to learning an $f$ so that $f_{\#}\mu = \rho$ for distributions $\rho$ and $\mu$.

# Important Qualifiers

I'm hand waving some important details. For example,

- most universality results require $\mathcal{F}$ to contain arbitrarily large (wide or deep) networks,

# Important Qualifiers

I'm hand waving some important details. For example,

- most universality results require $\mathcal{F}$ to contain arbitrarily large (wide or deep) networks,
- in practice we don't have access to $\mu$ or $\rho$ directly,

# Important Qualifiers

I'm hand waving some important details. For example,

- most universality results require $\mathcal{F}$ to contain arbitrarily large (wide or deep) networks,
- in practice we don't have access to $\mu$ or $\rho$ directly,
- loss functions have to be chosen carefully to encourage 'good' convergence,

# Important Qualifiers

I'm hand waving some important details. For example,

- most universality results require $\mathcal{F}$ to contain arbitrarily large (wide or deep) networks,
- in practice we don't have access to $\mu$ or $\rho$ directly,
- loss functions have to be chosen carefully to encourage 'good' convergence,
- it's not obvious that training (i.e. gradient descent on the loss function) selects a $\theta^* \in \Theta$ so that $f_{\theta^*,\#}\mu \approx \rho$,

# Important Qualifiers

I'm hand waving some important details. For example,

- most universality results require $\mathcal{F}$ to contain arbitrarily large (wide or deep) networks,
- in practice we don't have access to $\mu$ or $\rho$ directly,
- loss functions have to be chosen carefully to encourage 'good' convergence,
- it's not obvious that training (i.e. gradient descent on the loss function) selects a $\theta^* \in \Theta$ so that $f_{\theta^*,\#}\mu \approx \rho$,

All points (and more) are valid and require elaboration, but I only have 45 minutes today.

# Pivot

I would now like to pivot form a general discussion on deep learning, and talk more about a very specific problem involving likelihood estimation.

# Change of Variables, Likelihood Estimation

If $\mathrm{Y} = f(\mathrm{X})$ and $f$ is bijective and smooth with smooth inverse, then

$$p_{\mathrm{Y}}(f(x)) = p_{\mathrm{X}}(x) \left|\det(\nabla f(x))\right|^{-1}.$$

If we can ensure that $f$ is bijective, smooth and estimate $\left|\det(\nabla f)\right|$, then we can estimate $p_{\mathrm{Y}}(f(x))$ by evaluating the r.h.s. of the above equation.

# Triangular Mappings

A triangular mapping $T\colon \mathbb{R}^n \to \mathbb{R}^n$ is one such that

$$T\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}\right) = \begin{bmatrix} T_1(x_1) \\ T_2(x_1, x_2) \\ \vdots \\ T_n(x_1, \dots, x_n) \end{bmatrix}.$$

They are so-called because their Jacobians are lower-triangular. Thus

$$|\det(\nabla T(x))| = \left|\prod_{i=1}^{n} \frac{\partial T_i}{\partial x_i}\right|,$$

# Flow Networks

In deep learning, it is common to use triangular mappings to construct 'flow networks.' Flow networks are bijective and have simple Jacobins. Examples include coupling flows[8] or autoregressive flows[9].

[8] Dinh, Krueger, and Bengio, "Nice: Non-linear independent components estimation".

[9] Kingma et al., "Improving variational inference with inverse autoregressive flow"; Huang et al., "Neural autoregressive flows".

[10] Teshima et al., "Coupling-based invertible neural networks are universal diffeomorphism approximators".

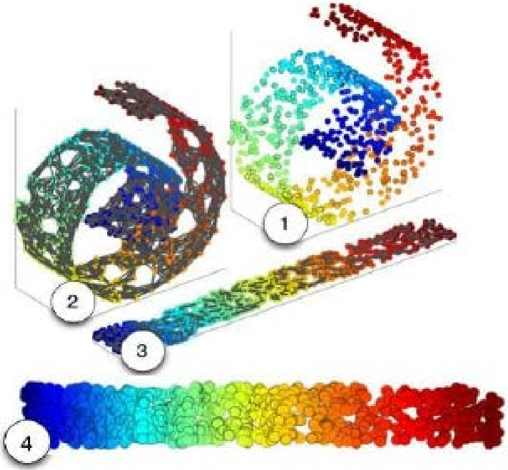
# Flow Networks

In deep learning, it is common to use triangular mappings to construct 'flow networks.' Flow networks are bijective and have simple Jacobins. Examples include coupling flows[8] or autoregressive flows[9].
Such networks are universal approximators of compact diffeomorphisms[10].
Diffeomorphsisms are smooth invertible maps whose gradient is always full-rank, i.e. functions for which the change of variables formula applies.

[8] Dinh, Krueger, and Bengio, "Nice: Non-linear independent components estimation".
[9] Kingma et al., "Improving variational inference with inverse autoregressive flow"; Huang et al., "Neural autoregressive flows".
[10] Teshima et al., "Coupling-based invertible neural networks are universal diffeomorphism approximators".

# Manifold Learning

Manifold learning is guided by the manifold hypothesis, the mantra that "high dimensional data are usually clustered around a low-dimensional manifold[a]." Figure from[b]

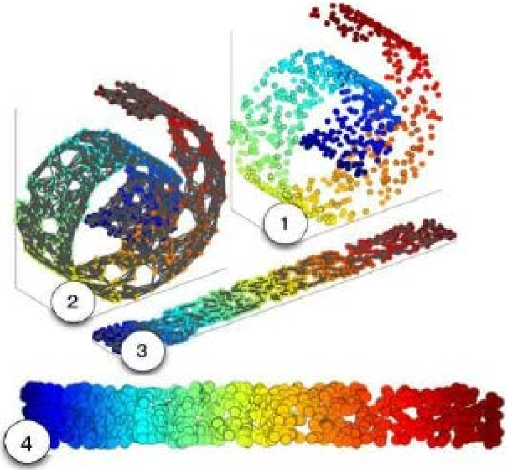



[a]Tenenbaum et al., "Mapping a manifold of perceptual observations".

[b]Weinberger and Saul, "Unsupervised learning of image manifolds by semidefinite programming".

# Manifold Learning

Manifold learning is guided by the manifold hypothesis, the mantra that “high dimensional data are usually clustered around a low-dimensional manifold[a].” Figure from[b] The prior universality results on flow networks applies to maps from $\mathbb{R}^n \to \mathbb{R}^n$, but such maps don't take advantage of the manifold hypothesis.

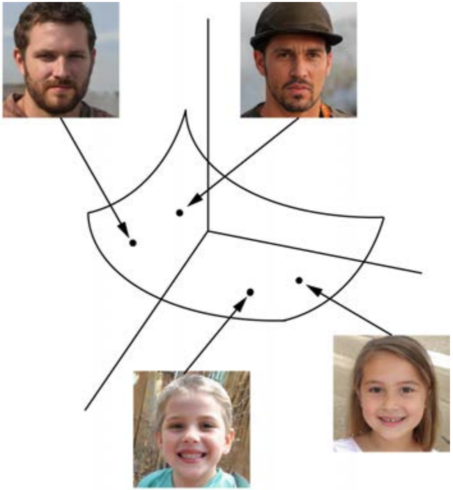



[a] Tenenbaum et al., “Mapping a manifold of perceptual observations”.

[b] Weinberger and Saul, “Unsupervised learning of image manifolds by semidefinite programming”.

# Manifold Learning of Faces

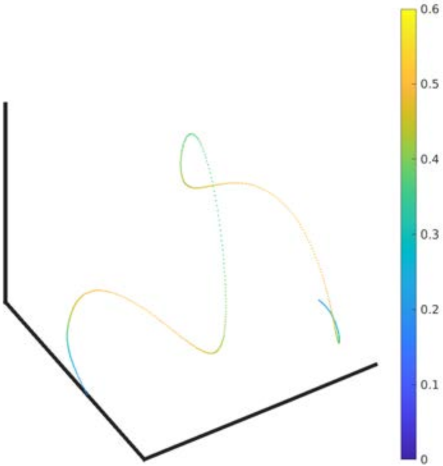
Applying the manifold hypothesis to the face distribution would looks like this.



# Manifold Learning, Smooth Embeddings

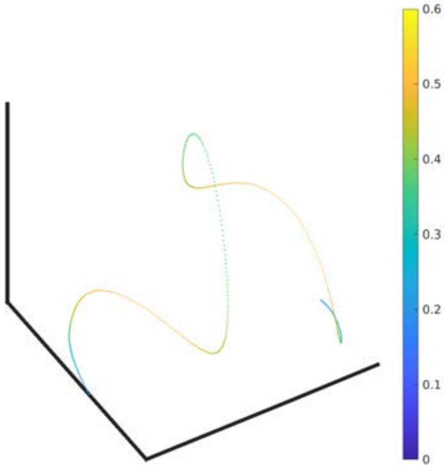
What about smooth embeddings $\mathbb{R}^n \hookrightarrow \mathbb{R}^m$ where $m >> n$? What types of networks are universal approximators w.r.t. smooth embeddings[a]? Do such networks allow for likelihood estimation?

[a] smooth embeddings are smooth bijections with smooth inverse



# Manifold Learning, Smooth Embeddings

What about smooth embeddings $\mathbb{R}^n \hookrightarrow \mathbb{R}^m$ where $m >> n$? What types of networks are universal approximators w.r.t. smooth embeddings[a]? Do such networks allow for likelihood estimation?
In P. et al. 2022[b], my coauthors and I answer this question.




---

[a] smooth embeddings are smooth bijections with smooth inverse

[b] Puthawala et al., "Universal Joint Approximation of Manifolds and Densities by Simple Injective Flows".

# Network Definition

If for $\ell = 1, \ldots, L$, $n_\ell \in \mathbb{N}$,

1. $\mathcal{T}_\ell^{n_\ell} \subset C(\mathbb{R}^{n_\ell}, \mathbb{R}^{n_\ell})$ is a flow network,
2. $\mathcal{R}_\ell^{n_{\ell-1}, n_\ell} \subset C(\mathbb{R}^{n_{\ell-1}}, \mathbb{R}^{n_\ell})$ is an injective ReLU layer,

then

$$\mathcal{E} = \mathcal{T}_L^{n_L} \circ \mathcal{R}_L^{n_{L-1}, n_L} \circ \cdots \circ \mathcal{T}_1^{n_1} \circ \mathcal{R}_1^{n_0, n_1} \circ \mathcal{T}_0^{n_0}$$

is always a family of injective mappings, where $\mathcal{H} \circ \mathcal{G} := \{h \circ g \colon h \in \mathcal{H}, g \in \mathcal{G}\}$.

# Network Definition

If for $\ell = 1, \dots, L$, $n_\ell \in \mathbb{N}$,

1. $\mathcal{T}_\ell^{n_\ell} \subset C(\mathbb{R}^{n_\ell}, \mathbb{R}^{n_\ell})$ is a flow network,
2. $\mathcal{R}_\ell^{n_{\ell-1}, n_\ell} \subset C(\mathbb{R}^{n_{\ell-1}}, \mathbb{R}^{n_\ell})$ is an injective ReLU layer,

then

$$\mathcal{E} = \mathcal{T}_L^{n_L} \circ \mathcal{R}_L^{n_{L-1}, n_L} \circ \cdots \circ \mathcal{T}_1^{n_1} \circ \mathcal{R}_1^{n_0, n_1} \circ \mathcal{T}_0^{n_0}$$

is always a family of injective mappings, where $\mathcal{H} \circ \mathcal{G} := \{h \circ g \colon h \in \mathcal{H}, g \in \mathcal{G}\}$.
When are these networks universal approximators?

# Embedding Gap

We call a function $f$ an embedding and denote if by $f \in \mathrm{emb}(X, Y)$ if $f : X \to Y$ is continuous, injective, and $f^{-1}\colon f(X) \to X$ is continuous.

## Definition (Embedding Gap)

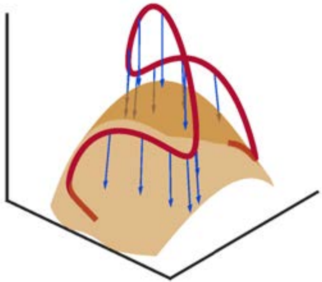
If,

- $K \subset \mathbb{R}^n$ and $W \subset \mathbb{R}^o$, both compact,
- $f \in \mathrm{emb}(K, \mathbb{R}^m)$, and $g \in \mathrm{emb}(W, \mathbb{R}^m)$

then we define

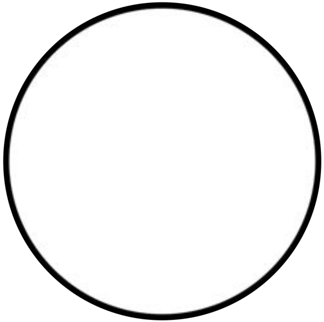
$$B_{K,W}(f,g) = \inf_{r \in \mathrm{emb}(f(K), g(W))} \|I - r\|_{L^\infty(f(K))}$$

where $I\colon f(K) \to f(K)$ is the identity function.

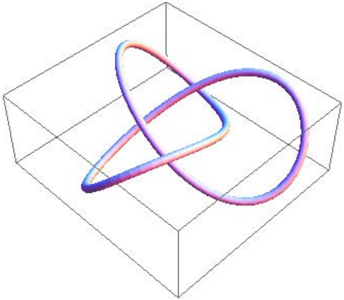


# Non-approximable manifolds

Let $K = S^1$ be a circle, and $f \in \mathrm{emb}(K, \mathbb{R}^3)$ an embedding of a trefoil knot into $\mathbb{R}^3$. There are no $E \in \mathcal{E} := \mathcal{T} \circ \mathcal{R}$ so that $E(K) = f(K)$.



(a) $K = S^1$



(b) Trefoil knot embedded in $\mathbb{R}^3$.

The trivial and trefoil knots are not equivalent.

# Extendable Embeddings

## Definition (Extendable Embedding)

With the above topological difficulty in mind, we define the set of extendable embeddings as

$$\mathcal{I}(\mathbb{R}^n, \mathbb{R}^m) := \{\Phi \circ R \in C(\mathbb{R}^n, \mathbb{R}^m) \colon R \in C(\mathbb{R}^n, \mathbb{R}^m), \Phi \in \mathbb{R}^m \to \mathbb{R}^m\}\,.$$

where $R$ is linear full-rank, and $\phi$ is a $C^1$-smooth diffeomorphism.

## Theorem (P. et al. 2022)

*When $m \geq 3n+1$ and $k \geq 1$, for any $C^k$ embedding $f \in \mathrm{emb}^k(\mathbb{R}^n, \mathbb{R}^m)$ and compact set $K \subset \mathbb{R}^n$, there is a map in the closures of the flow type neural network $E \in \mathcal{I}^k(\mathbb{R}^n, \mathbb{R}^m)$ such that $E(K) = f(K)$. Moreover,*

$$\mathcal{I}^k(K, \mathbb{R}^m) = \mathrm{emb}^k(K, \mathbb{R}^m)$$

# Manifold Universality

Let $\mathcal{F} = \mathrm{emb}(K, \mathbb{R}^m)$, or $\mathcal{F} = \mathcal{I}(K, \mathbb{R}^m)$.

**Theorem (P. et al. 2022)**

*Let $\mu \in \mathcal{P}(K)$ be an absolutely continuous measure w.r.t. Lebesgue measure and*

1. $\mathcal{R}_\ell^{n_{\ell-1}, n_\ell}$ *is injective,*
2. $\mathcal{T}_\ell^{n_\ell}$ *is injective, universal approximator of diffeomorphisms,*
3. $\mathcal{T}_0^{n}$ *is distributionally universal and injective*

*Then, there is a sequence of* $\{E_i\}_{i=1,\ldots,\infty} \subset \mathcal{E} := \mathcal{T}_L^{n_L} \circ \mathcal{R}_L^{n_{L-1}, n_L} \circ \cdots \circ \mathcal{R}_1^{n_0, n_1} \circ \mathcal{T}_0^{n_0}$ *such that*

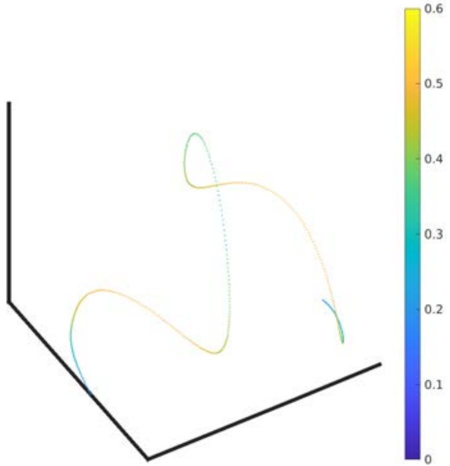
$$\lim_{i \to \infty} W_2 \left( F_{\#}\mu, E_{i\#}\mu \right) = 0.$$

# Decoupling

Optimality of layers of these deep neural networks can be established layer-by-layer.

Consider the problem of learning $\nu = f_{\#}\mu$ the following 1 dimensional distribution embedded in $\mathbb{R}^3$ with a network of the form

$$F_\theta(x) = T_2 \circ R_2 \circ T_1 \circ R_1 \circ T_0(x)$$

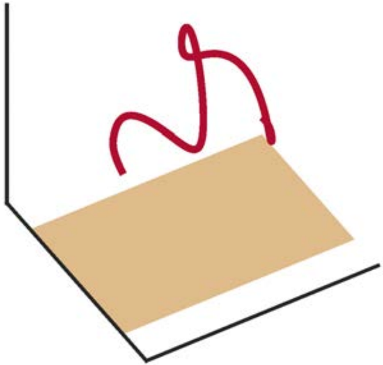


# Decoupling: Step one

First, we can update $T_2 \circ R_2$ to decrease

$$B_{K,W}(f, T_2 \circ R_2)$$

$$F_\theta(x) = T_2 \circ R_2 \circ T_1 \circ R_1 \circ T_0(x)$$

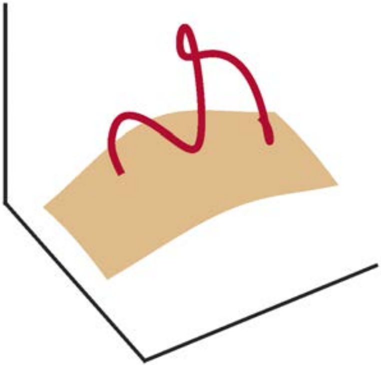


# Decoupling: Step one

First, we can update $T_2 \circ R_2$ to decrease

$$B_{K,W}(f, T_2 \circ R_2)$$

$$F_\theta(x) = T_2 \circ R_2 \circ T_1 \circ R_1 \circ T_0(x)$$

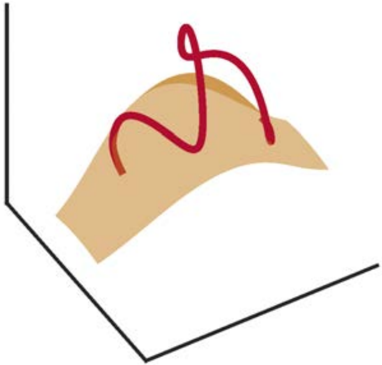


# Decoupling: Step one

First, we can update $T_2 \circ R_2$ to decrease

$$B_{K,W}(f, T_2 \circ R_2)$$

$$F_\theta(x) = T_2 \circ R_2 \circ T_1 \circ R_1 \circ T_0(x)$$

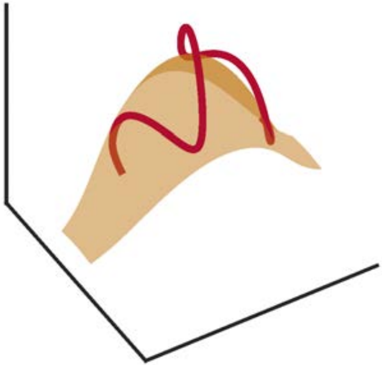


# Decoupling: Step one

First, we can update $T_2 \circ R_2$ to decrease

$$B_{K,W}(f, T_2 \circ R_2)$$

$$F_\theta(x) = T_2 \circ R_2 \circ T_1 \circ R_1 \circ T_0(x)$$

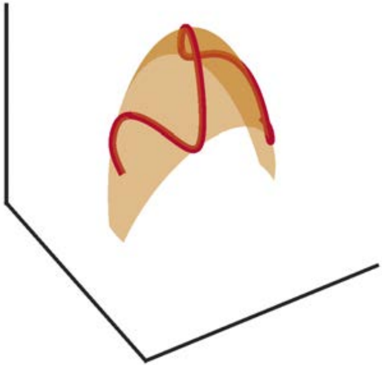


# Decoupling: Step one

First, we can update $T_2 \circ R_2$ to decrease

$$B_{K,W}(f, T_2 \circ R_2)$$

$$F_\theta(x) = T_2 \circ R_2 \circ T_1 \circ R_1 \circ T_0(x)$$

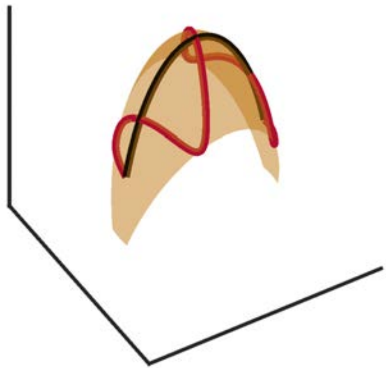


# Decoupling: Step two

Now fix $T_2 \circ R_2$ and update $T_1 \circ R_1$ to decrease

$$B_{K,W}(f, T_2 \circ R_2 \circ T_1 \circ R_1)$$

$$F_\theta(x) = T_2 \circ R_2 \circ T_1 \circ R_1 \circ T_0(x)$$

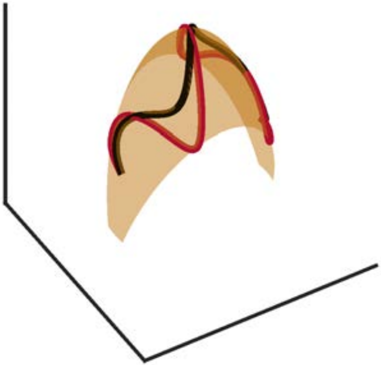


# Decoupling: Step two

Now fix $T_2 \circ R_2$ and update $T_1 \circ R_1$ to decrease

$$B_{K,W}(f, T_2 \circ R_2 \circ T_1 \circ R_1)$$

$$F_\theta(x) = T_2 \circ R_2 \circ T_1 \circ R_1 \circ T_0(x)$$

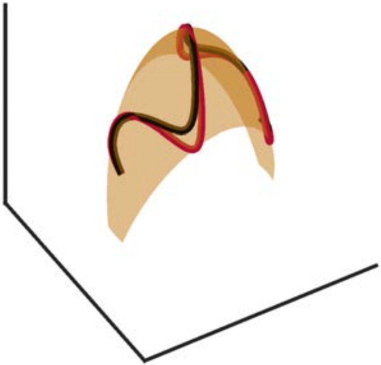


# Decoupling: Step two

Now fix $T_2 \circ R_2$ and update $T_1 \circ R_1$ to decrease

$$B_{K,W}(f, T_2 \circ R_2 \circ T_1 \circ R_1)$$

$$F_\theta(x) = T_2 \circ R_2 \circ T_1 \circ R_1 \circ T_0(x)$$

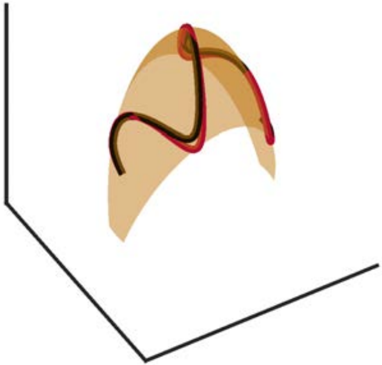


# Decoupling: Step two

Now fix $T_2 \circ R_2$ and update $T_1 \circ R_1$ to decrease

$$B_{K,W}(f, T_2 \circ R_2 \circ T_1 \circ R_1)$$

$$F_\theta(x) = T_2 \circ R_2 \circ T_1 \circ R_1 \circ T_0(x)$$

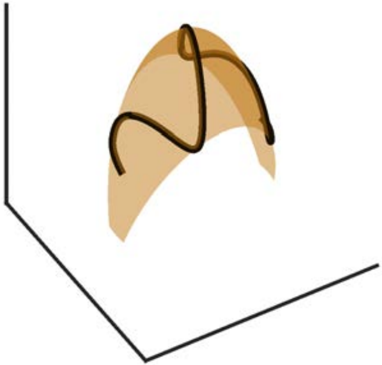


# Decoupling: Step two

Now fix $T_2 \circ R_2$ and update $T_1 \circ R_1$ to decrease

$$B_{K,W}(f, T_2 \circ R_2 \circ T_1 \circ R_1)$$

$$F_\theta(x) = T_2 \circ R_2 \circ T_1 \circ R_1 \circ T_0(x)$$

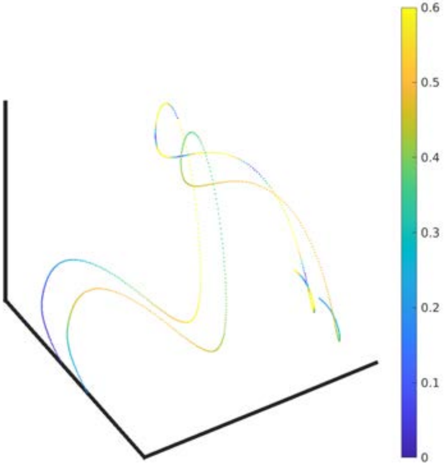


# Decoupling: Step three

Now fix $T_2 \circ R_2 \circ T_1 \circ R_1$ and update $T_0$ to decrease

$$W_2(\nu, F_{\theta,\#}\mu)$$

$$F_\theta(x) = T_2 \circ R_2 \circ T_1 \circ R_1 \circ T_0(x)$$

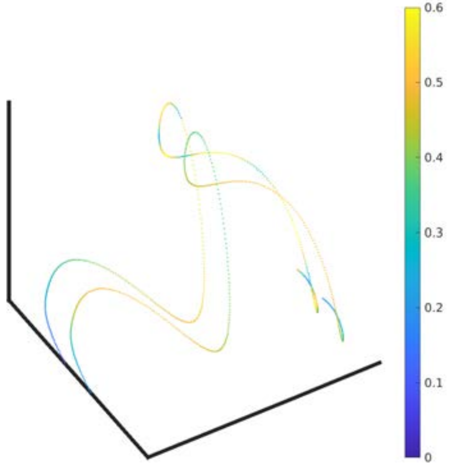


# Decoupling: Step three

Now fix $T_2 \circ R_2 \circ T_1 \circ R_1$ and update $T_0$ to decrease

$$W_2(\nu, F_{\theta,\#}\mu)$$

$$F_\theta(x) = T_2 \circ R_2 \circ T_1 \circ R_1 \circ T_0(x)$$

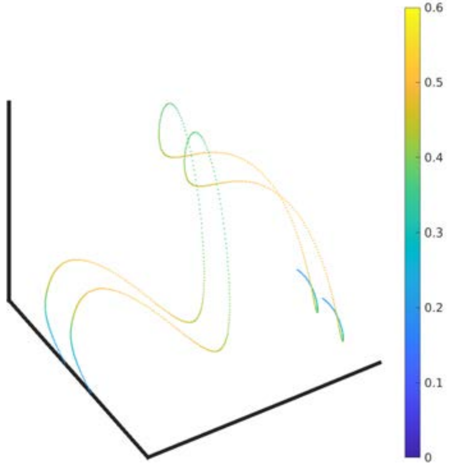


# Decoupling: Step three

Now fix $T_2 \circ R_2 \circ T_1 \circ R_1$ and update $T_0$ to decrease

$$W_2(\nu, F_{\theta,\#}\mu)$$

$$F_\theta(x) = T_2 \circ R_2 \circ T_1 \circ R_1 \circ T_0(x)$$



# Theory of Invertible and Injective Deep Neural Networks for Likelihood Estimation and Uncertainty Quantification

Michael Puthawala [1]

Matti Lassas[2], Ivan Dokmanić[3], Pekka Pankka[2], Maarten de Hoop[4]

[1]South Dakota State University, [2]University of Helsinki, [3]University of Basel, [4]Rice University

June 14, 2023