Deep learning and dynamical systems: applications in neuroimaging

François Rousseau

Institut Mines-Télécom

« The ever-increasing technological demands of our modern society require innovative approaches to highly demanding control problems. Artificial neural networks with their massive parallelism and learning capabilities offer the promise of better solutions, at least to some problems. »

« Interest in neural networks has made a comeback in this decade after a period of relative inactivity following the shortcomings of early neural networks… »

« Today, the need to control, in a better way, increasingly complex dynamical systems under significant uncertainty has led to a reevaluation of the conventional control methods … »

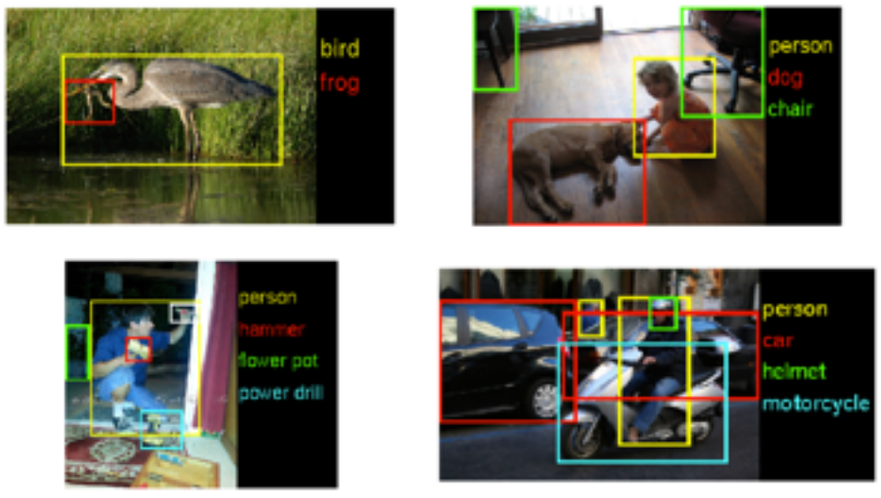
Quotes from
PJ Antsaklis, Neural networks for control systems
IEEE Transactions on Neural Networks 1 (2), 242-244, 1990
(Special issue)

Bialasiewicz & Soloway, Neural network modeling of dynamical systems, 1990, (20 citations)
Narendra & Parthasarathy, Identification and control of dynamical systems using neural networks, 1990, (10775 citations)
...

About Deep Learning…

- Why is deep learning so popular?
  - Very good performance
  - End-to-end approach
- Why now?
  - Automatic differentiation
  - Hardware: GPU
  - Datasets: ImageNet, Kaggle, etc.
  - User friendly libraries: PyTorch or Keras (and previously Caffe & Theano)

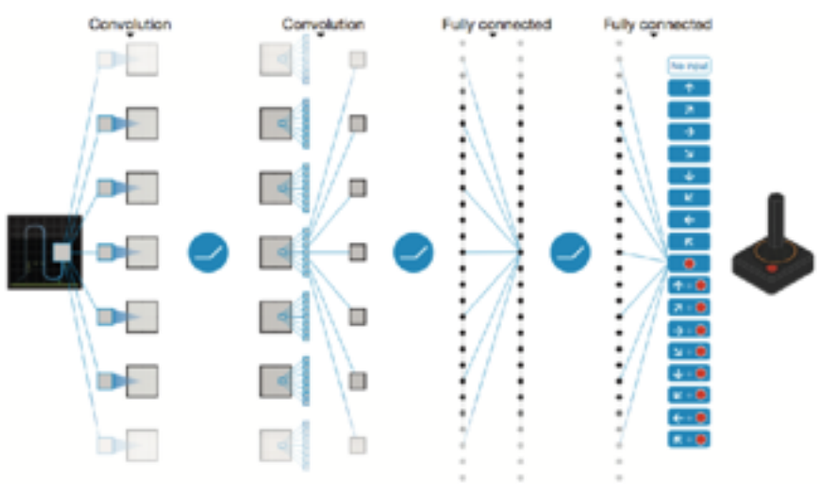
Large Scale Visual Recognition Challenge



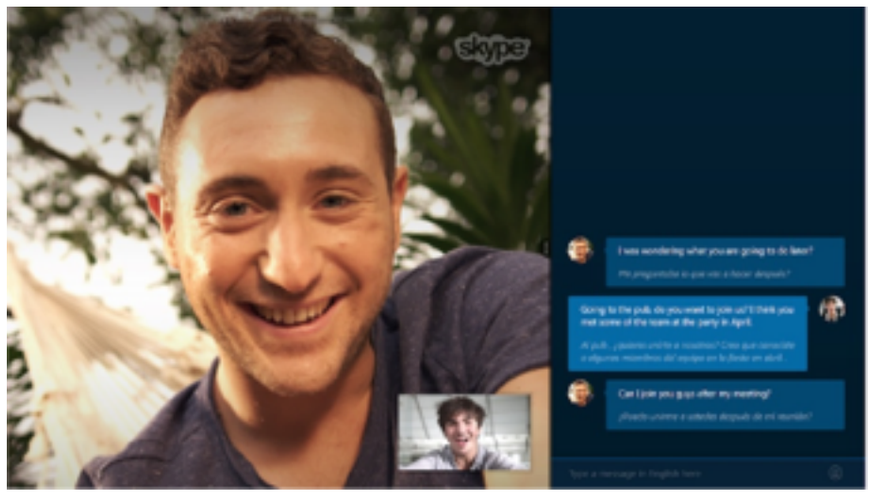
Real Time Recognition

Image captioning

Algorithm playing ATARI



Real Time Translation

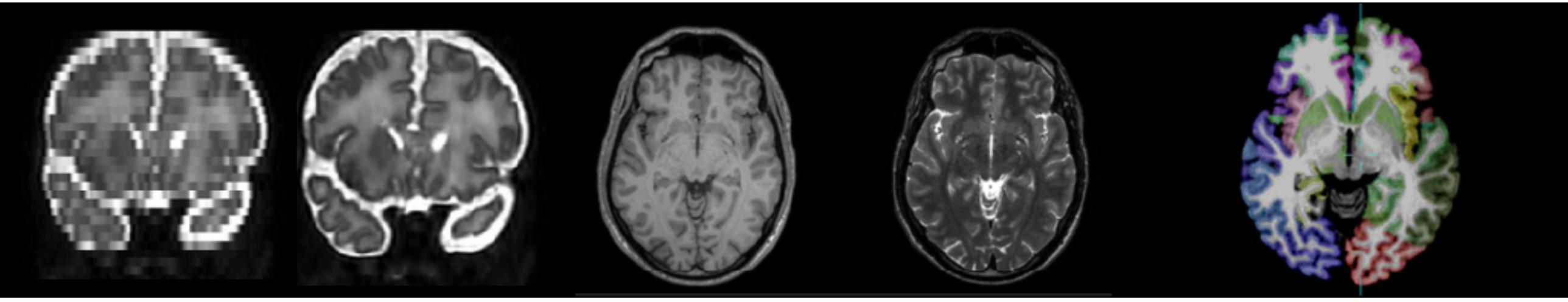


Text generator

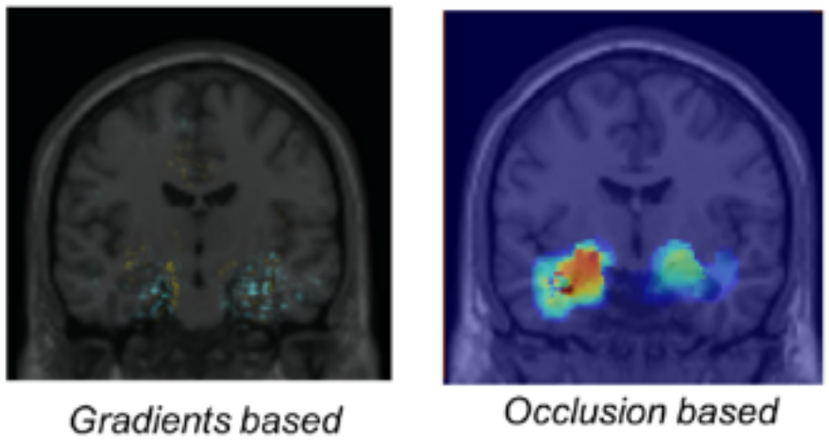
Super-resolution

Image synthesis

Image segmentation



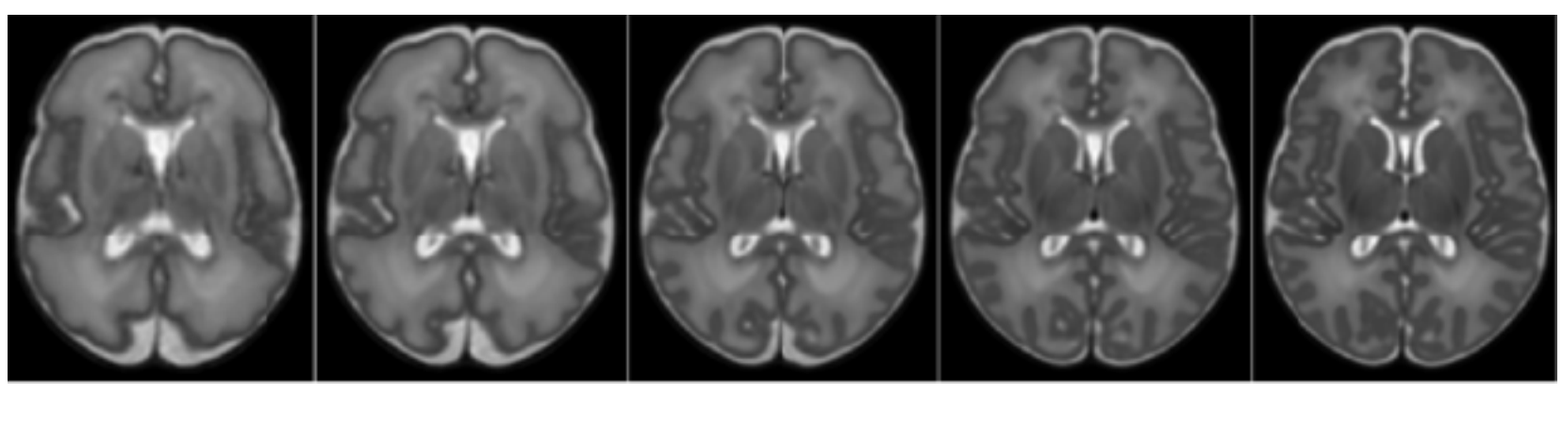
Class-sensitive visualization



Gradients based

Occlusion based

Conditional template



Zhou et al. A review of deep learning in medical imaging: Imaging traits, technology trends, case studies with progress highlights, and future promises, 2020
Burgos et al. Deep learning for brain disorders: from data processing to disease treatment, 2021.

Overview

Dynamical systems

Dynamical systems

- A dynamical system is a triple $(\mathcal{S}, \mathcal{T}, \Phi)$

  $\mathcal{S}$ is the state space

  $\mathcal{T}$ is the parameter space

  $\Phi : (\mathcal{T}, \mathcal{S}) \to \mathcal{S}$ is the evolution (also called flow)

Dynamical systems can be useful for prediction and discovery of laws (flows).

Dynamical systems

- Consider the following equations:

$$\frac{dx(t)}{dt} = f(x(t), t)$$

$$x(t_0) = x_0$$

- A *initial value problem* (IVP) consists in finding a function x solution to the ODE given f and x_0

- A IVP is a prediction problem.

Dynamical systems

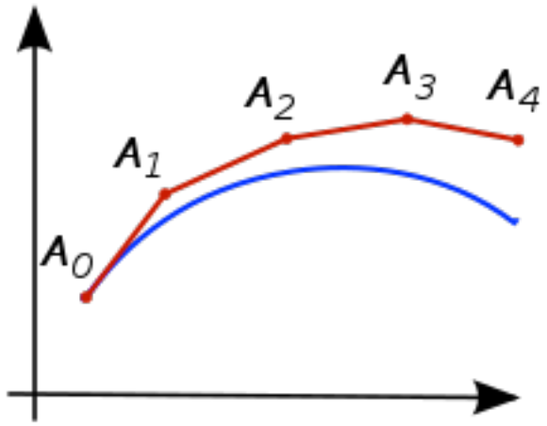
- Goal: find numerical solutions of IVP

$$\frac{dx(t)}{dt} = f(x(t), t)$$

$$x(t_0) = x_0$$

- Euler method: let define a step-size $h > 0$ and compute the trajectory as follows

$$x_{n+1} = x_n + hf(x_n(t), t)$$
$$t_{n+1} = t_n + h$$



(from Wikipedia)

Dynamical systems

- Consider the following equations:

$$\frac{dx(t)}{dt} = f(x(t), t)$$

$$x(t_0) = x_0$$

- Goal: given a point x_1, find f such that $x(t_1) = x_1$

- Finding an « optimal » f is related to *inverse problems* and *optimal control*.

- The purpose of the dynamical system point of view is to study the trajectories (also called flow).

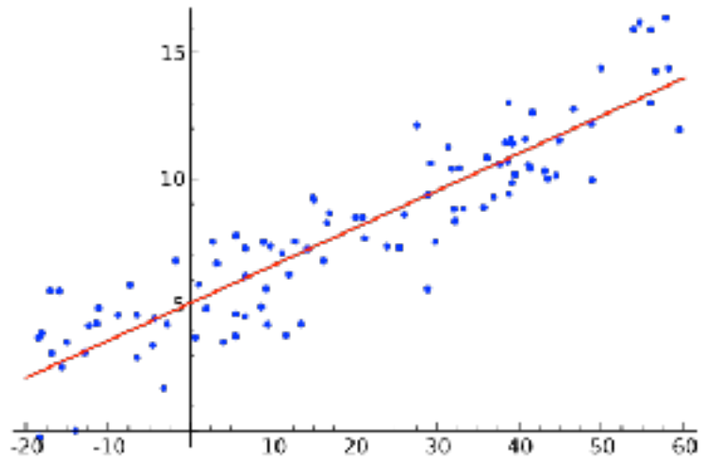
Overview

Dynamical systems

Neural Networks

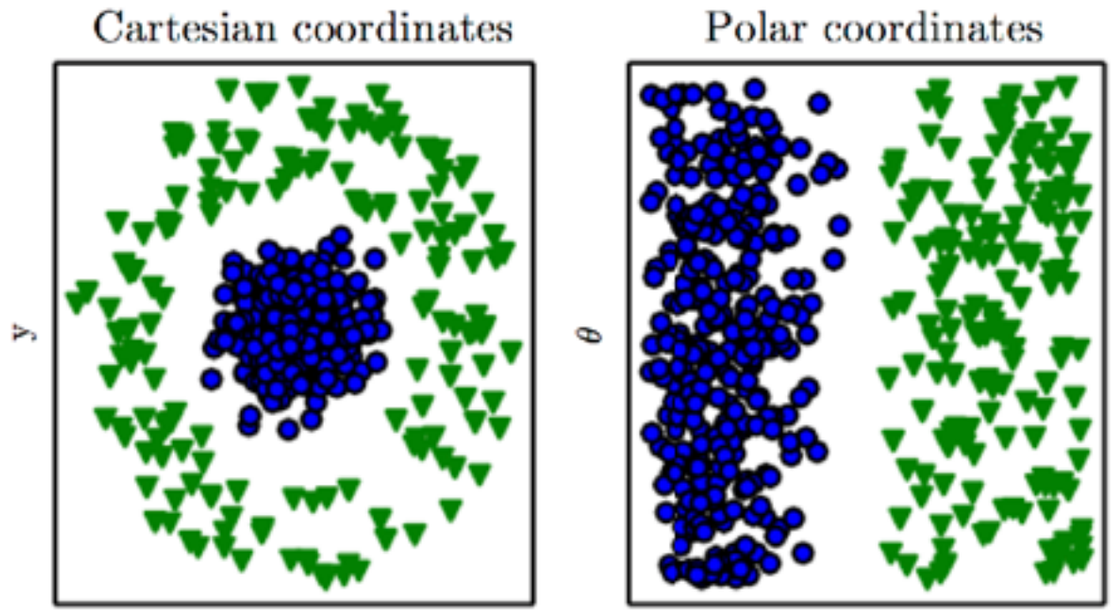
Machine learning vocabulary

- Data: input x and output y
- Task: regression, classification
- Experience: supervised (x_i, y_i) or unsupervised (x_i)
- Performance measure: accuracy of the model or error rate



Classification

- Data: input x and output y
- Task: classification
- Experience: supervised (x_i, y_i) or unsupervised (x_i)
- Performance measure: accuracy of the model or error rate



$$\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{w}) = \phi(\boldsymbol{x}; \boldsymbol{\theta})^\top \boldsymbol{w}$$

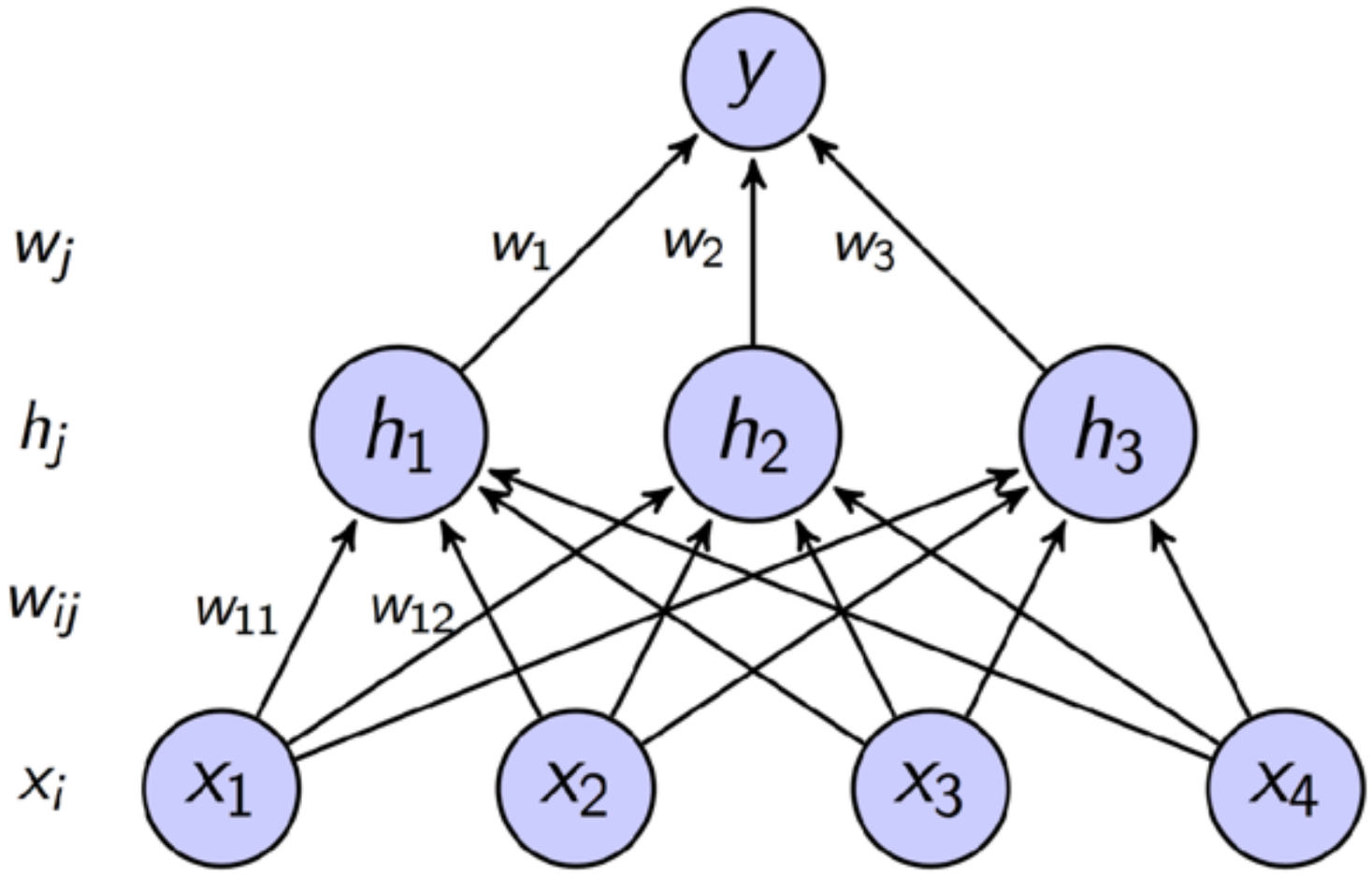
Neural networks

$$\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$$

Parameters to learn

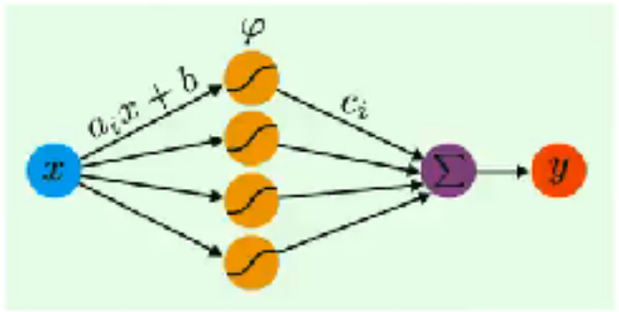
- Objective: approximate a function f^*
- NN: composition of functions $f = f_n \circ \ldots \circ f_2 \circ f_1$
- How: estimate parameters θ of the neural network via a minimization problem: $\arg\min_{\theta} \sum_{i} (f(x_i; \theta) - y_i)^2$

Neural networks



$$f(x) = \sigma(\sum_j w_j \cdot h_j) = \sigma(\sum_j w_j \cdot \sigma(\sum_i w_{ij} x_i))$$

Universal Approximation Theorem

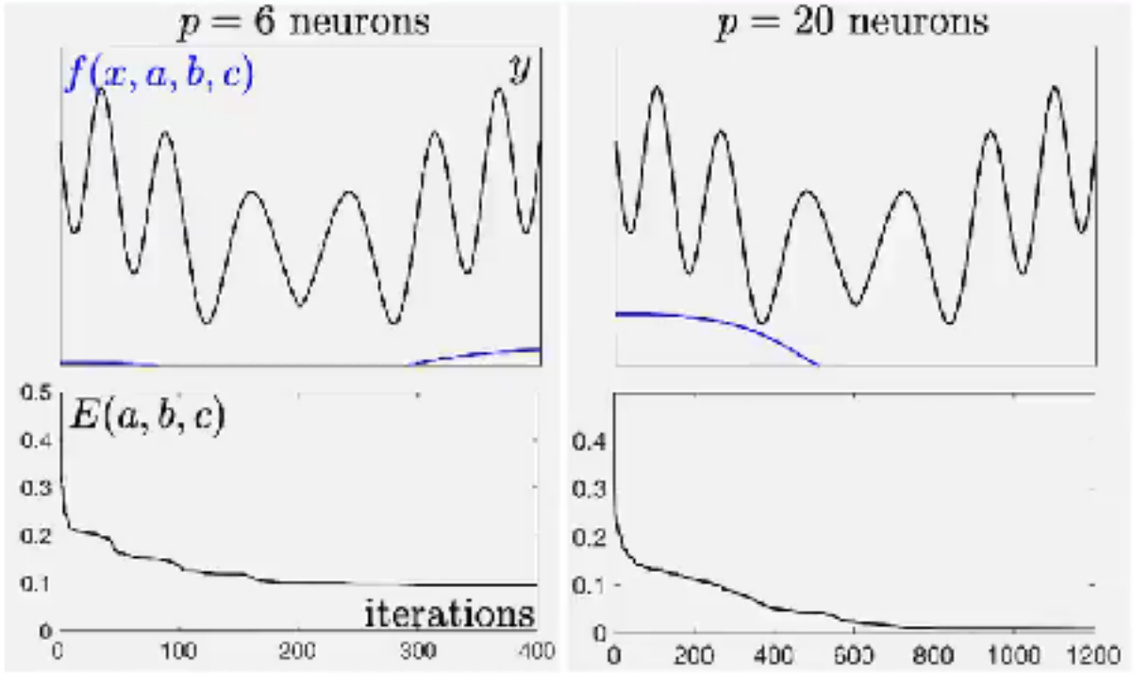


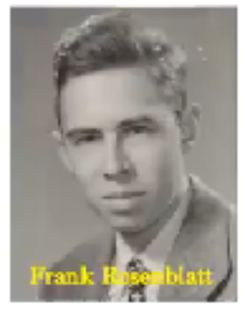
1 hidden layer perceptron:

$$y \approx f(x,a,b,c) \overset{\text{def.}}{=} \sum_{i=1}^{p} c_i \varphi(a_i x + b_i)$$

Training:

$$\min_{a,c,b} E(a,b,c) \overset{\text{def.}}{=} \frac{1}{2n} \sum_{k=1}^{n} |y_k - f(x_k,a,b,c)|^2$$

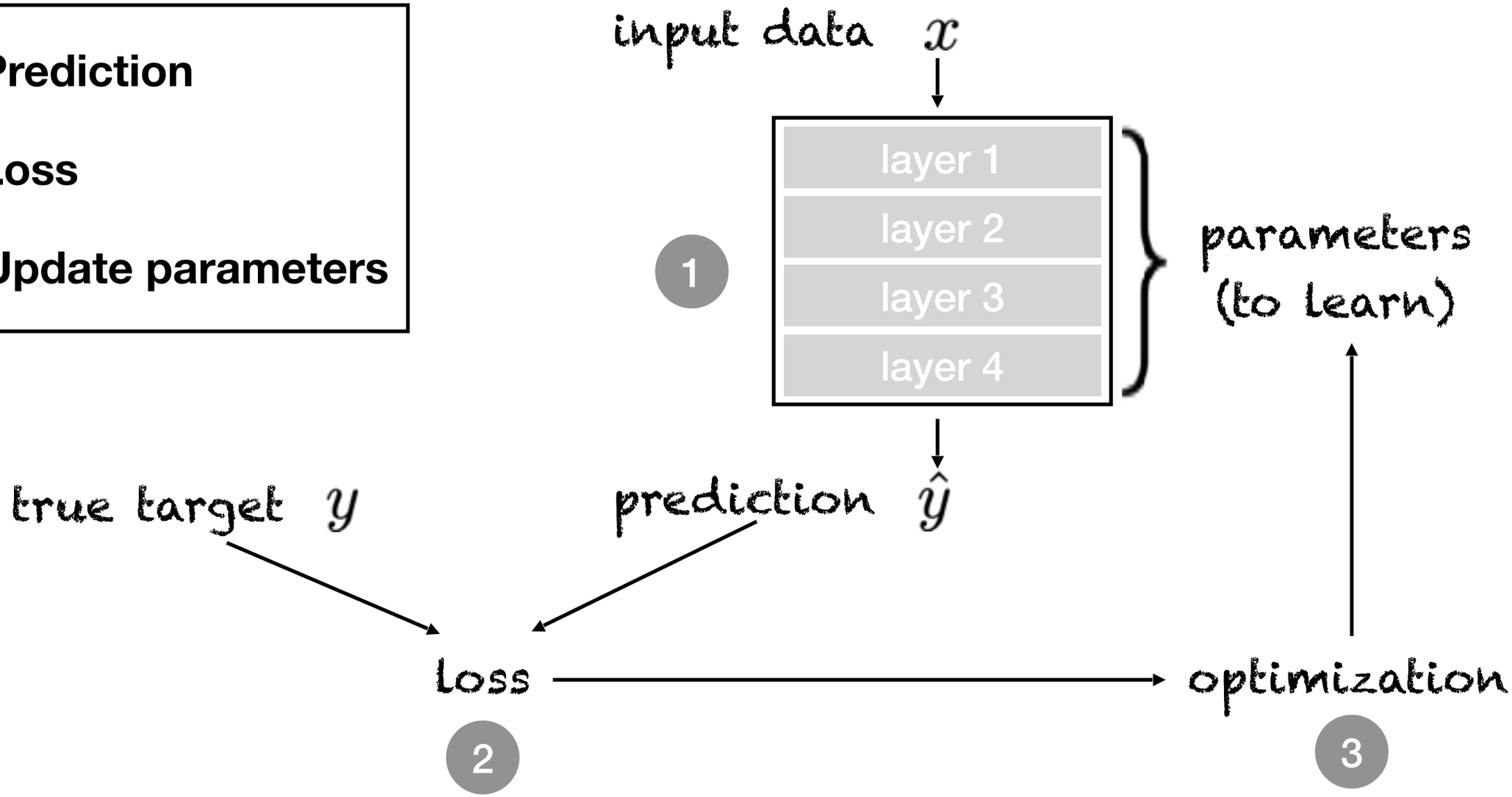




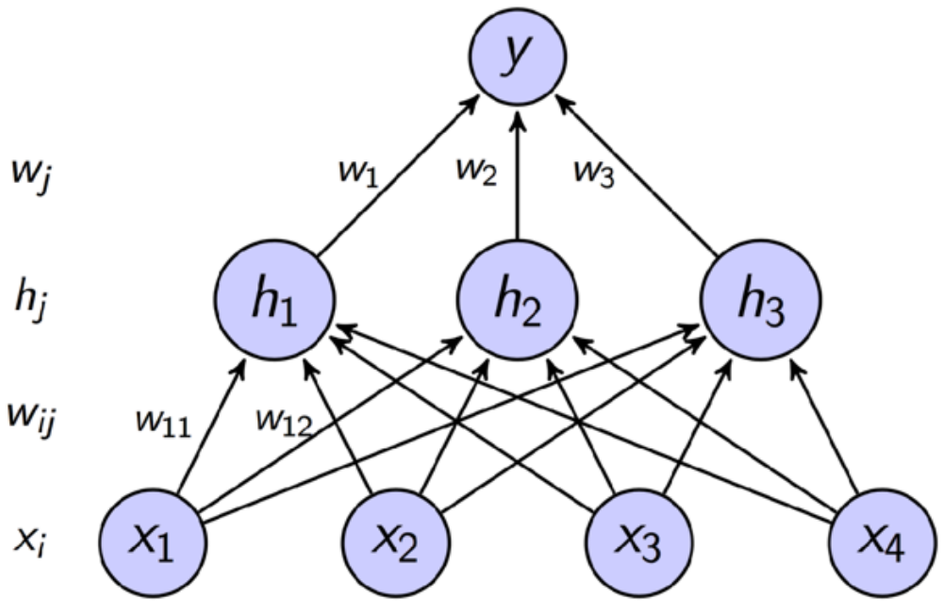
[From Gabriel Peyré]

Overview of DL

1. **Prediction**
2. **Loss**
3. **Update parameters**



In practice




```
import torch
import torch.nn as nn
```

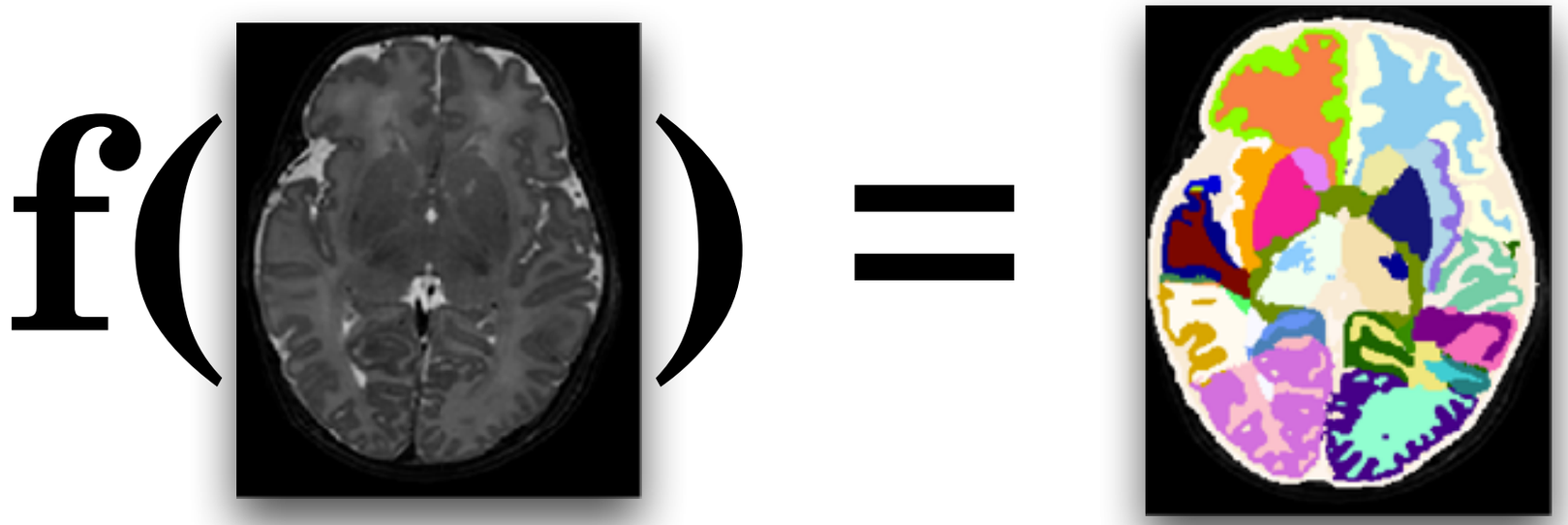
```
class Net(nn.Module):

  def __init__(self):
    super(Net, self).__init__()
    self.fc1 = nn.Linear(4,3)
    self.fc2 = nn.Linear(3,1)

  def forward(self, x):
    x = self.fc1(x)
    x = F.relu(x)
    x = self.fc2(x)
    x = F.softmax(x)
    return x

net = Net()
```

```
criterion = nn.MSELoss()
```

```
optimizer = torch.optim.SGD(net.parameters(), lr=0.01)
```

```
for i in range(epochs):       # training loop
  y_pred = net(x)             # compute the prediction using current parameters
  loss = criterion(y_pred,y) # compute loss
  optimizer.zero_grad()       # initialize the gradient to zero
  loss.backward()             # back propagation
  optimizer.step()            # update the parameters values using the gradient
```

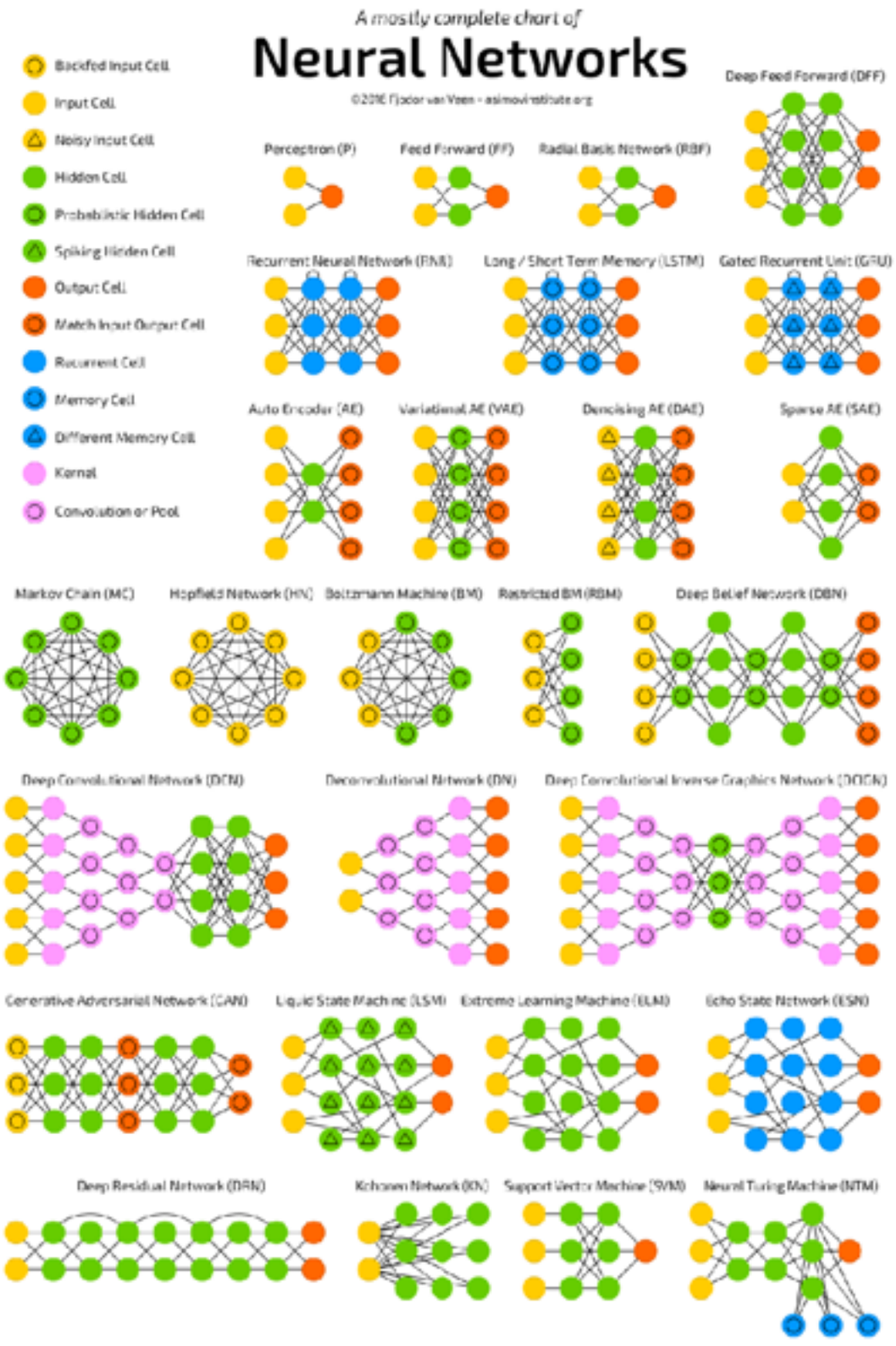
Deep Neural Networks

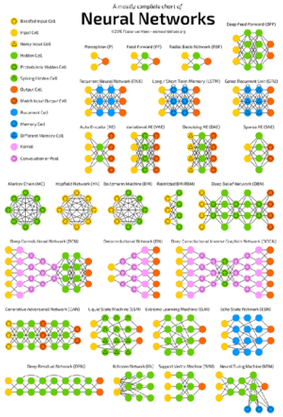


f can be approximated by a deep neural network :

$$f(\mathbf{X}, \{\mathbf{W}_i\})_{i=1}^{n}, \{\mathbf{b}_i\})_{i=1}^{n}) = \sigma\left(\mathbf{W}_n^T \sigma(\ldots \sigma(\mathbf{W}_2^T \sigma(\mathbf{W}_1^T \mathbf{X} + \mathbf{b}_1) + \mathbf{b}_2)\ldots) + \mathbf{b}_n\right)$$

Network Zoo

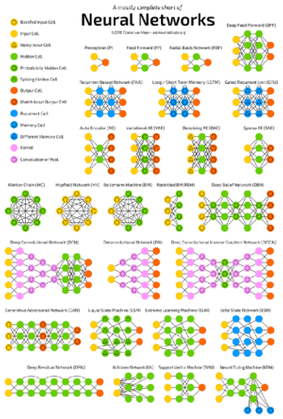

A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probabilistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)



CNN Zoo

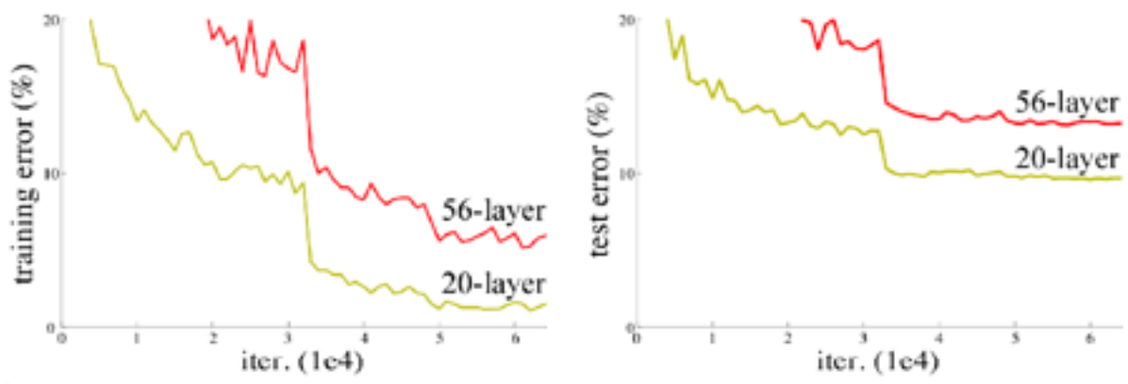
LeNet 28×28 (1998)	AlexNet 224×224 (2012)	VGG 224×224 (9/2014)	GoogLeNet 224×224 (9/2014)	Inception BN 224×224 (2/2015)	Inception V3 299×299 (12/2015)	Resnet (n=9, 56 Layers) 28×28 (12/2015)
						

http://josephpcohen.com



CNN Zoo

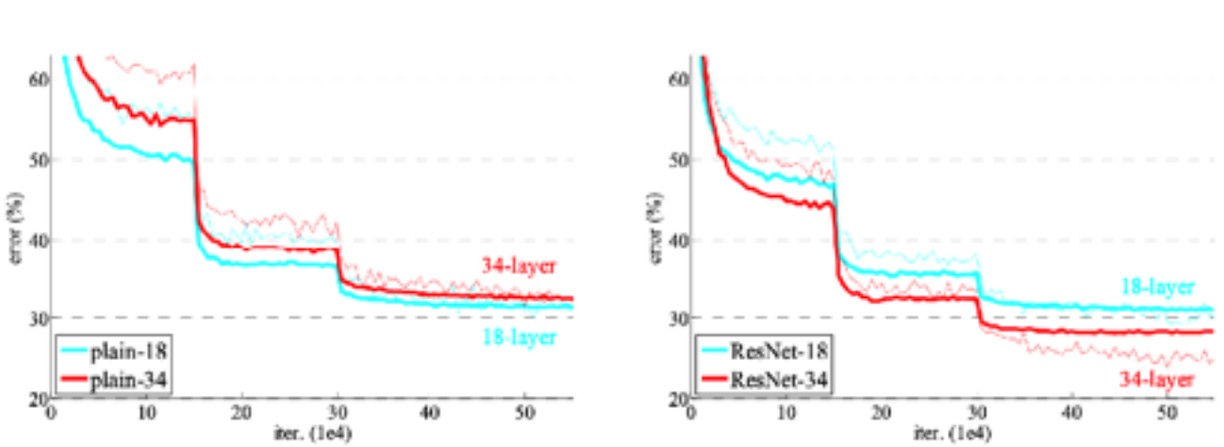
LeNet 28×28 (1998)	AlexNet 224×224 (2012)	VGG 224×224 (9/2014)	GoogLeNet 224×224 (9/2014)	Inception BN 224×224 (2/2015)	Inception V3 299×299 (12/2015)	Resnet (n=9, 56 Layers) 28×28 (12/2015)
						



Training error (left) and test error (right) on CIFAR 10 with « plain » networks.

Residual networks

- Motivations: « depth of representations is of central importance » and « deeper networks are more difficult to train »

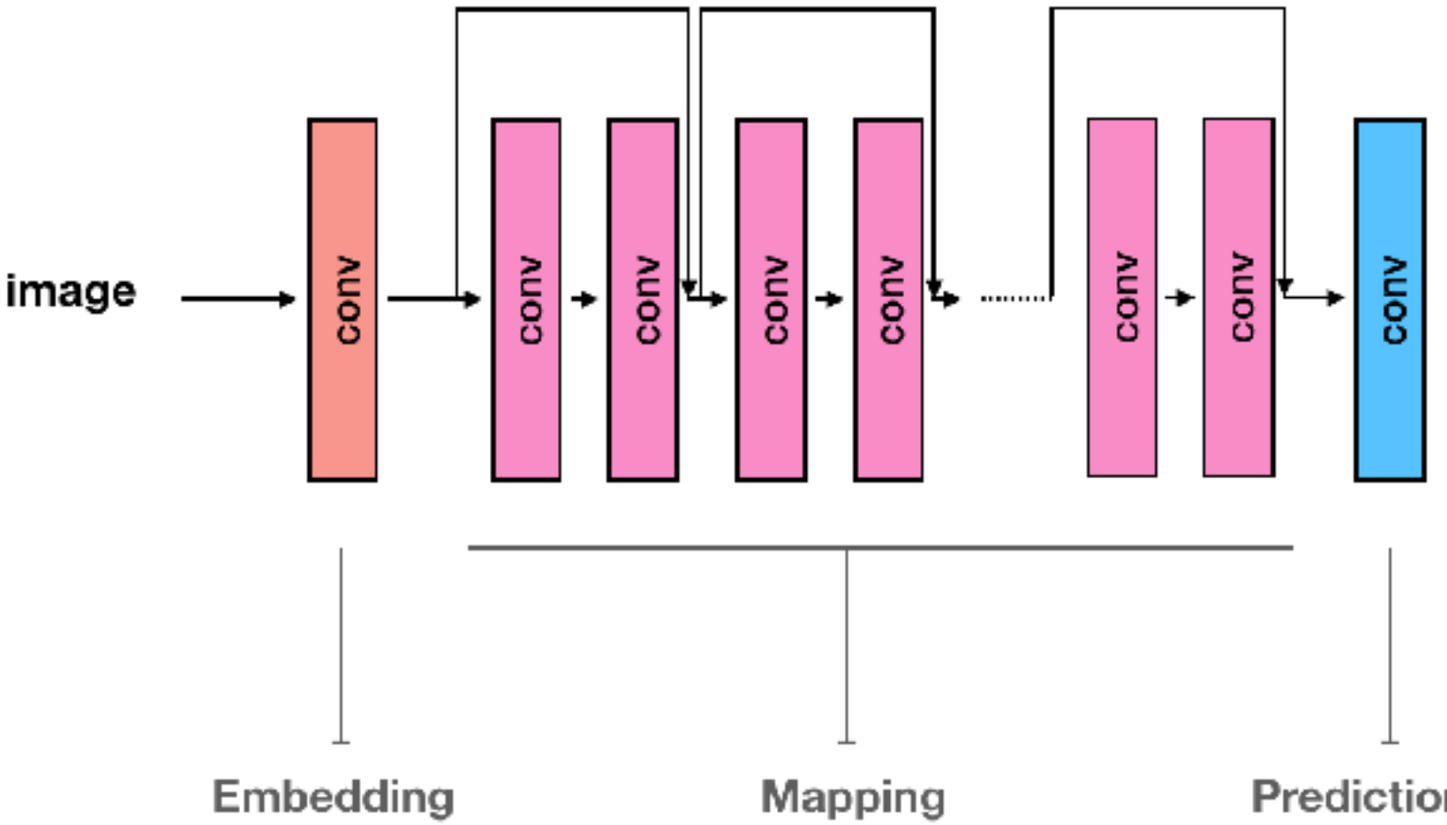


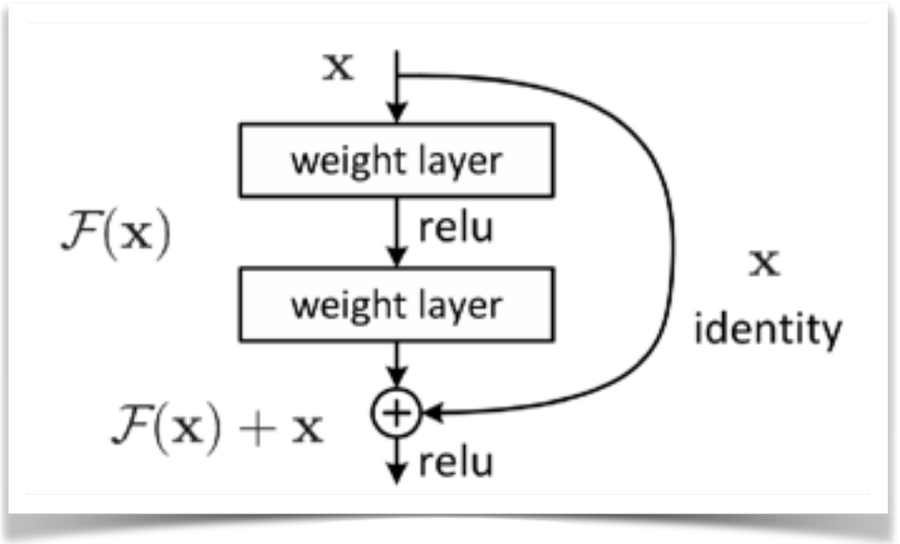
Training on ImageNet. Plain networks (left) vs residual networks (right).

- Exploring over 1000 layers: no optimization difficulty

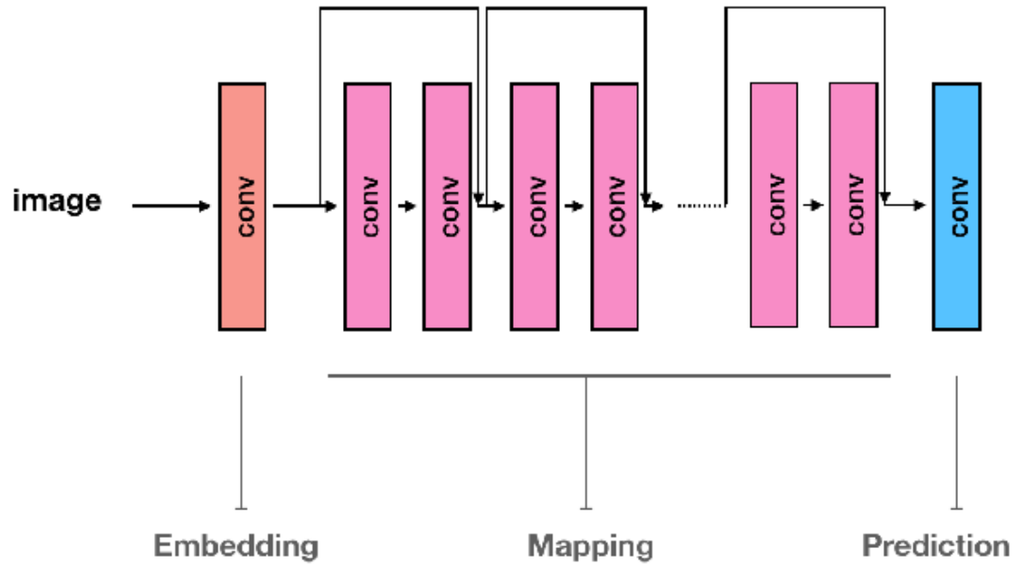
He et al. *Deep residual learning for image recognition*, CVPR 2015.

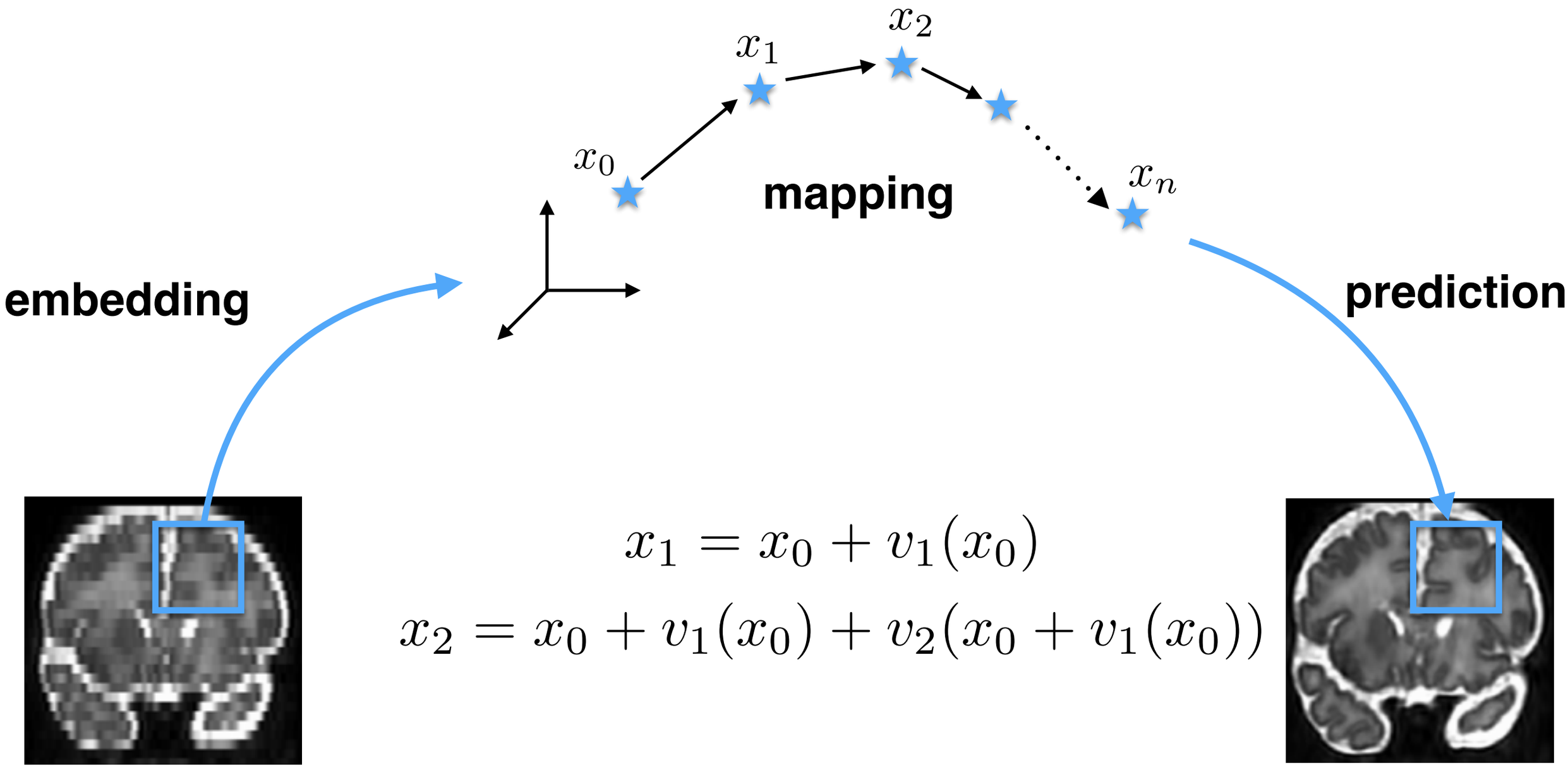
Overview of Residual Networks





He et al. *Deep residual learning for image recognition*, CVPR 2015.

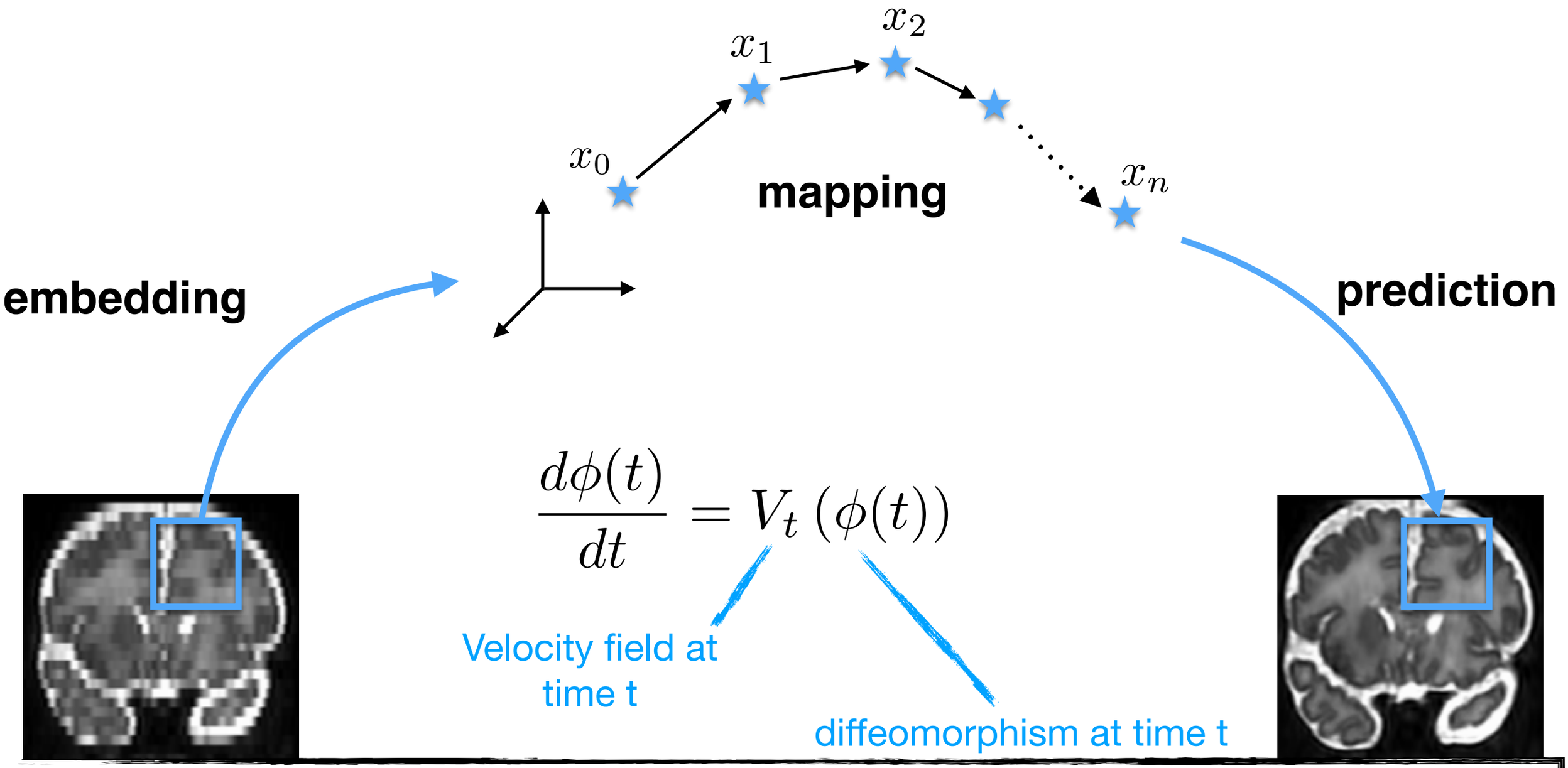




$$x_1 = x_0 + v_1(x_0)$$

$$x_2 = x_0 + v_1(x_0) + v_2(x_0 + v_1(x_0))$$

ResNets as a dynamical system



Rousseau et al. *Residual networks as flows of diffeomorphisms,* JMIV, 2019.

Diffeomorphic mapping

- A diffeomorphism is an invertible function that maps one differentiable manifold to another such that both the function and its inverse are smooth.
- Large deformation diffeomorphic metric mapping:

$$\frac{d\phi(t)}{dt} = V_t\left(\phi(t)\right)$$

velocity vector field at time t

diffeomorphism at time t

- Stationary velocity fields:

$$\frac{d\phi(t)}{dt} = V\left(\phi(t)\right)$$

Beg et al. *Computing Large Deformation Metric Mappings via Geodesic Flows of Diffeomorphisms,* IJCV, 2005.
Arsigny *Processing Data in Lie Groups: an Algebraic Approach*, PhD thesis 2006.

Diffeomorphic mapping

- Estimation of a diffeomorphism:

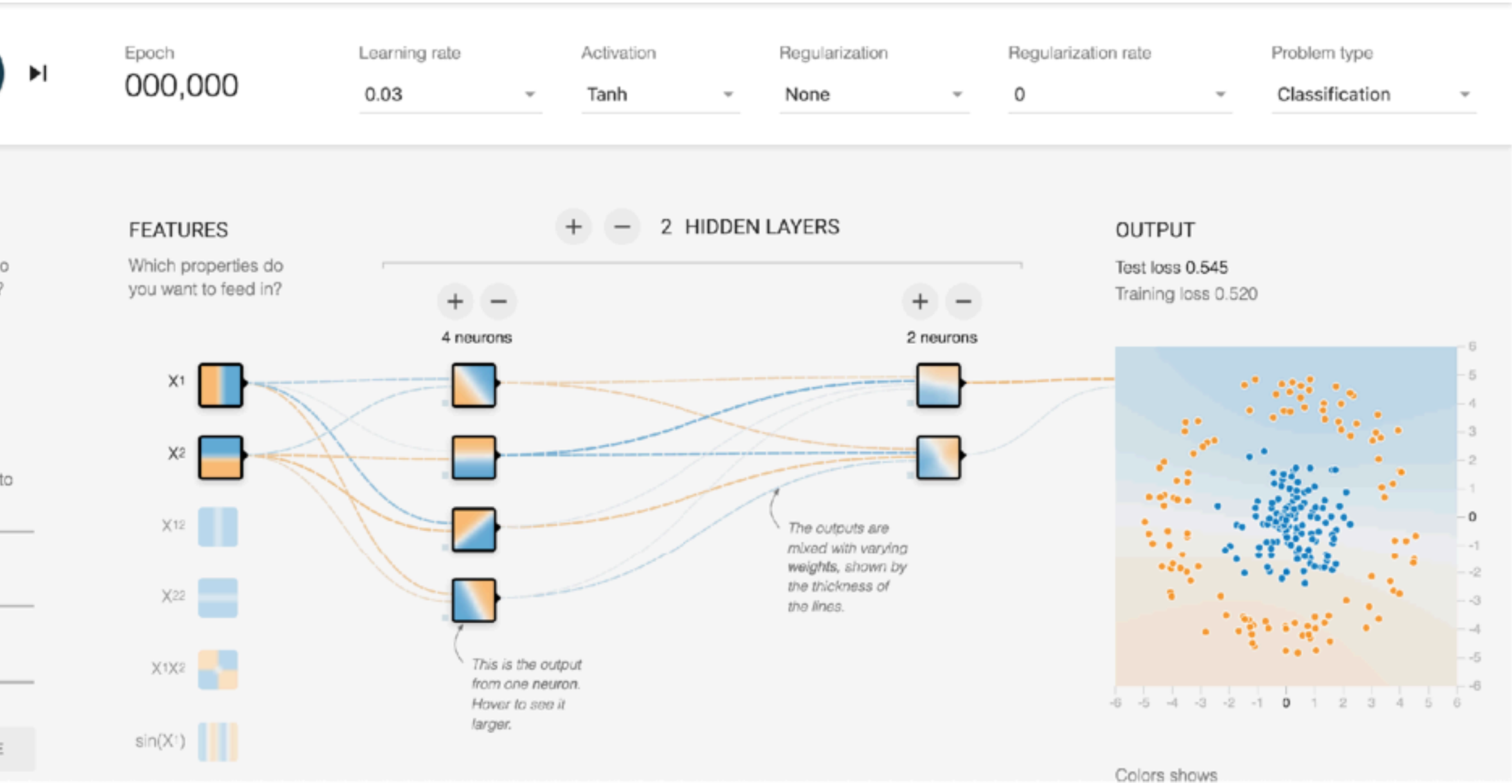
$$\widehat{(V_t)}, \widehat{\theta} = \arg\min_{(V_t),\theta} loss\left(\left\{\mathcal{L}_\theta\left(\phi(1)\left(X_i\right)\right), Y_i\right\}_i\right)$$

$$\text{subject to} \begin{cases} \dfrac{d\phi(t)}{dt} &= V_t\left(\phi(t)\right) \\ \\ \phi(0) &= I \end{cases}$$

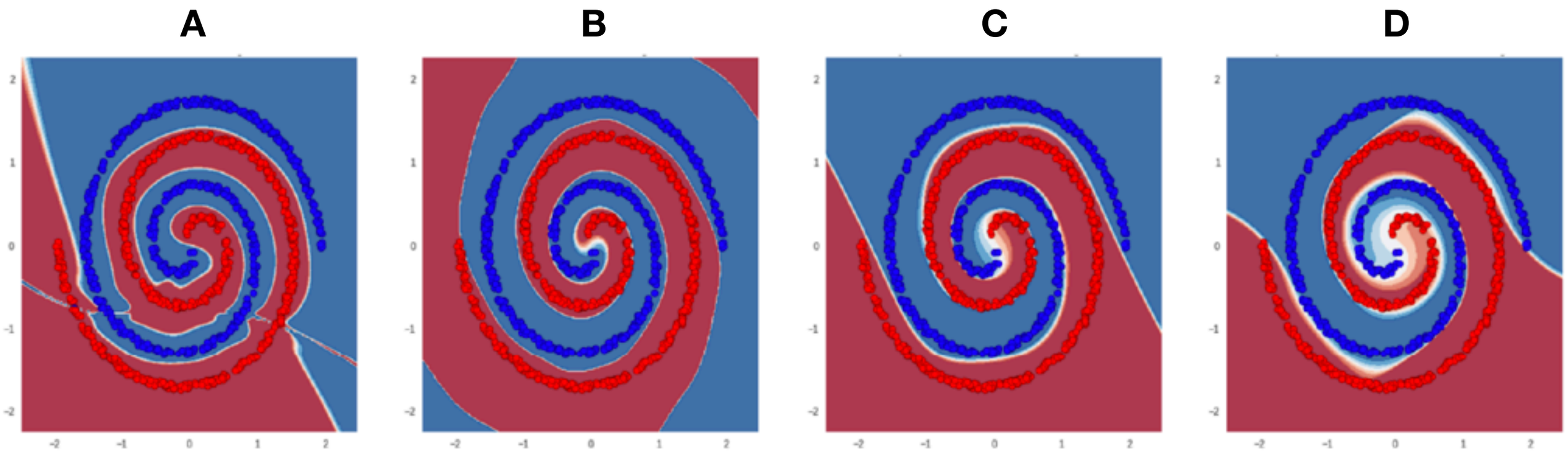
- Discrete parametrization of the velocity field: $V_t(\mathbf{x}) = \sum_i \nu_{t,i} f_{t,i}(\mathbf{x})$

- Inverse consistency: $\phi(1) \circ \phi(-1) = \phi(-1) \circ \phi(1) = \phi(0)$

Tensorflow Playground



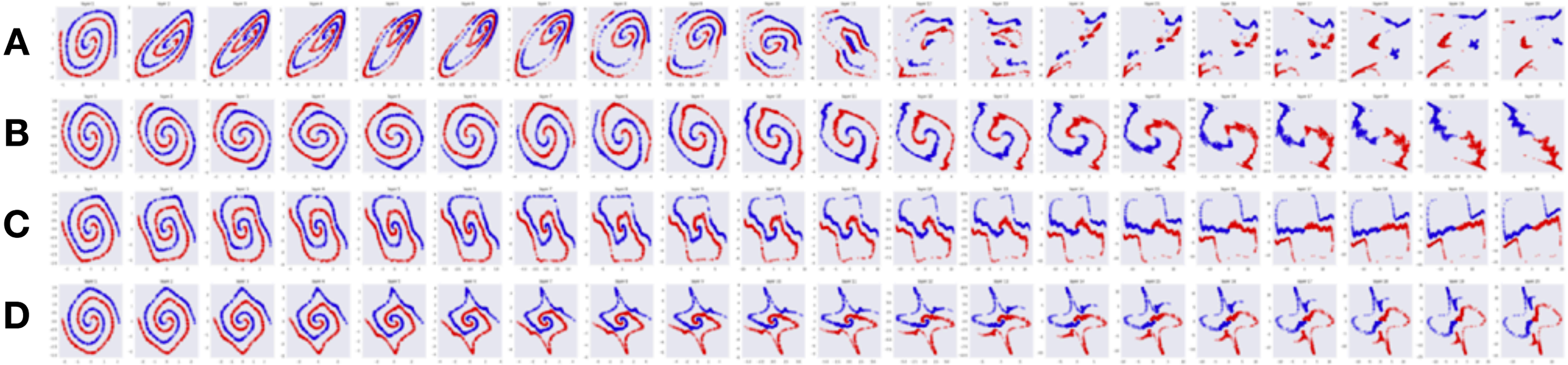
Experiments: decision boundaries



A: ResNet without shared weights
B: ResNet with shared weights
C: Data-driven symmetric ResNet with shared weights
D: Domain-driven symmetric ResNet with shared weights

Rousseau et al. *Residual networks as flows of diffeomorphisms,* JMIV, 2020.

Experiments: evolution of spatial configuration

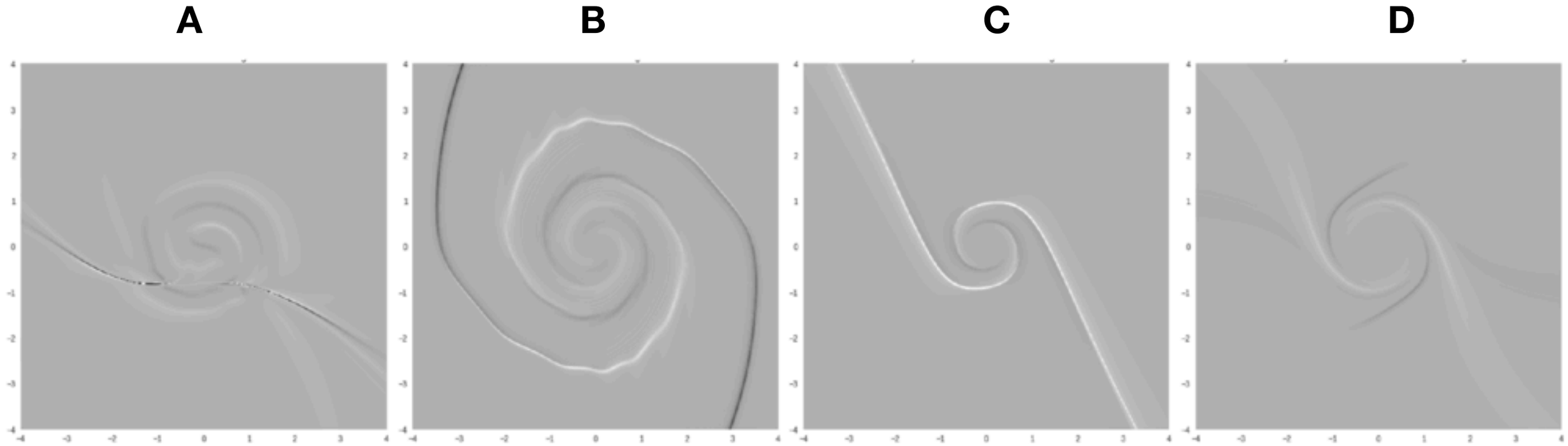


A: ResNet without shared weights
B: ResNet with shared weights
C: Data-driven symmetric ResNet with shared weights
D: Domain-driven symmetric ResNet with shared weights

Rousseau et al. *Residual networks as flows of diffeomorphisms,* JMIV, 2020.

Experiments: Jacobian maps

$$D\Phi(x) := \left(\frac{\partial \phi_i(x)}{\partial x_j} \right)_{i,j}$$



A: ResNet without shared weights
B: ResNet with shared weights
C: Data-driven symmetric ResNet with shared weights
D: Domain-driven symmetric ResNet with shared weights

Rousseau et al. *Residual networks as flows of diffeomorphisms,* JMIV, 2020.

DL as dynamical systems

- Residual Networks have revitalized the link between NN and dynamical systems

- Recurrent neural networks (RNN) are dynamical systems

Liao and Poggio. *Bridging the Gaps Between Residual Learning, Recurrent Neural Networks and Visual Cortex*, 2016.
Haber and Ruthotto. *Stable architectures for deep neural networks*, Inverse problems 2017.
E. *A Proposal on Machine Learning via Dynamical Systems*, CMS 2017
Lu et al. *Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations*, ICML 2018.
Ruthotto and Haber. *Deep neural networks motivated by partial differential equations*, JMIV 2020.

So far…

- Understanding residual networks as dynamical systems
- Decision boundary is informative but complementary study is required (such as the properties of the flow)
- The dynamical system-based approach is a way to open the « black box » of deep learning

Lipschitz networks

- Cauchy-Lipschitz (or also Picard-Lindelöf): solution to the ODE exists and is unique if the velocity field is Lipschitz.

- NN with low Lipschitz constants have better generalization

- Constraint on Lipschitz constants help optimization

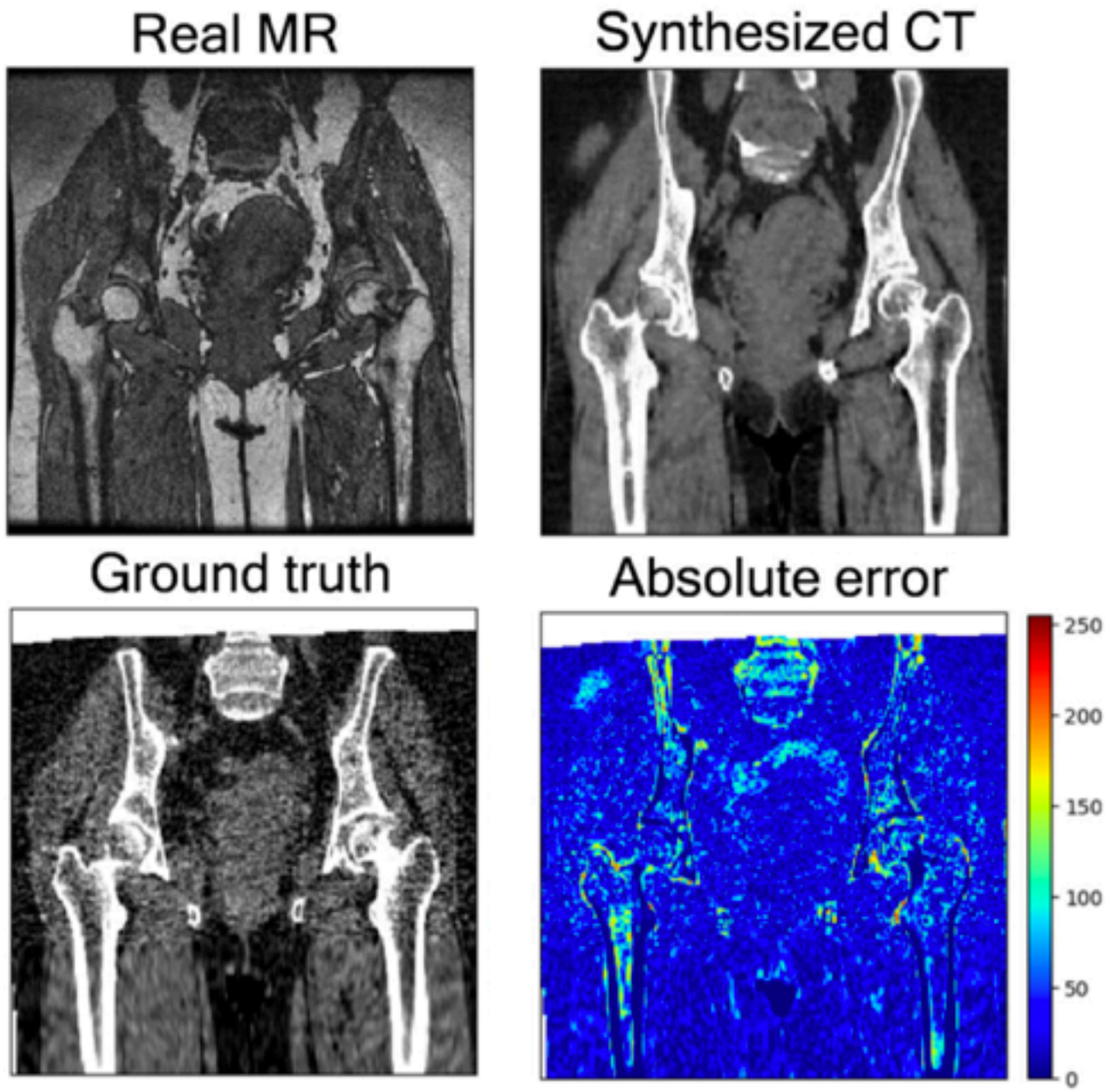
Bartlett. The sample complexity of pattern classification with neural networks, IEEE Trans. IT, 1998.
Oberman and Calder. Lipschitz regularized deep neural networks converge and generalize, Arxiv 2018.

Gouk et al.Regularisation of Neural Networks by Enforcing Lipschitz Continuity, Arxiv 2018.

Terjék. Adversarial Lipschitz Regularization, ICLR 2020.

Image synthesis

- MR to CT



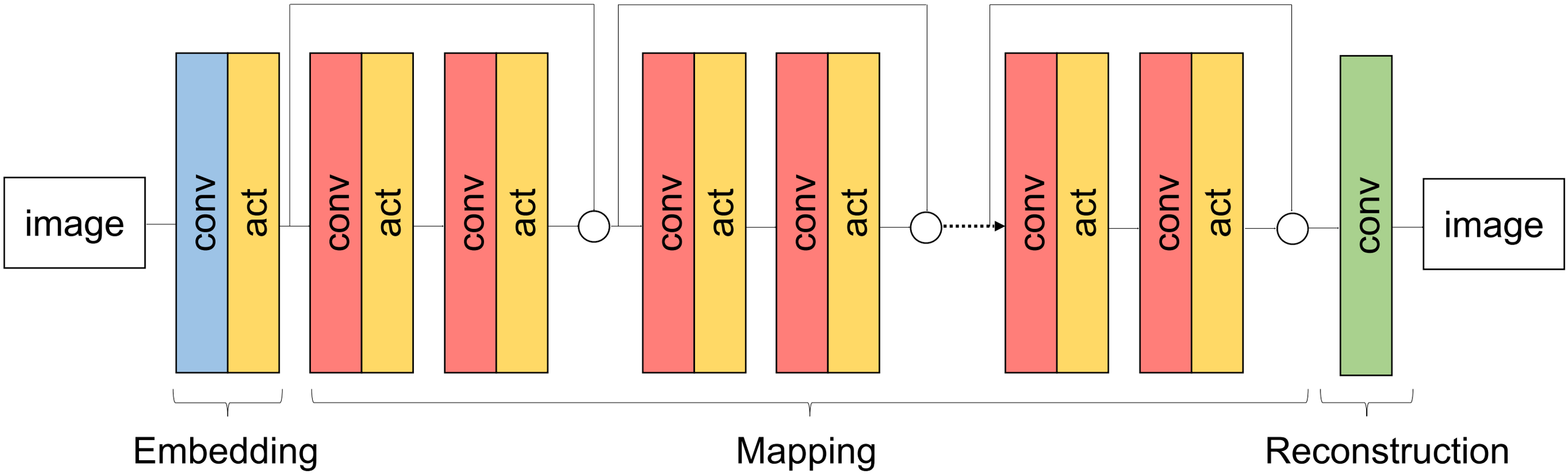
Hiasa et al. Cross-modality image synthesis from unparied data using CycleGAN, 2018

Image synthesis

- What for?
  - Imaging protocol harmonization
  - Handling uncertainty and incomplete data
  - Normalization and intensity correction
  - Simulation of large dataset
  - Super-resolution, denoising, segmentation, …

- Image synthesis = image regression

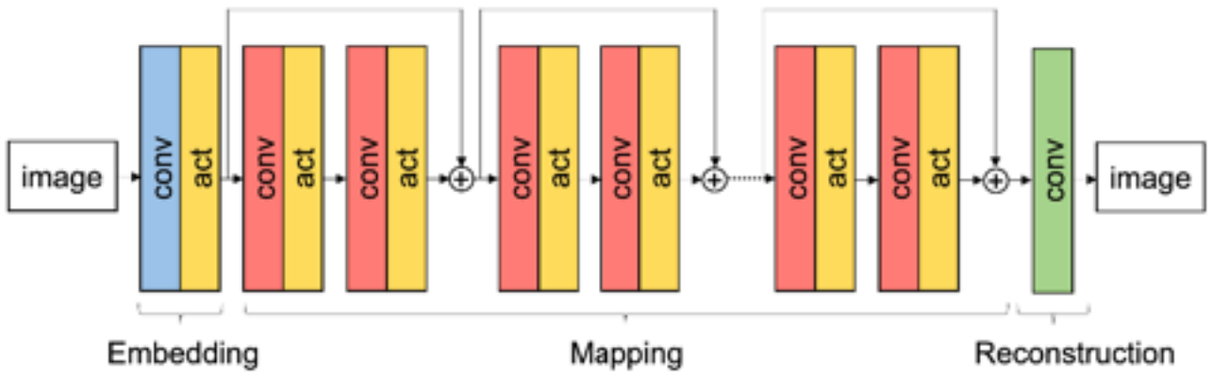
Interpretation of ResNets

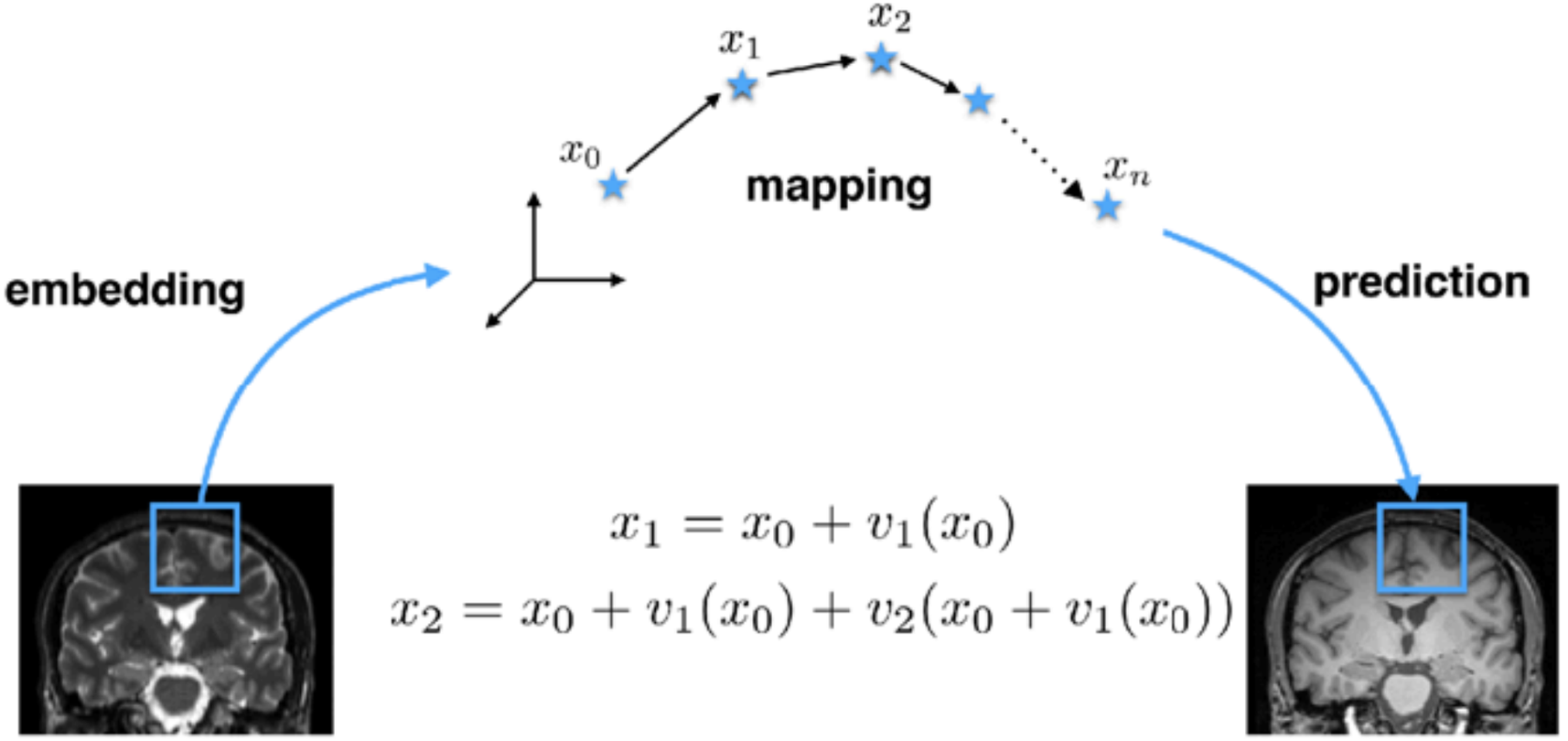


$$y = r \circ m \circ f(x)$$

He et al. Deep residual learning for image recognition, CVPR 2015.

Interpretation of ResNets

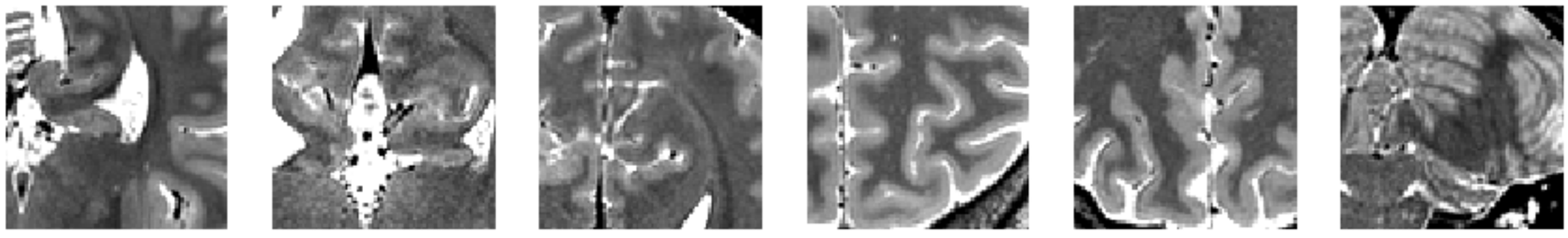




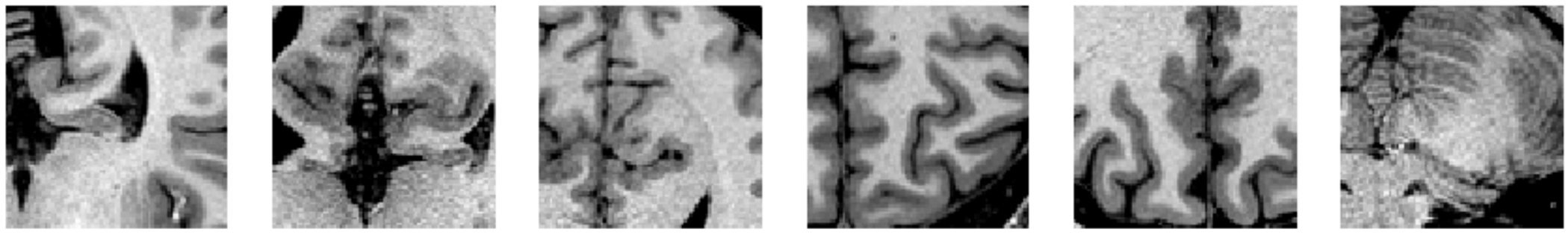
Rousseau et al. *Residual networks as flows of diffeomorphisms,* JMIV, 2020.

Results on HCP dataset

T2w

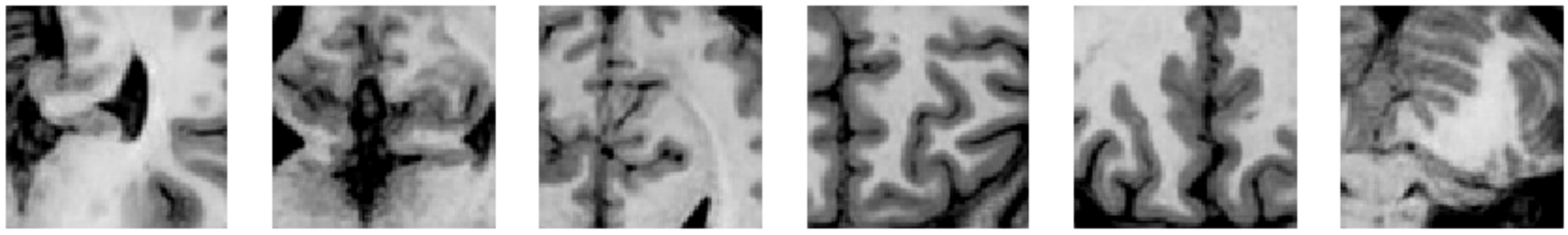


T1w

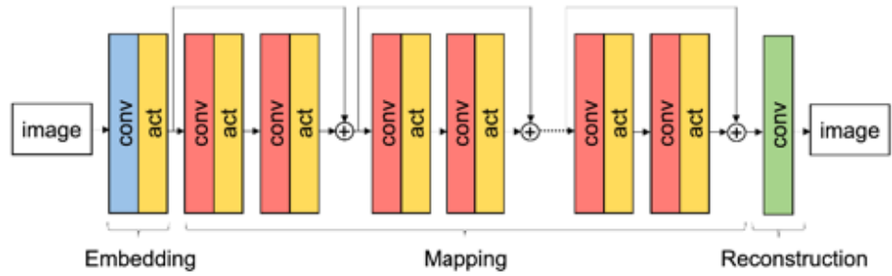


simulated T1w

$y = r \circ m \circ f(x)$



Reversible networks



$$y = r \circ m \circ f(x)$$

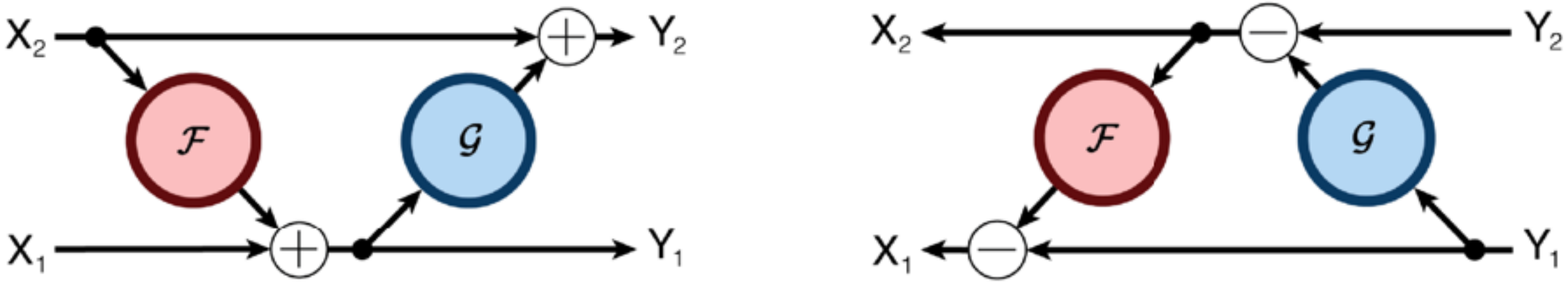
$$x = r \circ m^{-1} \circ f(y)$$

Each residual mapping block: $x + u(x)$

To compute the inverse mapping, we have to solve: $(I + u) \circ (I + v) = I$

leading to: $v = \sum_{n=1}^{\infty} (-1)^n u^n$

Reversible networks



FORWARD

$$y_1 = x_1 + \mathcal{F}(x_2)$$
$$y_2 = x_2 + \mathcal{G}(y_1)$$

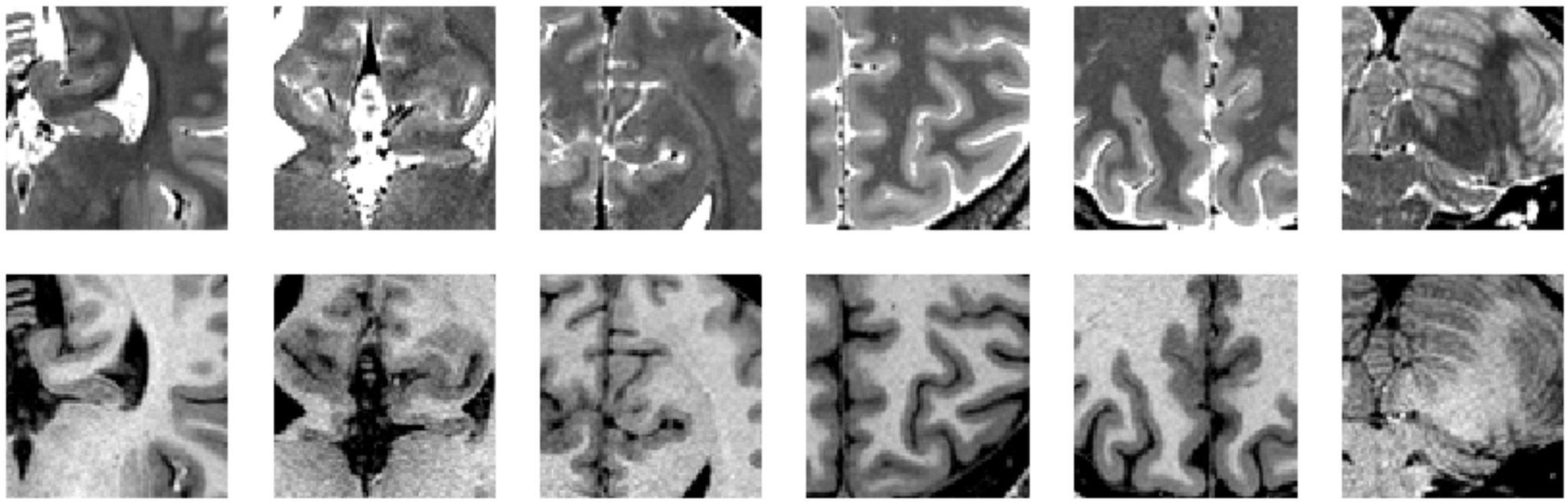
BACKWARD

$$x_2 = y_2 - \mathcal{G}(y_1)$$
$$x_1 = y_1 - \mathcal{F}(x_2)$$

Deco et al. Nonlinear higher-order statistical decorrelation by volume-conserving neural architectures. Neural Networks, 1995.
Gomez et al. The reversible residual network: backpropagation without storing activations. NIPS, 2017.

Results on HCP dataset

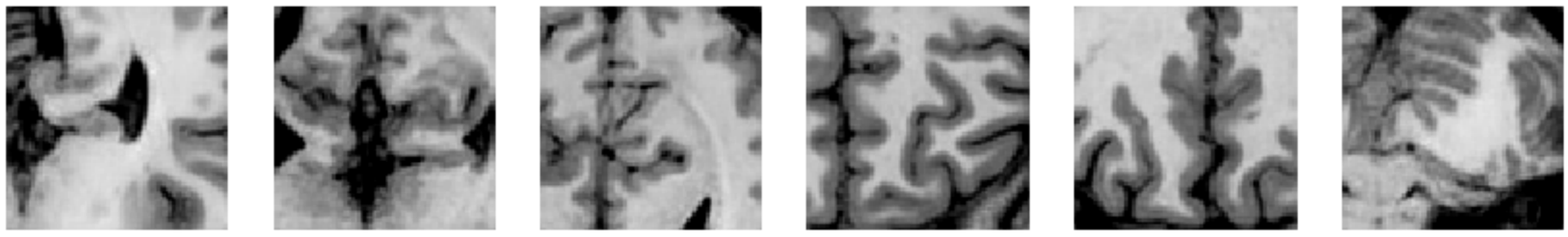
T2w



T1w

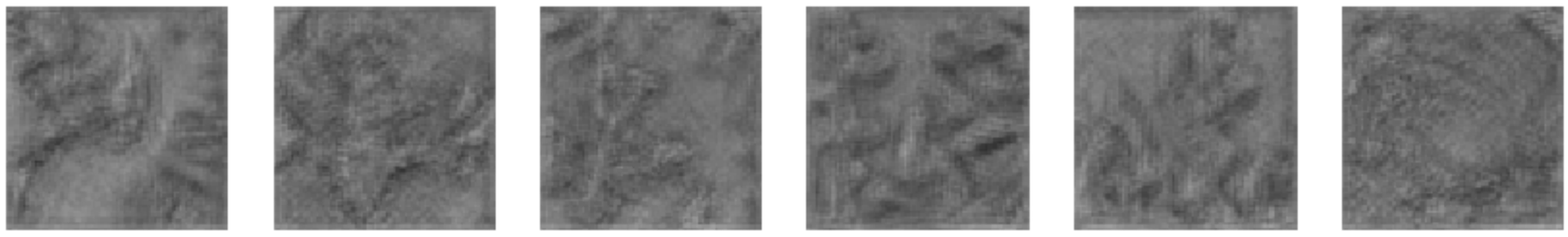
simulated T1w

$y = r \circ m \circ f(x)$

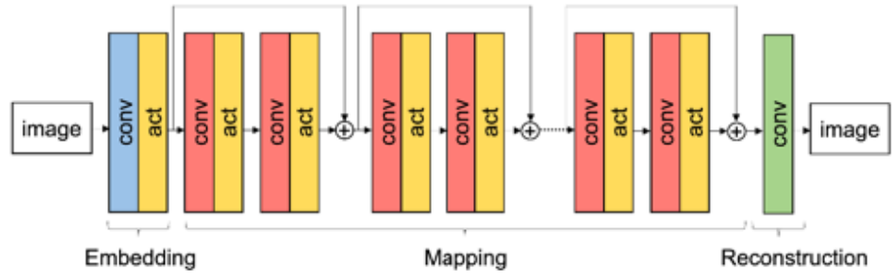


simulated T2w

$x = r \circ m^{-1} \circ f(y)$



Desired properties



- Mapping

$$y = r \circ m \circ f(x)$$

$$x = r \circ m^{-1} \circ f(y)$$

- Cycle consistency

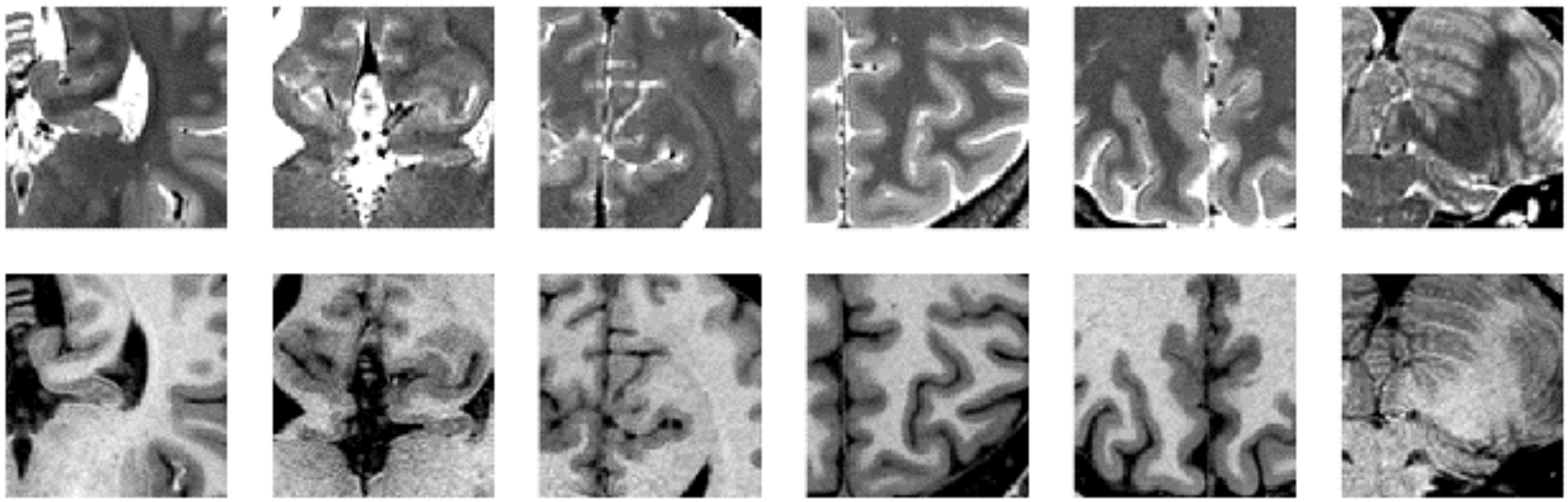
$$x = r \circ m^{-1} \circ f \circ r \circ m \circ f(x)$$

$$y = r \circ m \circ f \circ r \circ m^{-1} \circ f(y)$$

Results on HCP dataset

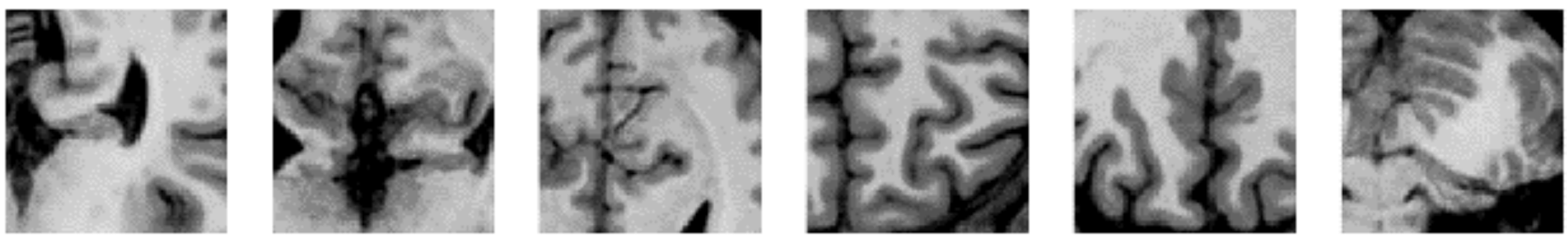
T2w

T1w



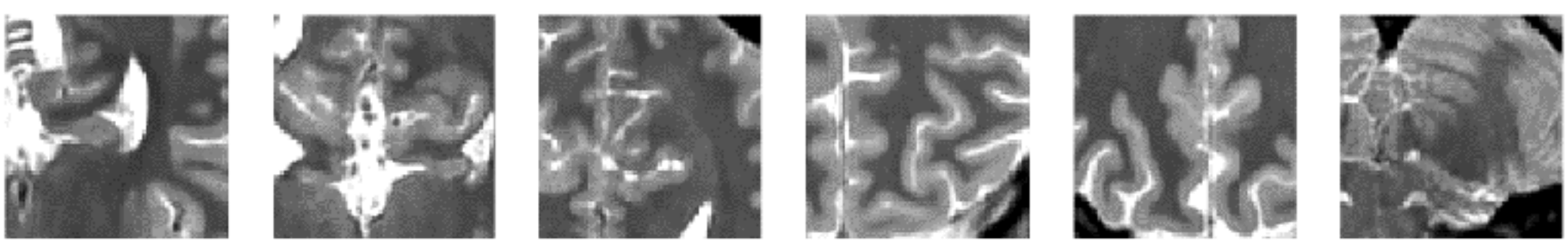
simulated
T1w

$$y = r \circ m \circ f(x)$$

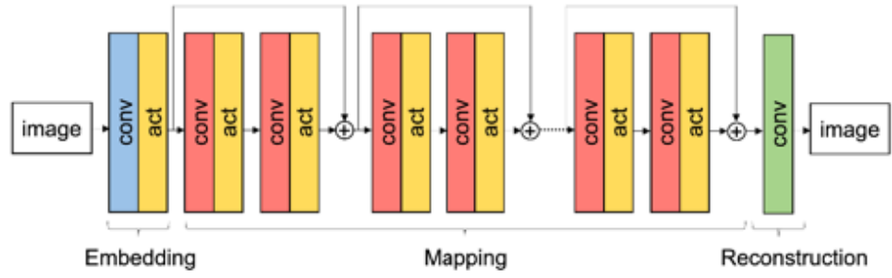


simulated
T2w

$$x = r \circ m^{-1} \circ f(y)$$



Desired properties



- Identity mapping

$$y = r \circ m \circ f(y)$$
$$x = r \circ m^{-1} \circ f(x)$$

- Autoencoder

$$x = r \circ f(x)$$
$$y = r \circ f(y)$$

- In feature space

$$f(y) = m \circ f(x)$$
$$f(x) = m^{-1} \circ f(y)$$

Deep learning and dynamical systems

- Dynamical systems for studying deep learning
  - Efficient architectures (i.e. numerical solvers (Euler, RK4, etc.)), generalization, adversarial examples

- Deep learning for solving dynamical systems
  - DL as efficient solvers

Overview

Dynamical systems

Neural Networks

Neuroimaging

Brain Mapping

Structures and functions

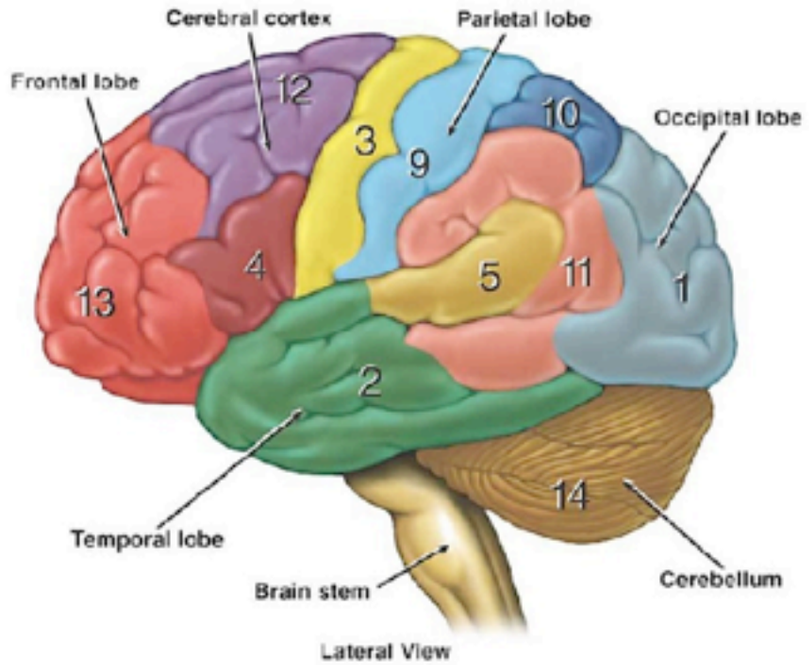
Functions

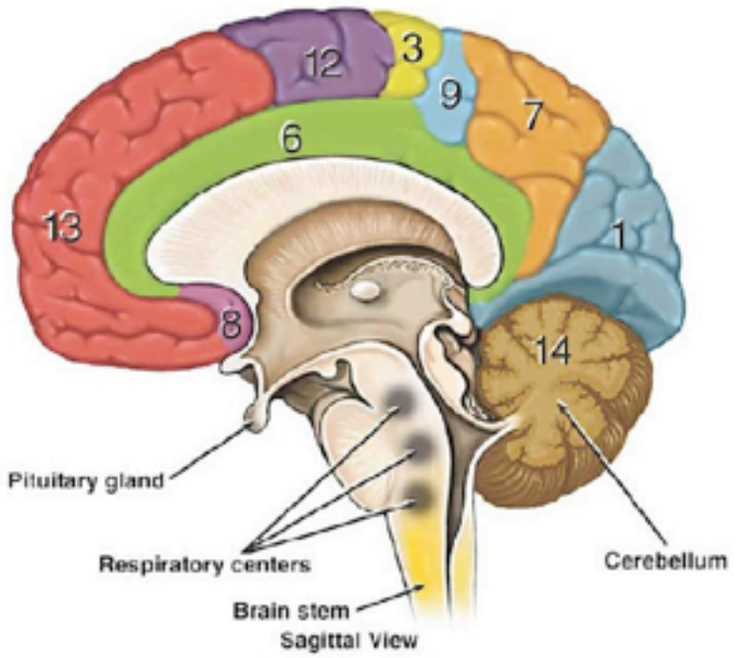
Functional Areas of the Cerebral Cortex

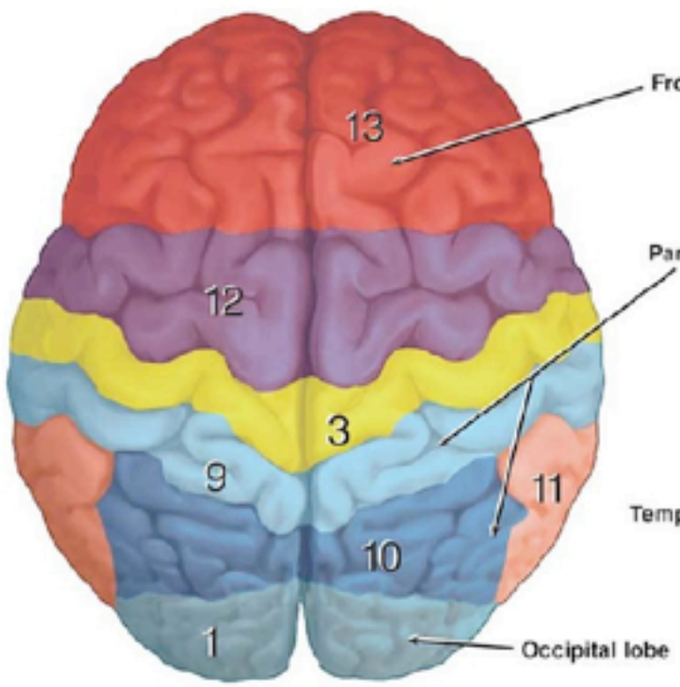
1. **Visual Area:** Sight, Image recognition, Image perception
2. **Association Area** — Short-term memory, Equilibrium, Emotion
3. **Motor Function Area** — Initiation of voluntary muscles
4. **Broca's Area** — Muscles of speech
5. **Auditory Area** — Hearing
6. **Emotional Area** — Pain, Hunger, "Fight or flight" response
7. **Sensory Association Area**
8. **Olfactory Area** — Smelling
9. **Sensory Area** — Sensation from muscles and skin
10. **Somatosensory Association Area** — Evaluation of weight, texture, temperature, etc. for object recognition
11. **Wernicke's Area** — Written and spoken language comprehension
12. **Motor Function Area** — Eye movement and orientation
13. **Higher Mental Functions** — Concentration, Planning, Judgment, Emotional expression, Creativity, Inhibition

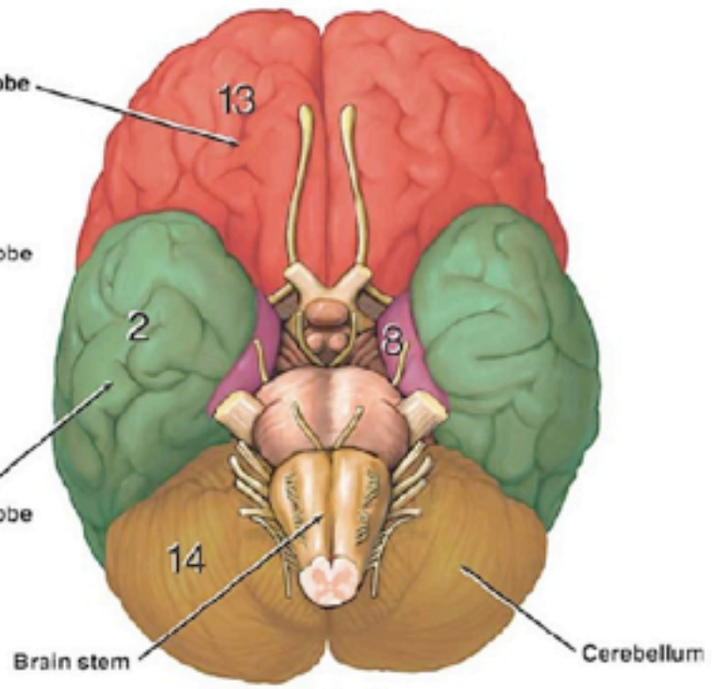
Functional Areas of the Cerebellum

14. **Motor Functions** — Coordination of movement, Balance and equilibrium, Posture

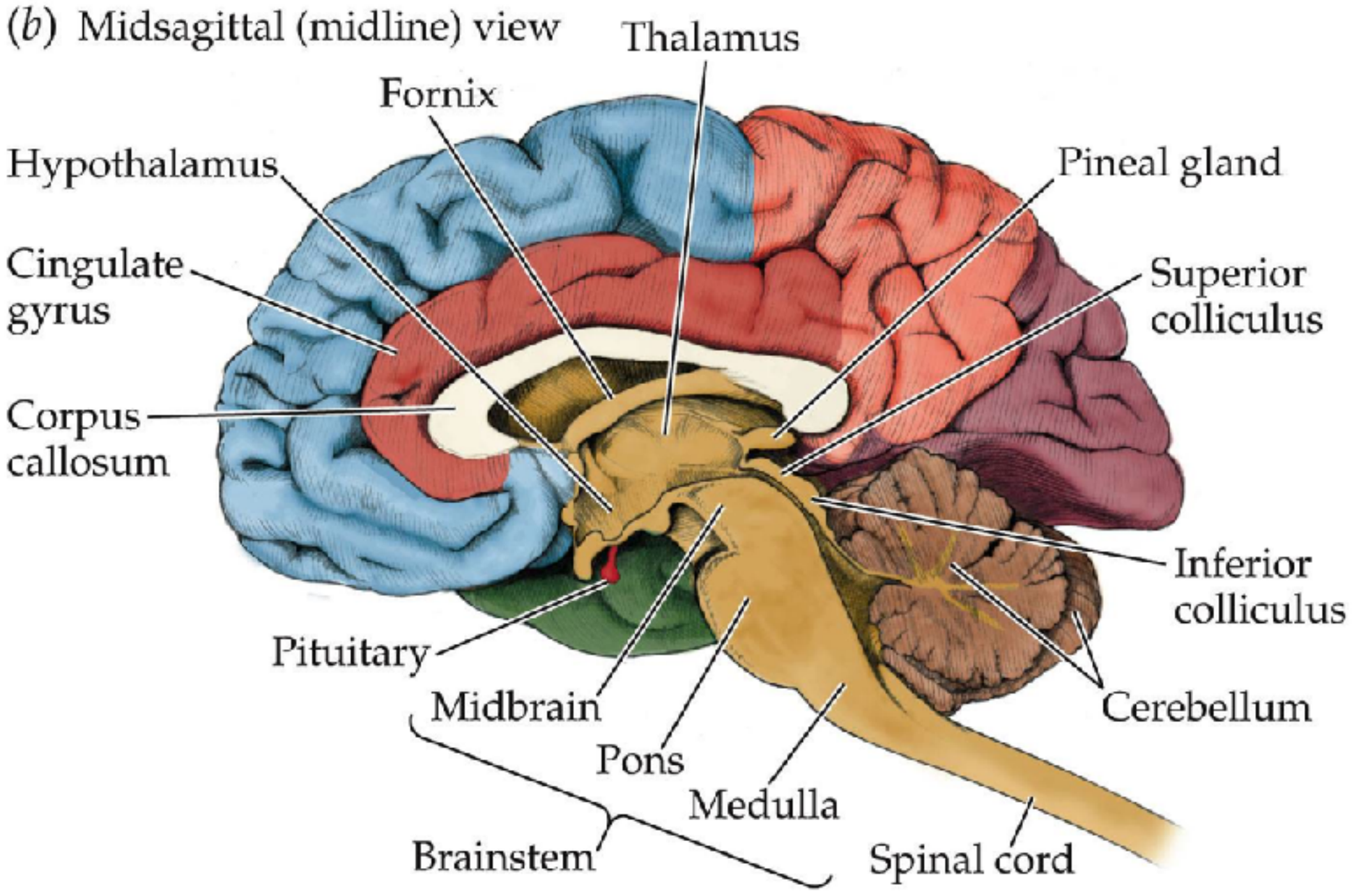






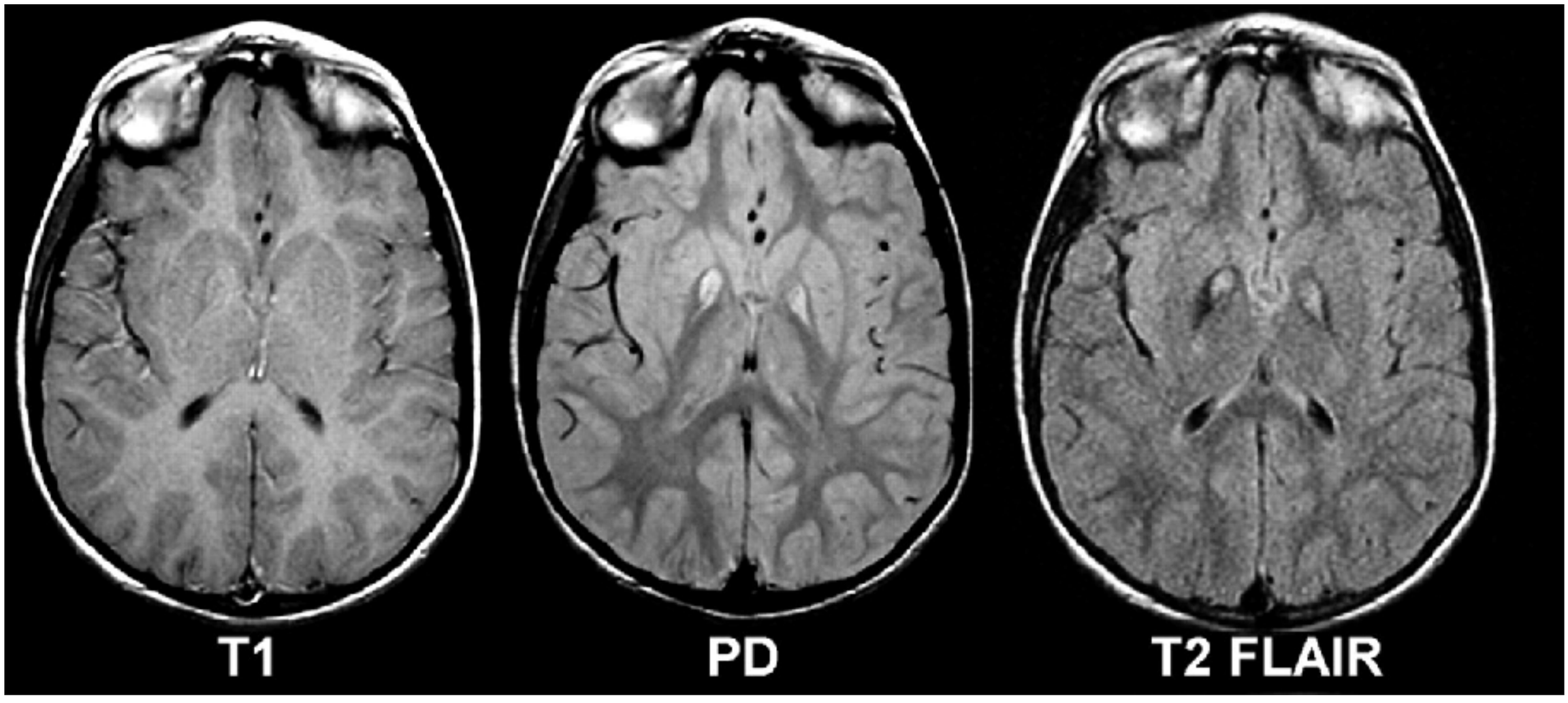


Structures



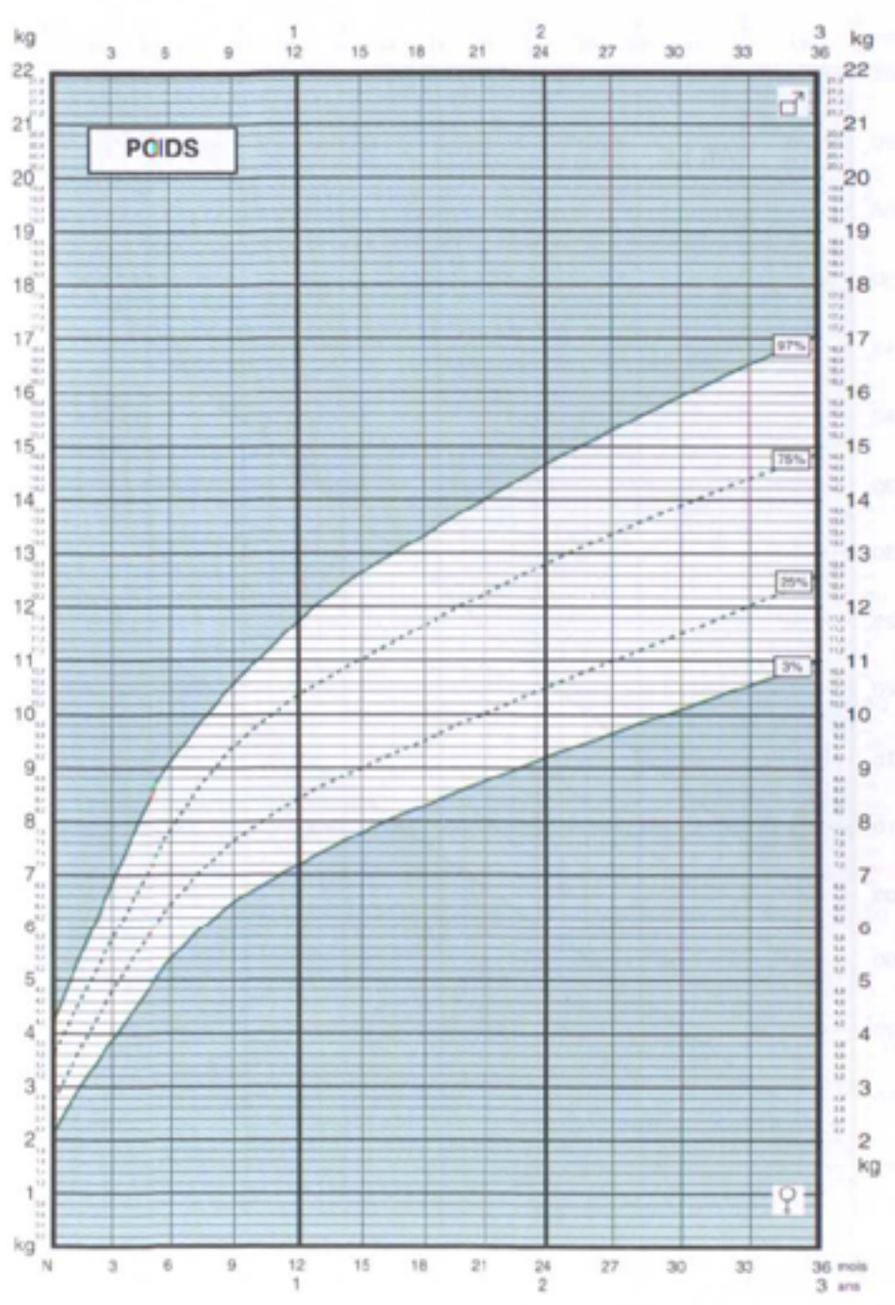
***Biological Psychology* 6e, Figure 2.12 (Part 2)**

Structural MRI

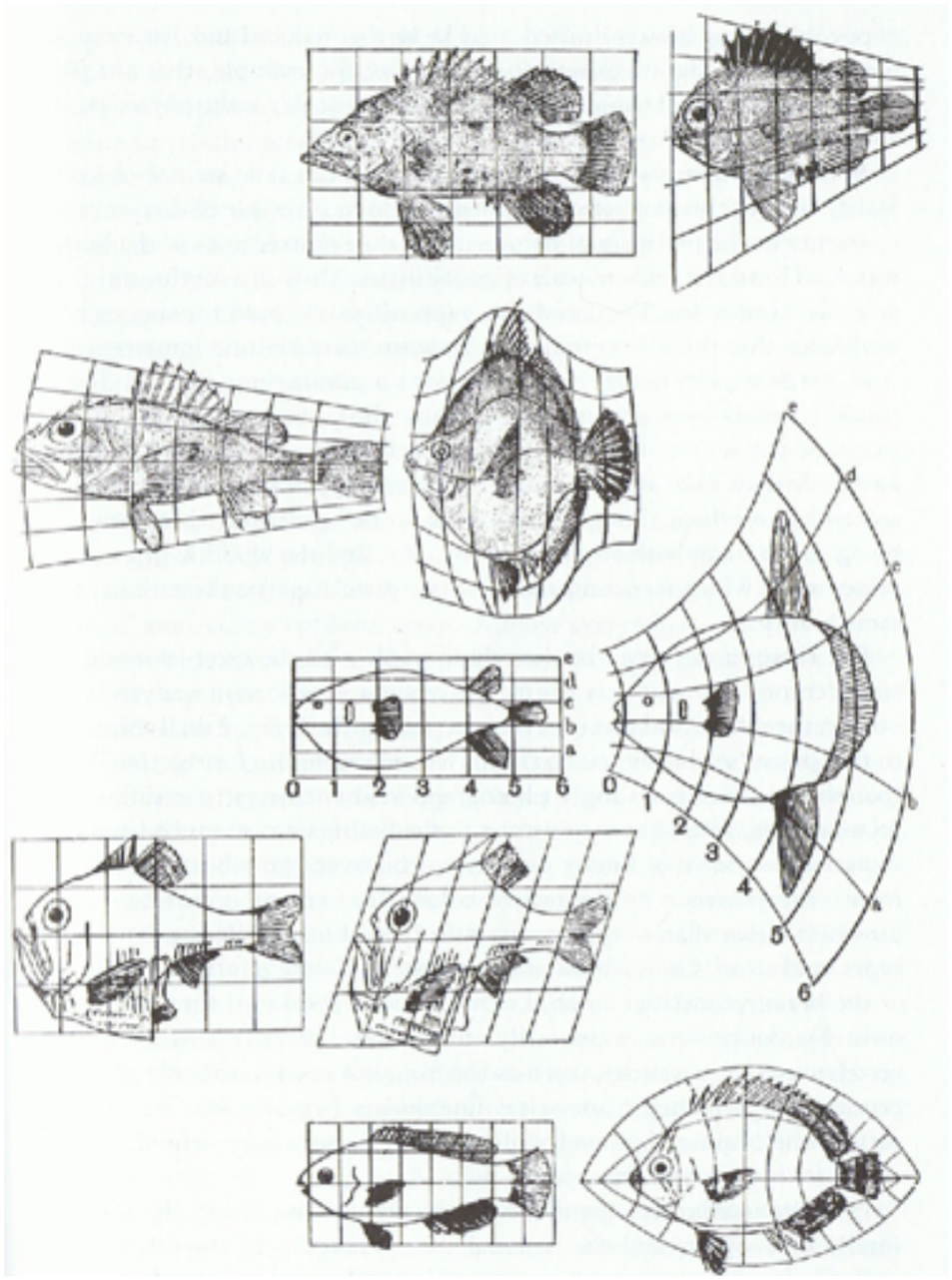


Morphology

- Volume
- Morphometry



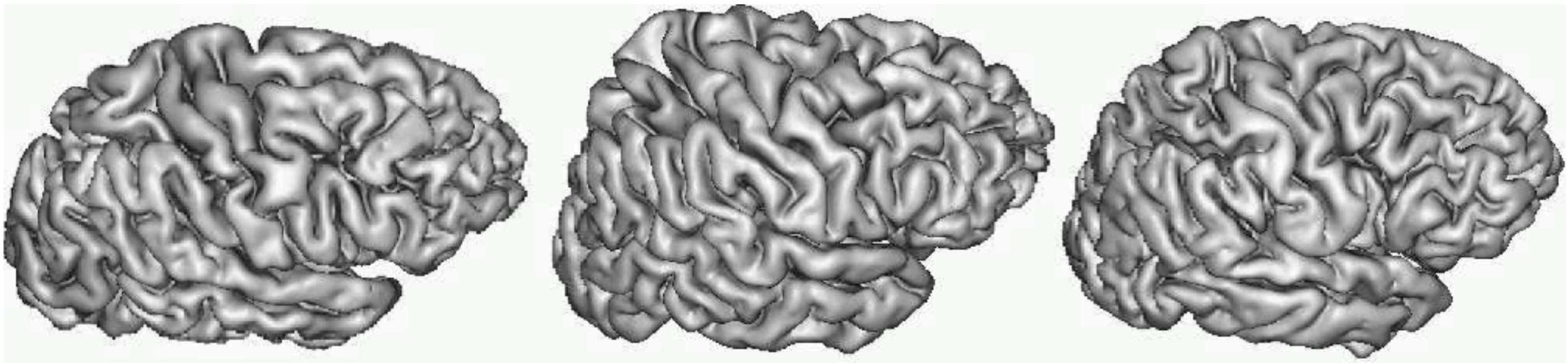
Morphometry



[D’Arcy Thompson 1917]

Morphometry is the study and analysis of the **geometry** of objects.

Complexity and variability



[Mangin 2005]

What for?

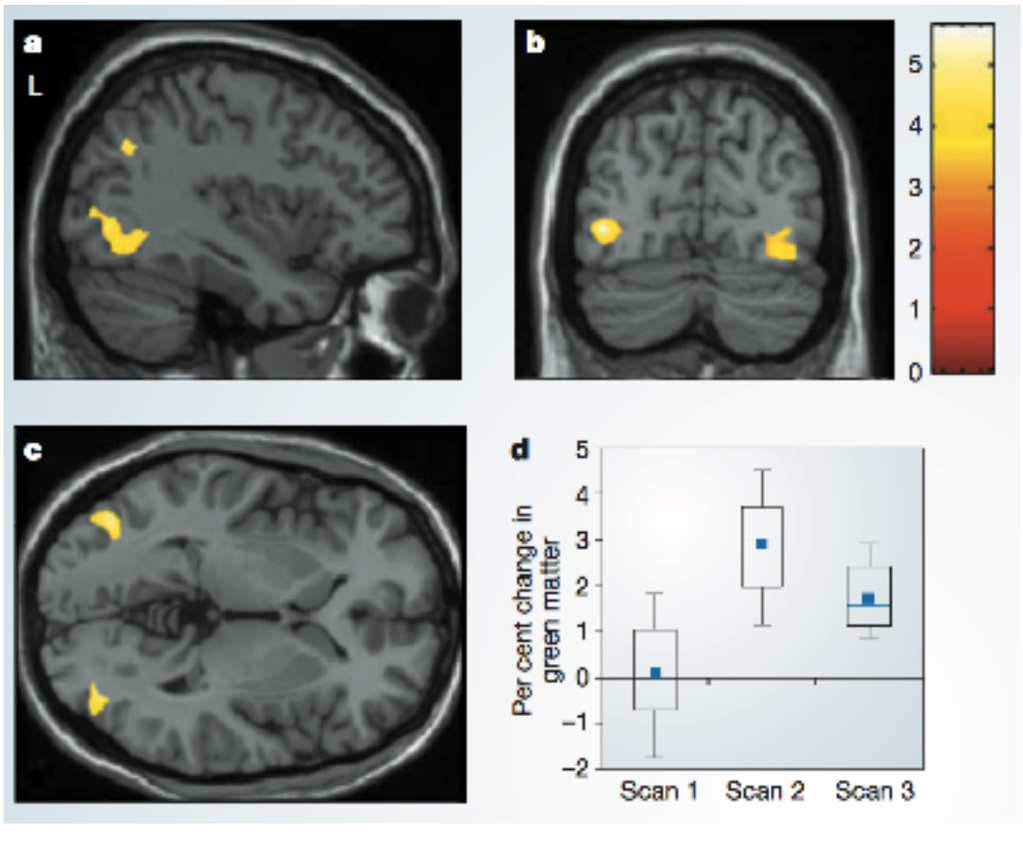
- Which areas atrophy with age?
- How does the brain reorganize itself when learning new tasks?
- How does the brain develop?

Example: Brain plasticity

How does the brain reorganize itself when learning new tasks?

24 subjects do not know how to juggle
12 subjects are learning

Significant increase in
in grey matter in jugglers



[Draganski et al. 2004]

3 items

define a way to compare	matching
unit of measure	a reference frame (model brain)
quantify differences	operations on transformations

Overview

Dynamical systems

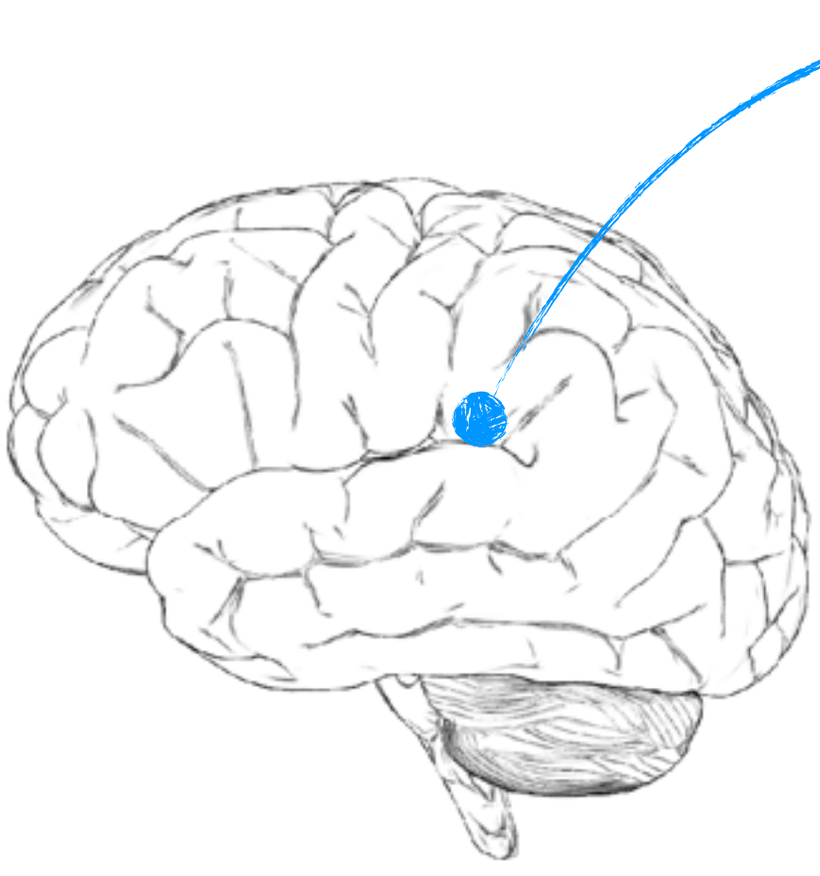
Neural Networks

Neuroimaging

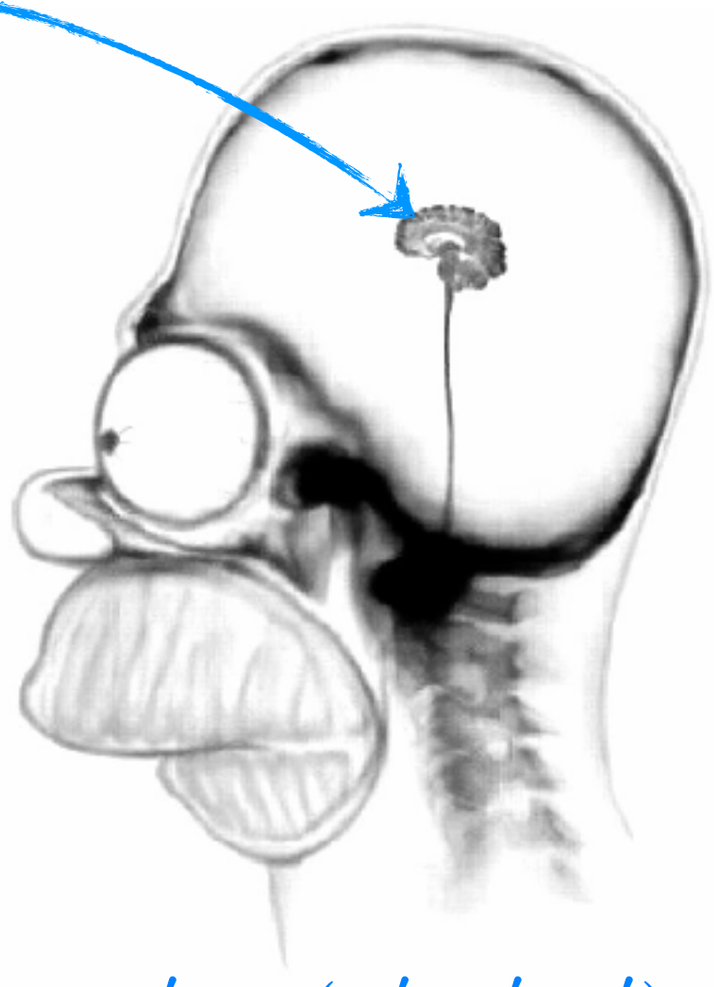
Image registration

Matching (or registration)

$$\mathbf{x}' = T(\mathbf{x})$$

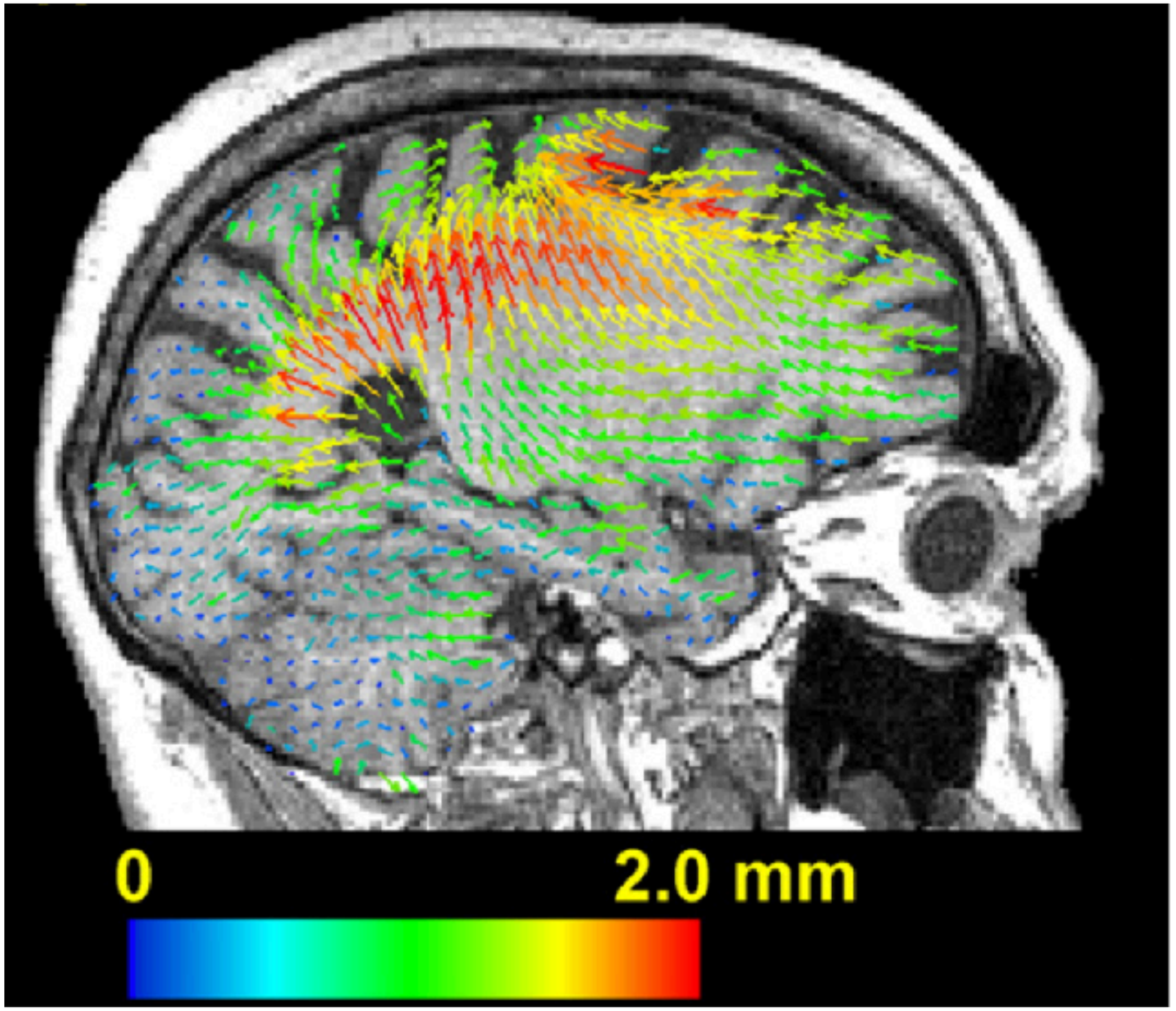


$$\mathbf{x} = (x, y, z)$$



$$\mathbf{x}' = (x', y', z')$$

Dense deformation fields

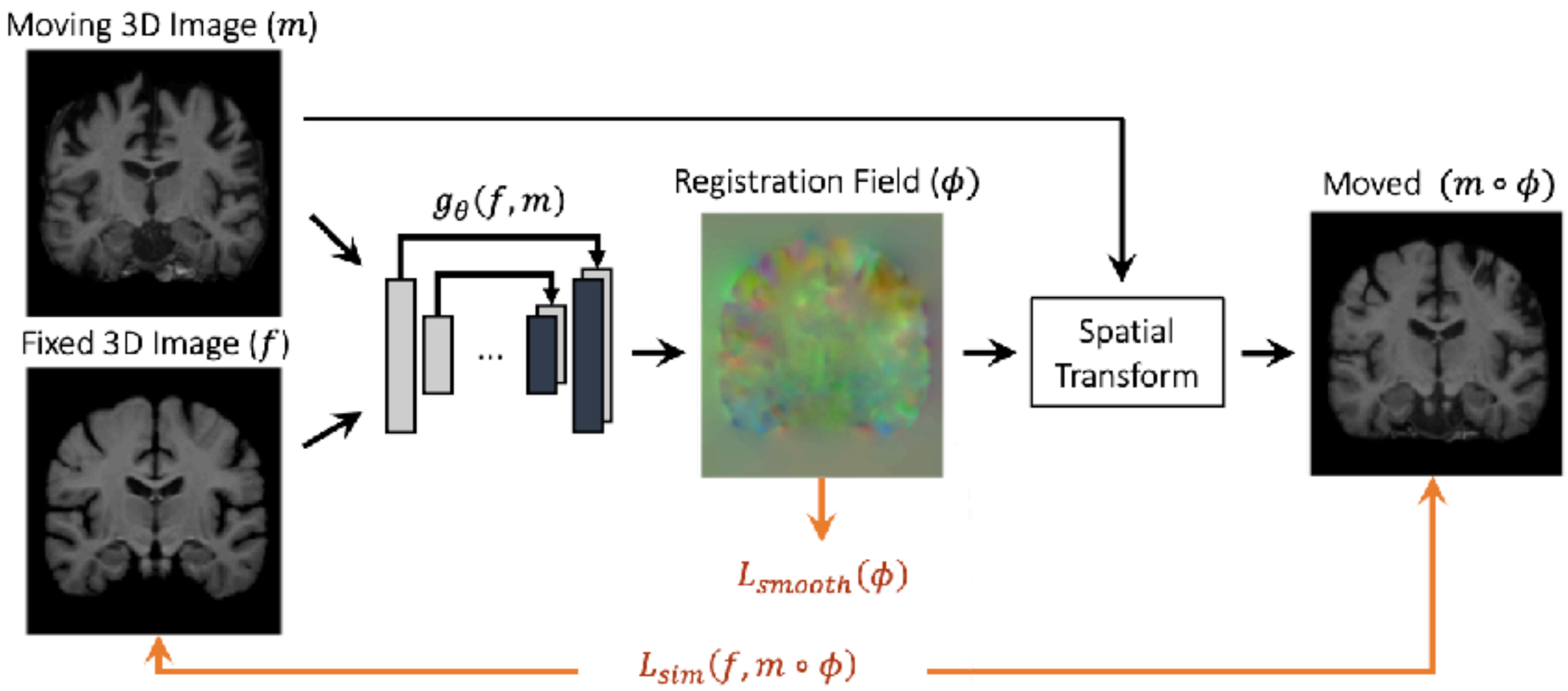


[Pieperhoff et al. 2008]

How to design registration in a deep learning framework?

- Inputs: a source image and a target image
- Goal: estimate a deformation field such that the warped source image is similar to a target image
- Spatial transformer: apply a deformation field to an image
- Losses:
  - similarity between the warped source image and the target image (MSE or CC for instance)
  - smoothness of the deformation field

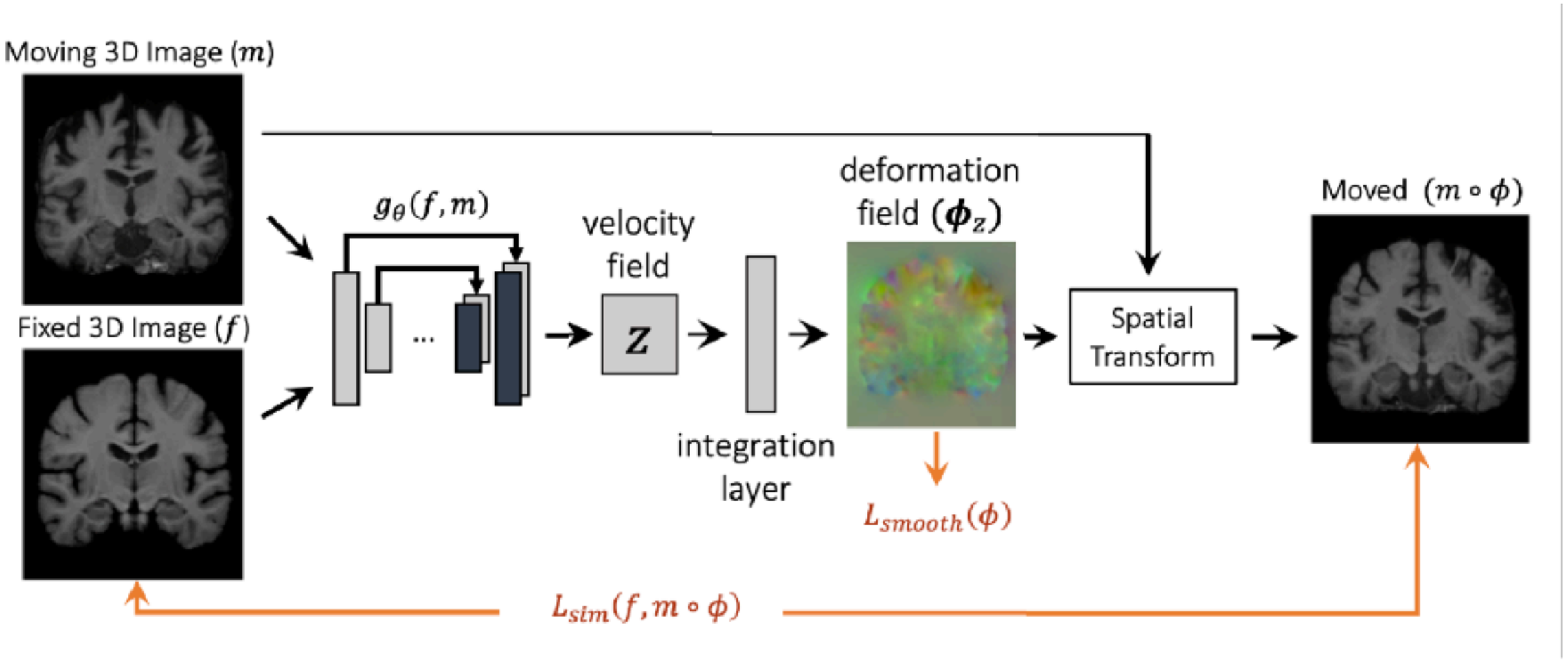
DL-based registration



[Balakrishnan et al. VoxelMorph: A Learning Framework for Deformable Medical Image Registration, IEEE TMI 2019]

DL-based registration

Use a dynamical formulation for the deformation field: $\frac{d\phi(t)}{dt} = V\left(\phi(t)\right)$



Adapted from Dalca et al. Unsupervised Learning of Probabilistic Diffeomorphic Registration for Images and Surfaces, Medical Image Analysis 2019

Overview

Dynamical systems

Neural Networks

Neuroimaging

Image registration

Template estimation

Reference frames

Talairach space

- «A **standard brain space** is needed for consistent spatial localization in both individual and composite images, and this is provided by the **Talairach space**.»

- «low degree-of-freedom regional method»

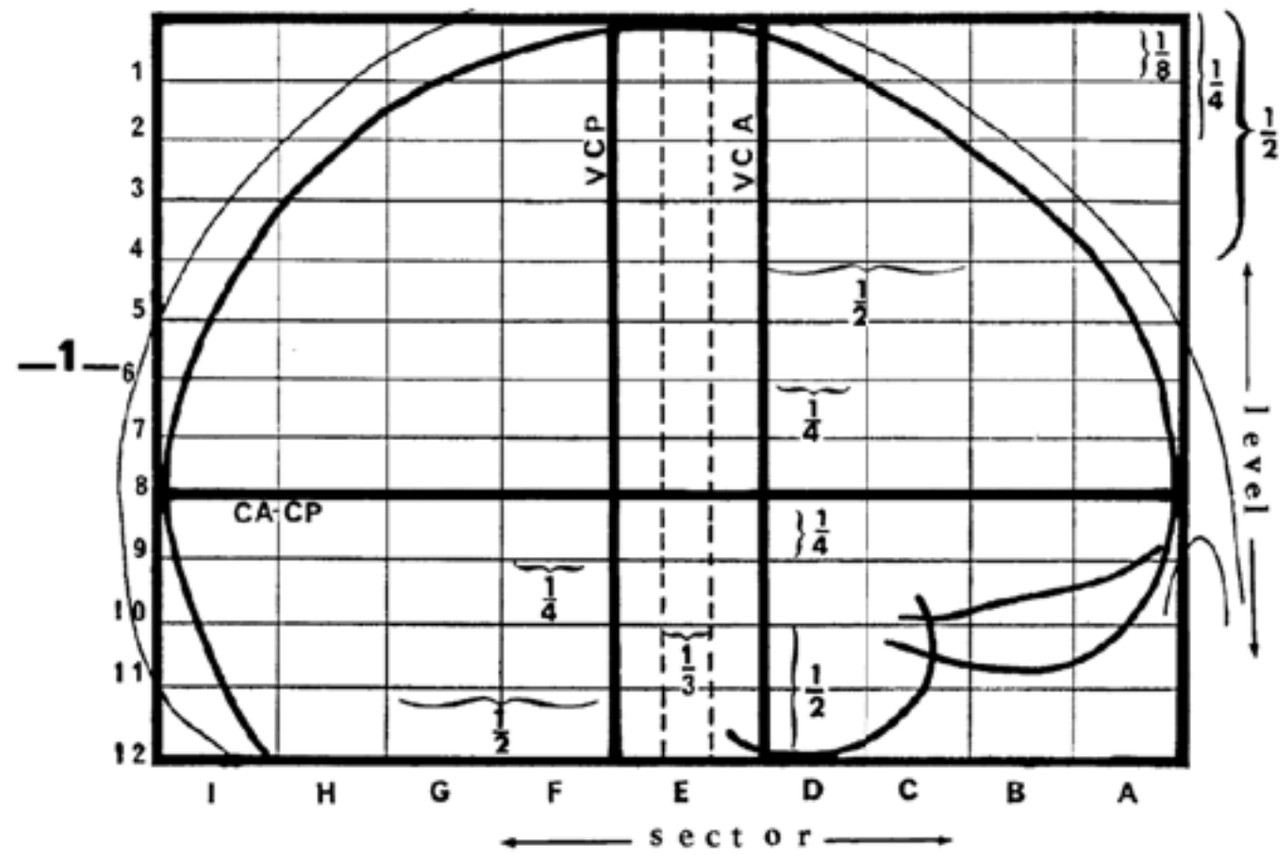
[Lancaster & Fox 2008]

Talairach space

3D oriented frame

Piecewise linear deformations

Anatomical template

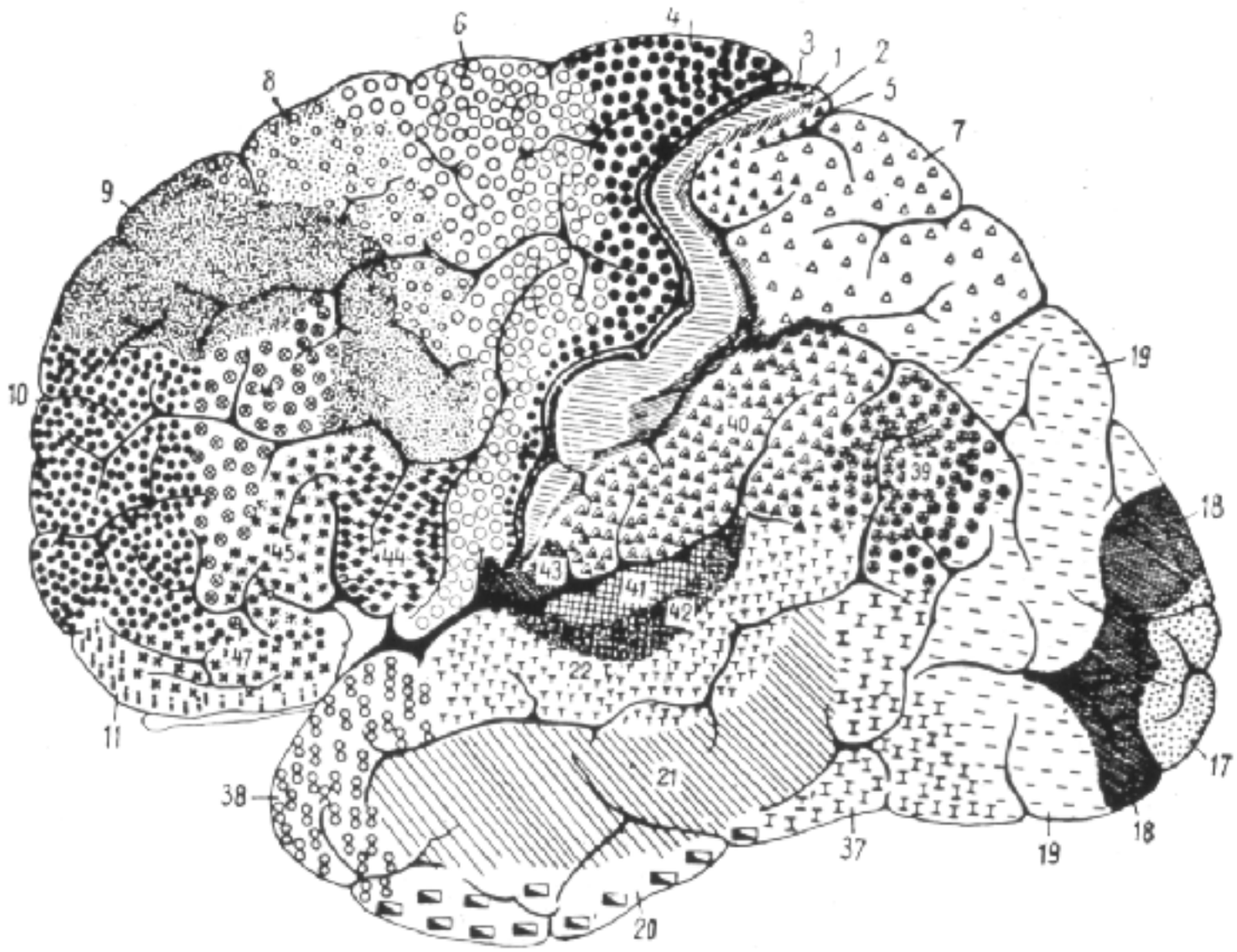


[Talairach & Tournoux 1988]

Brodmann areas

cytoarchitecture

52 cortical regions



[Brodmann 1909]

- a single reference image cannot represent the structural variability

- the variability is encoded in the deformation fields

- the choice of the reference introduces a bias

- an average image can be calculated for a given population

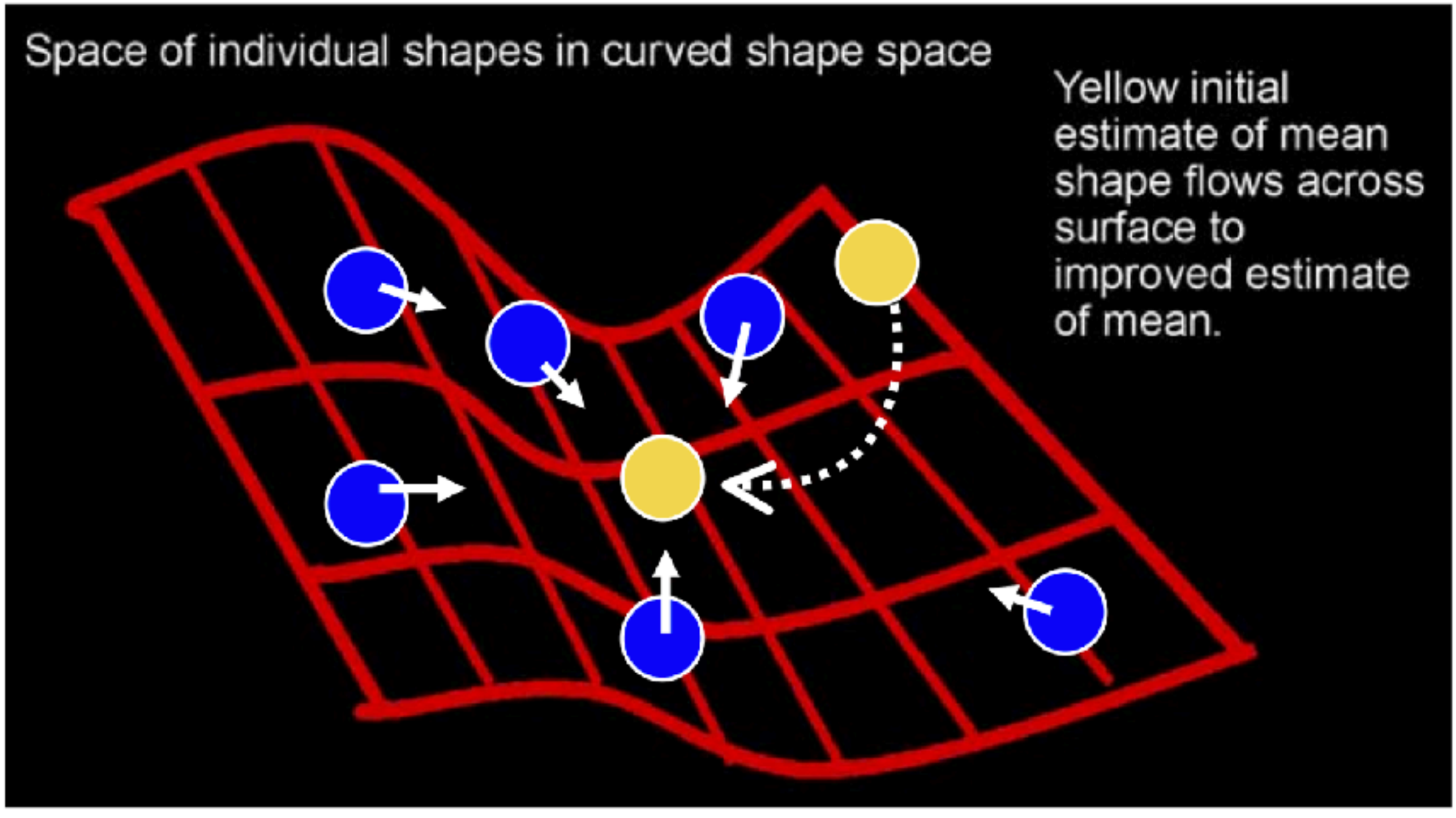
Mean in a vector space

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

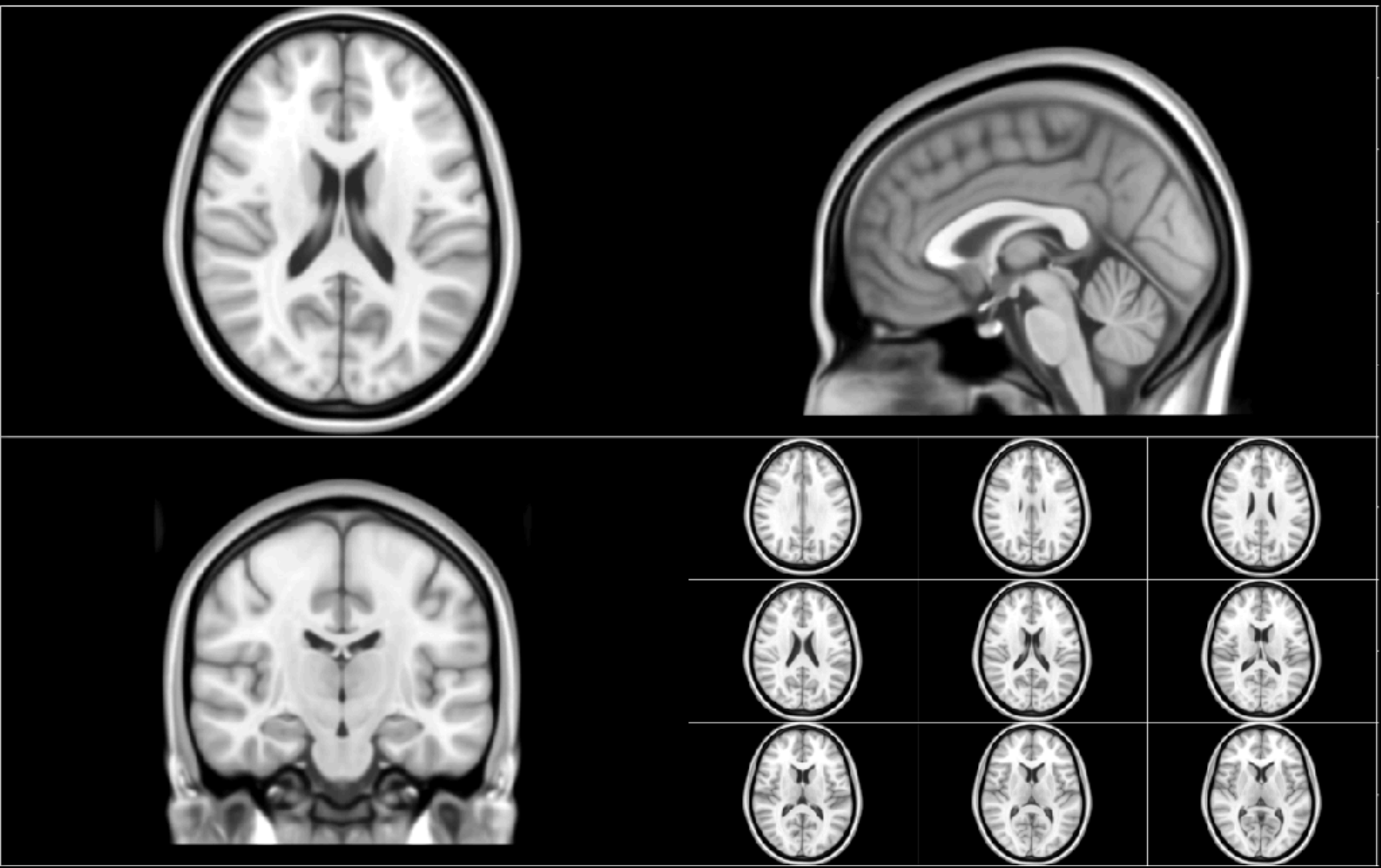
Fréchet mean in a metric space M

$$\mu = \arg\min_{x \in M} \sum_{i=1}^{N} d(x, x_i)^2$$

Estimation of the mean image



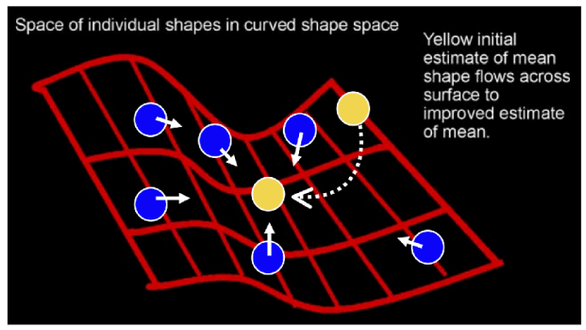
[Avants et al. 2004]



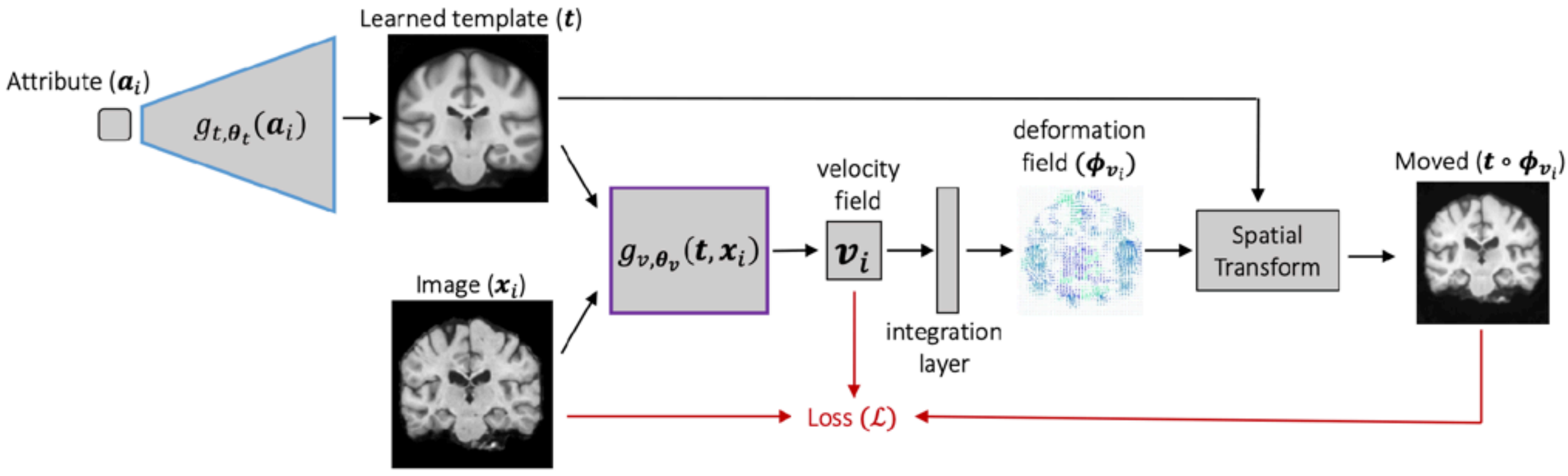
FSL MNI152 T1 [Grabner et al. 2006]

How to estimate a template in a deep learning framework?

- Inputs: set of images
- Goal: estimate a template image (« barycenter » of the image set)
- Use of a registration framework
- A template generator network
- Add a loss related to « barycenter » constraint: the sum of the deformation fields should be minimal

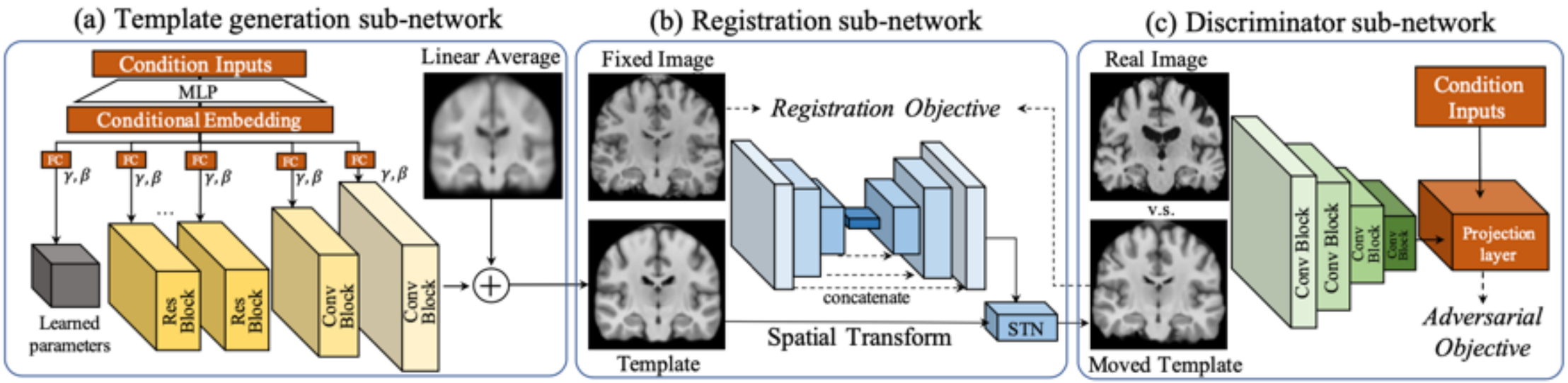


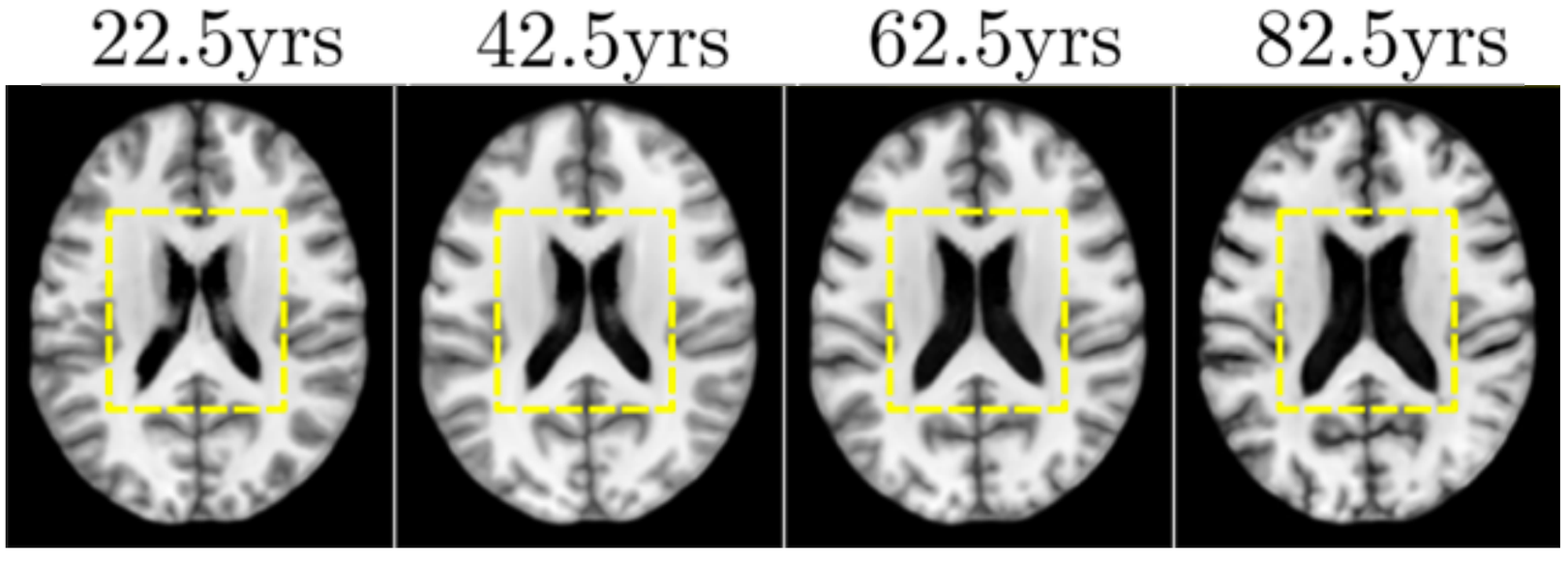
DL-based atlas building



[Dalca et al. Learning Conditional Deformable Templates with Convolutional Networks, NeurIPS 2019]

Conditional atlas building





Dey et al. *Generative adversarial registration for improved conditional deformable templates,* ICCV 2021.

Overview

Dynamical systems

Neural Networks

Neuroimaging

Image registration

Template estimation

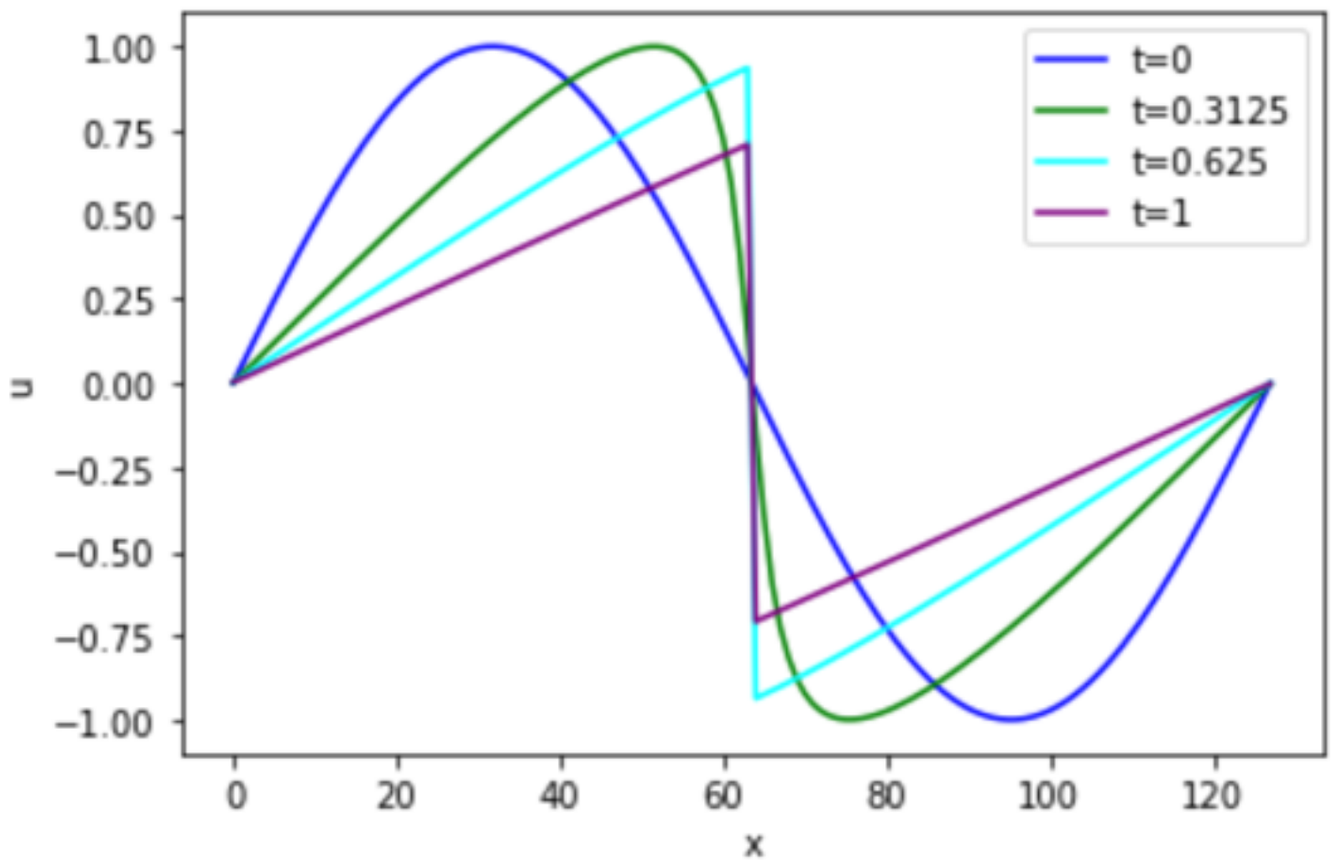
DL & Physics

What about physics?

- Differential equations are a key element in physics
- How to introduce physical contraints in neural networks?
- How to use neural networks for physics problems?

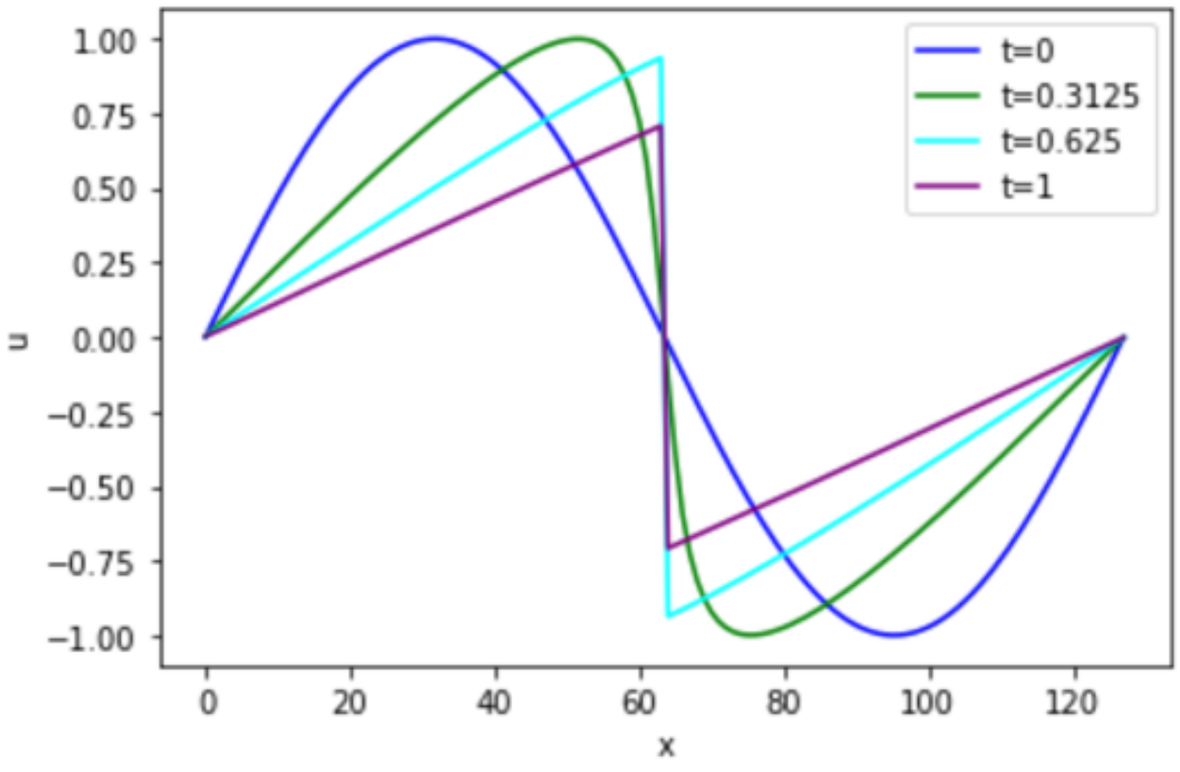
Burgers' equation

$$\frac{\partial u}{\partial t} + u\nabla u = \nu\nabla \cdot \nabla u$$

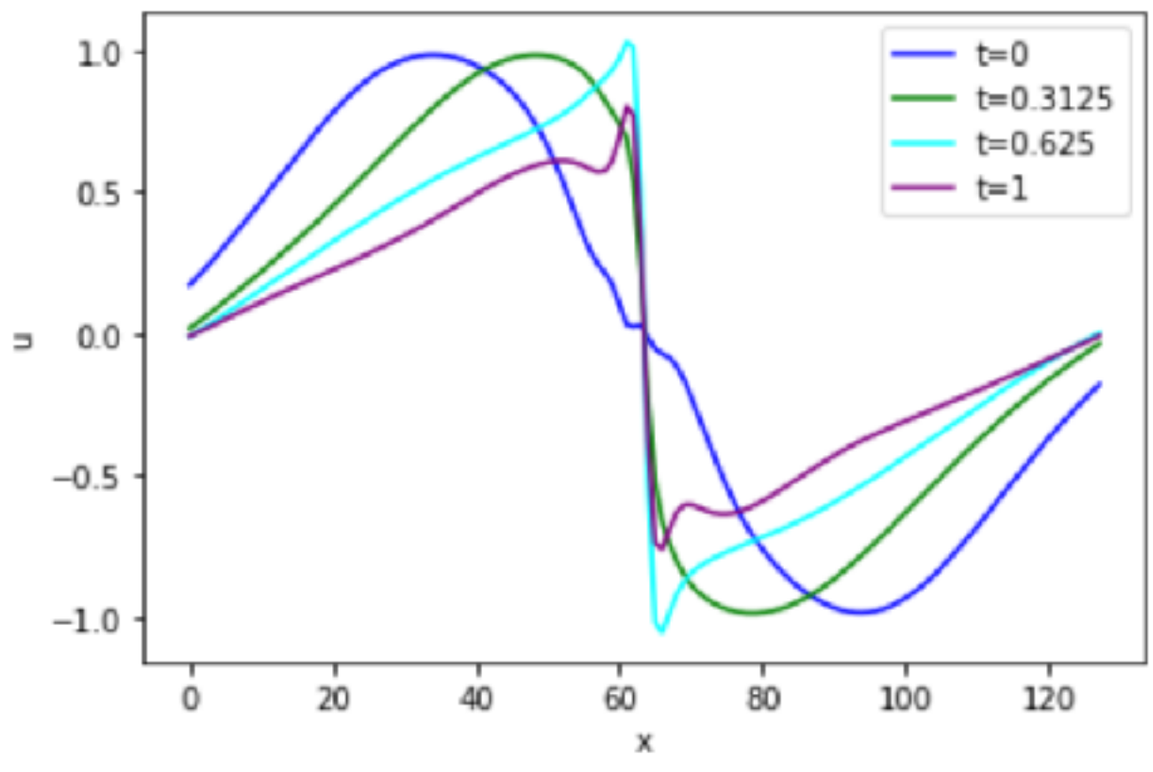


Burgers' equation

- Fully supervised approach (using MLP)



Ground truth



Supervised MLP

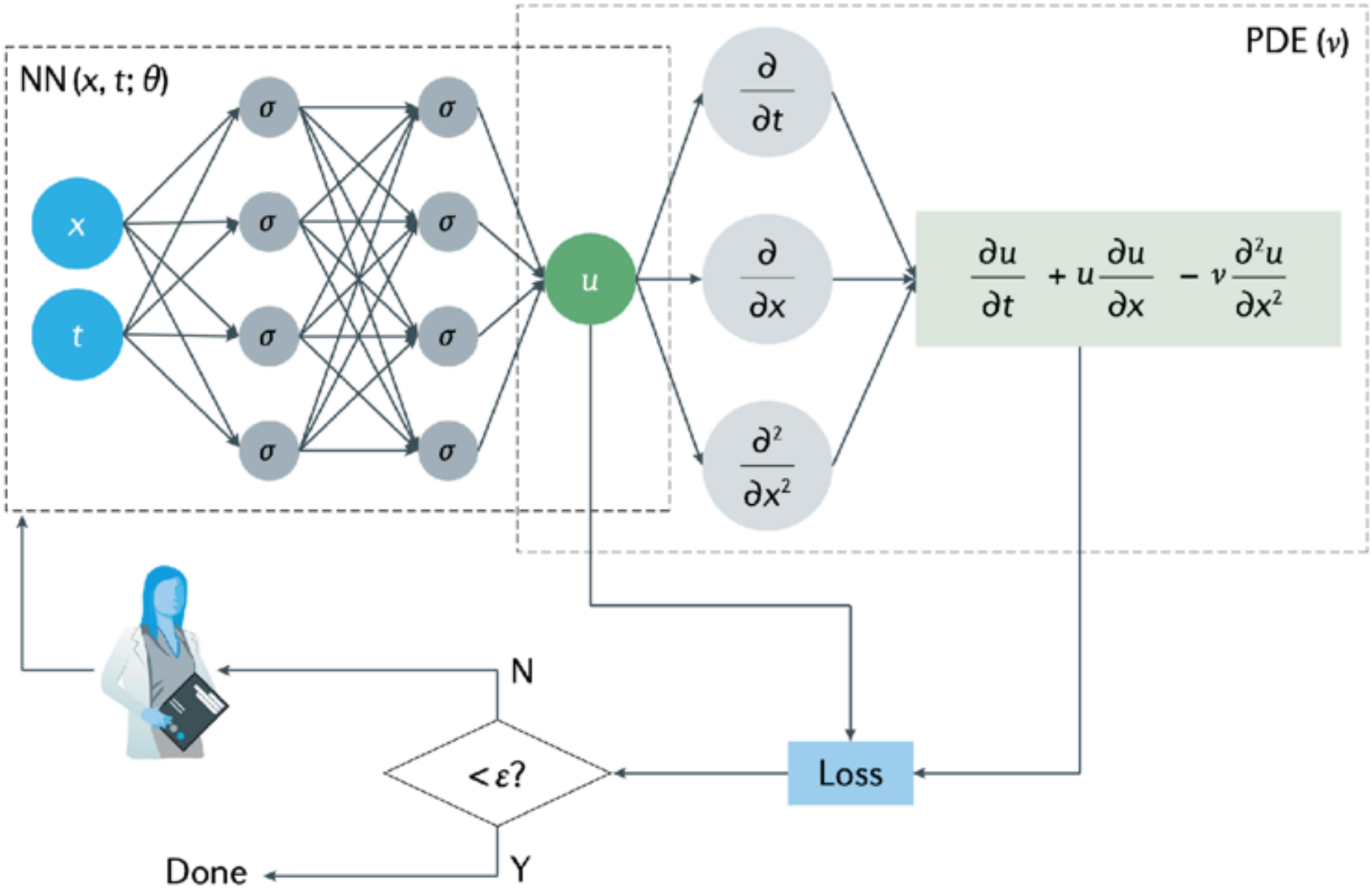
Physics-informed NN

- Given a PDE for $\mathbf{u}(\mathbf{x}, t)$: $\mathbf{u}_t = \mathscr{F}(\mathbf{u}_\mathbf{x}, \mathbf{u}_\mathbf{xx}, \dots)$
- To approximate $\mathbf{u}$ with a neural network, we want to minimize a residual term: $R = \mathbf{u}_t - \mathscr{F}(\mathbf{u}_\mathbf{x}, \mathbf{u}_\mathbf{xx}, \dots)$

- In a supervised setting, the training objective becomes:

$$\arg\min_\theta \sum_i \alpha_0 (f(x_i; \theta) - y_i)^2 + \alpha_1 R(f(x_i; \theta))$$

Physics-informed NN



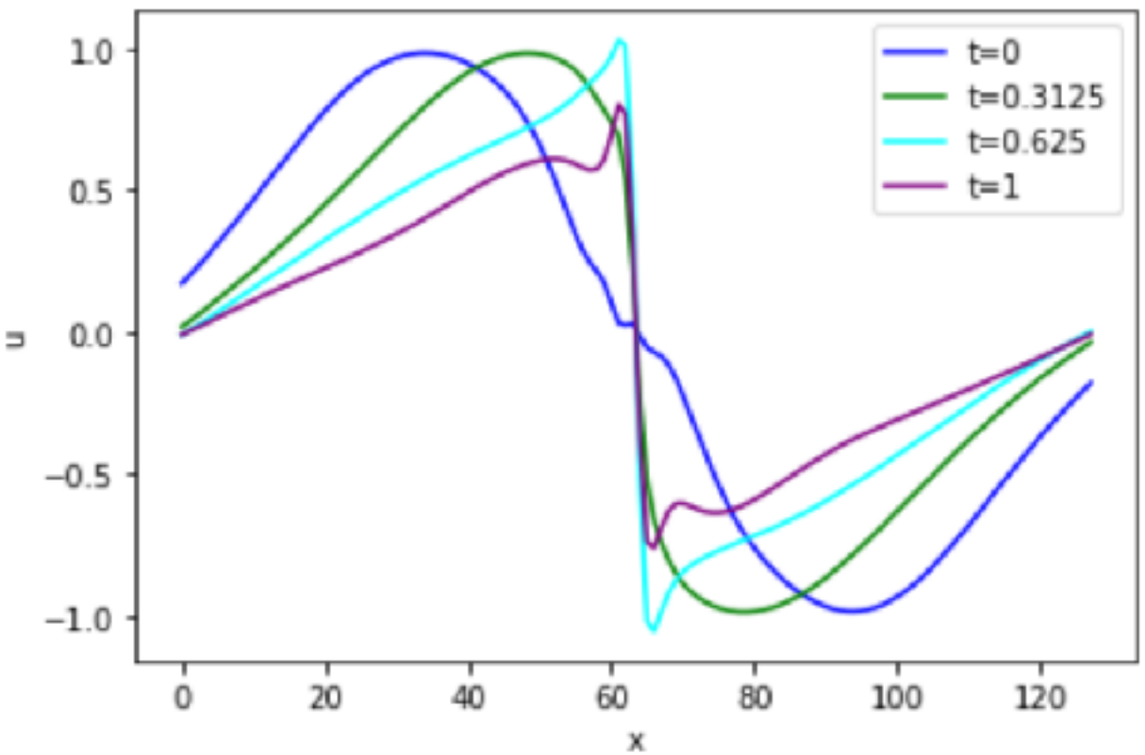
Karniadakis et al. Physics-informed machine learning. Nature Reviews Physics, 2021

Physics-informed NN

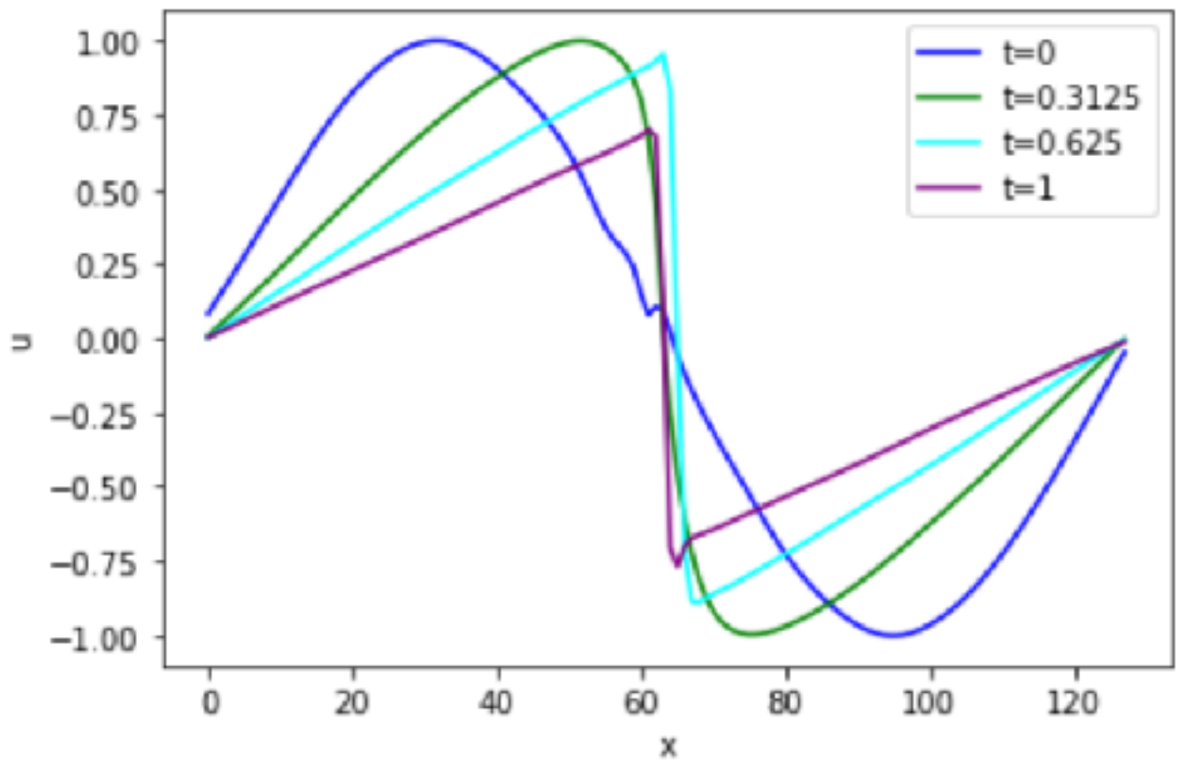
- Residual code in PyTorch

```
def residual_function(u,x,t):
  u_t = torch.autograd.grad(torch.sum(u),t, create_graph=True, retain_graph=True)[0]
  u_x = torch.autograd.grad(torch.sum(u),x, create_graph=True, retain_graph=True)[0]
  u_xx = torch.autograd.grad(torch.sum(u_x),x, create_graph=True, retain_graph=True)[0]
  return u_t + u*u_x - (0.01 / np.pi) * u_xx
```

- Results for 1D Burger's equation



Supervised MLP



PINN

About Physics-informed NN

- Physical equations are included in the form of soft constraints
- Automatic differentiation can be use to compute high-order derivatives in PDEs
- Each derivative computation can be memory expensive (and slow)
- PINN is a inverse problem with physical regularization

Differentiable physics

- Consider a continuous formulation $\mathscr{P}(\mathbf{x}, \nu)$ of a physical quantity of interest $\mathbf{u}(\mathbf{x}, t) : \mathbb{R}^d \times \mathbb{R}^+ \rightarrow \mathbb{R}^d$.

- Assume that $\mathscr{P}(\mathbf{x}, \nu)$ can be decomposed as a sequence of operations $\mathscr{P}_1, \mathscr{P}_2, \ldots \mathscr{P}_m$ such that:

$$\mathbf{u}(t + \Delta t) = \mathscr{P}_1 \circ \mathscr{P}_2 \circ \ldots \circ \mathscr{P}_m(\mathbf{u}(t), \nu)$$

- Objective: find $\mathbf{u}$ that minimizes a loss given hard physical constraints (given by $\mathscr{P}$)

Differentiable physics

- Physical model $\mathscr{P}$: $$\frac{\partial d}{\partial t} + \mathbf{u}\nabla d = 0$$
- Evolution equation: $$d(t + \Delta t) = \mathscr{P}(d(t), \mathbf{u}, t + \Delta t)$$
- Objective: find a motion $\mathbf{u}$ that transform a given initial state d^0 to a target density d^{target} at final time t^e
- Minimization problem:

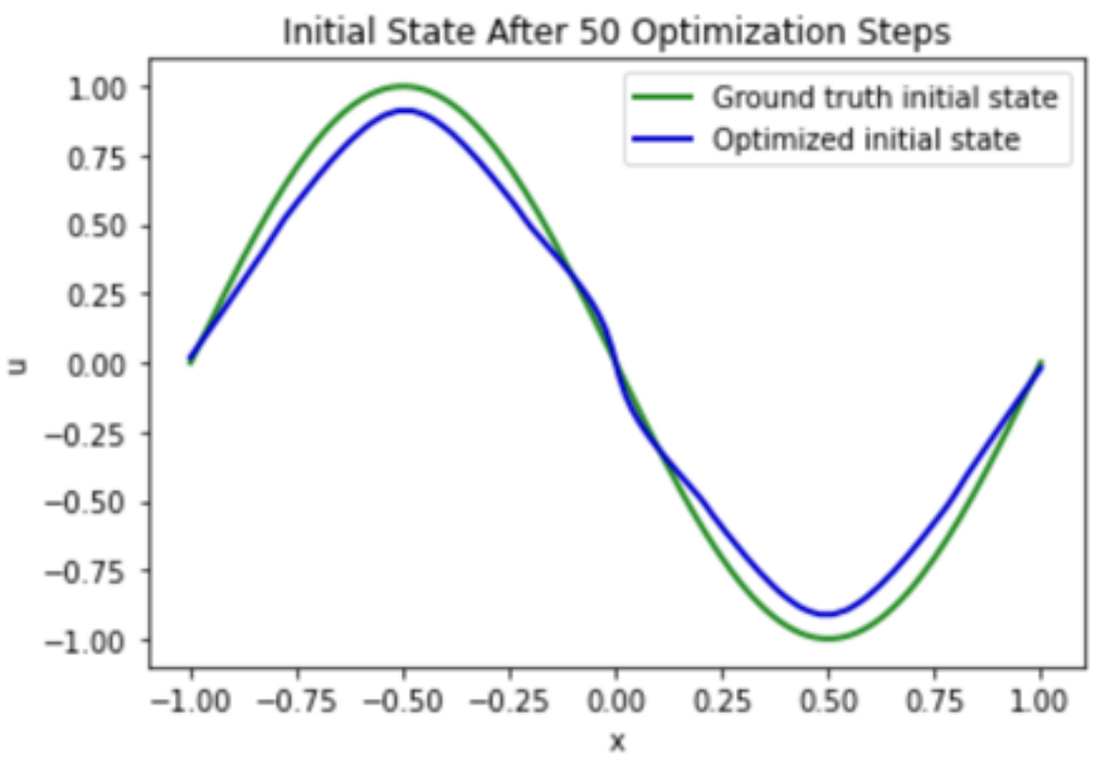
$$\arg\min_{\mathbf{u}} \|\mathscr{P}(d^0, \mathbf{u}, t^e) - d^{target}\|^2$$

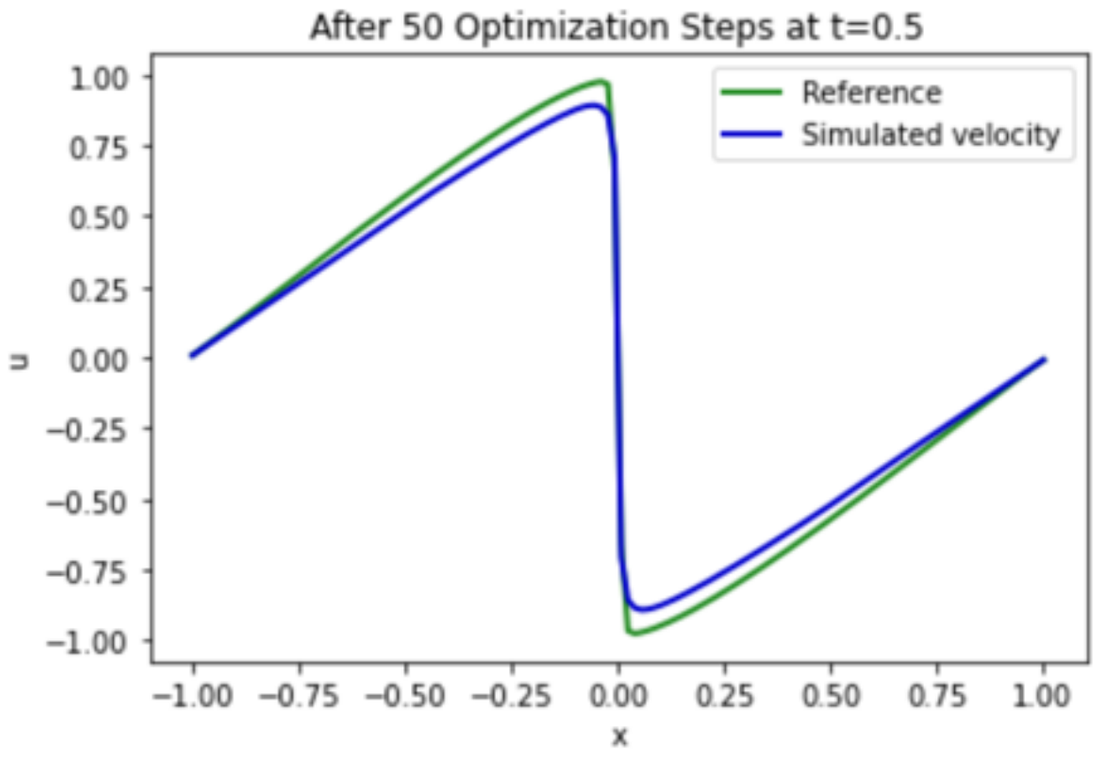
Burgers' equation and Differentiable physics

PhiFlow code

```
LR = 5.

grads=[]
for optim_step in range(5):
    velocities = [velocity]
    with math.record_gradients(velocity.values):
        for time_step in range(STEPS):
            v1 = diffuse.explicit(1.0*velocities[-1], NU, DT)
            v2 = advect.semi_lagrangian(v1, v1, DT)
            velocities.append(v2)

        loss = field.l2_loss(velocities[16] - SOLUTION_T16)*2./N # MSE
        print('Optimization step %d, loss: %f' % (optim_step,loss))

        grads.append( math.gradients(loss, velocity.values) )

    velocity = velocity - LR * grads[-1]
```

Burgers' equation and Differentiable physics





About Differentiable physics

- Makes use of physical model and numerical methods for discretization
- Efficient evaluation of simulation and derivatives
- More complicated implementation (than PINN)

https://www.physicsbased deeplearning.org

So far …

- Clear link between dynamical systems and network architecture design

- Images and velocity fields are discrete, and a neural network is a discrete numerical solver of ODE

- Automatic differentiation from NN toolbox can be used for physics

- What about continuous representations*?

** the choice of the representation is related to the choice « discretize-optimize » vs « optimize-discretize »*

Overview

Dynamical systems

Neural Networks

Neuroimaging

Image registration

Template estimation

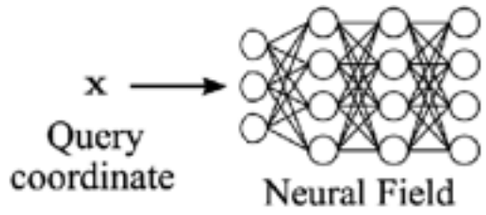
DL & Physics

Neural fields

Neural fields

- Neural fields continuously parametrize a « physical quantity » over space (and time) using a (coordinate-based) neural network

- Neural fields are often parametrized by MLPs



- Neural fields are also called implicit neural representations

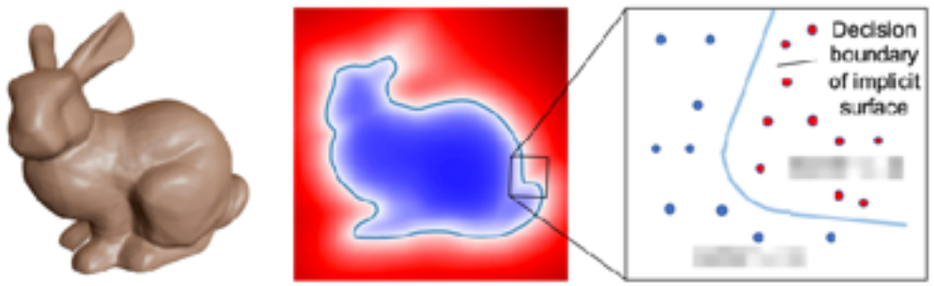
$$F\left(\mathbf{x}, \Phi, \nabla_{\mathbf{x}}\Phi, \nabla^2_{\mathbf{x}}\Phi, \ldots\right) = 0, \quad \Phi : \mathbf{x} \mapsto \Phi(\mathbf{x})$$

Example: signed distance fields for shape representation

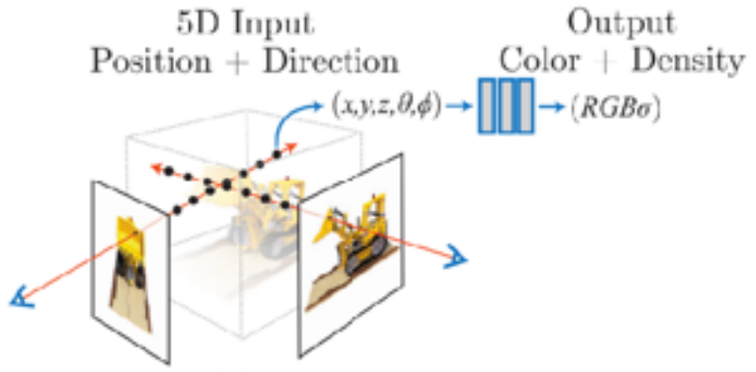
Xie et al. *Neural Fields in Visual Computing and Beyond*, Eurographics 2022.
Park et al. DeepSDF: Learning continuous signed distance functions for shape representation, CVPR 2019.

Neural fields

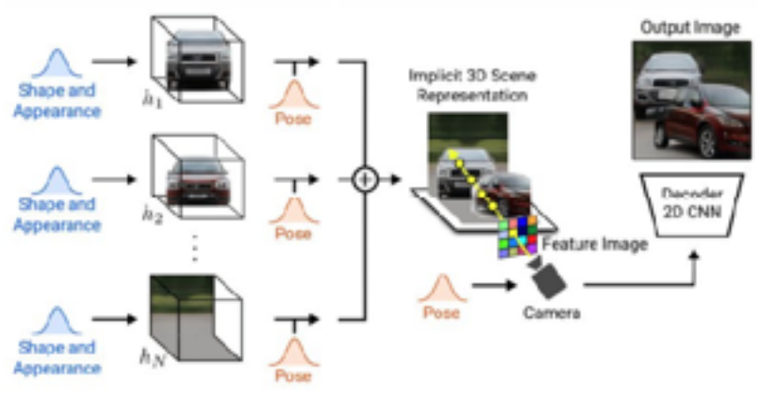
- Applications: regular or irregular grids (meshes, point clouds, etc.)



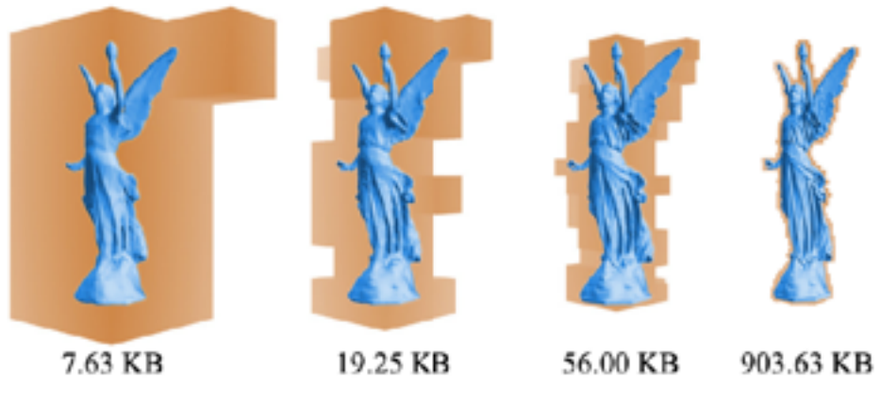
Surface parametrization



Volume rendering



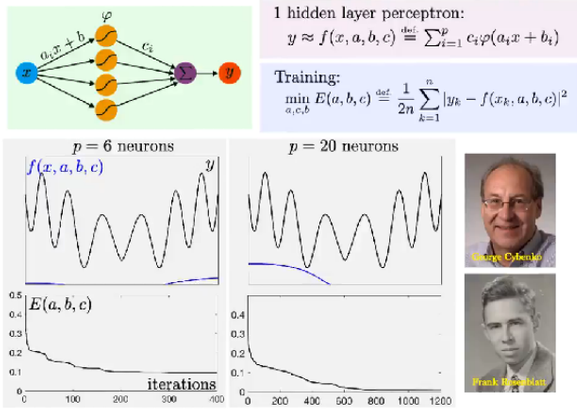
Generative models



Compression

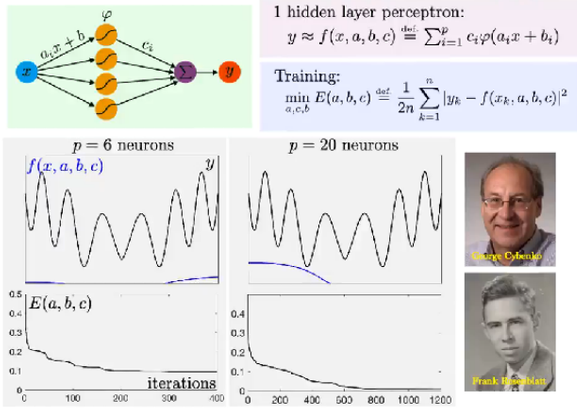
Xie et al. *Neural Fields in Visual Computing and Beyond*, Eurographics 2022.

Neural fields



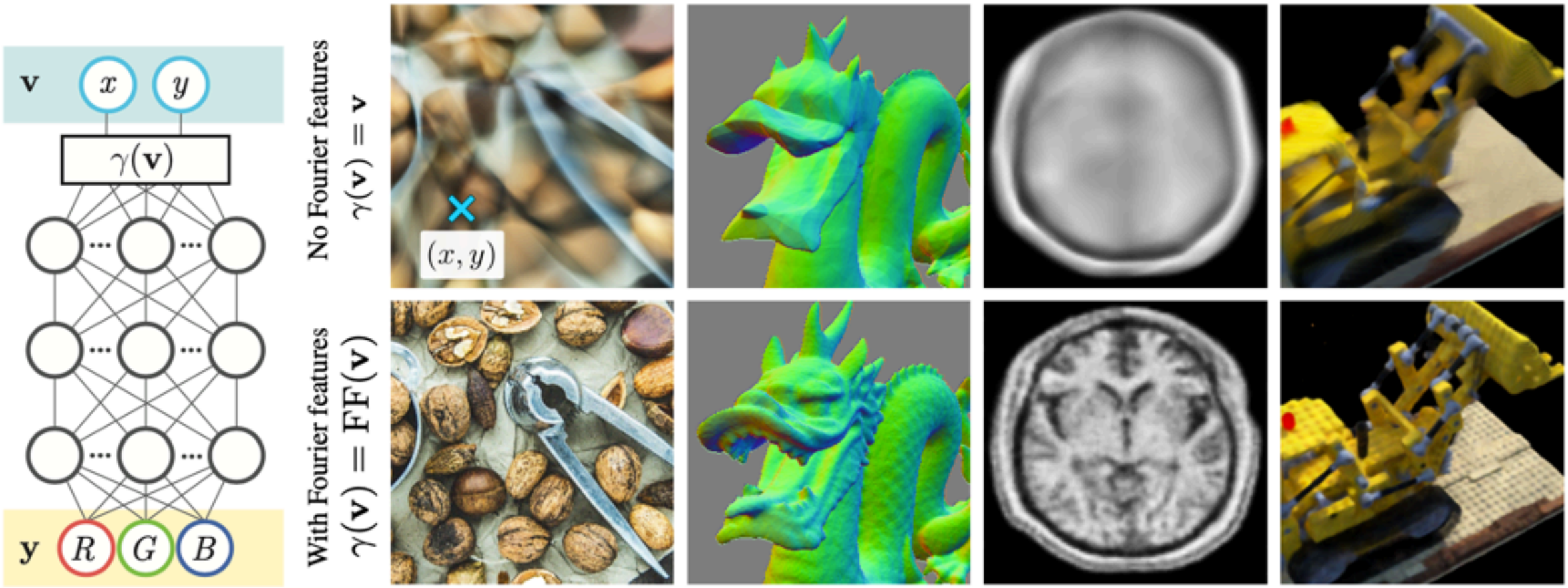
- Why neural fields?
  - More memory efficient than a discrete representation
  - Derivatives can be computed using automatic differentiation
  - Interesting for solving inverse problems
- So … neural fields = continuous representation using MLPs!

Neural fields



- But how to deal with high-frequency details with MLPs?
  - ReLU is a standard activation function, so …
  - … NN approximate functions with piecewise linear functions …
  - … which is not efficient to approximate high-frequency signals.

Neural fields: Fourier !



(a) Coordinate-based MLP

(b) Image regression
$(x,y) \rightarrow$ RGB

(c) 3D shape regression
$(x,y,z) \rightarrow$ occupancy

(d) MRI reconstruction
$(x,y,z) \rightarrow$ density

(e) Inverse rendering
$(x,y,z) \rightarrow$ RGB, density

$$\gamma(\mathbf{v}) = \left[a_1 \cos(2\pi \mathbf{b}_1^{\mathrm{T}} \mathbf{v}), a_1 \sin(2\pi \mathbf{b}_1^{\mathrm{T}} \mathbf{v}), \ldots, a_m \cos(2\pi \mathbf{b}_m^{\mathrm{T}} \mathbf{v}), a_m \sin(2\pi \mathbf{b}_m^{\mathrm{T}} \mathbf{v})\right]^{\mathrm{T}}$$

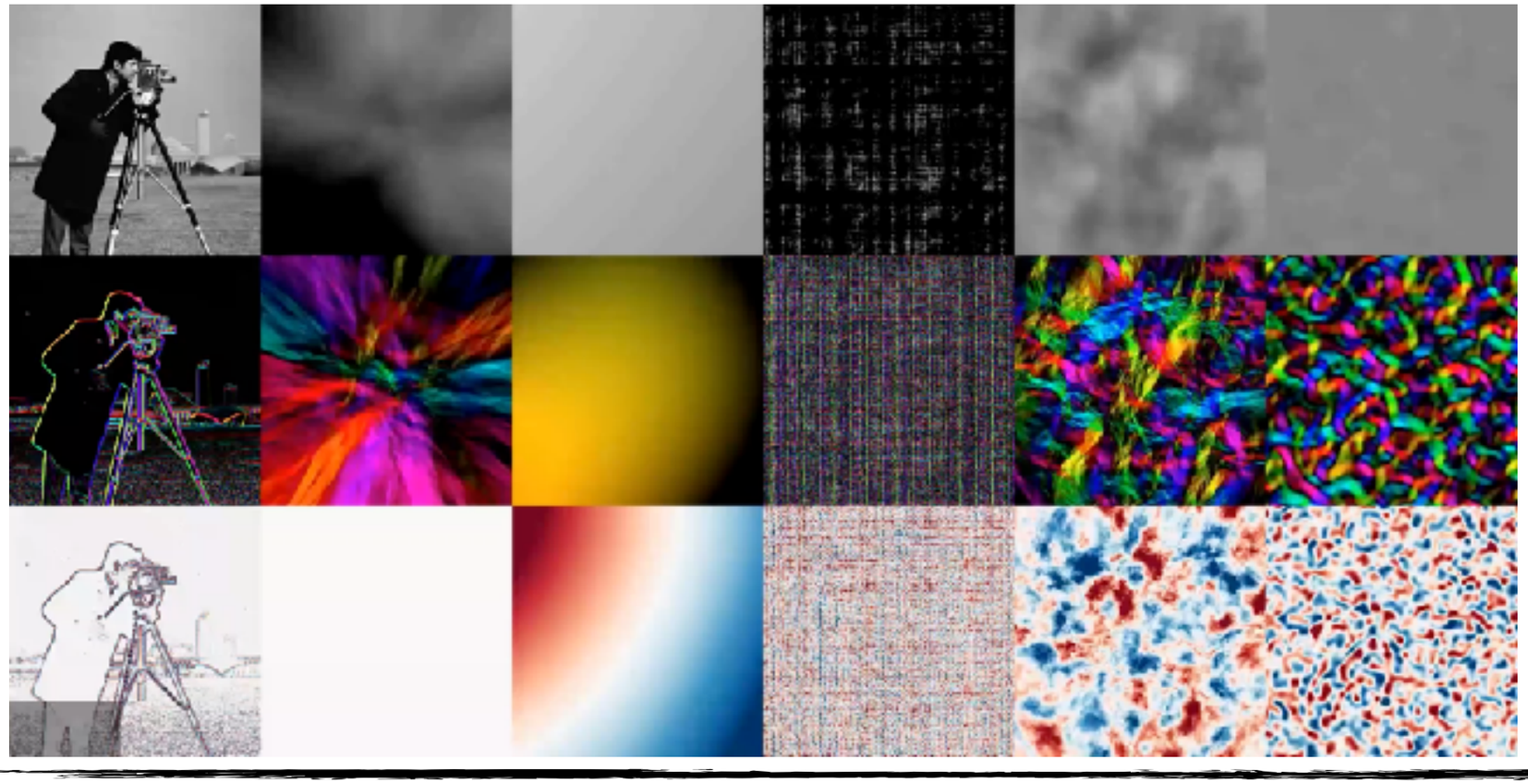
Tannic et al. *Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains*, NeurIPS 2020.

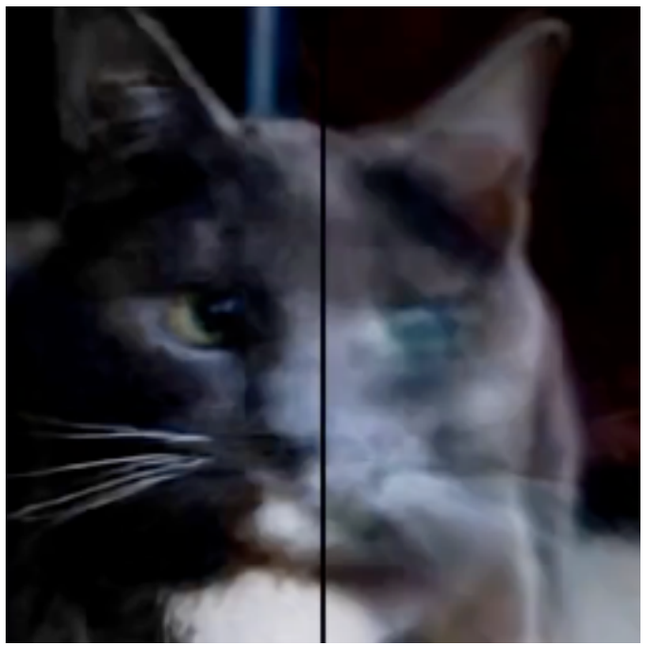
Neural fields: SIREN

- SIREN: MLP with sine activation functions

$$F\left(\mathbf{x}, \Phi, \nabla_{\mathbf{x}}\Phi, \nabla^2_{\mathbf{x}}\Phi, \ldots\right) = 0, \quad \Phi : \mathbf{x} \mapsto \Phi(\mathbf{x})$$

$$\Phi\left(\mathbf{x}\right) = \mathbf{W}_n \left(\phi_{n-1} \circ \phi_{n-2} \circ \ldots \circ \phi_0\right)\left(\mathbf{x}\right) + \mathbf{b}_n, \quad \mathbf{x}_i \mapsto \phi_i\left(\mathbf{x}_i\right) = \sin\left(\mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i\right)$$



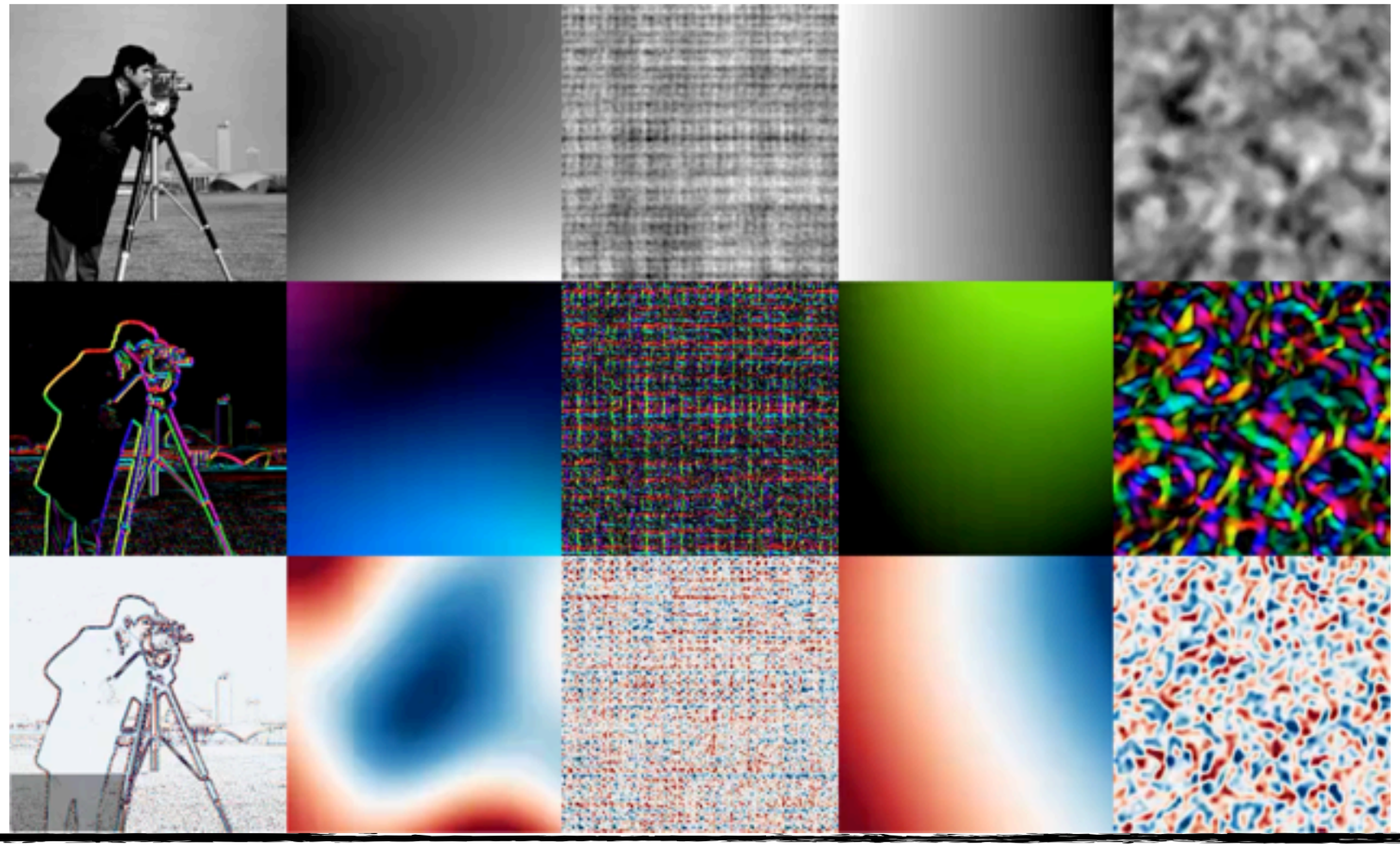


Stizmann et al. *Implicit Neural Representations with Periodic Activation Functions*, NeurIPS 2020.
https://www.vincentsitzmann.com/siren/

Neural fields: SIREN

- Solving Poisson's equation

$$F\left(\mathbf{x}, \Phi, \nabla_{\mathbf{x}}\Phi, \nabla^2_{\mathbf{x}}\Phi, \ldots\right) = 0, \quad \Phi : \mathbf{x} \mapsto \Phi(\mathbf{x})$$

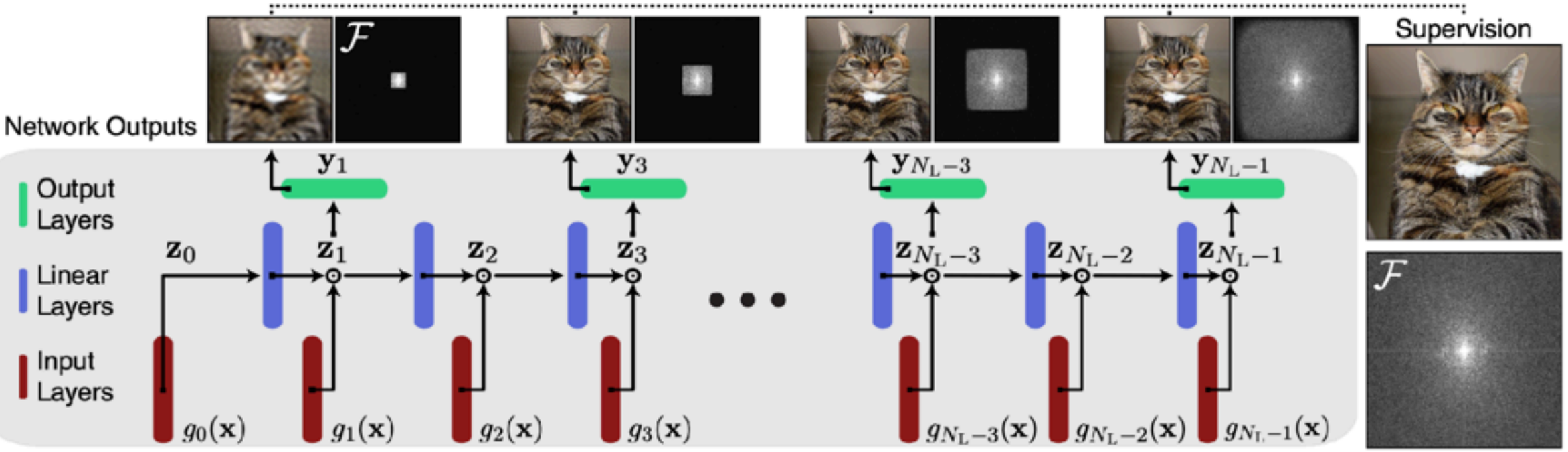


Stizmann et al. *Implicit Neural Representations with Periodic Activation Functions*, NeurIPS 2020
https://www.vincentsitzmann.com/siren/

Neural fields: BACON

- Motivation: control the spectral characteristics of estimated fields

$$
\begin{aligned}
&\mathbf{z}_0 = g_0(\mathbf{x}) \\
&\mathbf{z}_i = g_i(\mathbf{x}) \circ (\mathbf{W}_i \mathbf{z}_{i-1} + \mathbf{b}_i)\,, \quad 0 \leq i < N_\mathrm{L} \quad \text{with } g_i(\mathbf{x}) = \sin(\boldsymbol{\omega}_i \mathbf{x} + \boldsymbol{\phi}_i) \\
&\mathbf{y}_i = \mathbf{W}_i^{\text{out}} \mathbf{z}_i + \mathbf{b}_i^{\text{out}},
\end{aligned}
$$



Lindell et al. *BACON: Band-limited Coordinate Networks for Multiscale Scene Representation*, CVPR 2022.

Neural fields: BACON

- Motivation: control the spectral characteristics of estimated fields

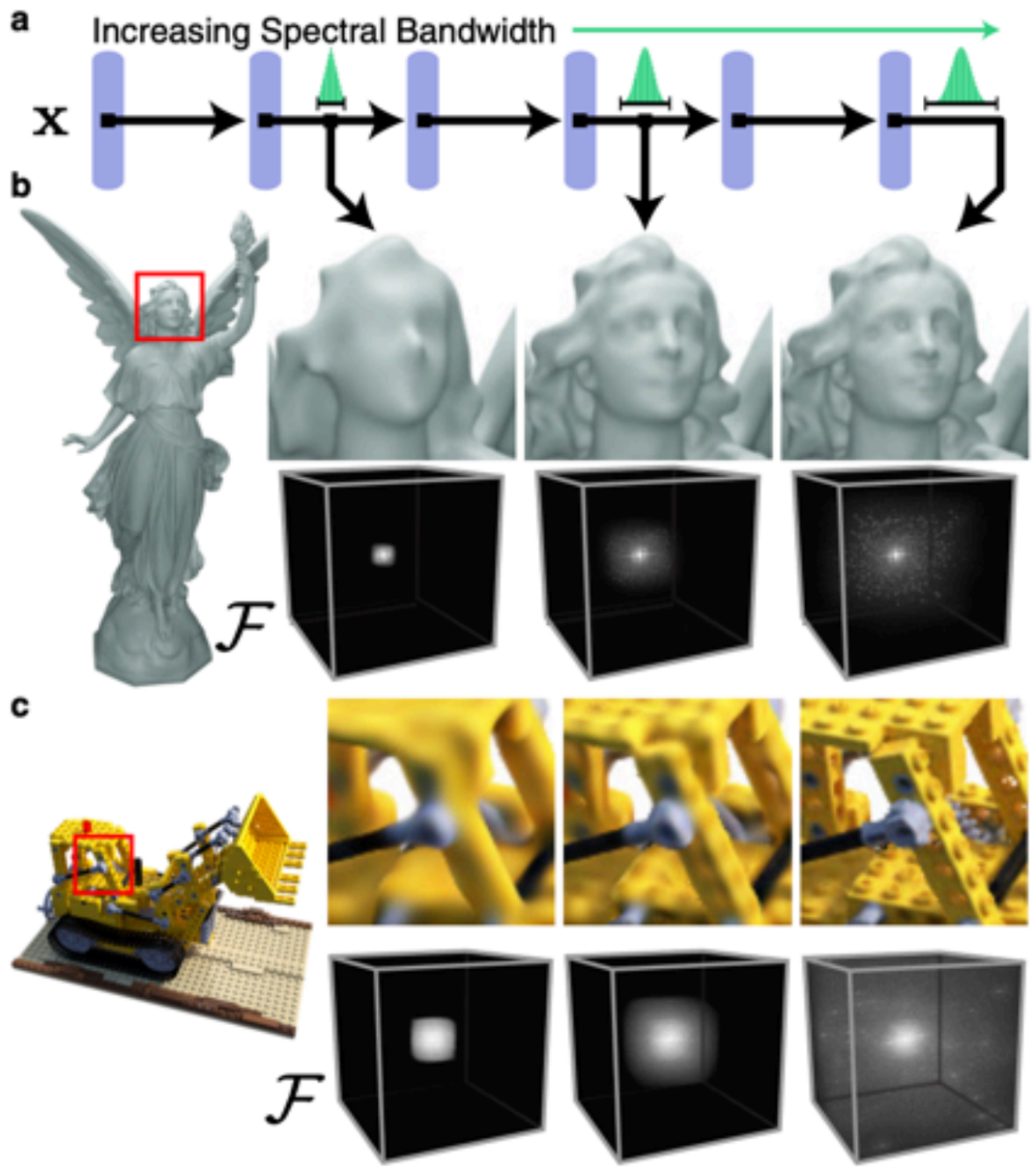
$$\mathbf{z}_0 = g_0(\mathbf{x})$$
$$\mathbf{z}_i = g_i(\mathbf{x}) \circ (\mathbf{W}_i \mathbf{z}_{i-1} + \mathbf{b}_i)\,, \quad 0 \leq i < N_\mathrm{L} \quad \text{with } g_i(\mathbf{x}) = \sin(\boldsymbol{\omega}_i \mathbf{x} + \boldsymbol{\phi}_i)$$
$$\mathbf{y}_i = \mathbf{W}_i^\mathrm{out} \mathbf{z}_i + \mathbf{b}_i^\mathrm{out},$$

- The network output is a sum of sines

$$\mathbf{y}_i = \sum_{j=0}^{N_\mathrm{sine}^{(i)}-1} \bar{\alpha}_j \sin(\bar{\boldsymbol{\omega}}_j \mathbf{x} + \bar{\phi}_j)$$

Lindell et al. *BACON: Band-limited Coordinate Networks for Multiscale Scene Representation*, CVPR 2022.

Neural fields: BACON



It learns a multi-resolution decomposition of the data.

Lindell et al. *BACON: Band-limited Coordinate Networks for Multiscale Scene Representation,* CVPR 2022.

Neural fields: BACON

Training at resolution x1, and reconstruction at 1/4 and 4x resolution.

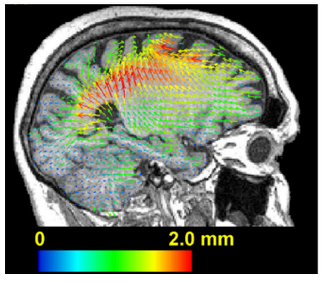
Lindell et al. *BACON: Band-limited Coordinate Networks for Multiscale Scene Representation,* CVPR 2022.

Implicit deformable image registration

- Motivation: use a continuous representation of the deformation field for flexible regularization terms (and to avoid numerical errors)

- Jacobian regularizer: $\mathcal{S}^{\text{jac}}[\Phi] = \int_{\Omega} |1 - \det \nabla \Phi| d\mathbf{x}.$

- Implicit representation of Φ using SIREN



Wolterink et al. *Implicit neural representations for deformable image registration,* MIDL 2022.

So Far…

- Neural fields: implicit continuous representation using neural network (MLPs)

- Memory efficient alternative to discrete representation (CNN etc.)

- Information is encoded in the weights of the NN

- Challenge of generalization (latent code, meta-learning…)

- How to define a continuous representation of the dynamical system?

Overview

Dynamical systems

Neural Networks

Neuroimaging

Image registration

Template estimation

DL & Physics

Neural fields

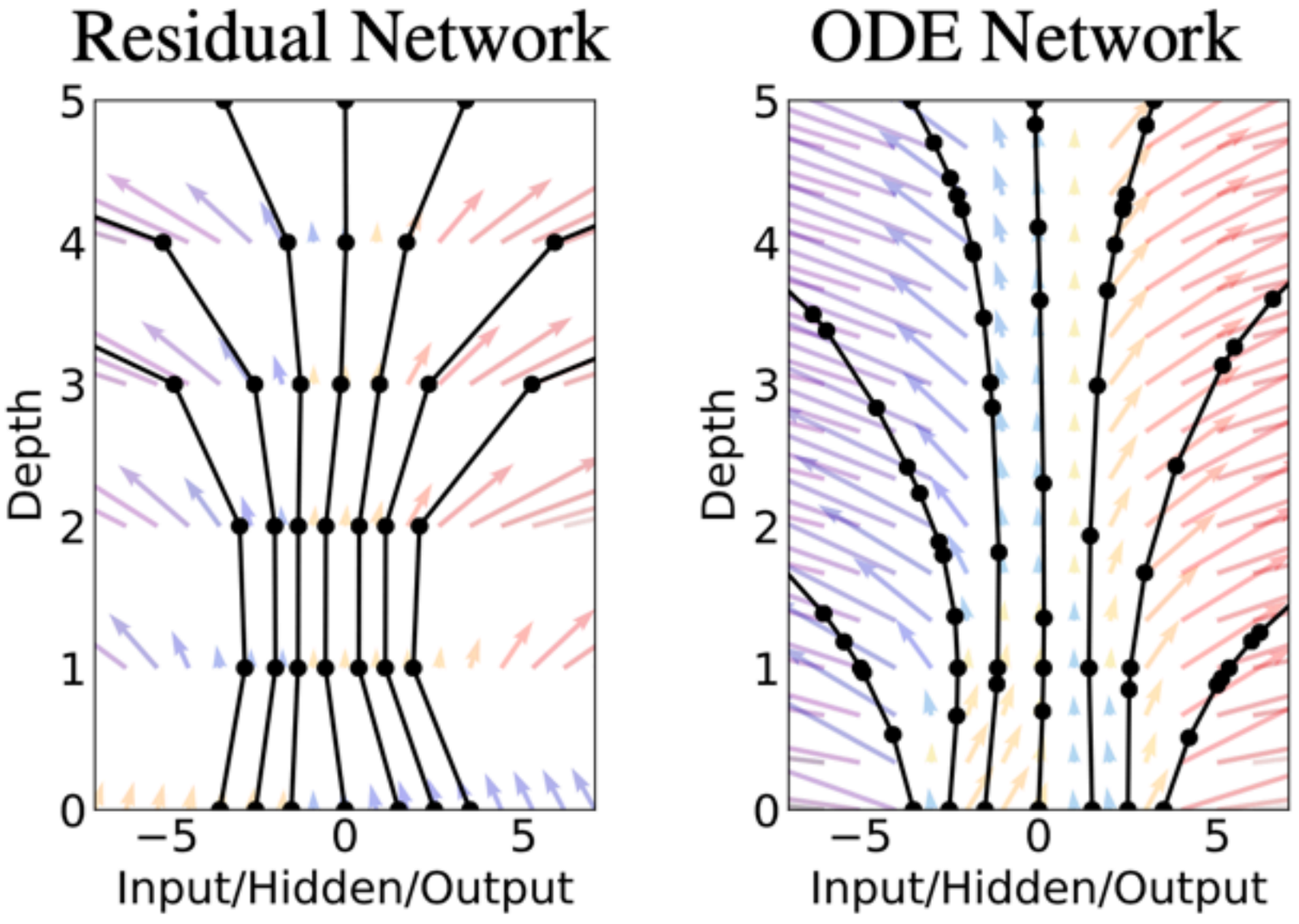
Neural ODE

Neural ODE

- ResNet = Euler discretization of continuous formulation
- Increasing the number of layers in ResNet tends to continuous formulation, but …
- … this incurs a high-memory cost to save all intermediate activations of the solver for the backpropagation step.
- Idea of neural ODE is to replace the discretization step (i.e. ResNet blocks) by a ODE solver (of a IVP)
- Reminder: IVP consists in finding a function x solution to the ODE given f and x_0

$$\frac{dx(t)}{dt} = f(x(t), t) \qquad x(t_0) = x_0$$

Neural ODE



$$\frac{dx(t)}{dt} = f(x(t), t)$$

In ODE net, f is a NN.

Chen et al. *Neural Ordinary Differential Equations,* NeurIPS 2018.

Neural ODE

- Given a ODE solver, f and x_0, we can compute x_1. But how to perform the backpropagation step?

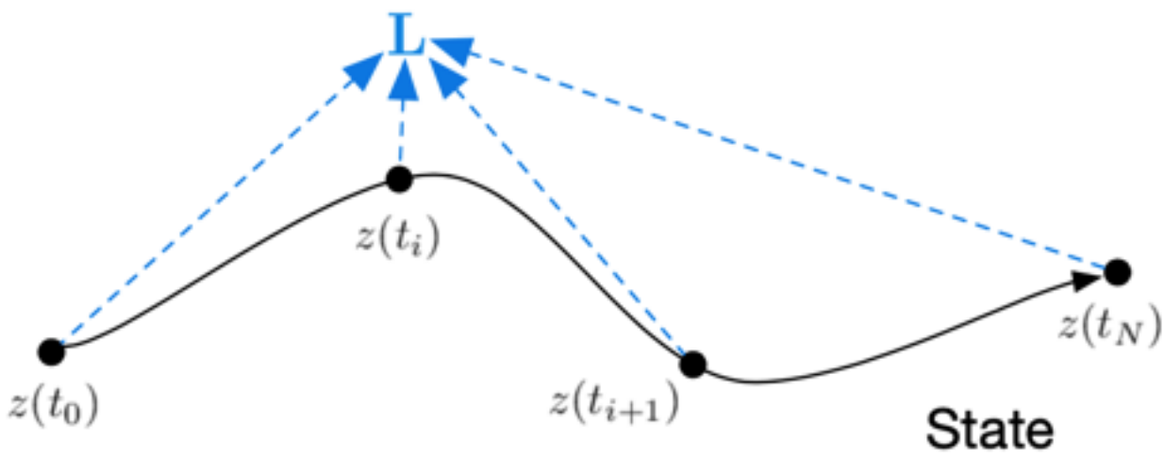
- Backpropagation: optimize the weights of the NN by minimizing the loss function (by gradient descent).

$$\theta_{k+1} = \theta_k - \epsilon_k \frac{\partial L(\theta_k)}{\partial \theta_k}$$

Neural ODE

- Consider the following loss function:

$$L(\mathbf{z}(t_1)) = L\left(\mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t, \theta)dt\right) = L\left(\text{ODESolve}(\mathbf{z}(t_0), f, t_0, t_1, \theta)\right)$$



Neural ODE

- Consider the following loss function:

$$L(\mathbf{z}(t_1)) = L\left(\mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t, \theta)dt\right) = L\left(\text{ODESolve}(\mathbf{z}(t_0), f, t_0, t_1, \theta)\right)$$

- The adjoint: $\mathbf{a}(t) = \dfrac{\partial L}{\partial \mathbf{z}(t)}$

- The gradient of the loss wrt the parameters is given by

$$\frac{dL}{d\theta} = \int_{t_1}^{t_0} \mathbf{a}(t)^\mathsf{T} \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \theta} dt$$

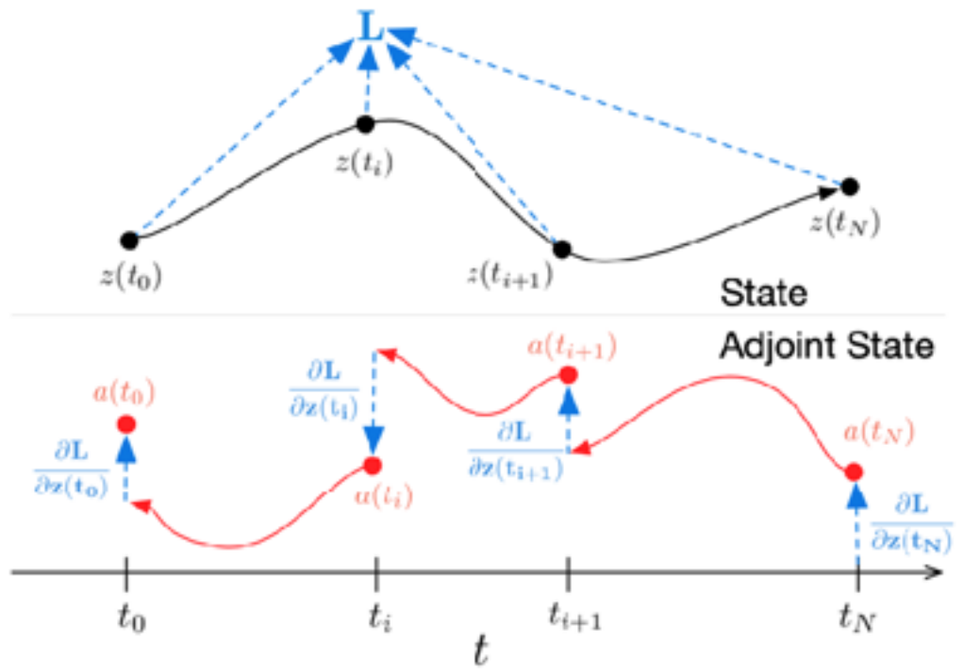
- We need to compute $\mathbf{z}(t)$ and the adjoint

Neural ODE

- The dynamics of the adjoint is given by the following ODE:

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^\mathsf{T} \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \mathbf{z}}$$

- The adjoint sensitivity method solves an augmented ODE backwards in time



Pontryagin et al. *The mathematical theory of optimal processes*, 1962
Chen et al. *Neural Ordinary Differential Equations,* NeurIPS 2018.

Neural ODE

Algorithm 1 Reverse-mode derivative of an ODE initial value problem

Input: dynamics parameters θ, start time t_0, stop time t_1, final state $\mathbf{z}(t_1)$, loss gradient $\partial L / \partial \mathbf{z}(t_1)$

$s_0 = [\mathbf{z}(t_1), \frac{\partial L}{\partial \mathbf{z}(t_1)}, \mathbf{0}_{|\theta|}]$ ▷ Define initial augmented state

def aug_dynamics$([\mathbf{z}(t), \mathbf{a}(t), \cdot], t, \theta)$: ▷ Define dynamics on augmented state

  return $[f(\mathbf{z}(t), t, \theta), -\mathbf{a}(t)^\mathsf{T} \frac{\partial f}{\partial \mathbf{z}}, -\mathbf{a}(t)^\mathsf{T} \frac{\partial f}{\partial \theta}]$ ▷ Compute vector-Jacobian products

$[\mathbf{z}(t_0), \frac{\partial L}{\partial \mathbf{z}(t_0)}, \frac{\partial L}{\partial \theta}] = \text{ODESolve}(s_0, \text{aug\_dynamics}, t_1, t_0, \theta)$ ▷ Solve reverse-time ODE

return $\frac{\partial L}{\partial \mathbf{z}(t_0)}, \frac{\partial L}{\partial \theta}$ ▷ Return gradients

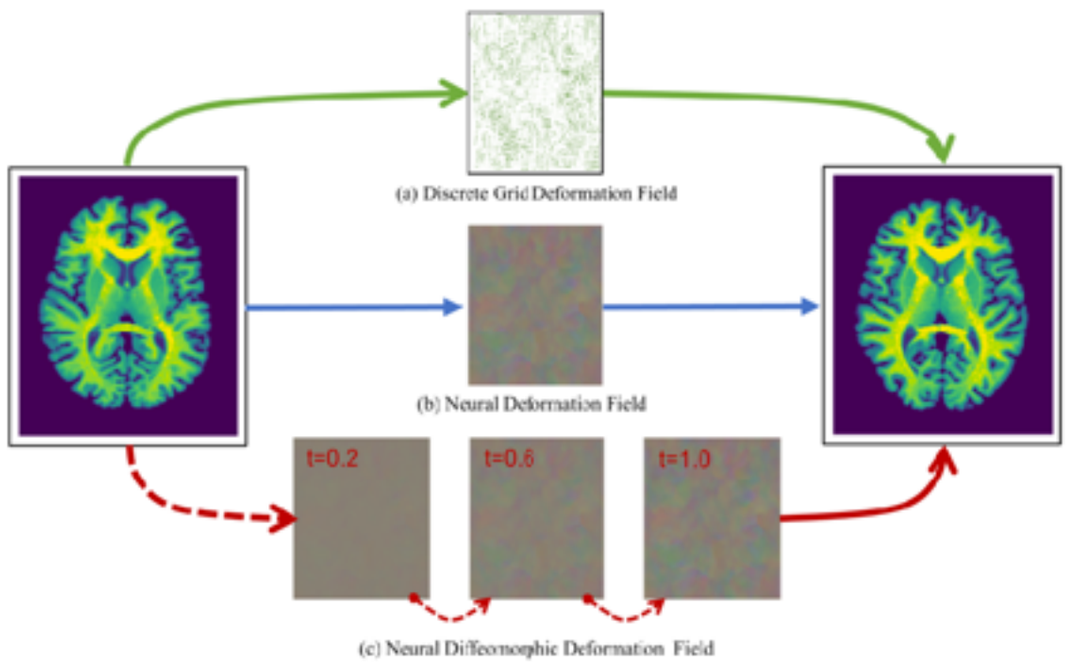
Chen et al. *Neural Ordinary Differential Equations,* NeurIPS 2018.

Medical image registration via neural fields

- Diffeomorphic mapping using stationary velocity fields:

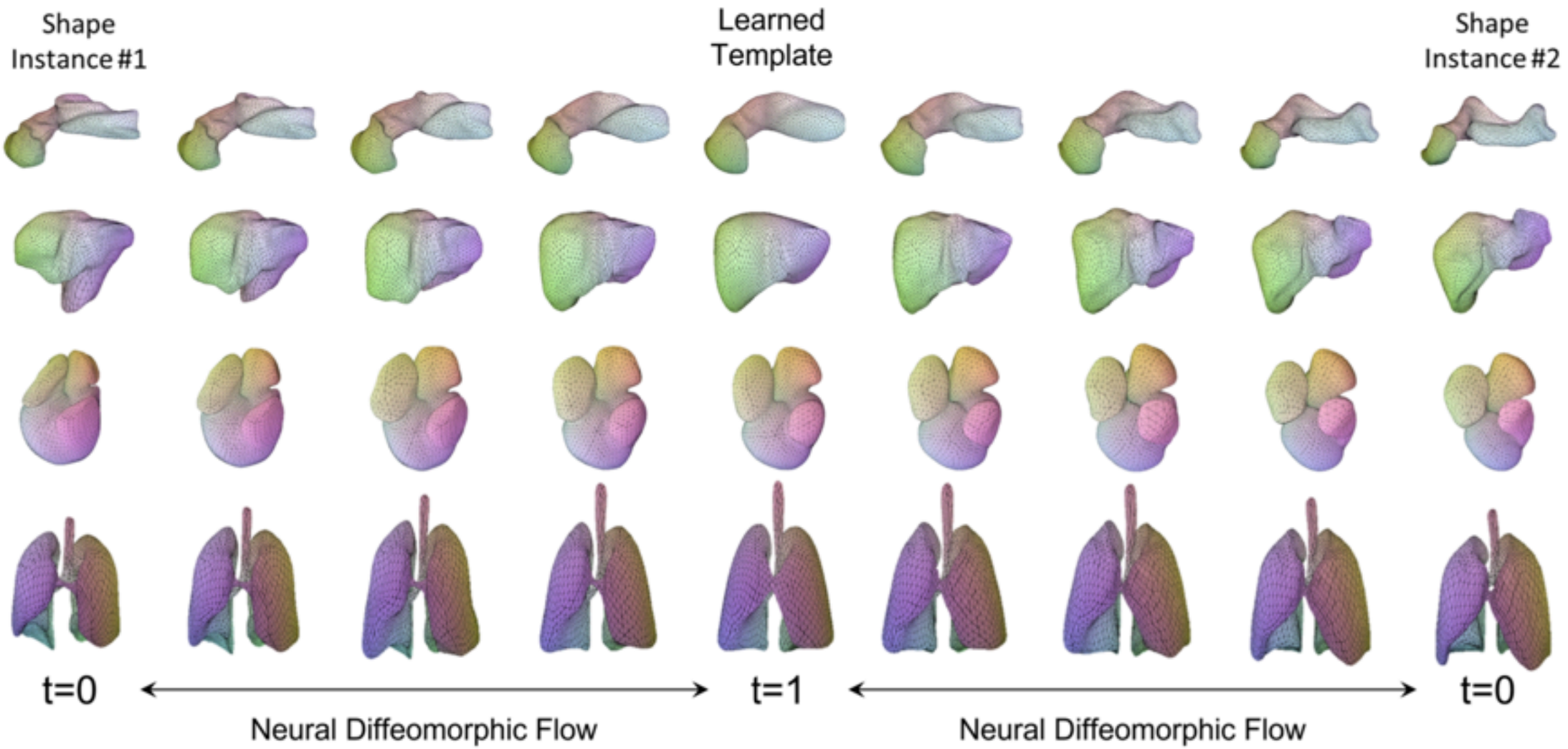
$$\Phi_i(\boldsymbol{p}, 1) = \Phi_i(\boldsymbol{p}, 0) + \int_0^1 \boldsymbol{v}(\Phi(\boldsymbol{p}, t))\mathrm{d}t$$

- Neural ODE is used to estimate the velocity field modeled with SIREN



Sun et al. *Medical image registration via neural fields,* Arxiv 2022.

Implicit Atlas



Sun et al. *Topology-Preserving Shape Reconstruction and Registration via Neural Diffeomorphic Flow*, CVPR 2022.
Yang et al. *ImplicitAtlas: Learning Deformable Shape Templates in Medical Imaging,* CVPR 2022.

Overview

Dynamical systems

Neural Networks

Neuroimaging

Image registration

Template estimation

DL & Physics

Neural fields

Neural ODE

Coffee break?

Take home messages

- Neural networks and dynamical systems can be interconnected

- Image registration for morphometry in neuroimaging can be modeled as a dynamical system

- Discrete vs continuous representations (implicit « stuff »)

- NN and dynamical systems are also related to optimal control, optimal transport, stochastic modeling etc.

Deep learning and dynamical systems: applications in neuroimaging

François Rousseau

Institut Mines-Télécom