Les mathématiques de l'intelligence artificielle

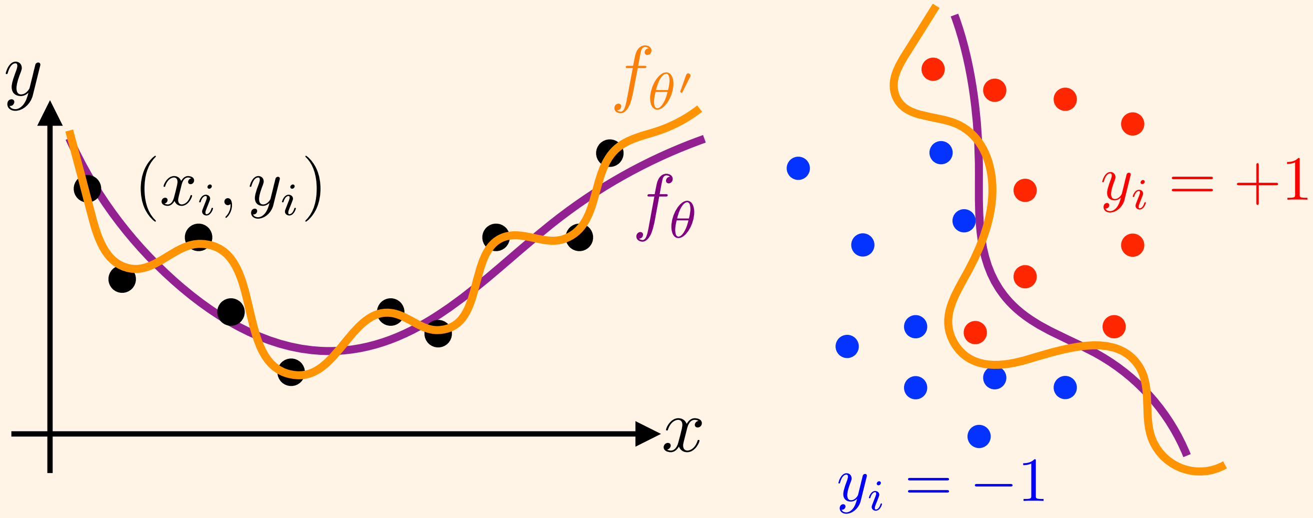
Gabriel Peyré

www.numerical-tours.com

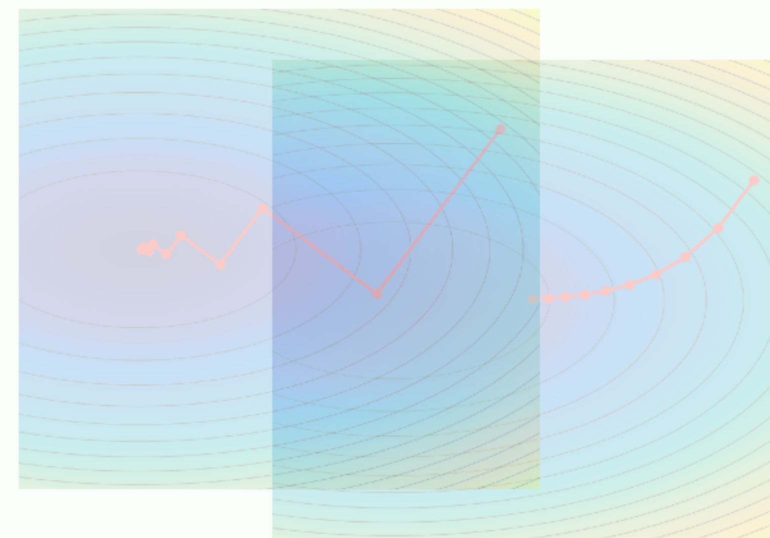
1794
ENS
ÉCOLE NORMALE SUPÉRIEURE

Overview

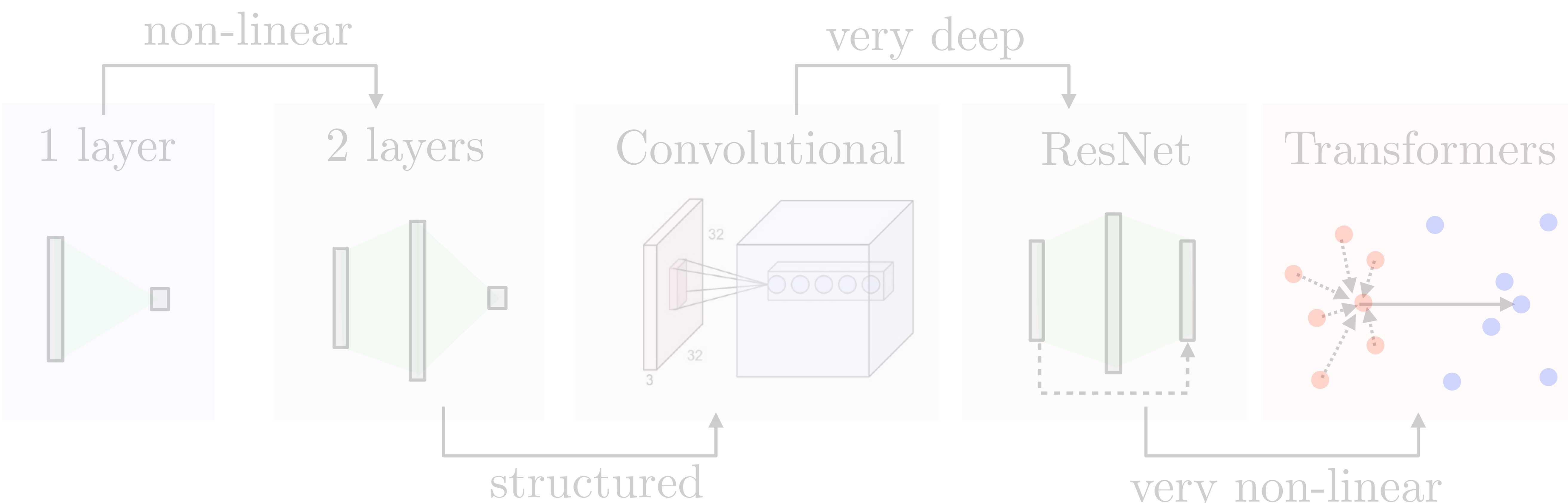
Machine Learning



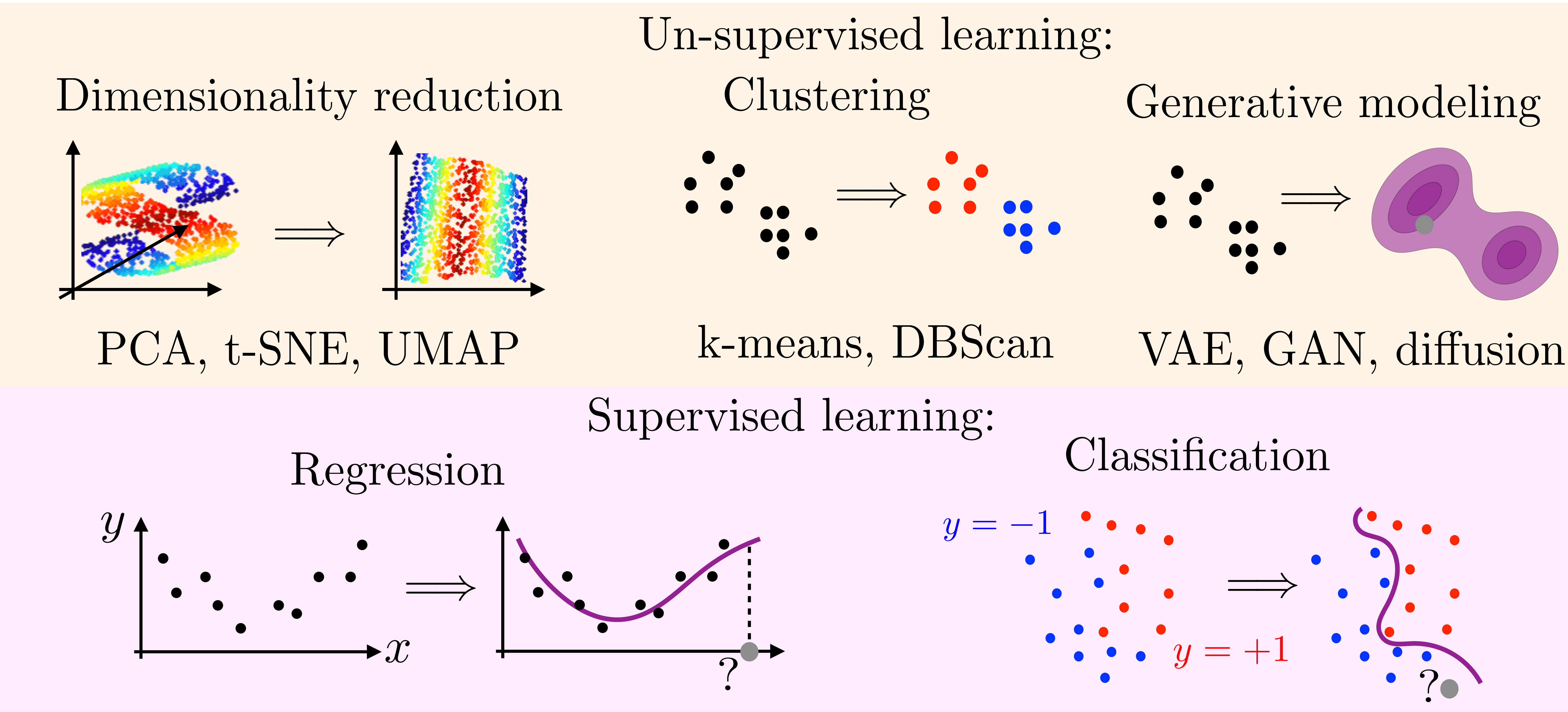
Optimization



Networks Architectures



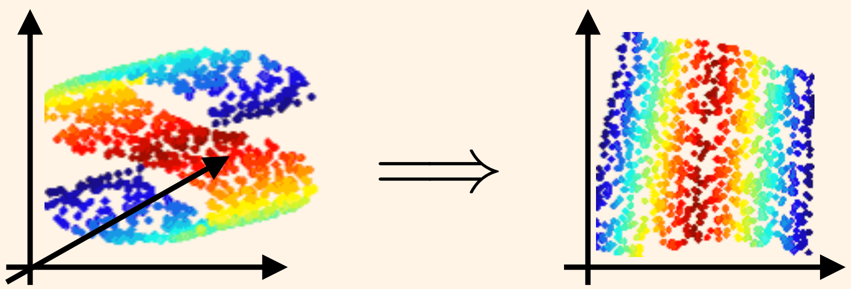
Supervised vs Unsupervised Learning



Supervised vs Unsupervised Learning

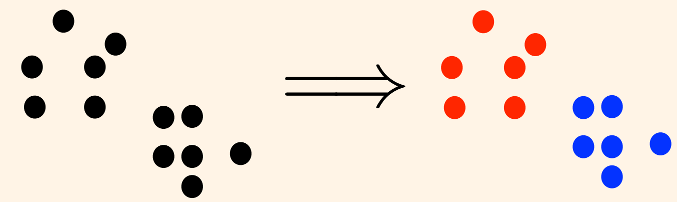
Un-supervised learning:

Dimensionality reduction



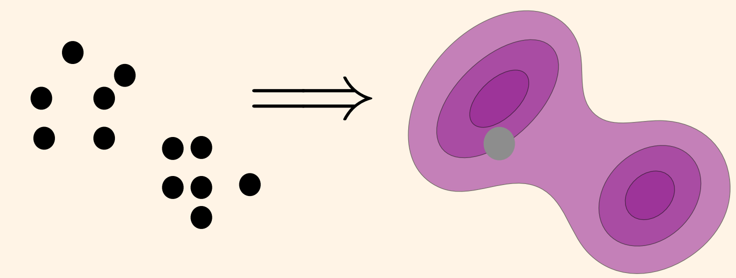
PCA, t-SNE, UMAP

Clustering



k-means, DBScan

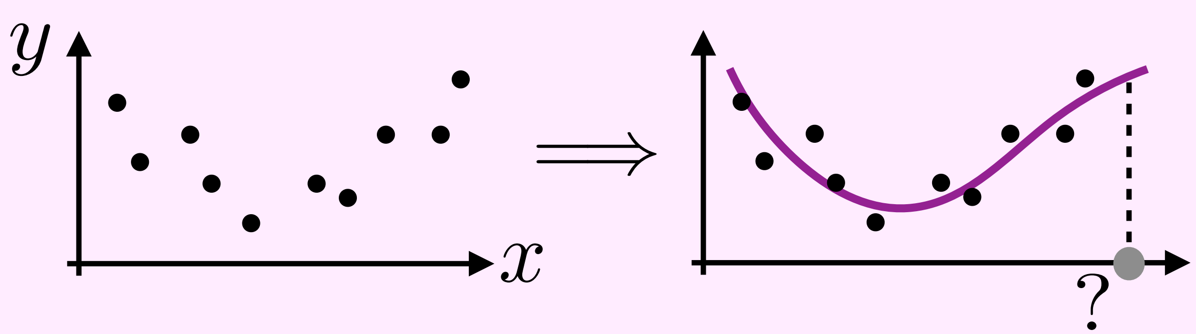
Generative modeling



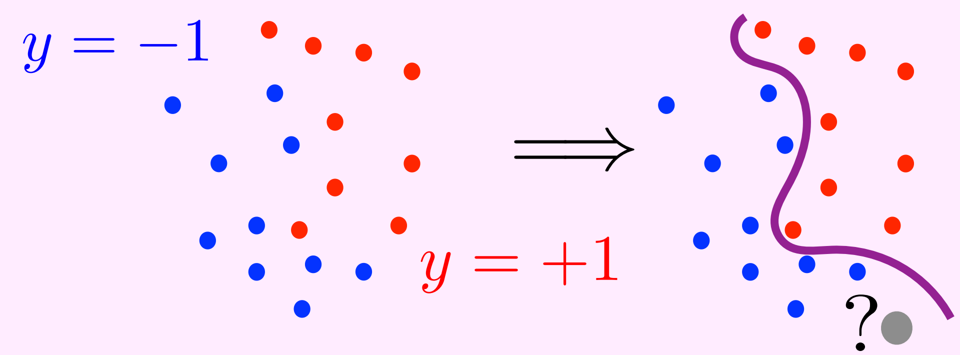
VAE, GAN, diffusion

Supervised learning:

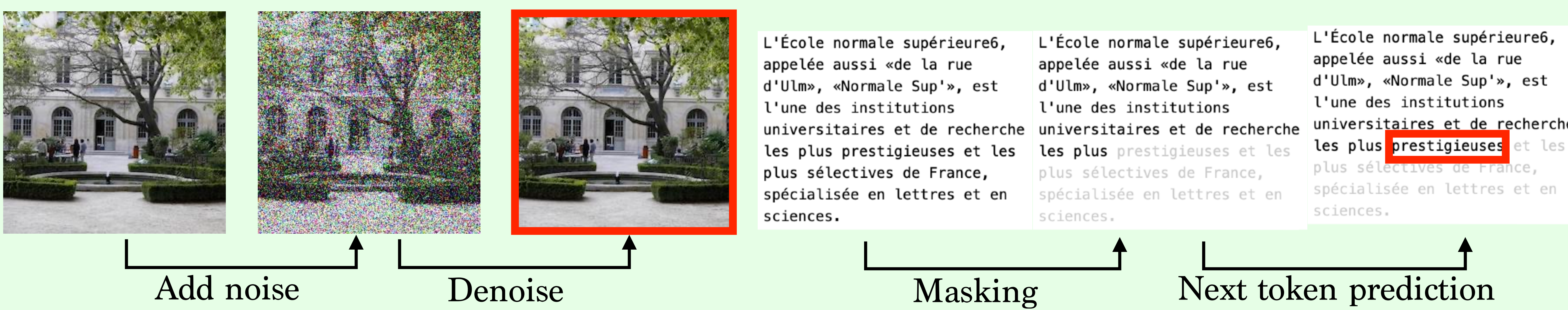
Regression



Classification



Self-supervised learning:



Empirical Risk Minimization

Dataset: $(x_i, y_i)_{i=1}^n$.

$x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R}$

Empirical risk minimization:

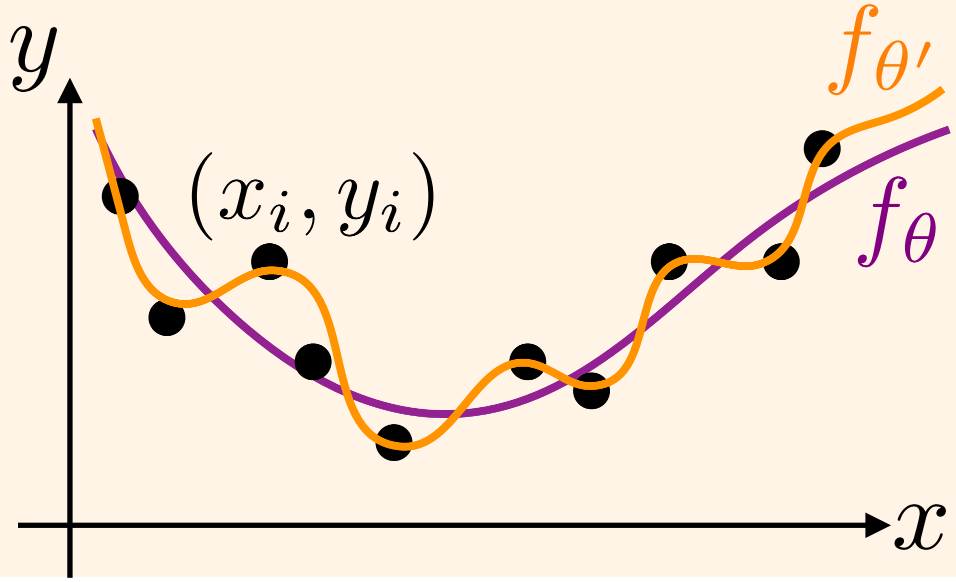
$$\min_\theta \frac{1}{n} \sum_{i=1}^n \ell(f_\theta(x_i), y_i)$$

Regression: $y_i \in \mathbb{R}$

$$y \approx f_\theta(x)$$

Least square:

$$\ell(y, y') = (y - y')^2$$

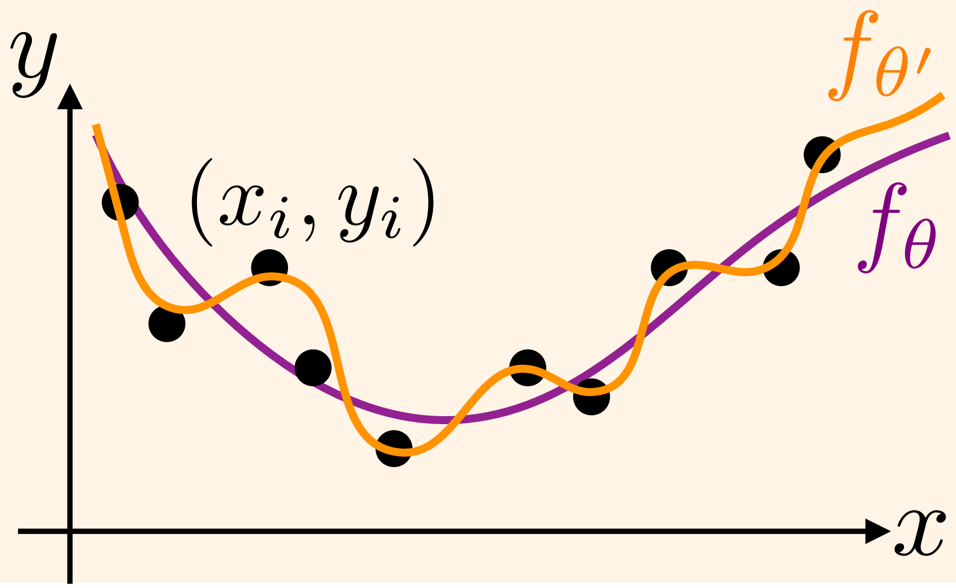


Empirical Risk Minimization

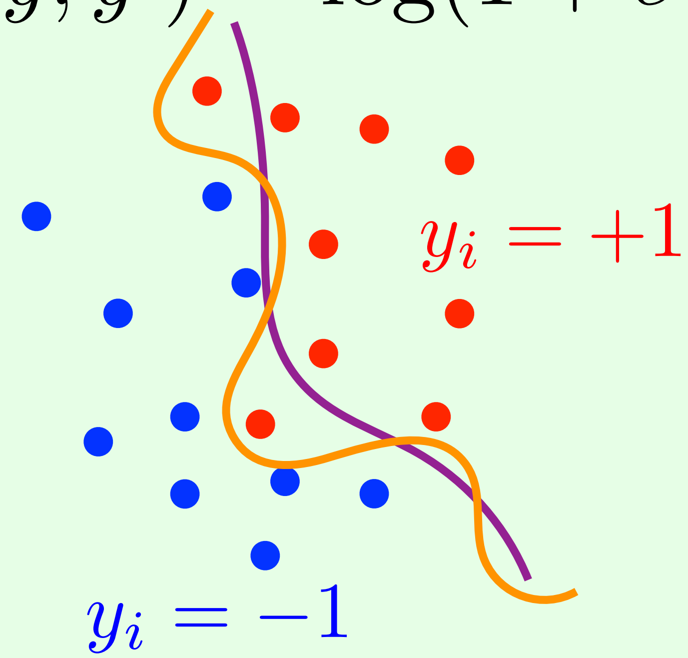
Dataset: $(x_i, y_i)_{i=1}^n$.
$x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R}$

Empirical risk minimization:
$$\min_\theta \frac{1}{n} \sum_{i=1}^n \ell(f_\theta(x_i), y_i)$$

Regression: $y_i \in \mathbb{R}$
$$y \approx f_\theta(x)$$
Least square:
$$\ell(y, y') = (y - y')^2$$



Classification: $y_i \in \{-1, 1\}$
$$y \approx \text{sign}(f_\theta(x))$$
Logistic:
$$\ell(y, y') = \log(1 + e^{-yy'})$$



Empirical Risk Minimization

Dataset: $(x_i, y_i)_{i=1}^n$.

$x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R}$

Empirical risk minimization:

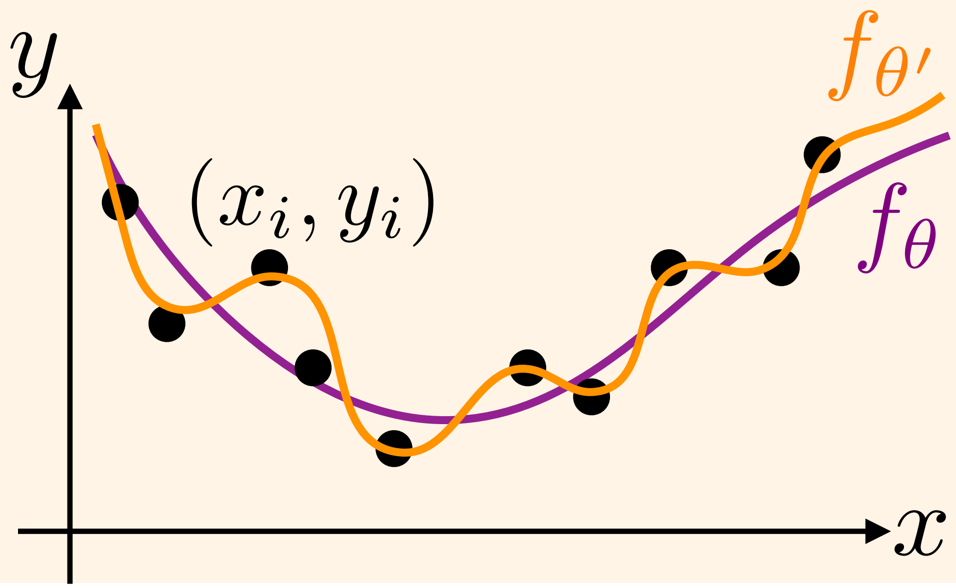
$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i)$$

Regression: $y_i \in \mathbb{R}$

$$y \approx f_\theta(x)$$

Least square:

$$\ell(y, y') = (y - y')^2$$

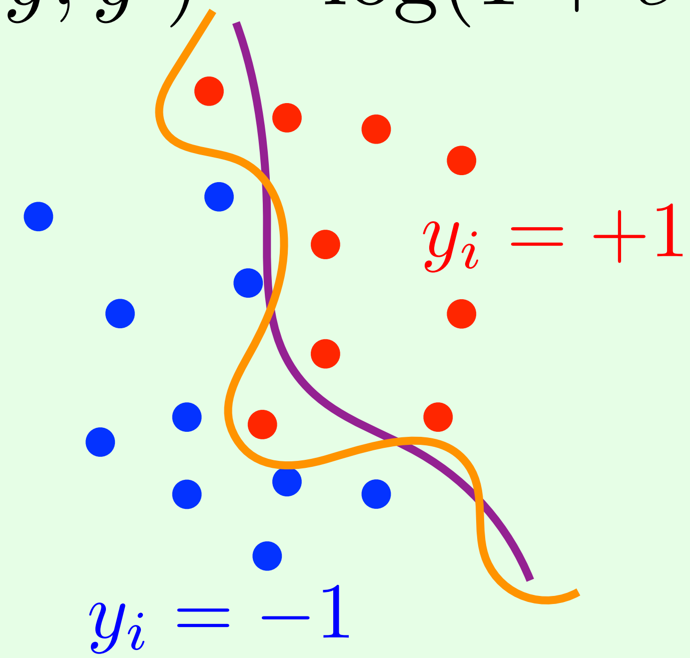


Classification: $y_i \in \{-1, 1\}$

$$y \approx \text{sign}(f_\theta(x))$$

Logistic:

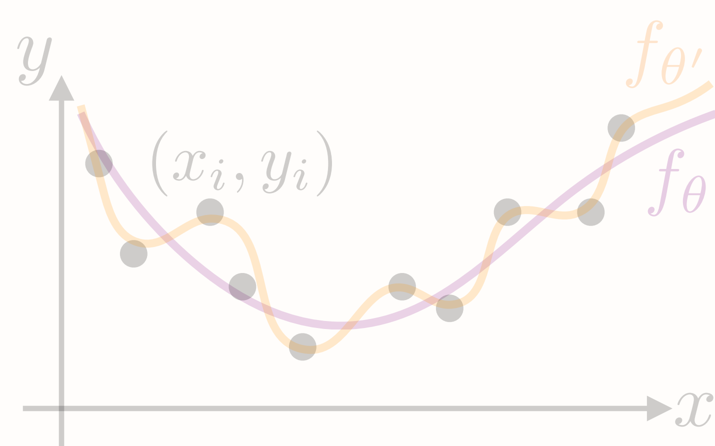
$$\ell(y, y') = \log(1 + e^{-yy'})$$



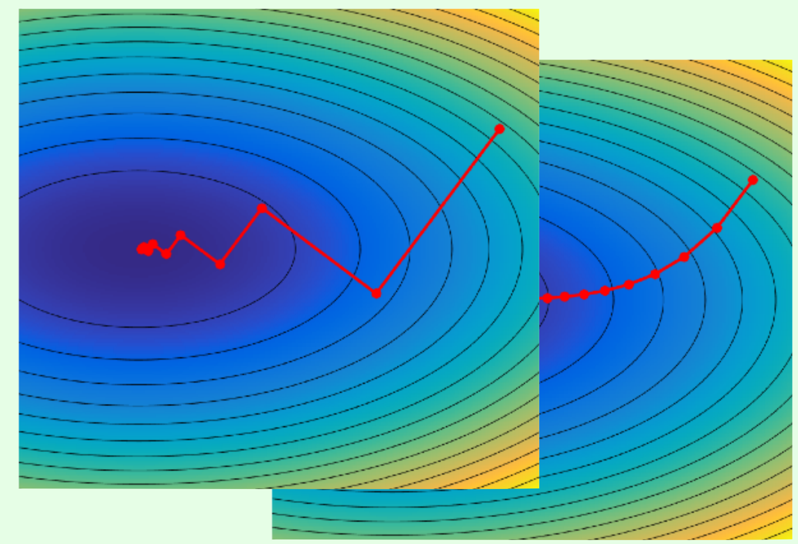
Overfitting, regularization, ...

Overview

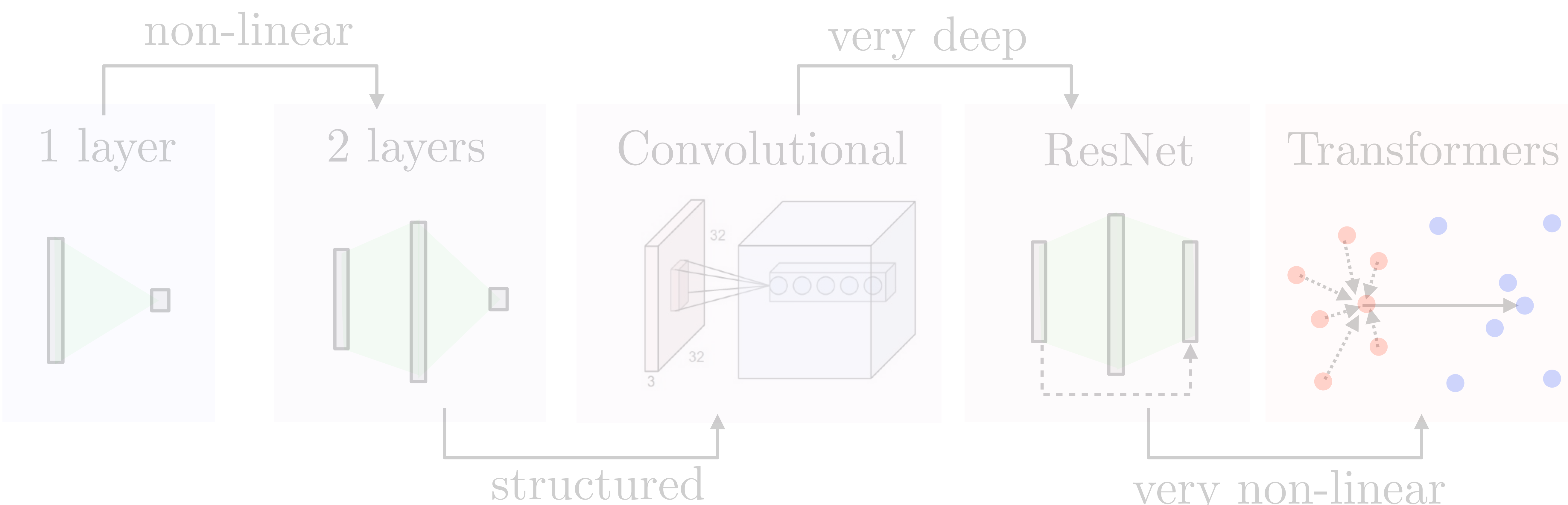
Machine Learning



Optimization



Networks Architectures

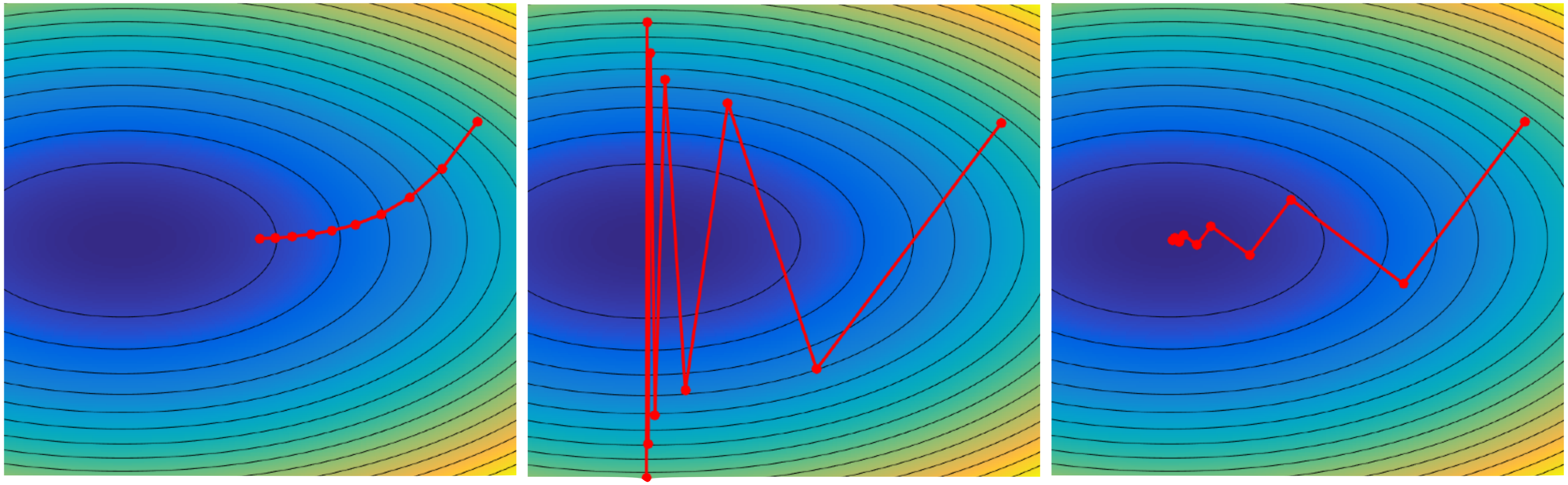


Gradient Descent

$$\min_\theta \mathcal{E}(\theta) \triangleq \frac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(x_i), y_i)$$

Gradient descent:

$$\theta_{\ell+1} = \theta_\ell - \tau_\ell \nabla \mathcal{E}(\theta_\ell)$$



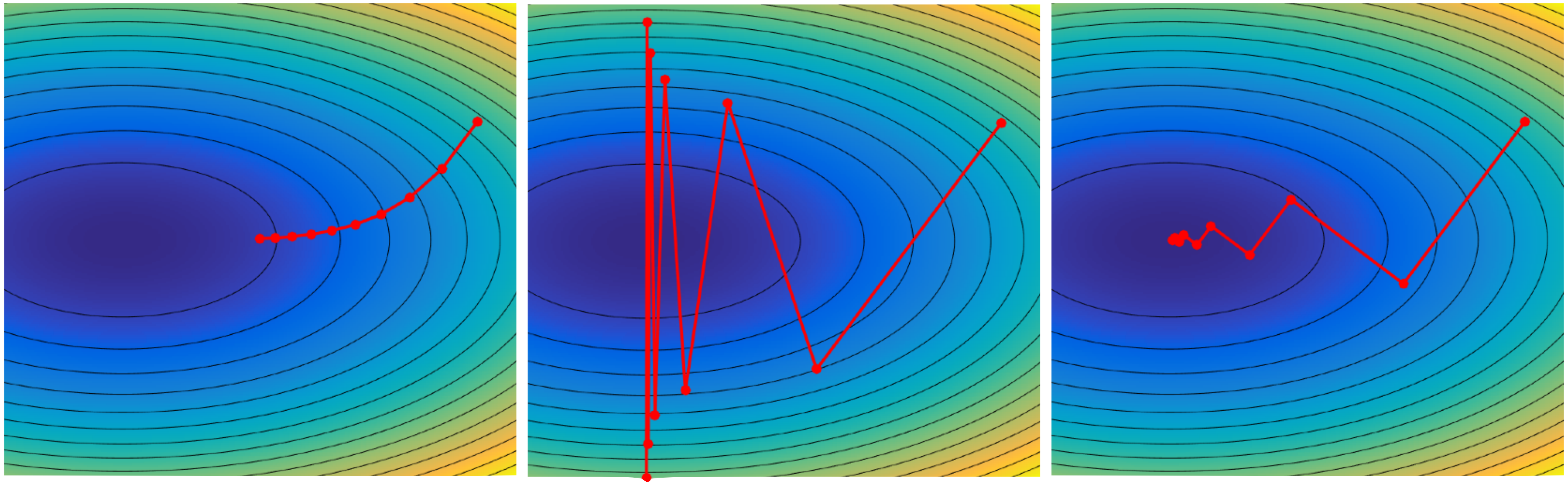
Small τ_ℓ Large τ_ℓ Optimal $\tau_\ell = \tau_\ell^\star$

Gradient Descent

$$\min_\theta \mathcal{E}(\theta) \triangleq \frac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(x_i), y_i)$$

Gradient descent:

$$\theta_{\ell+1} = \theta_\ell - \tau_\ell \nabla \mathcal{E}(\theta_\ell)$$



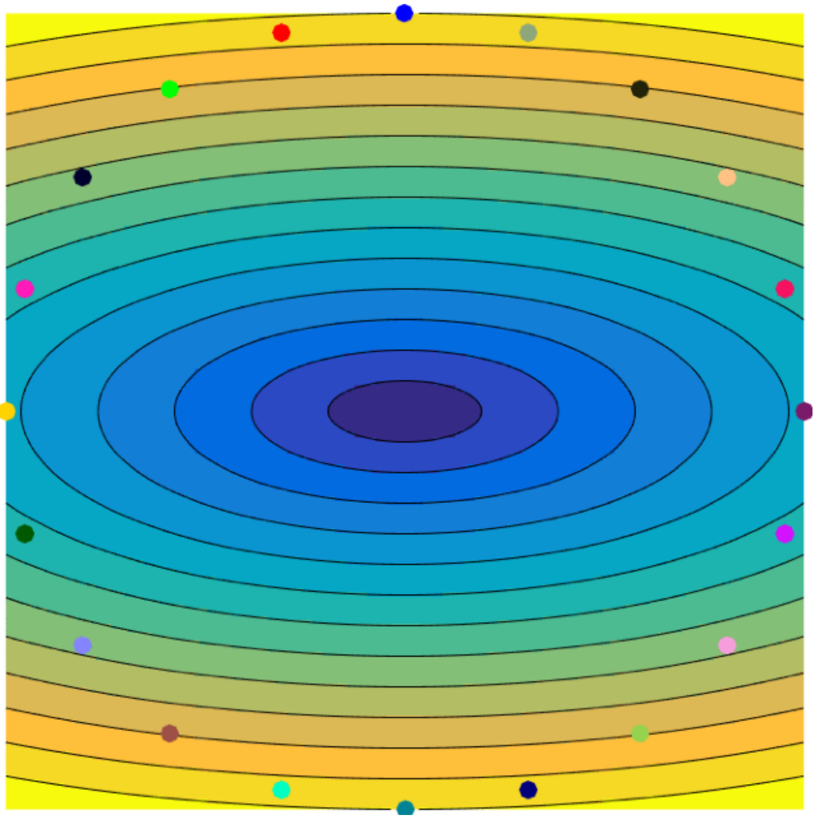
Small τ_ℓ Large τ_ℓ Optimal $\tau_\ell = \tau_\ell^\star$

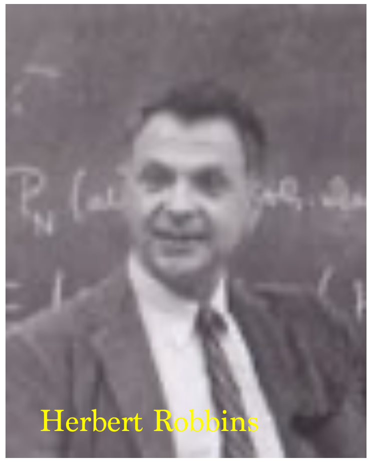
Stochastic gradient descent:

$$\theta_{\ell+1} = \theta_\ell - \frac{\tau}{\ell} \nabla \mathcal{E}_\ell(\theta_\ell)$$

$$i \leftarrow \texttt{rand}$$

$$\mathcal{E}_\ell(\theta) \triangleq \ell(f_\theta(x_i), y_i)$$





Sutton Monro

The Complexity of Gradient Computation

Setup: $\mathcal{E} : \mathbb{R}^d \to \mathbb{R}$ computable in K operations.

```
def ForwardNN(A,b,Z):
    X = []
    X.append(Z)
    for r in arange(0,R):
        X.append( rhoF( A[r].dot(X[r]) + tile(b[r],[1,Z.shape[1]]) ) )
    return X
```

Hypothesis: elementary operations $(a \times b, \log(a), \sqrt{a} \ldots)$ and their derivatives cost $O(1)$.

Question: What is the complexity of computing $\nabla\mathcal{E} : \mathbb{R}^d \to \mathbb{R}^d$?

The Complexity of Gradient Computation

Setup: $\mathcal{E} : \mathbb{R}^d \to \mathbb{R}$ computable in K operations.

```
def ForwardNN(A,b,Z):
    X = []
    X.append(Z)
    for r in arange(0,R):
        X.append( rhoF( A[r].dot(X[r]) + tile(b[r],[1,Z.shape[1]]) ) )
    return X
```

Hypothesis: elementary operations $(a \times b, \log(a), \sqrt{a} \ldots)$ and their derivatives cost $O(1)$.

Question: What is the complexity of computing $\nabla\mathcal{E} : \mathbb{R}^d \to \mathbb{R}^d$?

Finite differences:

$$\nabla\mathcal{E}(\theta) \approx \frac{1}{\varepsilon}(\mathcal{E}(\theta + \varepsilon\delta_1) - \mathcal{E}(\theta), \ldots \mathcal{E}(\theta + \varepsilon\delta_d) - \mathcal{E}(\theta))$$

$K(d+1)$ operations, intractable for large d.

The Complexity of Gradient Computation

Setup: $\mathcal{E} : \mathbb{R}^d \to \mathbb{R}$ computable in K operations.

```python
def ForwardNN(A,b,Z):
    X = []
    X.append(Z)
    for r in arange(0,R):
        X.append( rhoF( A[r].dot(X[r]) + tile(b[r],[1,Z.shape[1]]) ) )
    return X
```

Hypothesis: elementary operations $(a \times b, \log(a), \sqrt{a} \ldots)$ and their derivatives cost $O(1)$.

Question: What is the complexity of computing $\nabla \mathcal{E} : \mathbb{R}^d \to \mathbb{R}^d$?

Finite differences:

$$\nabla \mathcal{E}(\theta) \approx \frac{1}{\varepsilon}(\mathcal{E}(\theta + \varepsilon\delta_1) - \mathcal{E}(\theta), \ldots \mathcal{E}(\theta + \varepsilon\delta_d) - \mathcal{E}(\theta))$$

$K(d+1)$ operations, intractable for large d.

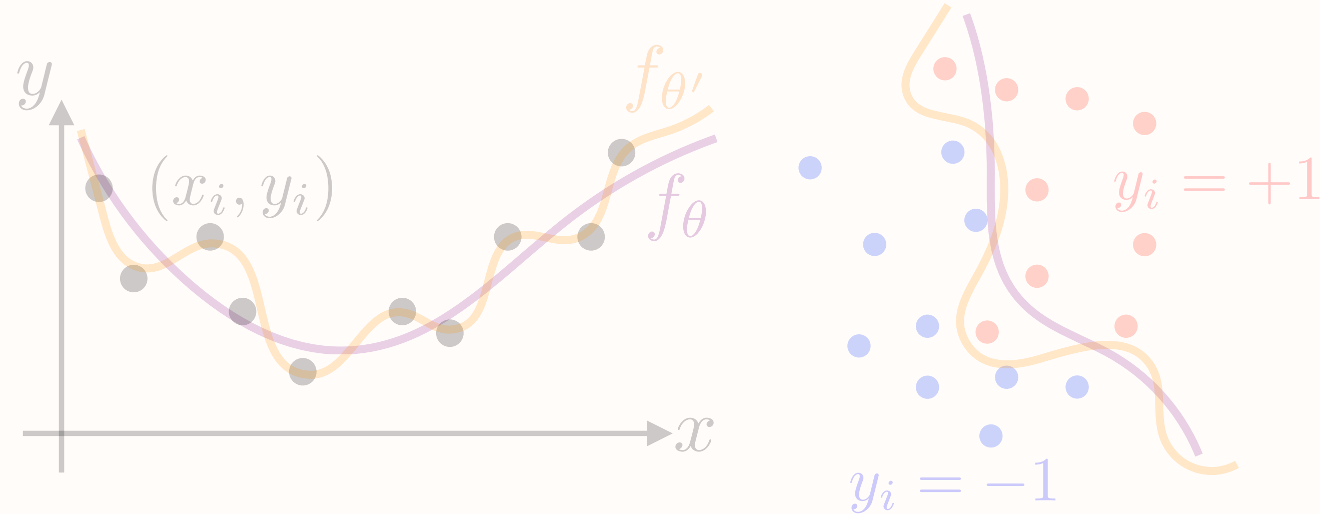
Theorem: there is an algorithm to compute $\nabla \mathcal{E}$ in $O(K)$ operations. [Seppo Linnainmaa, 1970]

This algorithm is reverse mode automatic differentiation

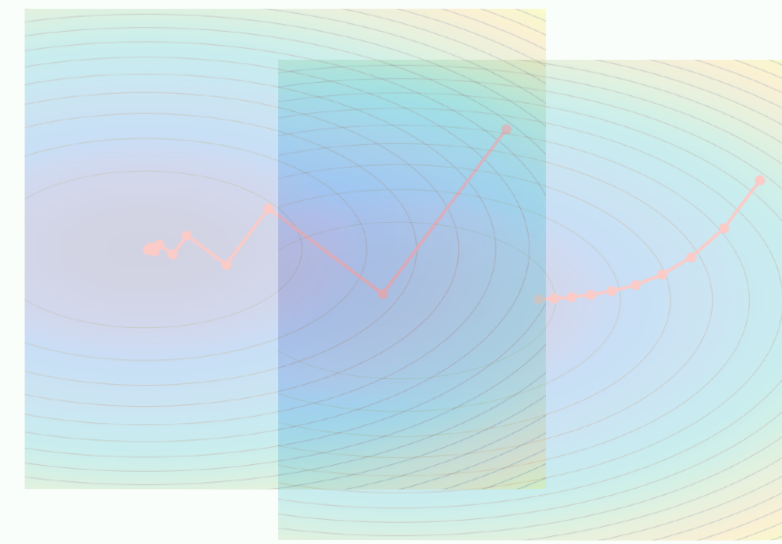
```python
def BackwardNN(A,b,X):
    gx = lossG(X[R],Y) # initialize the gradient
    for r in arange(R-1,-1,-1):
        M = rhoG( A[r].dot(X[r]) + tile(b[r],[1,n]) ) * gx
        gx = A[r].transpose().dot(M)
        gA[r] =  M.dot(X[r].transpose())
        gb[r] =  MakeCol(M.sum(axis=1))
    return [gA,gb]
```

Overview

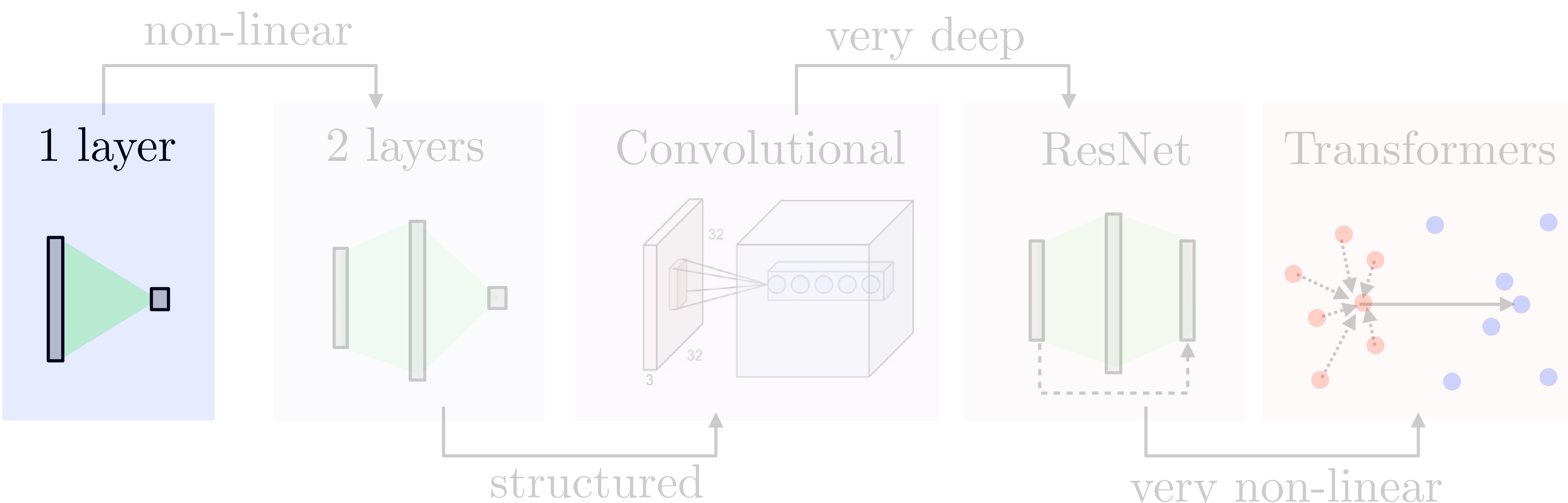
Machine Learning



Optimization

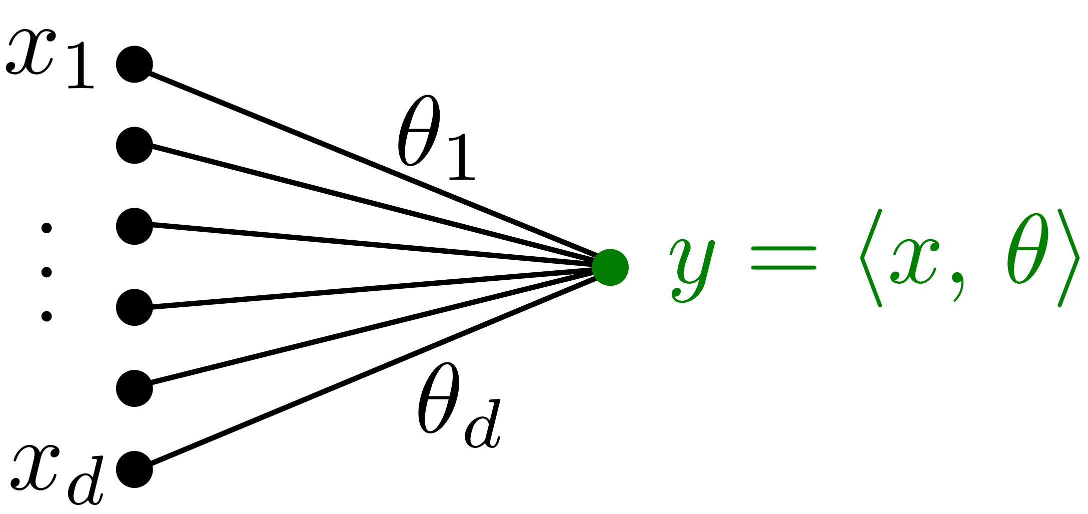


Networks Architectures

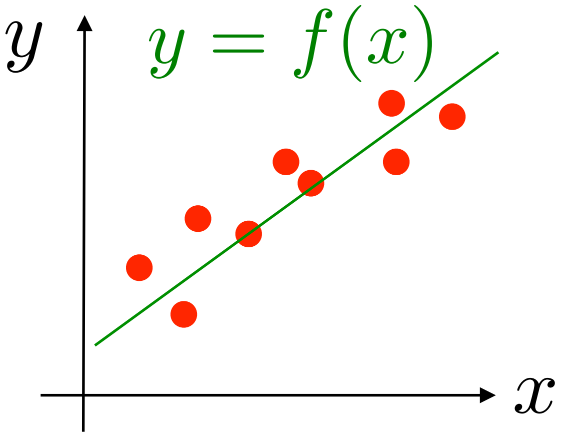


Linear model (1 layer)

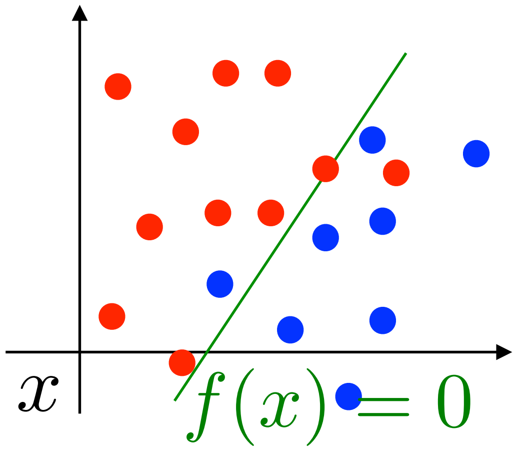
$$f_\theta(x) = \langle x,\, \theta \rangle = \sum_k x_k \theta_k$$



Regression

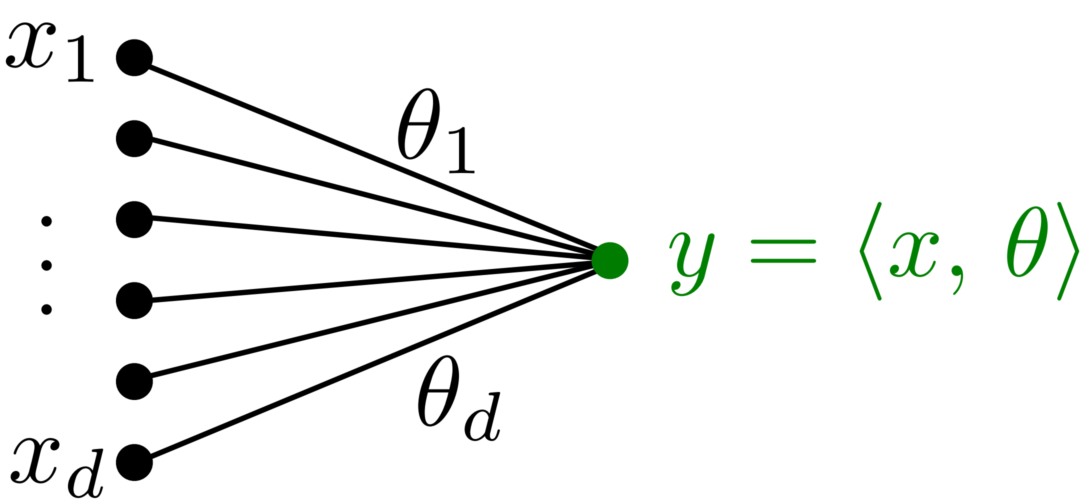


Classification:

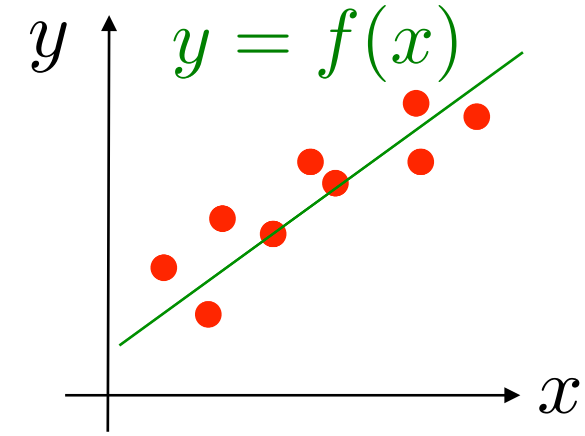


Linear model (1 layer)

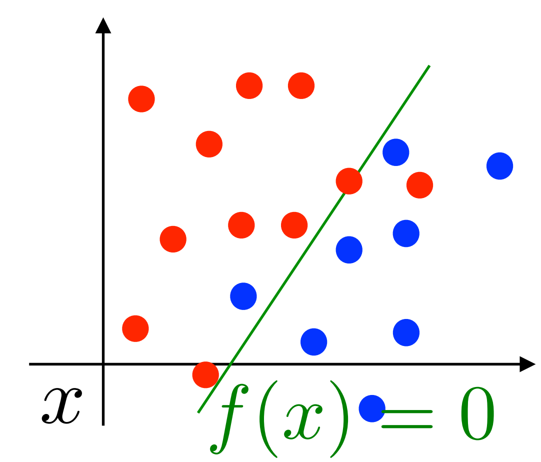
$$f_\theta(x) = \langle x,\, \theta\rangle = \textstyle\sum_k x_k \theta_k$$



Regression



Classification:

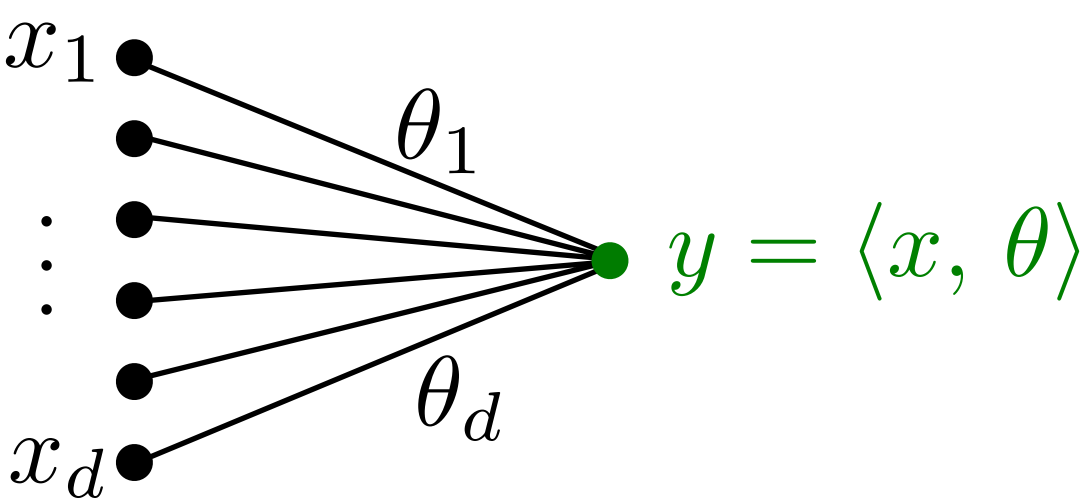


Convex optimization:

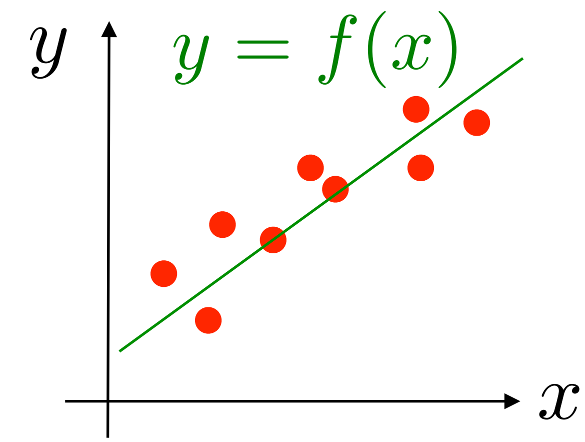
$$\min_\theta \triangleq \frac{1}{n} \sum_{i=1}^{n} \ell(\langle x_i,\, \theta_i\rangle, y_i)$$

Linear model (1 layer)

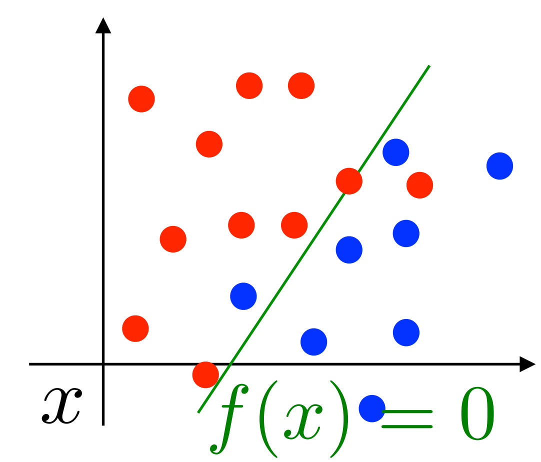
$$f_\theta(x) = \langle x,\, \theta \rangle = \sum_k x_k \theta_k$$



Regression



Classification:



Convex optimization:

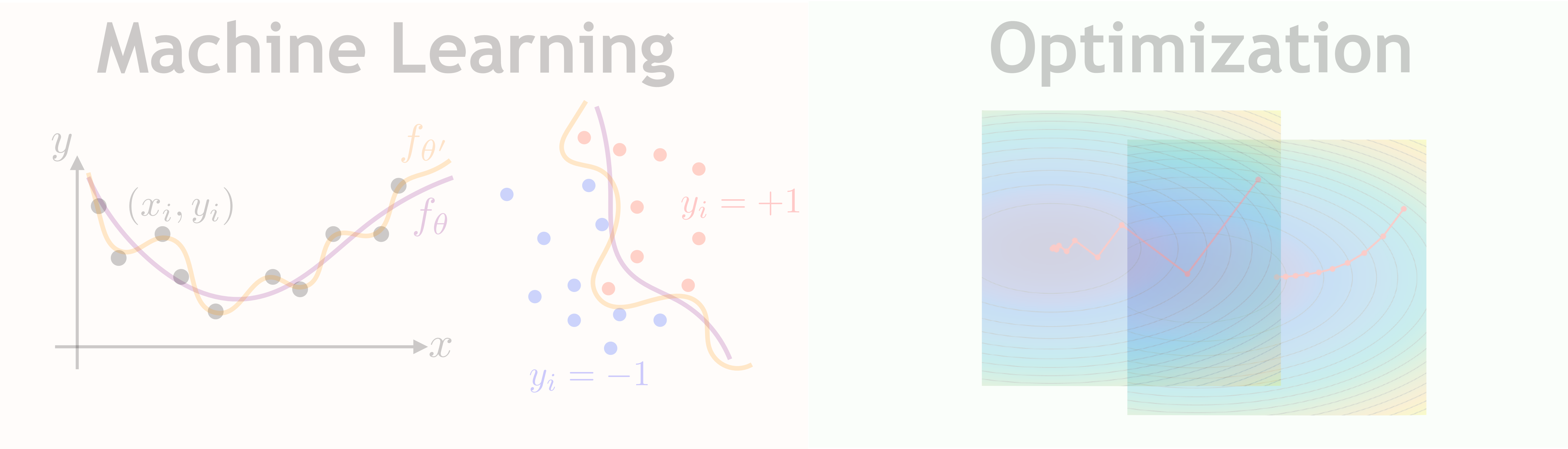
$$\min_\theta \triangleq \frac{1}{n} \sum_{i=1}^{n} \ell(\langle x_i,\, \theta_i \rangle, y_i)$$

Kernel methods: replace x by $\varphi(x) \in \mathbb{R}^D$

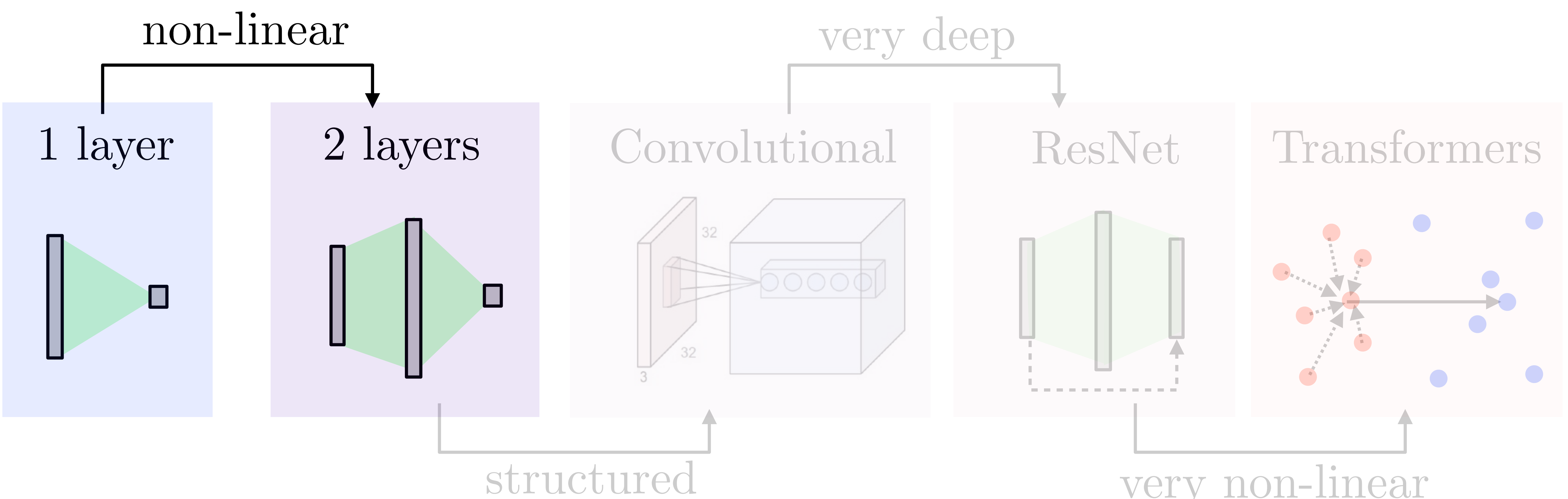
$$(D \gg d, \text{ even } D = \infty!)$$

Deep learning methods: learn $\varphi(x)$!

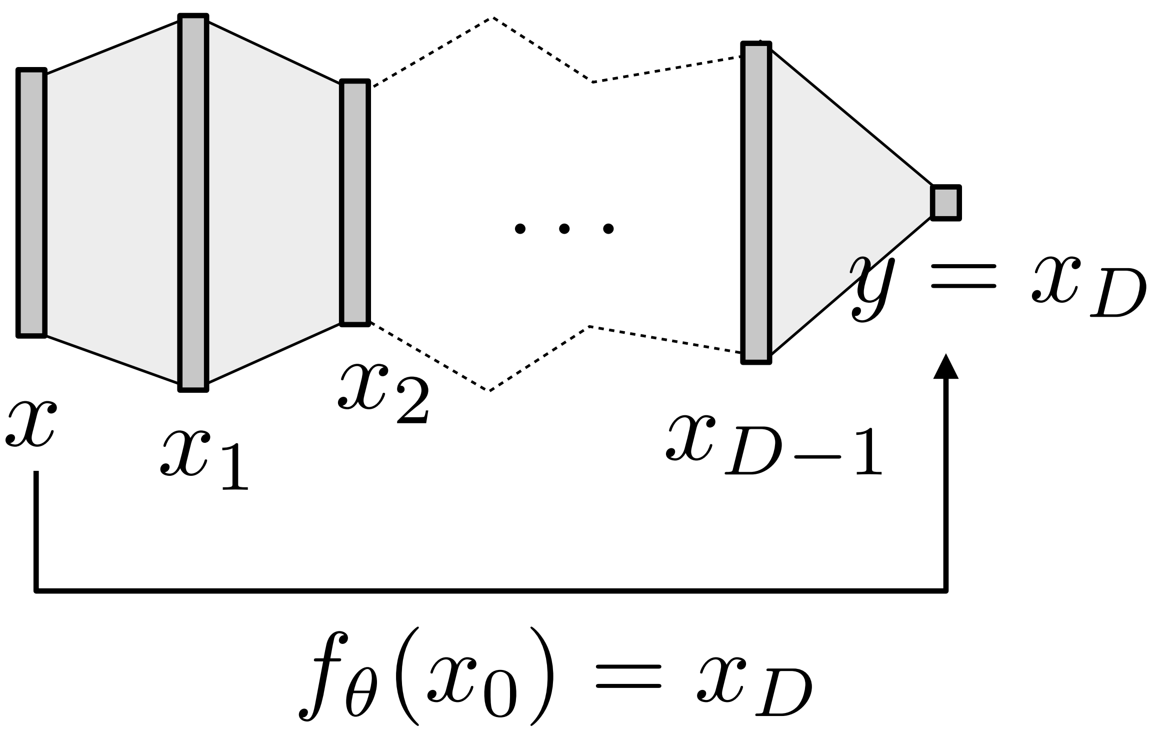
Overview



Networks Architectures



Multi-layer Perceptron



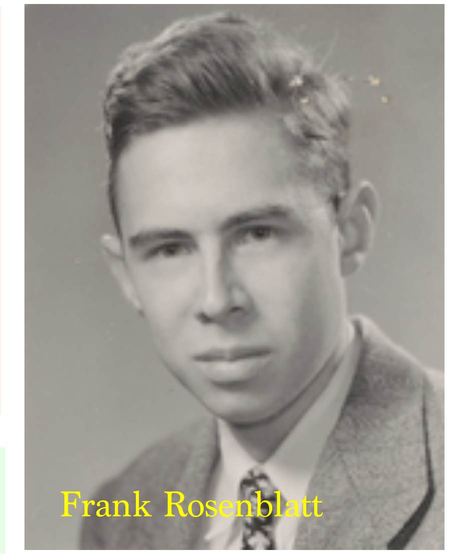
$$x_0 \leftarrow x$$

$$x_{k+1} \triangleq \sigma(W_k x_k + b_k)$$

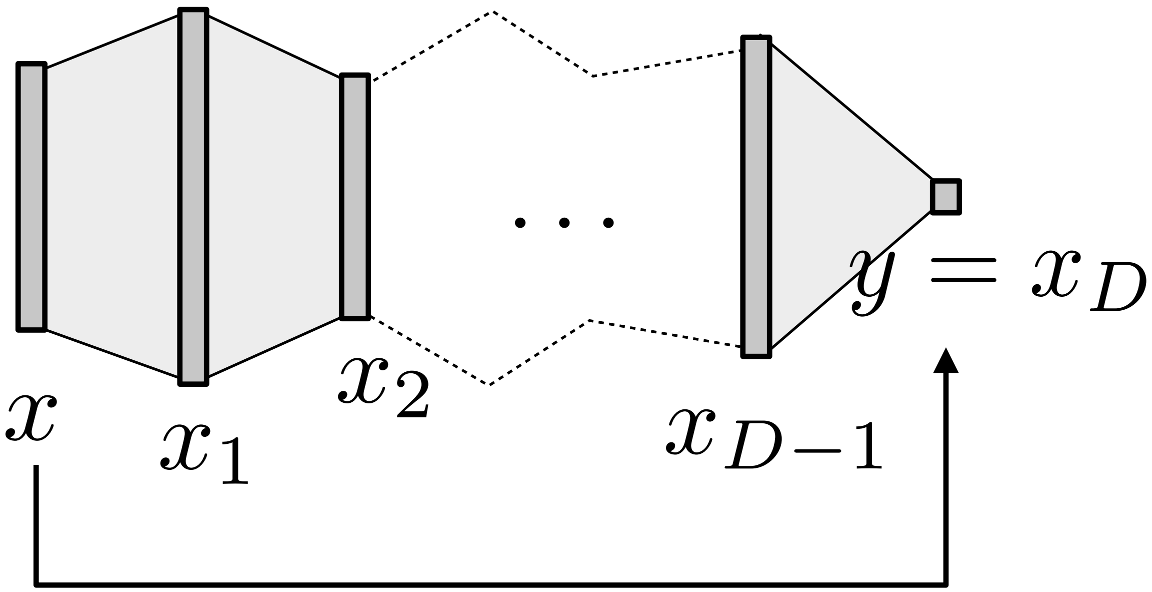
$$\theta = \{(W_k, b_k)\}_{k=0}^{D-1}$$

$$W_k \in \mathbb{R}^{d_{k+1} \times d_k}$$

$$b_k \in \mathbb{R}^{d_{k+1}}$$



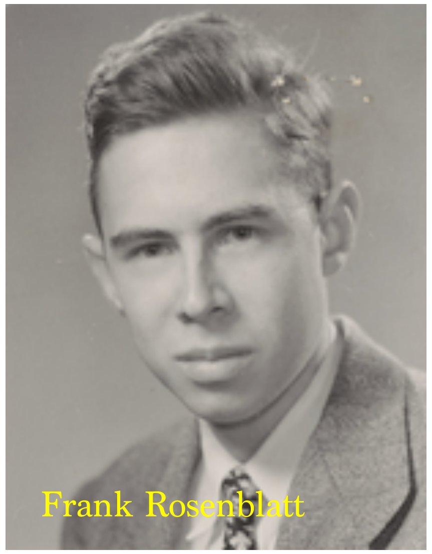
Multi-layer Perceptron



$$f_\theta(x_0) = x_D$$

$$x_0 \leftarrow x$$

$$x_{k+1} \triangleq \sigma(W_k x_k + b_k)$$

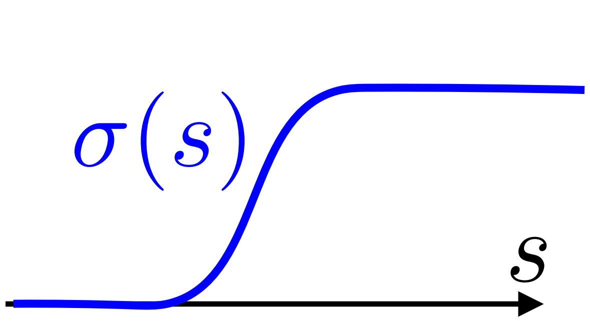


$$\theta = \{(W_k, b_k)\}_{k=0}^{D-1}$$

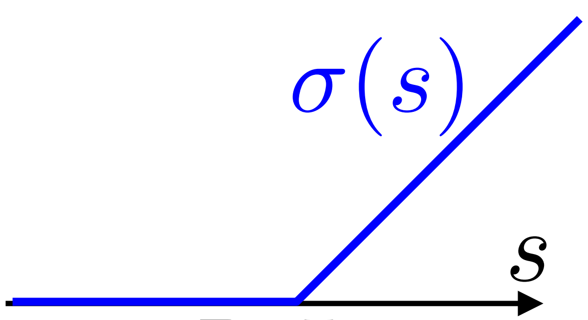
$$W_k \in \mathbb{R}^{d_{k+1} \times d_k}$$

$$b_k \in \mathbb{R}^{d_{k+1}}$$

Non-linearity: σ must be non-polynomial to increase expressivity.

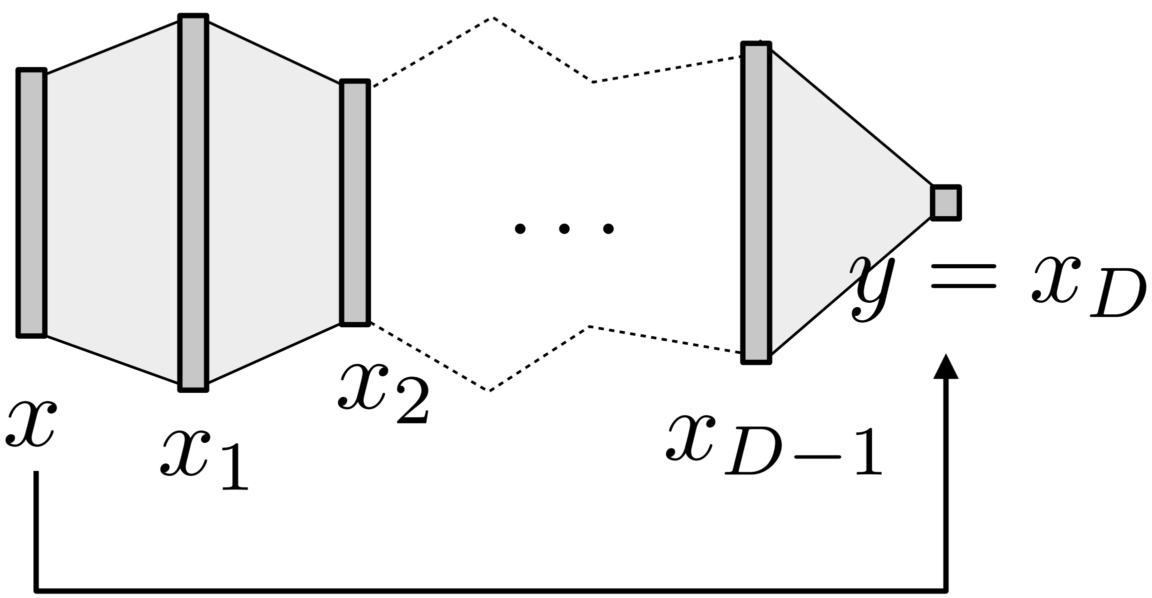


Sigmoid



ReLu

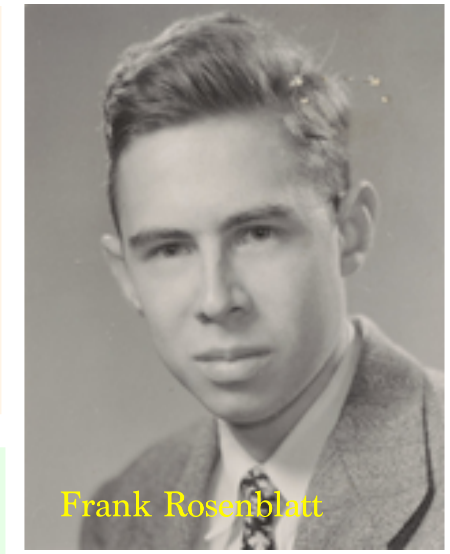
Multi-layer Perceptron



$$f_\theta(x_0) = x_D$$

$$x_0 \leftarrow x$$

$$x_{k+1} \triangleq \sigma(W_k x_k + b_k)$$

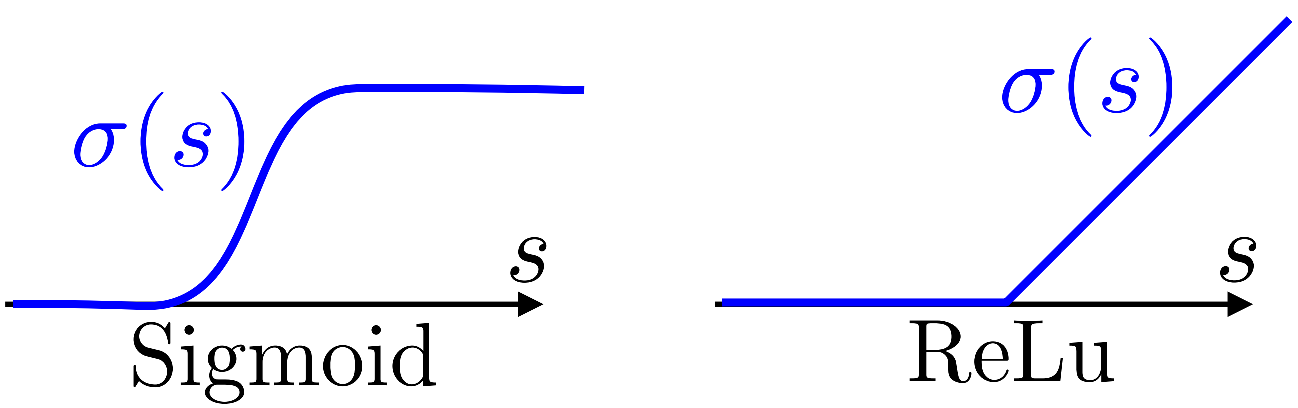


$$\theta = \{(W_k, b_k)\}_{k=0}^{D-1}$$

$$W_k \in \mathbb{R}^{d_{k+1} \times d_k}$$

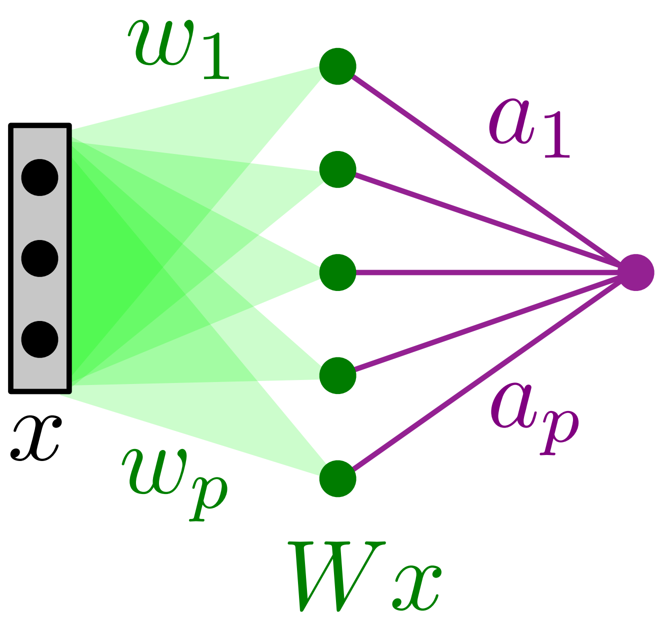
$$b_k \in \mathbb{R}^{d_{k+1}}$$

Non-linearity: σ must be non-polynomial to increase expressivity.



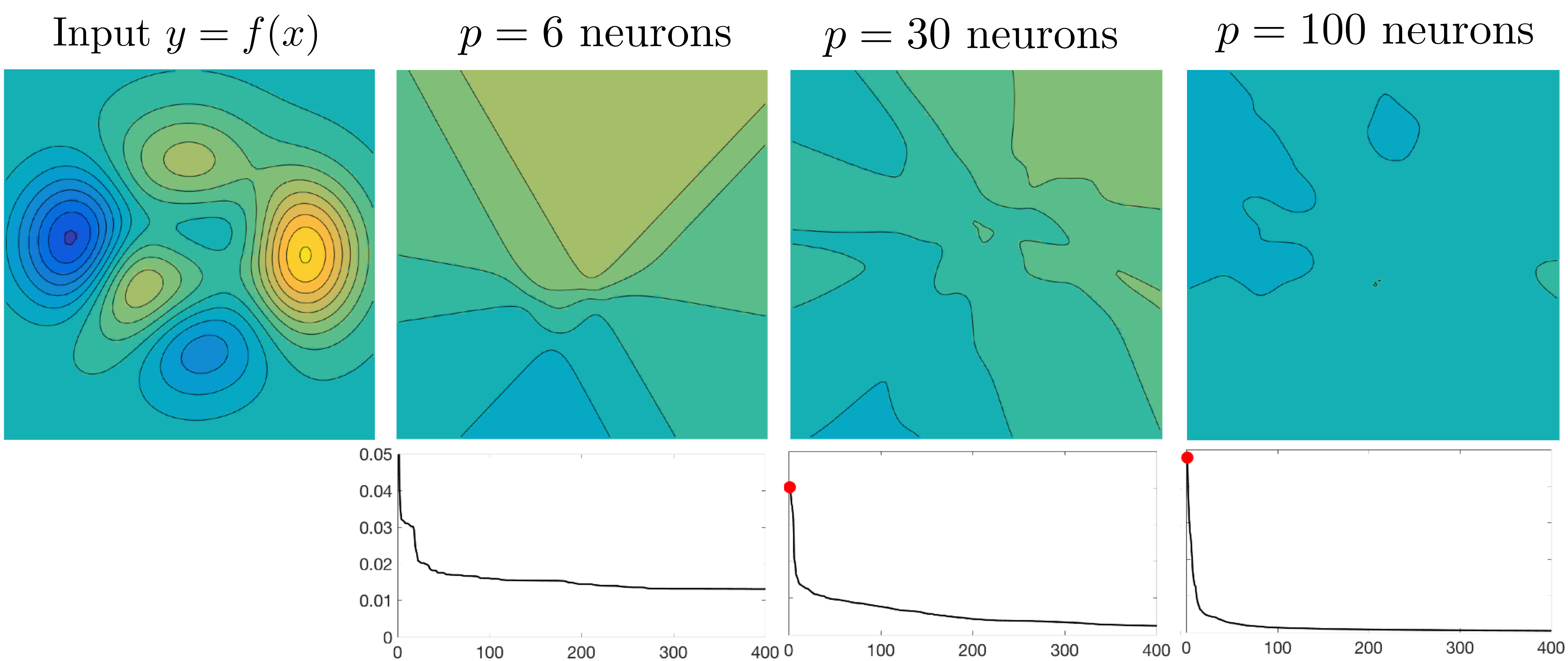
Weight matrix: needs extra constraints (e.g. convolution & sub-sampling)

Two Layers Perceptron

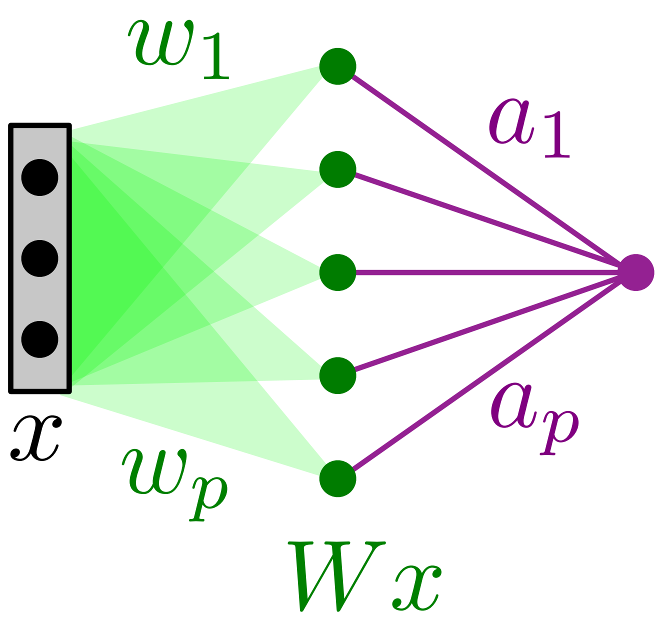


$$f_\theta(x) \triangleq \sum_{s=1}^{p} a_s \sigma(\langle x,\, w_s \rangle + b_s)$$

$\rightarrow$ sum of "ridge" functions $\sigma(\langle x,\, w \rangle + b)$



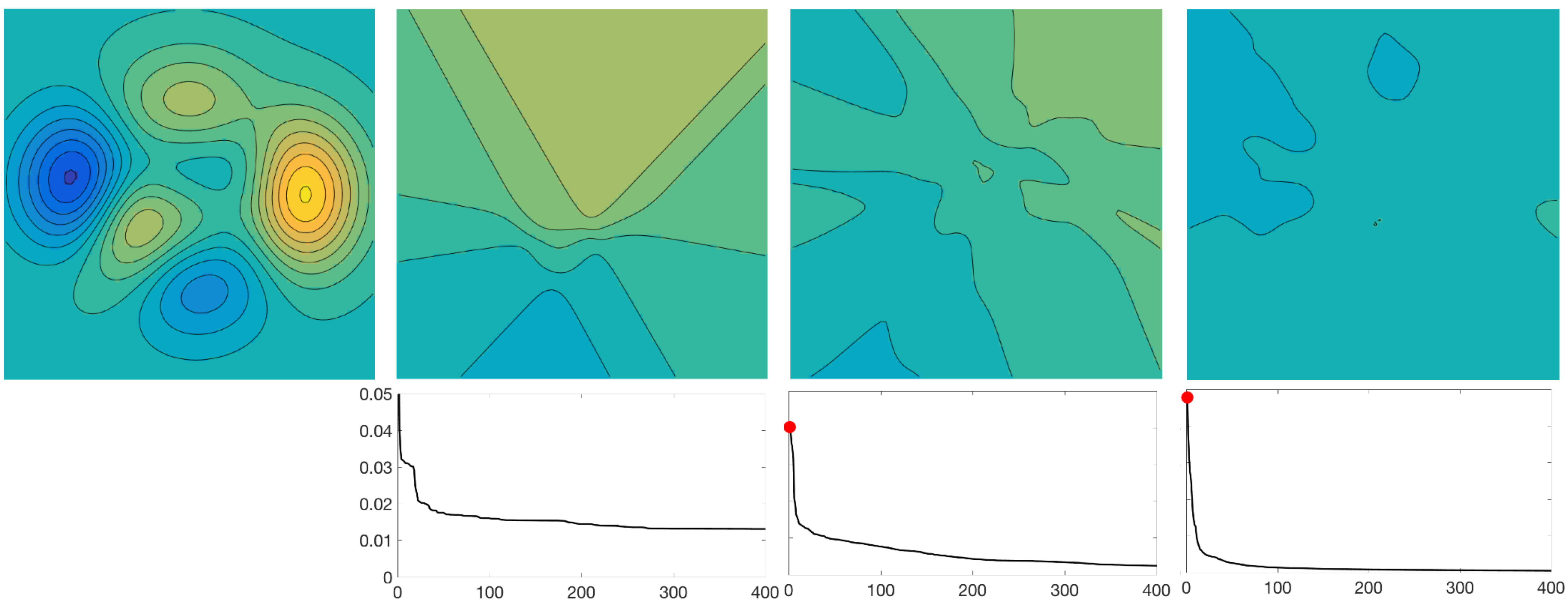
Two Layers Perceptron



$$f_\theta(x) \triangleq \sum_{s=1}^{p} a_s \sigma(\langle x,\, w_s \rangle + b_s)$$

$\rightarrow$ sum of "ridge" functions $\sigma(\langle x,\, w\rangle + b)$

Input $y = f(x)$ | $p = 6$ neurons | $p = 30$ neurons | $p = 100$ neurons



Universality of Perceptrons

Theorem: If f is continuous on a compact Ω, for all $\varepsilon > 0$ for p large enough, there exists θ such that

$$\forall\, x \in \Omega,\ |f_\theta(x) - f(x)| \leqslant \varepsilon$$

$\rightarrow$ non quantitative ... no free lunch.

Universality of Perceptrons

Theorem: If f is continuous on a compact Ω, for all $\varepsilon > 0$ for p large enough, there exists θ such that

$$\forall\, x \in \Omega,\ |f_\theta(x) - f(x)| \leqslant \varepsilon$$

$\rightarrow$ non quantitative … no free lunch.

Barron's functions: $\|f\|_B \triangleq \int_{\mathbb{R}^d} \|\omega\| |\hat{f}(\omega)| \mathrm{d}\omega < +\infty$

Theorem: for p large, there exists θ such that

$$\|f - f_\theta\|_{L^2(\Omega)} \leqslant \frac{2\mathrm{diam}(\Omega)\|f\|_B}{\sqrt{p}}$$

Universality of Perceptrons

Theorem: If f is continuous on a compact Ω, for all $\varepsilon > 0$ for p large enough, there exists θ such that

$$\forall\, x \in \Omega,\ |f_\theta(x) - f(x)| \leqslant \varepsilon$$

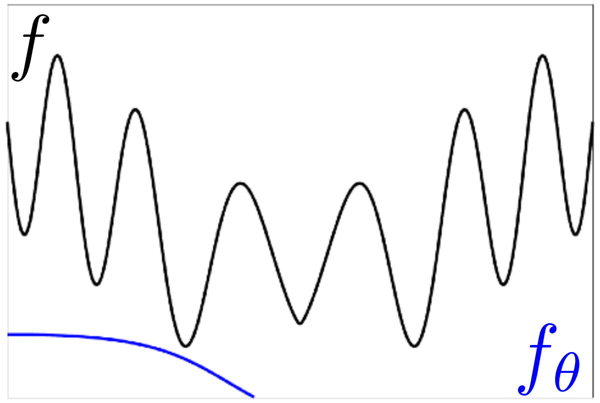
$\rightarrow$ non quantitative ... no free lunch.

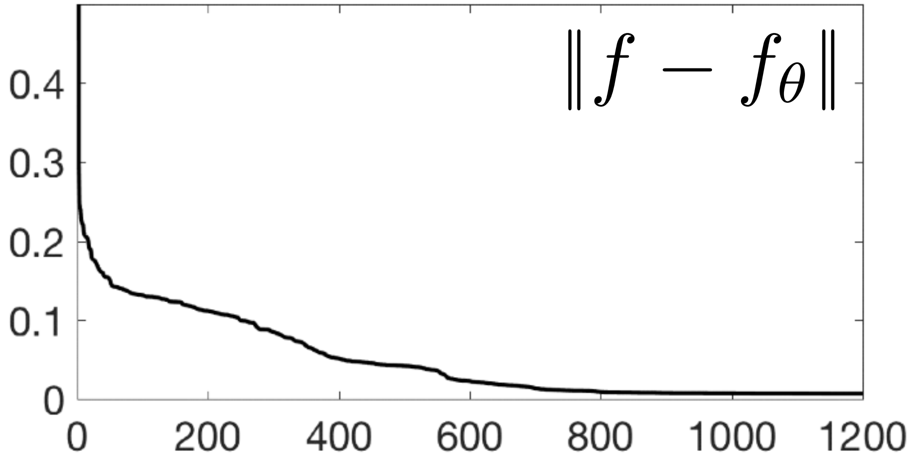
Barron's functions: $\|f\|_B \triangleq \int_{\mathbb{R}^d} \|\omega\| |\hat{f}(\omega)| \mathrm{d}\omega < +\infty$

Theorem: for p large, there exists θ such that

$$\|f - f_\theta\|_{L^2(\Omega)} \leqslant \frac{2\mathrm{diam}(\Omega)\|f\|_B}{\sqrt{p}}$$

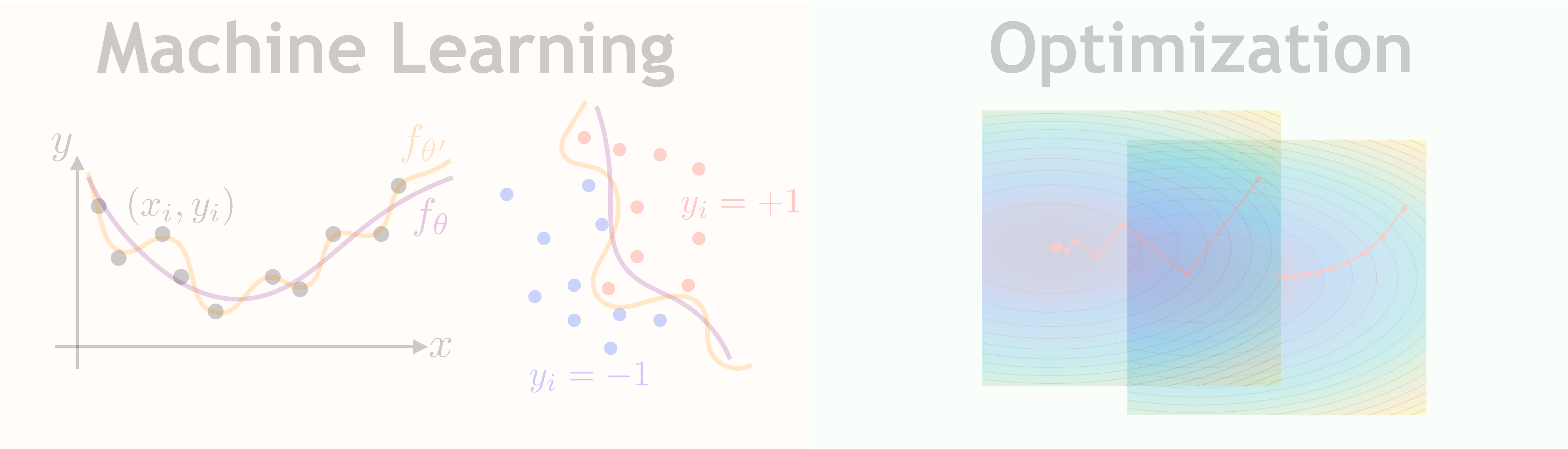
$\rightarrow$ non-constructive.



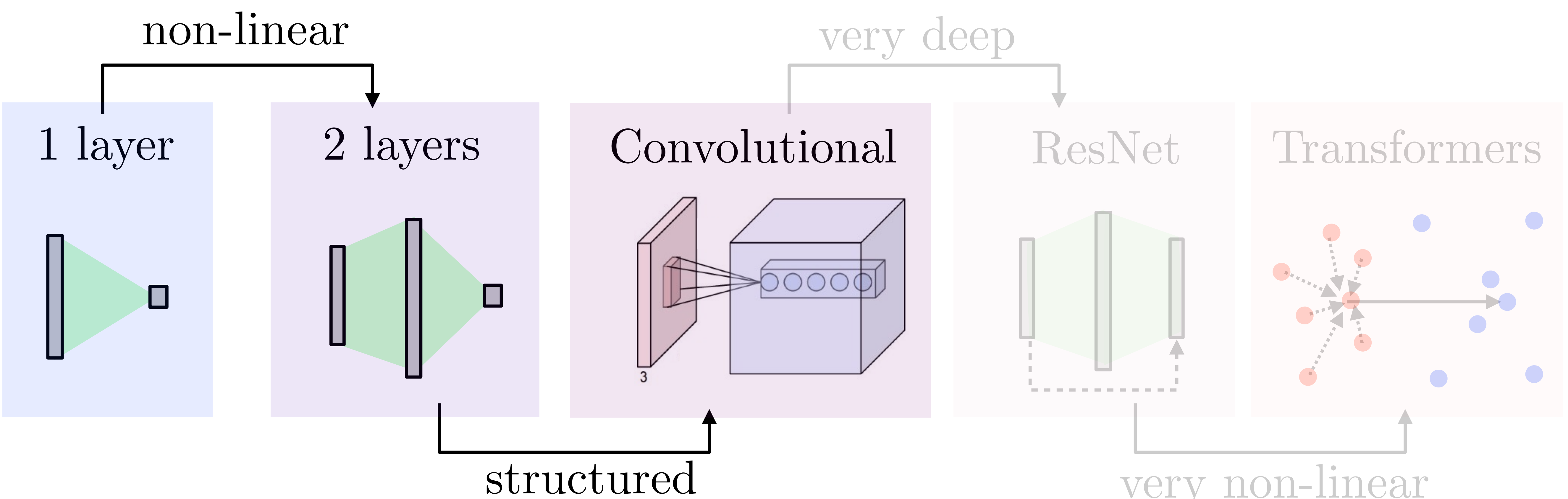


$\rightarrow$ for p "large enough" gradient descent works [Chizat-Bach 2018]

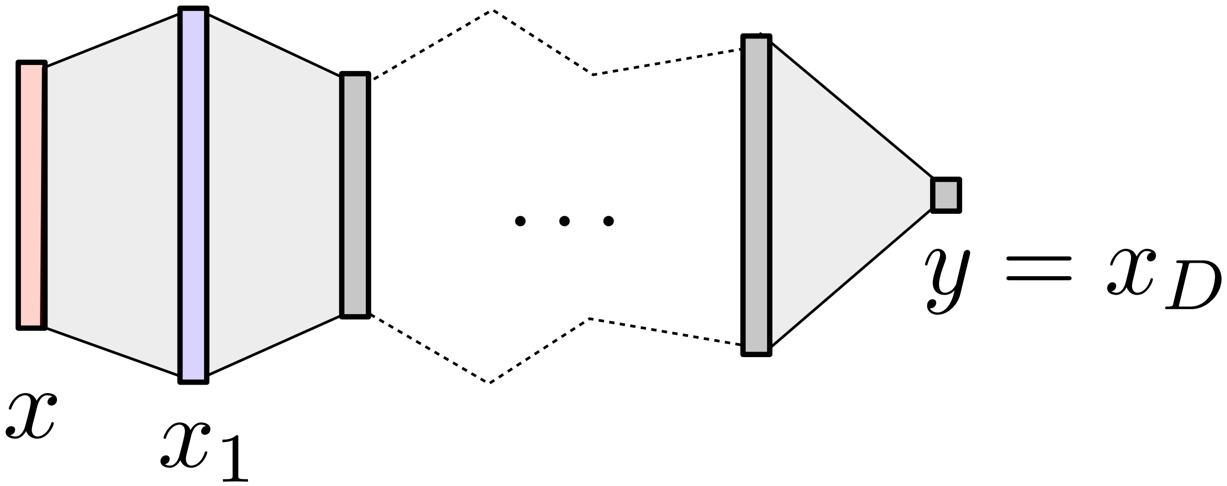
Overview



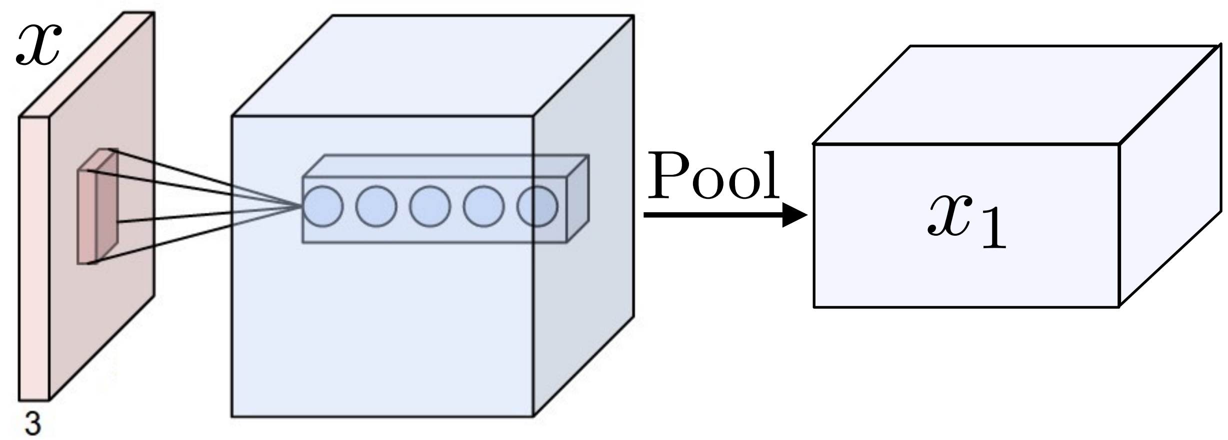
Networks Architectures



Convolutional CNN



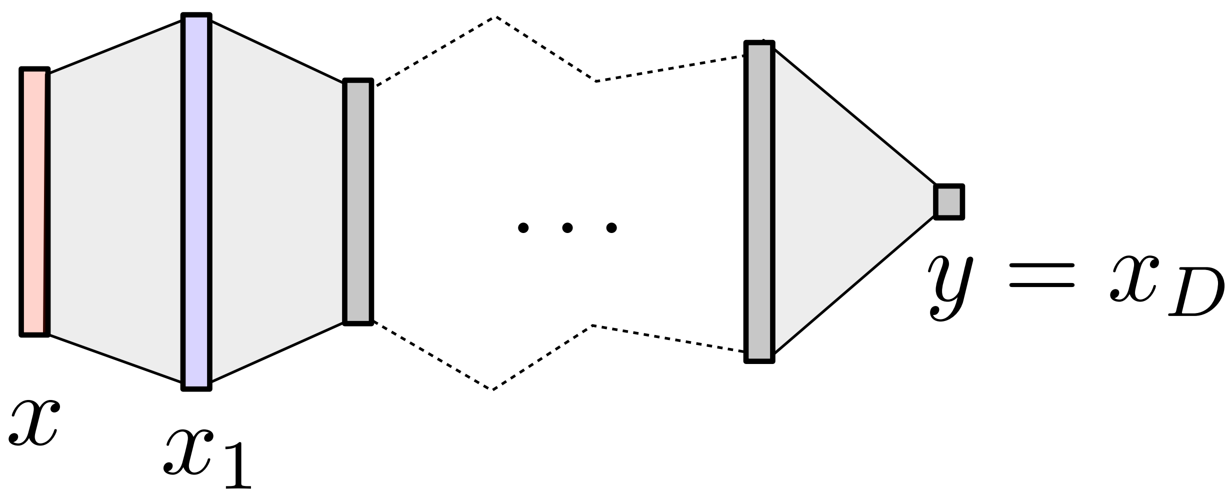
$$x_{k+1} \triangleq \sigma(W_k x_k + b_k)$$



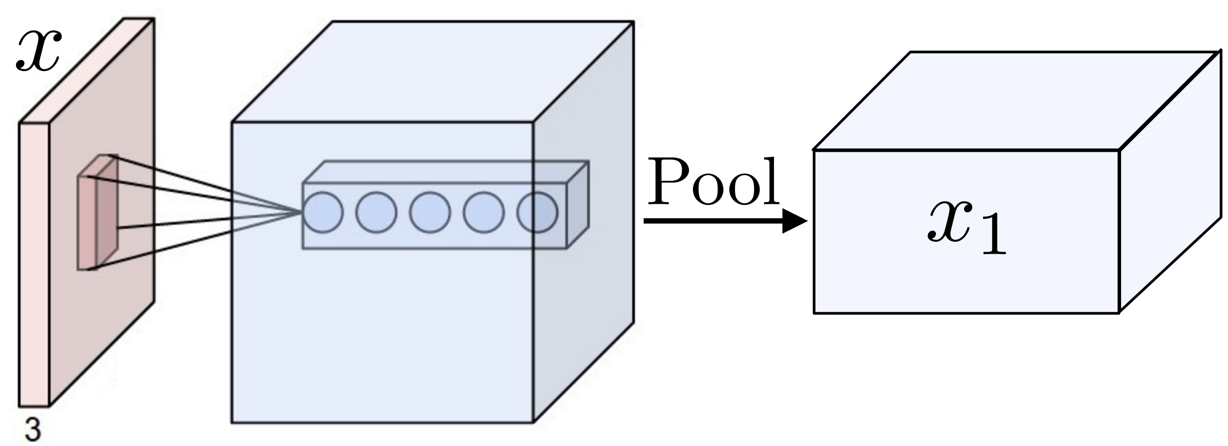
$\rightarrow$ Leverage translation invariance of images.

$\rightarrow$ Sub-sampling: breaks invariance but increase receptive fields.

Convolutional CNN

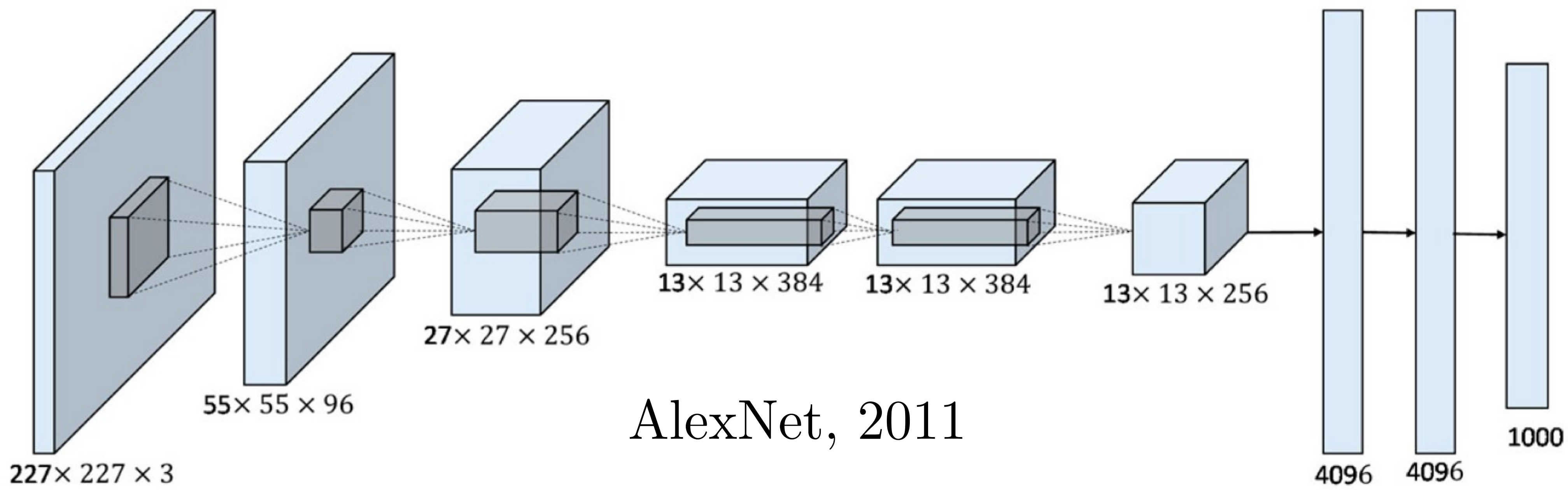


$$x_{k+1} \triangleq \sigma(W_k x_k + b_k)$$



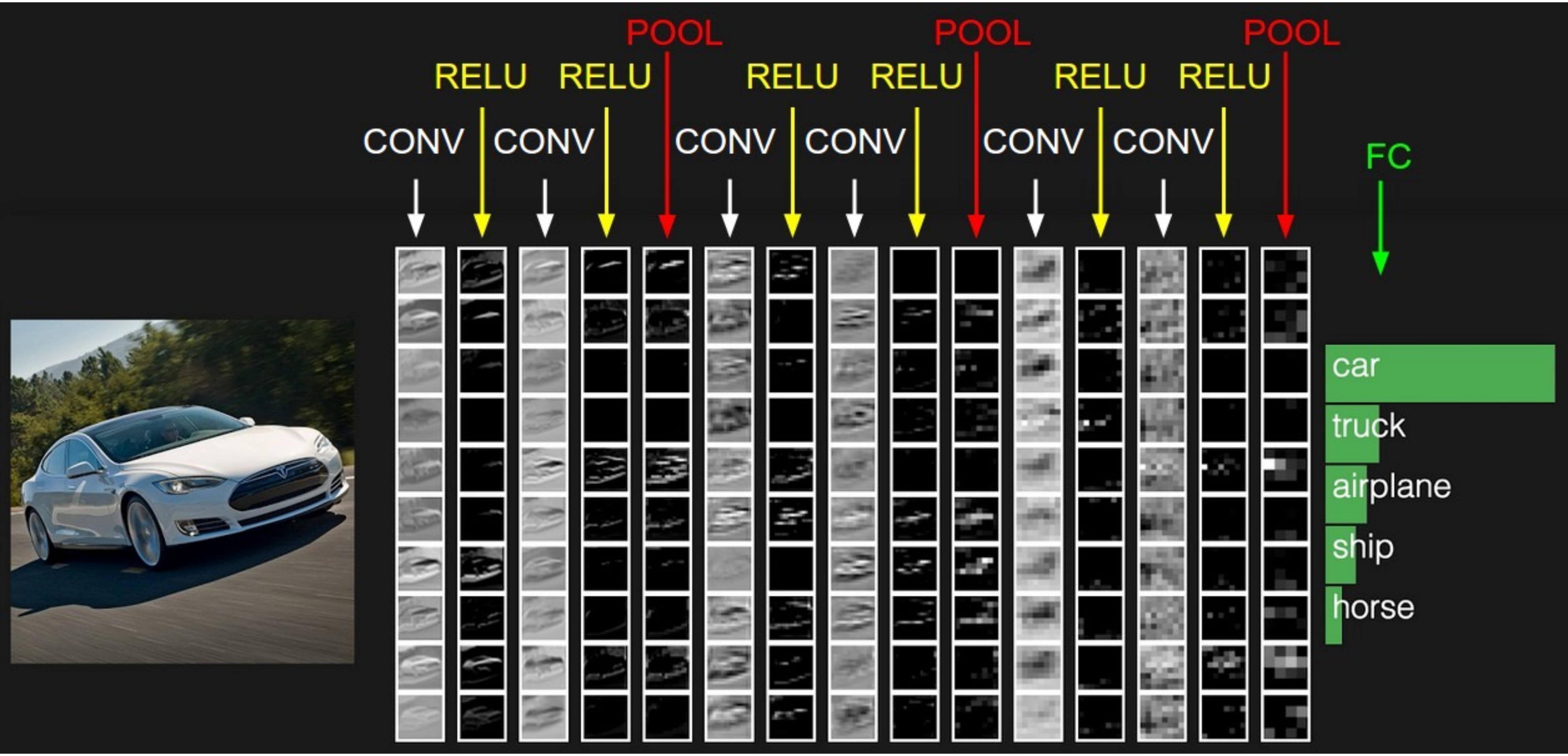
→ Leverage translation invariance of images.

→ Sub-sampling: breaks invariance but increase receptive fields.



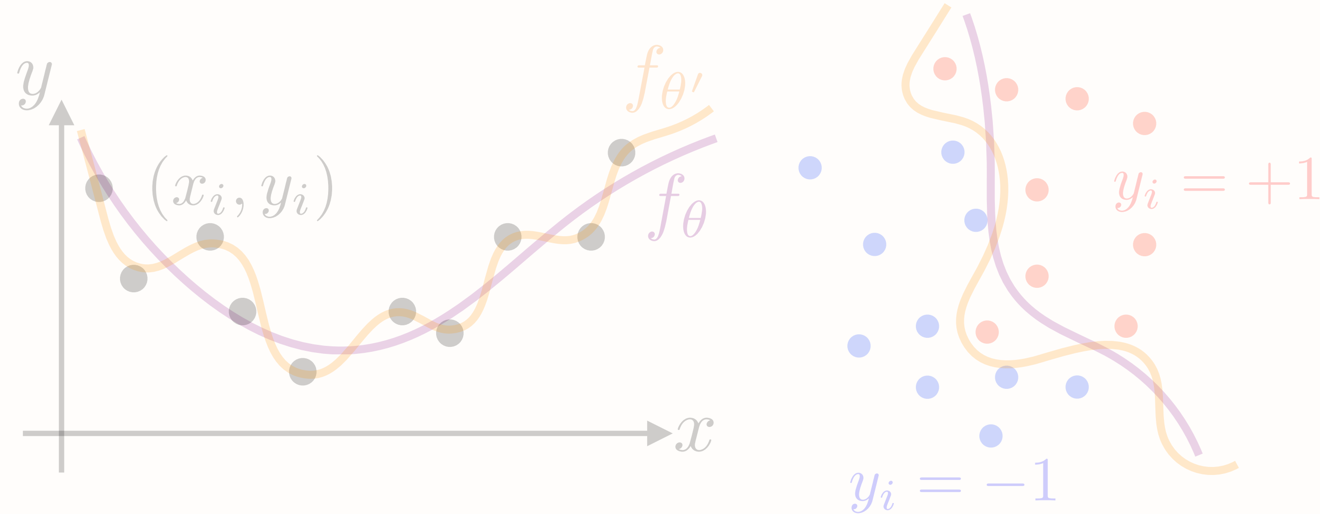
AlexNet, 2011

Example of Activations

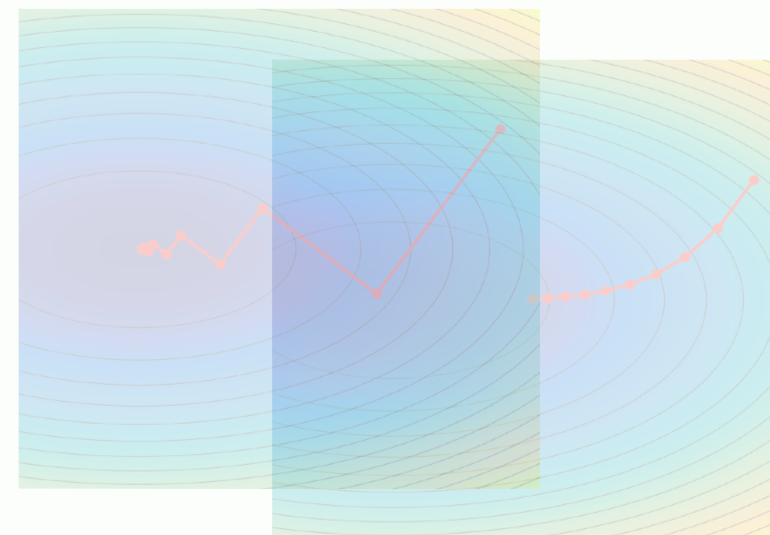


Overview

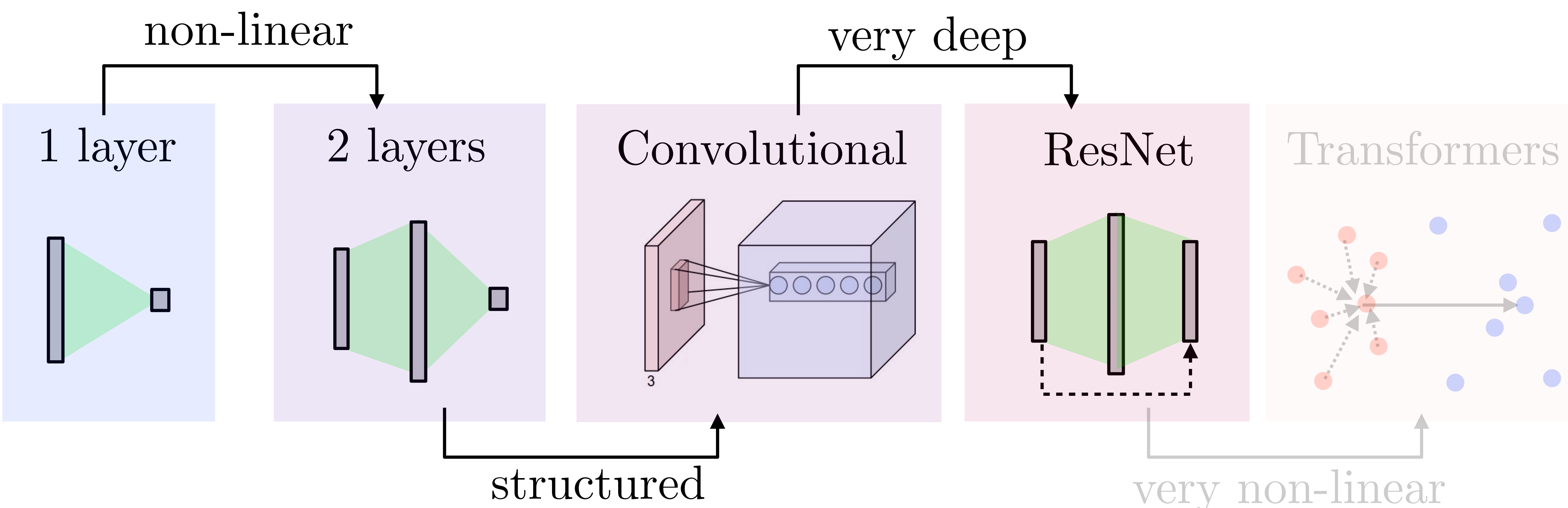
Machine Learning



Optimization

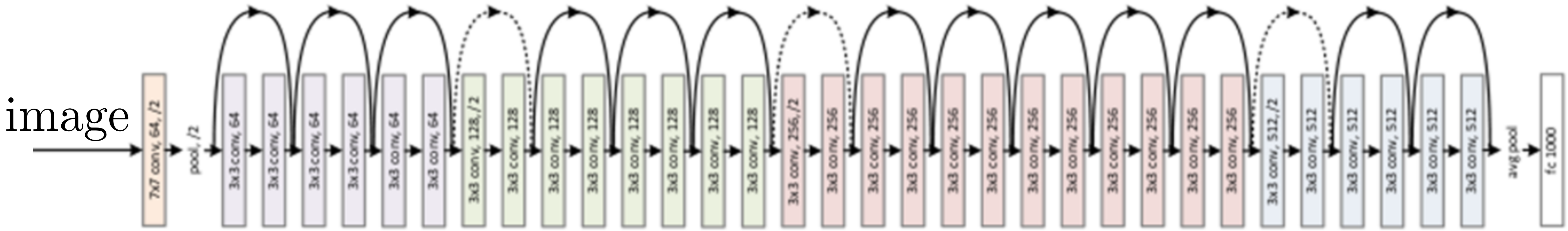


Networks Architectures

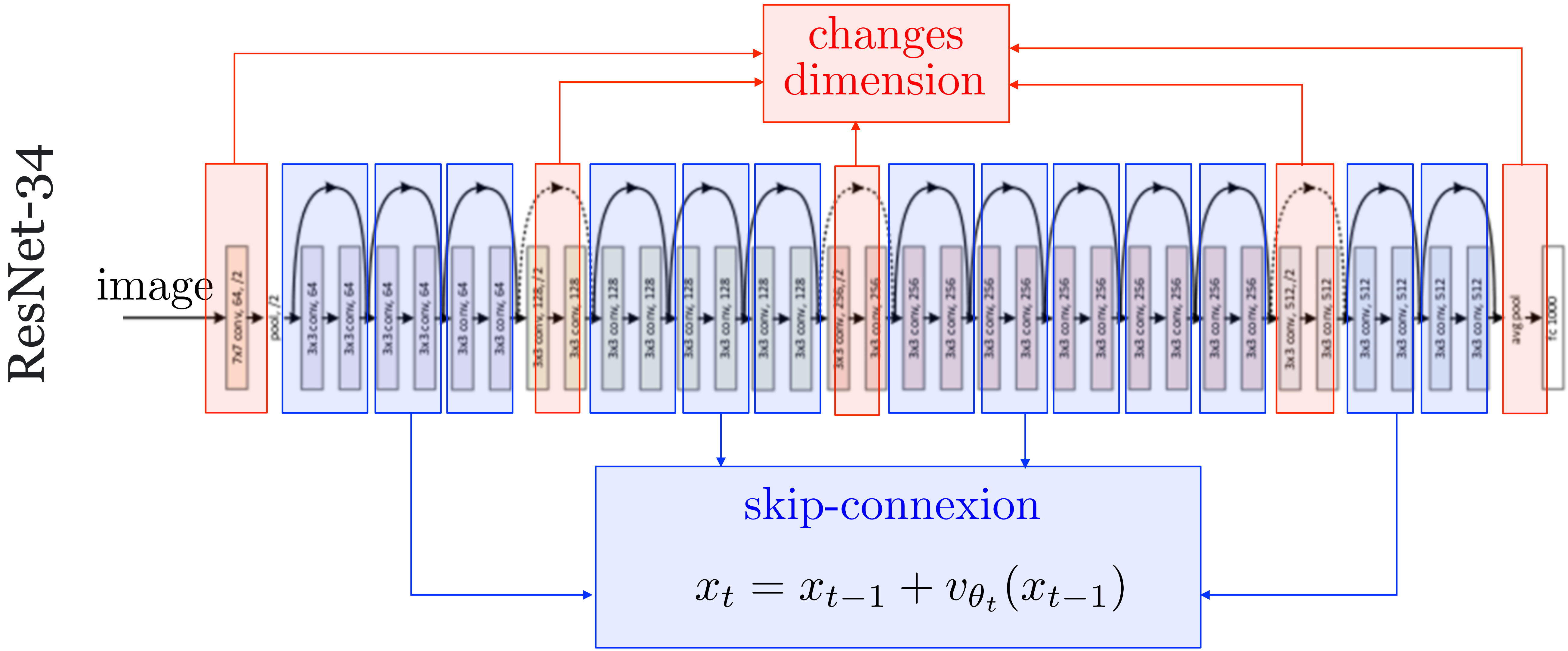


ResNet-type Architectures [He et al’ 16]

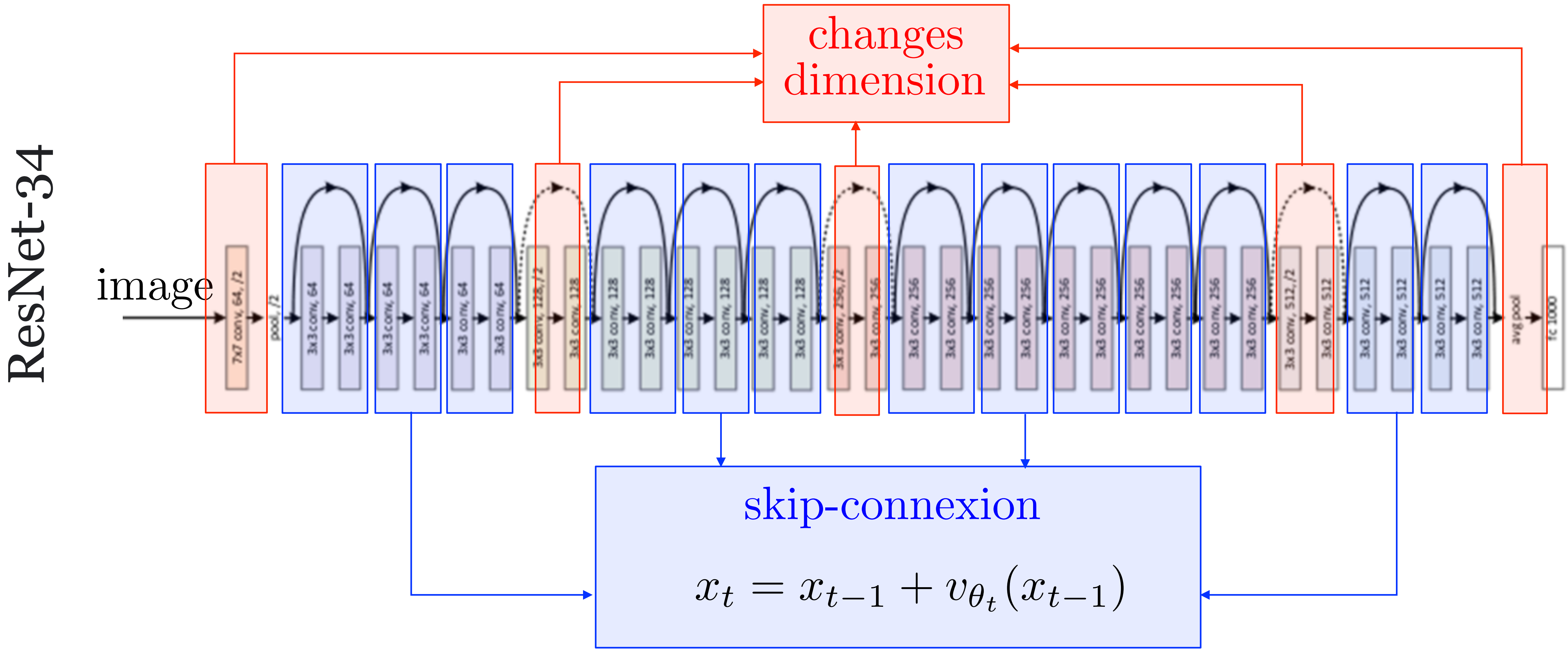
ResNet-34



ResNet-type Architectures [He et al' 16]

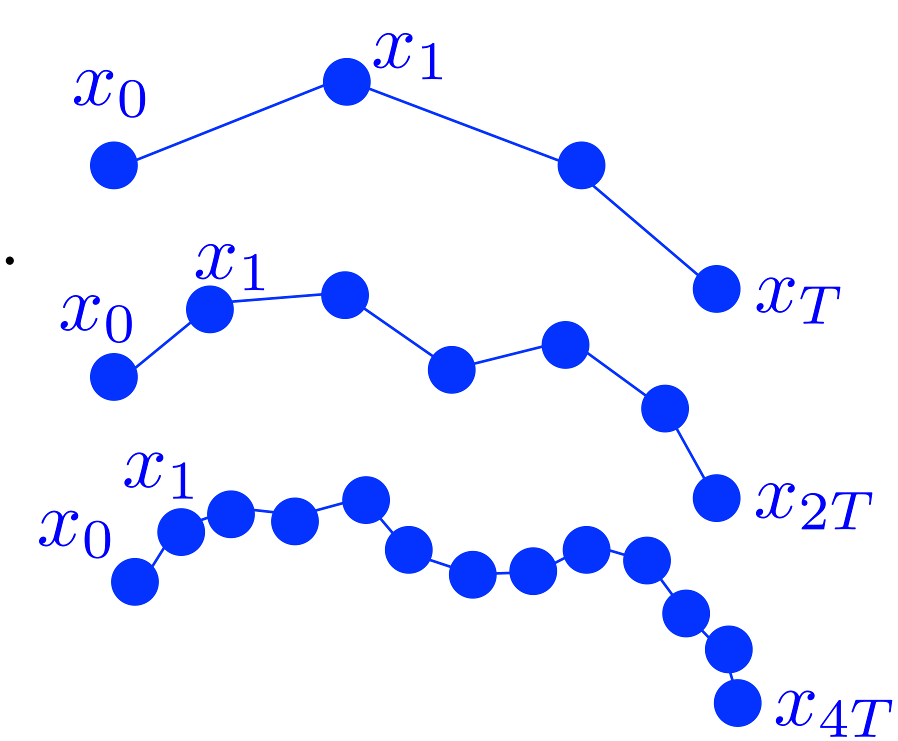


ResNet-type Architectures [He et al’ 16]



→ Makes the “infinite depth” limit non-degenerate.

→ Enable $v_\theta = 0$ initialization, i.e. identity map.

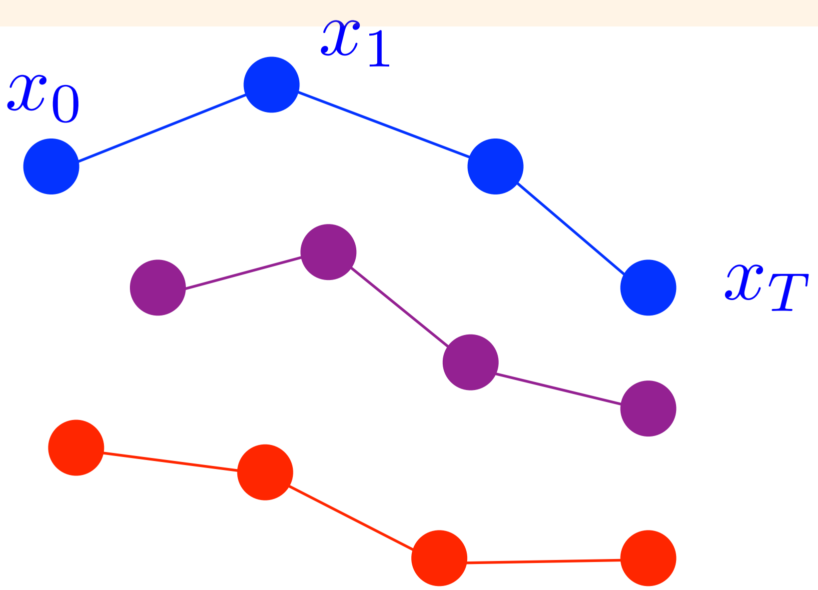


Infinite Depth and Neural-ODEs

ResNet [He et al, 2016]

$$\Phi_\theta(x_0) \triangleq x_T \quad \text{where}$$

$$x_{t+1} = x_t + \frac{1}{T}\, v_{\theta_t}(x_t)$$



Infinite Depth and Neural-ODEs

ResNet [He et al, 2016]

$$\Phi_\theta(x_0) \triangleq x_T \quad \text{where}$$

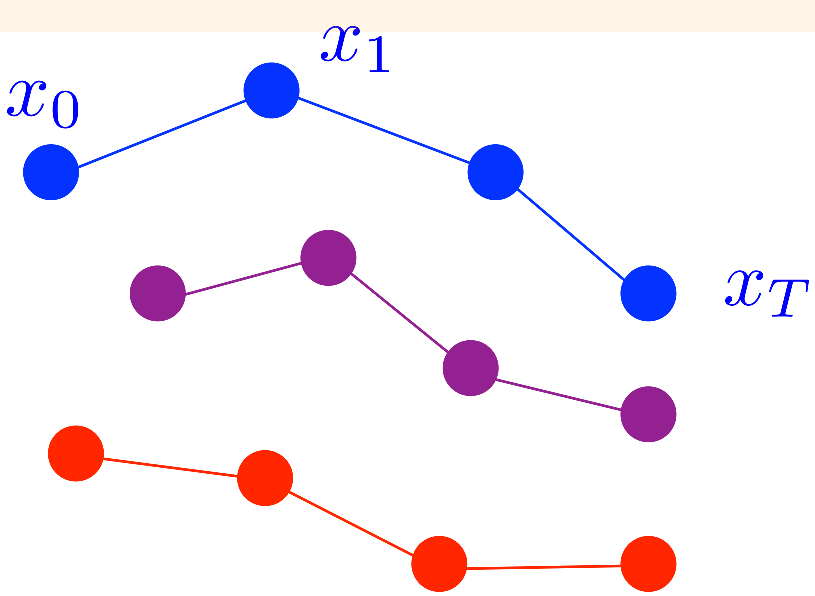
$$x_{t+1} = x_t + \frac{1}{T} v_{\theta_t}(x_t)$$

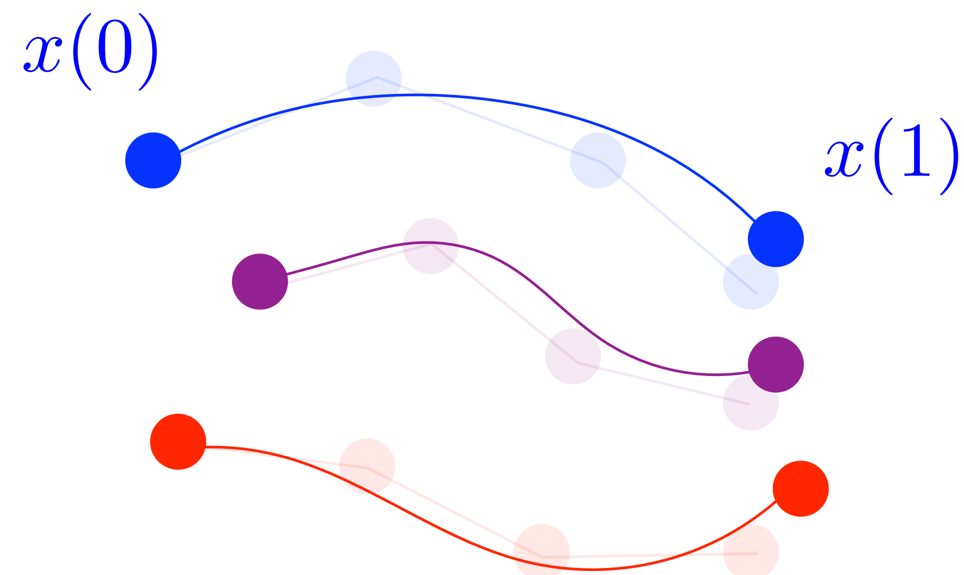
$T \to +\infty$

Neural ODE [Chen et al, 2018]

$$\Phi_\theta(x(0)) \triangleq x(1) \quad \text{where}$$

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = v_{\theta(t)}(x(t))$$





Infinite Depth and Neural-ODEs

ResNet [He et al, 2016]

$$\Phi_\theta(x_0) \triangleq x_T \quad \text{where}$$

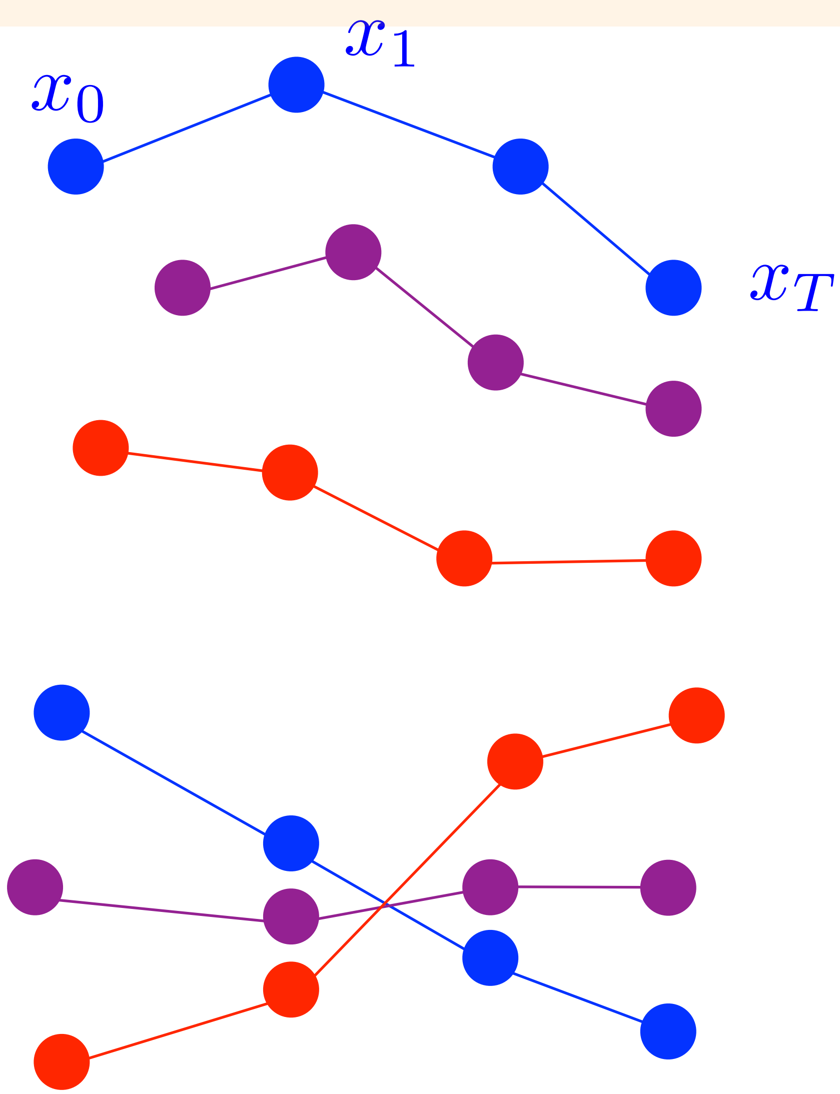
$$x_{t+1} = x_t + \frac{1}{T} v_{\theta_t}(x_t)$$

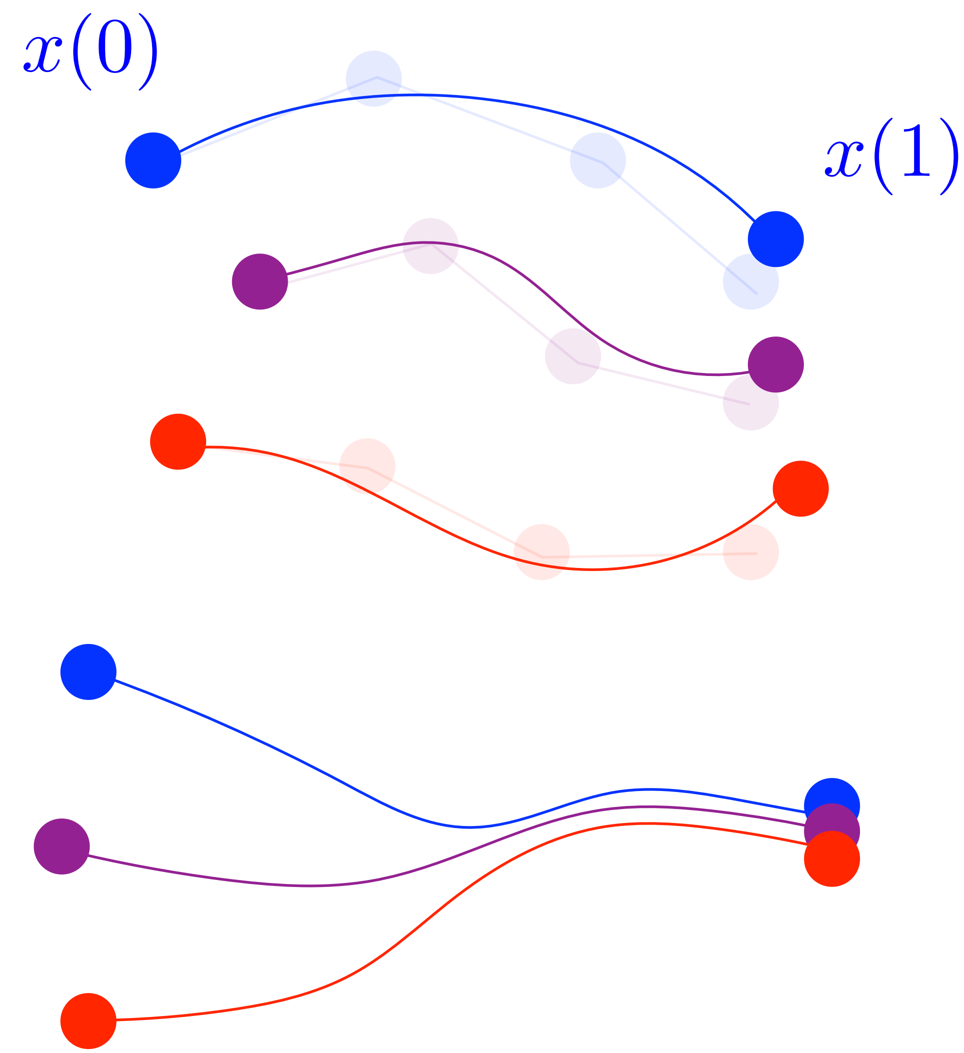
$T \to +\infty$

Neural ODE [Chen et al, 2018]

$$\Phi_\theta(x(0)) \triangleq x(1) \quad \text{where}$$

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = v_{\theta(t)}(x(t))$$



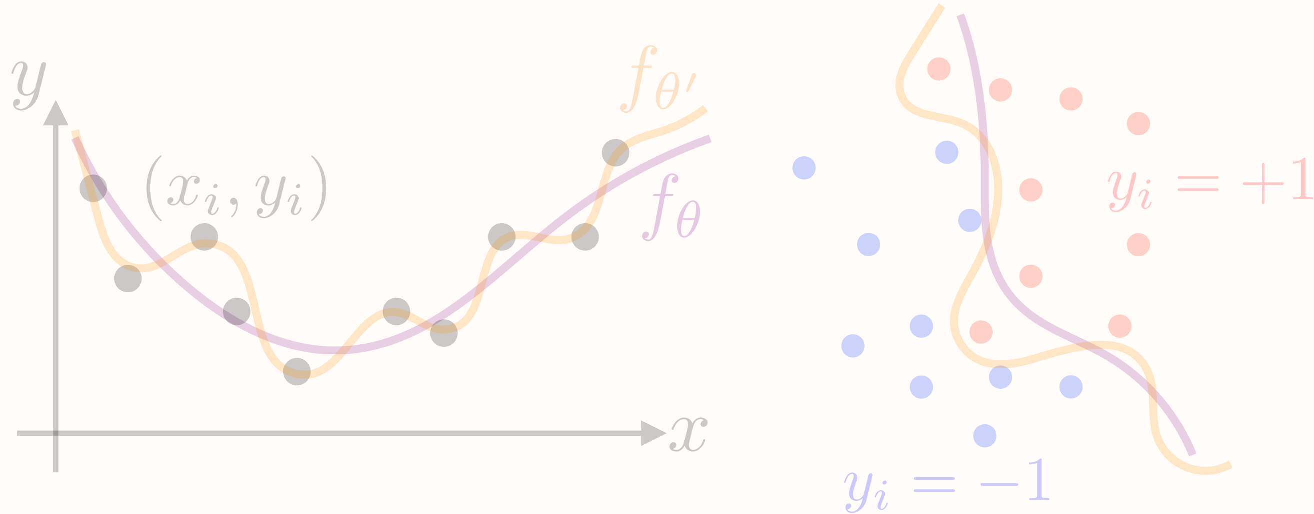


Trajectories cannot cross: Φ_θ defines a diffeomorphism.

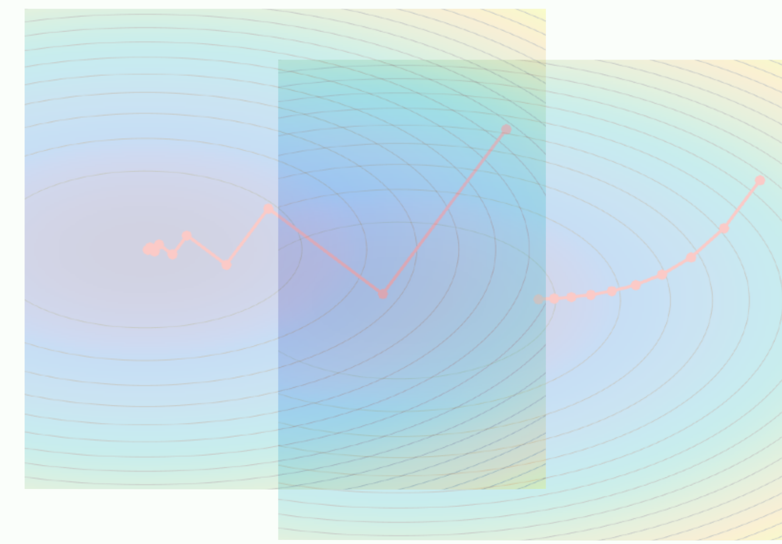
$T \to +\infty$ is a singular limit (θ can "explodes" during training)

Overview

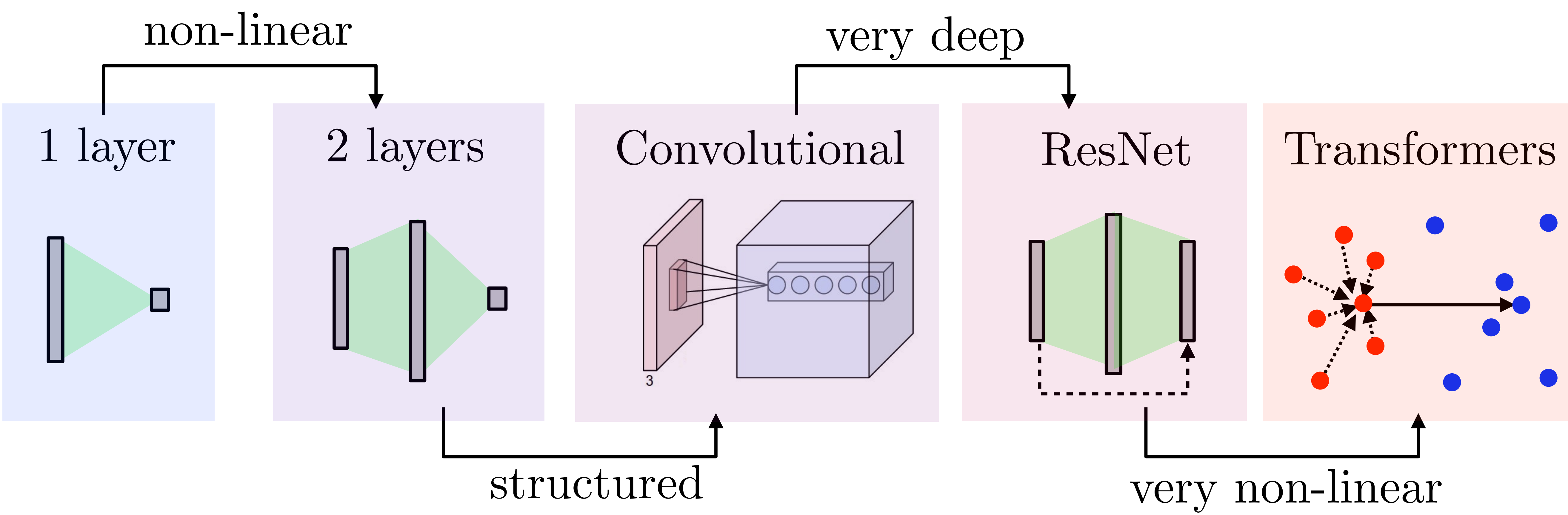
Machine Learning



Optimization

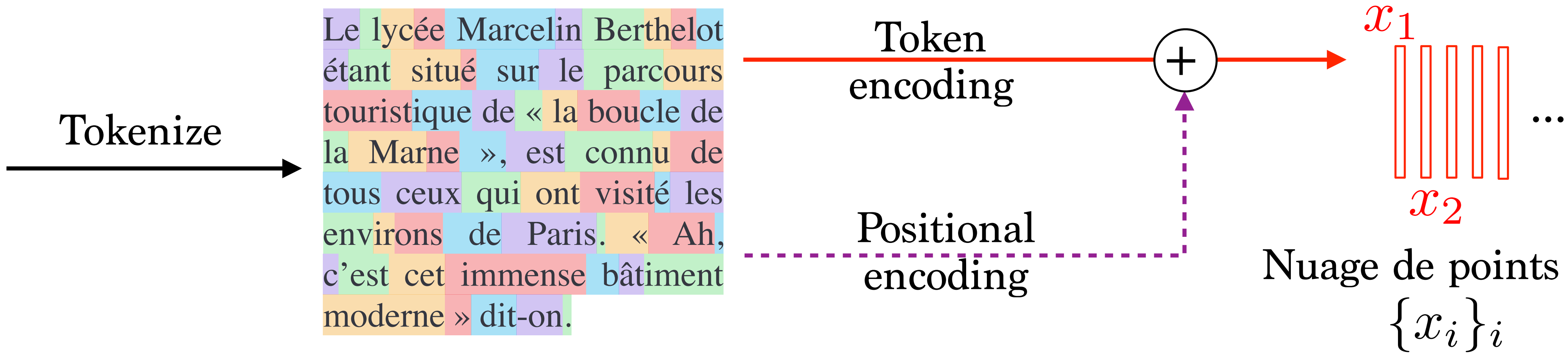


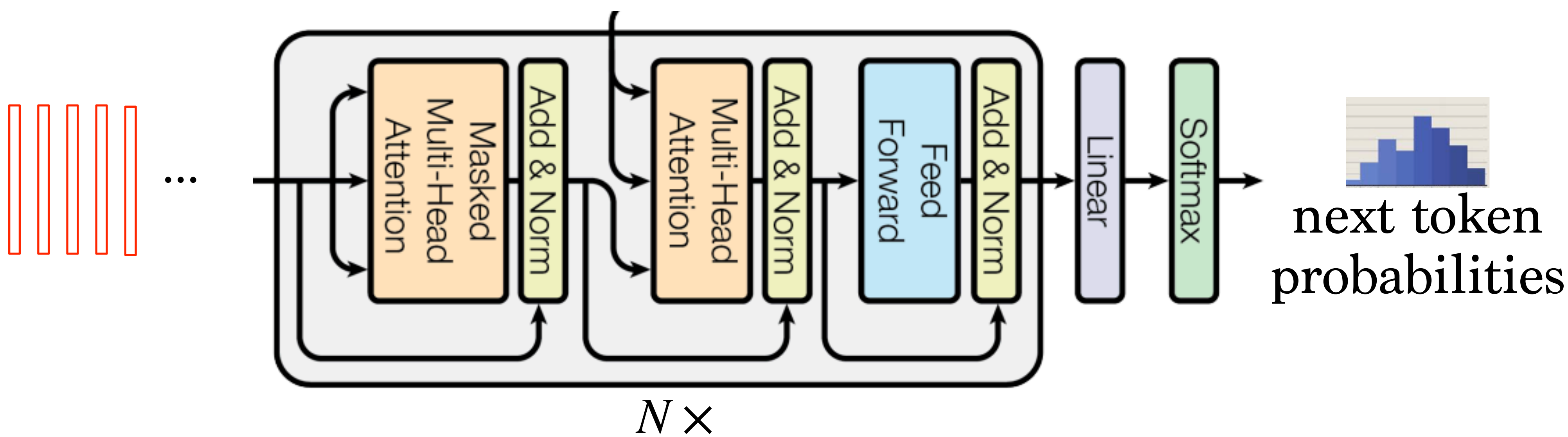
Networks Architectures



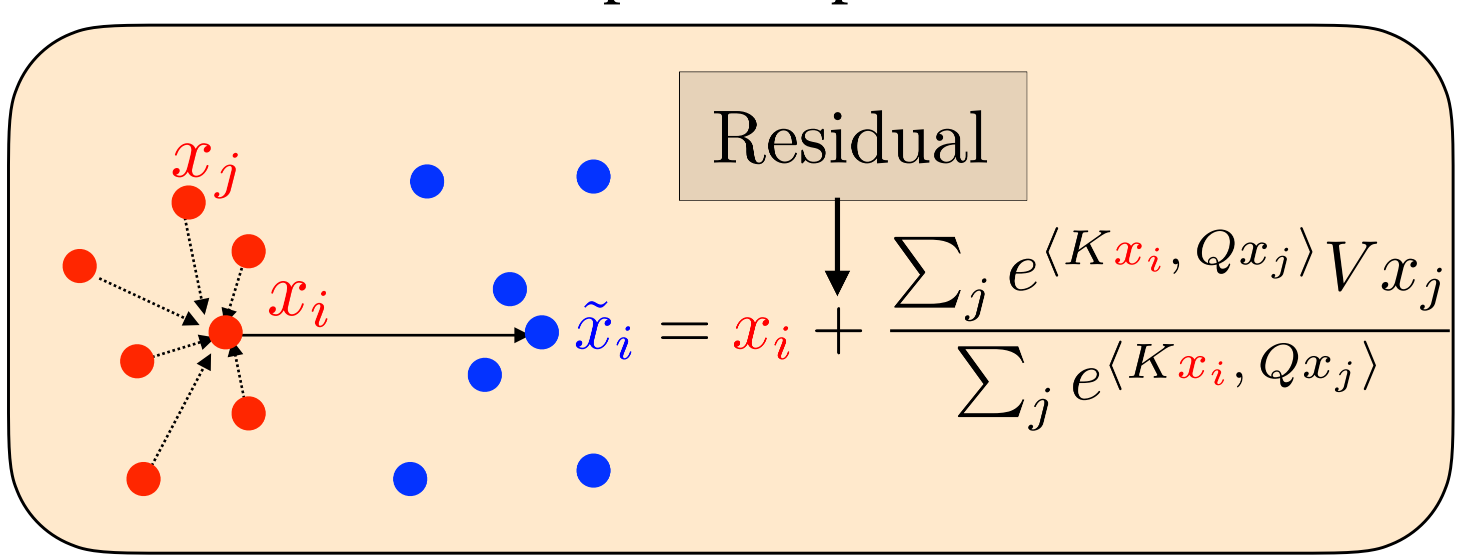
Les transformers et le méchanisme d'attention

Le lycée Marcelin Berthelot étant situé sur le parcours touristique de « la boucle de la Marne », est connu de tous ceux qui ont visité les environs de Paris. « Ah, c'est cet immense bâtiment moderne » dit-on.





Convolutions remplacées par des **attentions**



Large Language Models (LLM)

But: Pas seulement générer des données!
Répondre à des questions
Faire « tourner » des « algorithmes ».

Question:

Un transformer entrainé uniquement à faire du « next token prediction » est-il capable de raisonner?

Enjeu mathématique:

Modéliser et comprendre le phénomène de « in context learning ».

You

The Frobenius norm over matrices is $\langle S, T \rangle_F \coloneqq \sum_{i,j} S_{i,j} T_{i,j}$. Show that the Frobenius inner product can be written as $\langle S, T \rangle_F = \text{trace}(ST^\top)$.

ChatGPT

To demonstrate that the Frobenius inner product can be expressed as $\langle S, T \rangle_F = \text{trace}(ST^\top)$, let's start by defining the Frobenius inner product and the trace of a matrix.

1. **Frobenius Inner Product**: The Frobenius inner product of two matrices S and T is defined as $\langle S, T \rangle_F \coloneqq \sum_{i,j} S_{i,j} T_{i,j}$, where $S_{i,j}$ and $T_{i,j}$ are the elements of matrices S and T at the ith row and j^{th} column respectively.
2. **Trace of a Matrix**: The trace of a square matrix A is defined as the sum of its diagonal elements, i.e., $\text{trace}(A) = \sum_i A_{i,i}$.

Now, let's prove that $\langle S, T \rangle_F = \text{trace}(ST^\top)$:

- The product $ST^\top$ results in a matrix where each element $(ST^\top)_{i,j}$ is the dot product of the ith row of S and the jth row of T. That is, $(ST^\top)_{i,j} = \sum_k S_{i,k} T_{j,k}$.

14/20

Large Language Models (LLM)

But:
- Pas seulement générer des données!
- Répondre à des questions
- Faire « tourner » des « algorithmes ».

Question:

Un transformer entrainé uniquement à faire du « next token prediction » est-il capable de raisonner?

Enjeu mathématique:

Modéliser et comprendre le phénomène de « in context learning ».

You

The Frobenius norm over matrices is $\langle S, T \rangle_F \coloneqq \sum_{i,j} S_{i,j} T_{i,j}$. Show that the Frobenius inner product can be written as $\langle S, T \rangle_F = \text{trace}(ST^\top)$.

ChatGPT

To demonstrate that the Frobenius inner product can be expressed as $\langle S, T \rangle_F = \text{trace}(ST^\top)$, let's start by defining the Frobenius inner product and the trace of a matrix.

1. **Frobenius Inner Product**: The Frobenius inner product of two matrices S and T is defined as $\langle S, T \rangle_F := \sum_{i,j} S_{i,j} T_{i,j}$, where $S_{i,j}$ and $T_{i,j}$ are the elements of matrices S and T at the ith row and j^{th} column respectively.
2. **Trace of a Matrix**: The trace of a square matrix A is defined as the sum of its diagonal elements, i.e., $\text{trace}(A) = \sum_i A_{i,i}$.

Now, let's prove that $\langle S, T \rangle_F = \text{trace}(ST^\top)$:

- The product $ST^\top$ results in a matrix where each element $(ST^\top)_{i,j}$ is the dot product of the ith row of S and the jth row of T. That is, $(ST^\top)_{i,j} = \sum_k S_{i,k} T_{j,k}$.

14/20

Theorem [Sander, Peyré 2024]:

If one trains a transformer on geometric sequences $\{(x_1, x_2 = Ax_1, \ldots x_t = Ax_{t-1}) : A, x_1\}$ then for an input $(x_1, x_2 = Bx_1, \ldots, x_t = Bx_{t-1})$, the transformer:

1) estimates B 2) predicts $x_{t+1} = Bx_t$

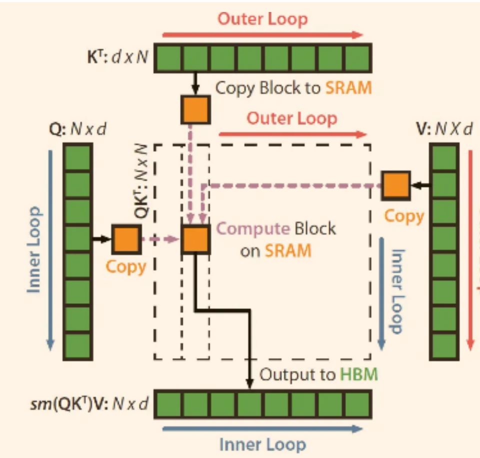
Le futur ?

Le machine learning (IA) change à toute vitesse. Impossible de faire des prédictions à 5 ans …

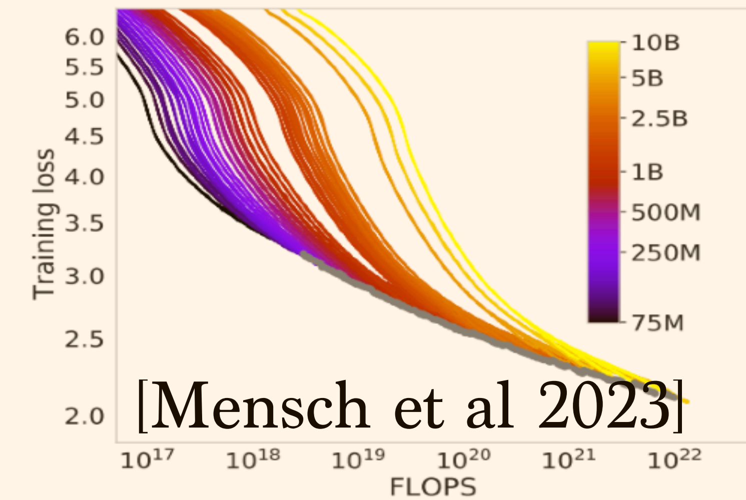
Numérique

High performance computing

Au delà de l'attention (meilleures scaling laws) ?



Flash Attention



[Mensch et al 2023]

Théorie

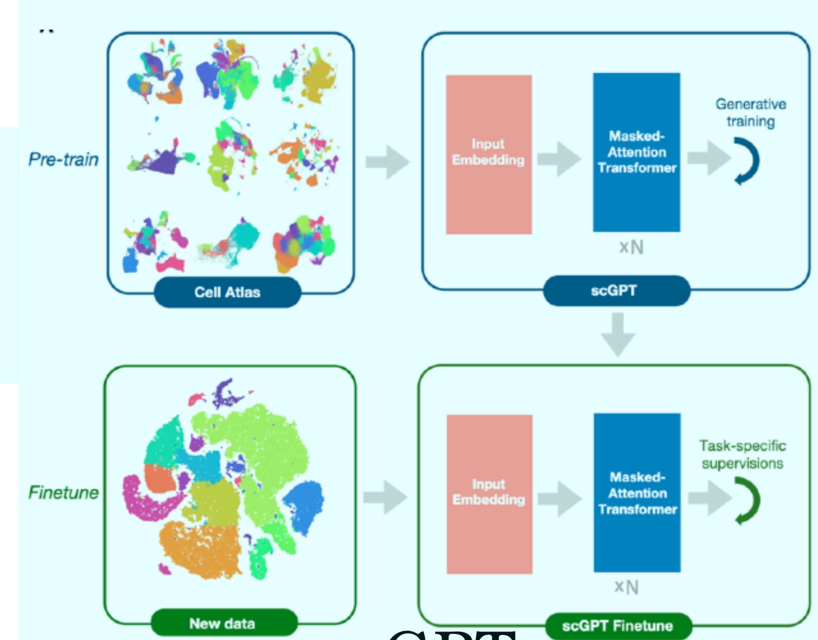
Connecter modèles de diffusion (continue) et LLM (discret)

Comprendre le processus de « in context learning ».

Applications

AI for science : au delà du next token prediction

Vers le multimodal (video, 3D, etc).



scGPT

Industrie

Open weights v.s. open source (données d'entrainement?)

Interface avec la vie quotidienne?