
Operator learning for multi-patch domains

Massimiliano Ghiotto

Departement of Mathematics, University of Pavia

Joint work with:

Carlo Marcati¹ Giancarlo Sangalli¹



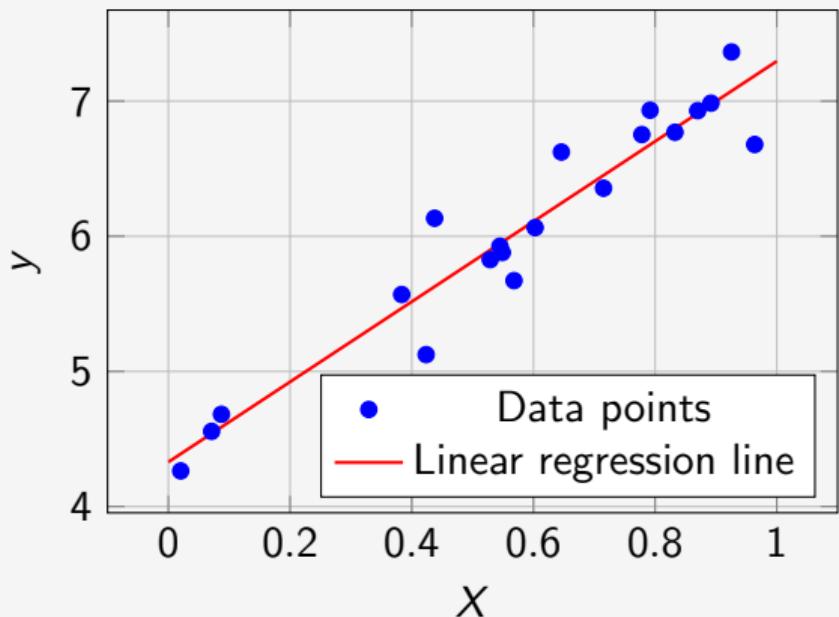
Table of Contents

- ▶ Deep Learning
- ▶ Applicazioni interessanti
- ▶ Risultati teorici
- ▶ Operatori neurali

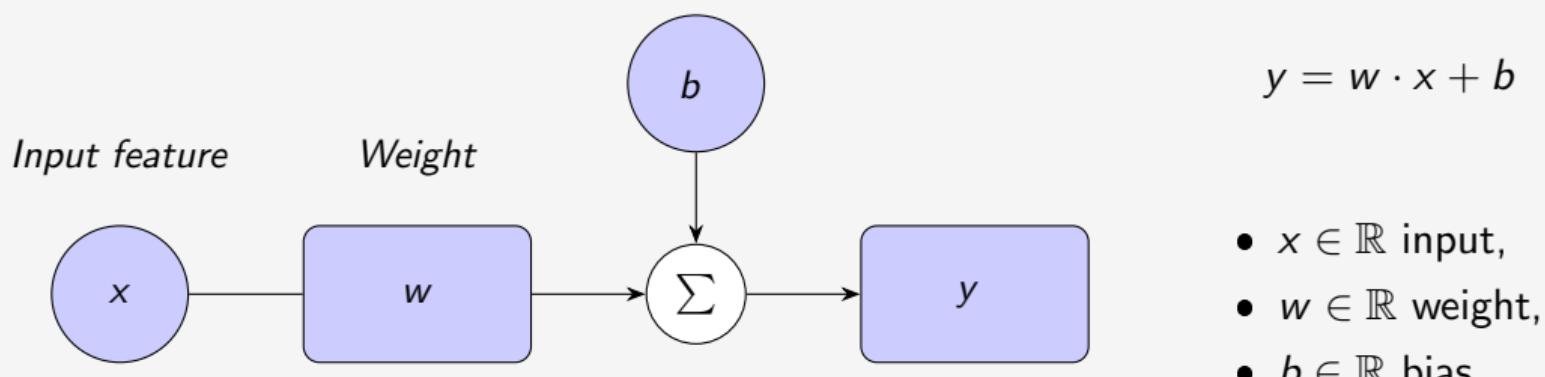


UNIVERSITÀ DI PAVIA

Linear Regression Example

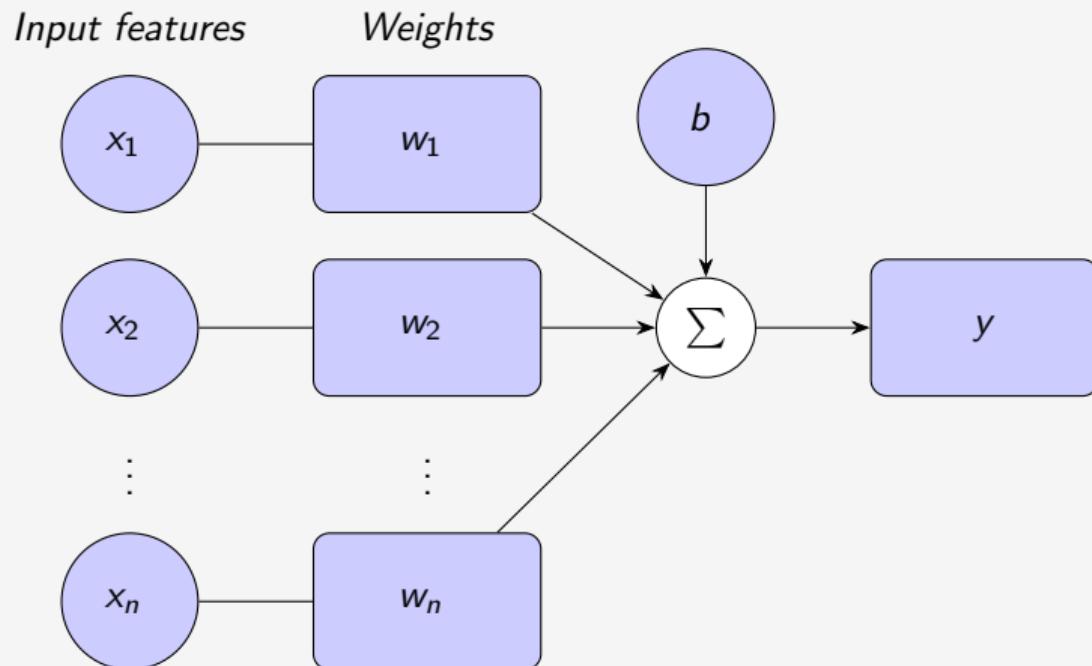


Dove $\{x_i\}_i \subset \mathbb{R}$ sono i dati di input e $\{y_i\}_i \subset \mathbb{R}$ sono i dati di output. Noi vogliamo trovare la retta $y = w \cdot x + b$ che meglio approssima i dati.



Regressione lineare

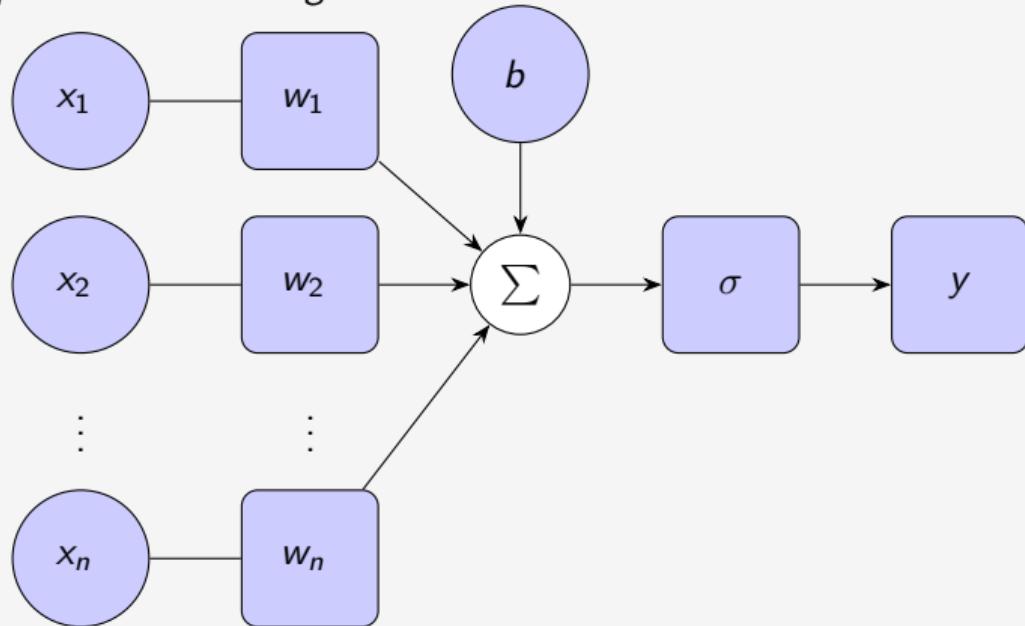
1 Deep Learning



$$y = \mathbf{w} \cdot \mathbf{x} + b$$

- $\mathbf{x} \in \mathbb{R}^n$ inputs,
- $\mathbf{w} \in \mathbb{R}^n$ weights,
- $b \in \mathbb{R}$ bias.

Input features Weights



Problema di classificazione.

$$y = \sigma(\mathbf{w} \cdot \mathbf{x} + b),$$

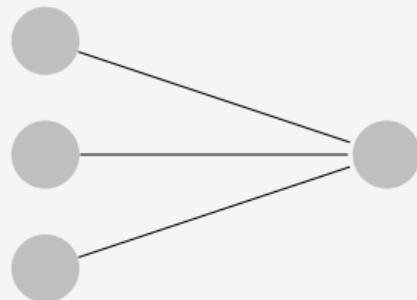
con $\sigma(z) = 1$ se $z \geq 0$ e nulla altrove.

Chiamiamo $\hat{y}(\mathbf{x})$ la label corretta associata ad \mathbf{x} , l'algoritmo per trovare \mathbf{w} e b è:

- calcolare $y(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$,
- se $y(\mathbf{x}) = \hat{y}(\mathbf{x})$ no aggiornamento,
- se $y(\mathbf{x}) = 1$ e $\hat{y}(\mathbf{x}) = 0$ allora aggiorniamo $\mathbf{w} = \mathbf{w} + \eta \mathbf{x}$ e $b = b + \eta$.
- se $y(\mathbf{x}) = 0$ e $\hat{y}(\mathbf{x}) = 1$ allora aggiorniamo $\mathbf{w} = \mathbf{w} - \eta \mathbf{x}$ e $b = b - \eta$.

Chiamiamo $\hat{y}(\mathbf{x})$ la label corretta associata ad \mathbf{x} , l'algoritmo per trovare \mathbf{w} e b è:

- calcolare $y(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$,
- se $y(\mathbf{x}) = \hat{y}(\mathbf{x})$ no aggiornamento,
- se $y(\mathbf{x}) = 1$ e $\hat{y}(\mathbf{x}) = 0$ allora aggiorniamo $\mathbf{w} = \mathbf{w} + \eta \mathbf{x}$ e $b = b + \eta$.
- se $y(\mathbf{x}) = 0$ e $\hat{y}(\mathbf{x}) = 1$ allora aggiorniamo $\mathbf{w} = \mathbf{w} - \eta \mathbf{x}$ e $b = b - \eta$.



Ogni cerchio rappresenta un neurone, a cui è associato un bias. Ogni freccia rappresenta una connessione tra due neuroni a cui è associato un peso.

Notiamo che se $x_0, x_1 \in \{0, 1\}$ allora con il percettrone si puo' rappresentare le operazioni logiche di negazione, congiunzione e disgiunzione.

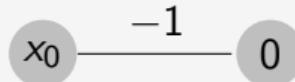


Figure: Negazione

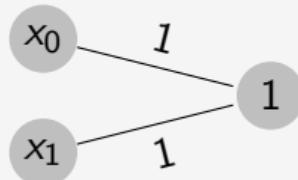


Figure: Congiunzione

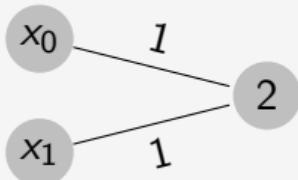


Figure: Disgiunzione

Notiamo che se $x_0, x_1 \in \{0, 1\}$ allora con il percettrone si puo' rappresentare le operazioni logiche di negazione, congiunzione e disgiunzione.

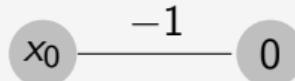


Figure: Negazione

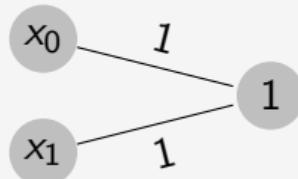


Figure: Congiunzione

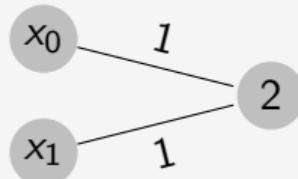


Figure: Disgiunzione

Ma non si puo' rappresentare la disgiunzione esclusiva (XOR).

Indicando lo XOR con \times vale che

$$x_0 \times x_1 = (x_0 \vee x_1) \wedge \neg(x_0 \wedge x_1) = (x_0 \vee x_1) \wedge (\neg x_0 \vee \neg x_1).$$

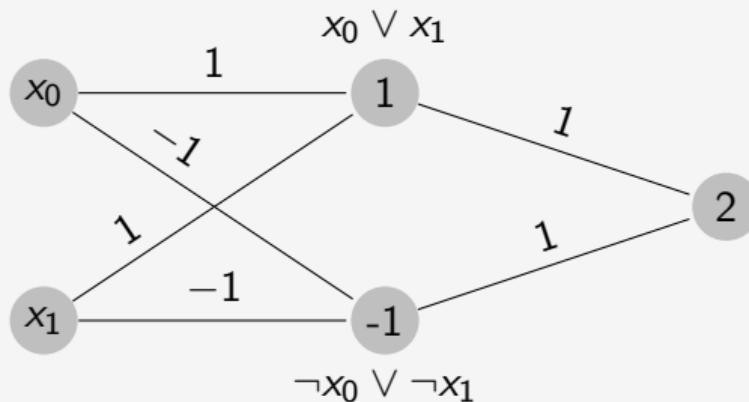
Multi-layer Perceptron

1 Deep Learning

Indicando lo XOR con \times vale che

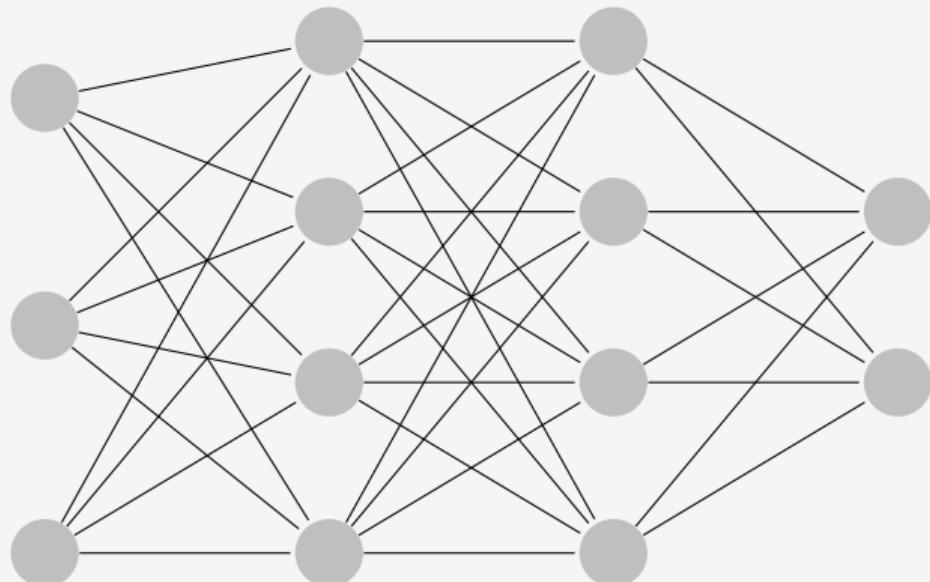
$$x_0 \times x_1 = (x_0 \vee x_1) \wedge \neg(x_0 \wedge x_1) = (x_0 \vee x_1) \wedge (\neg x_0 \vee \neg x_1).$$

Dunque lo possiamo rappresentare con



Multi-layer Perceptron

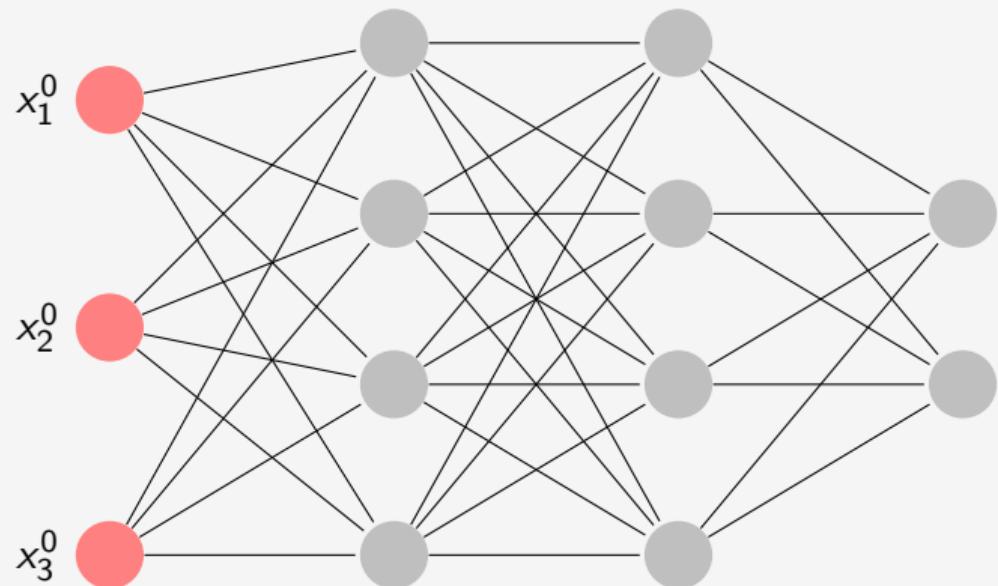
1 Deep Learning



Esempio di architettura di un Multi-layer Perceptron con 3 neuroni di input, 4 neuroni nel primo hidden layer, 4 neuroni nel secondo hidden layer e 2 neuroni di output.

Multi-layer Perceptron

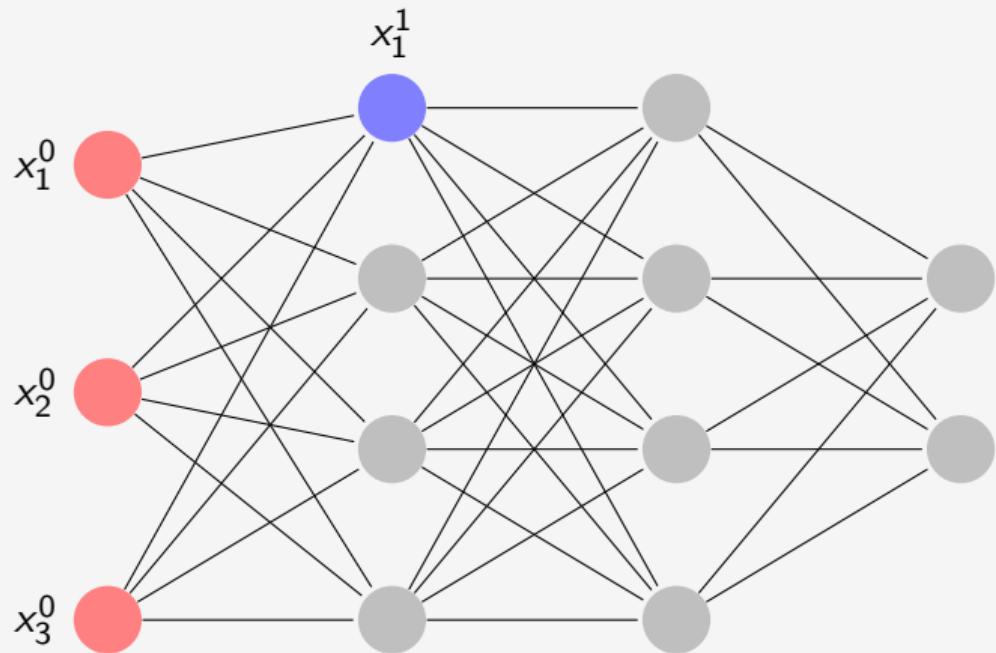
1 Deep Learning



Layer di input con 3 neuroni.

Multi-layer Perceptron

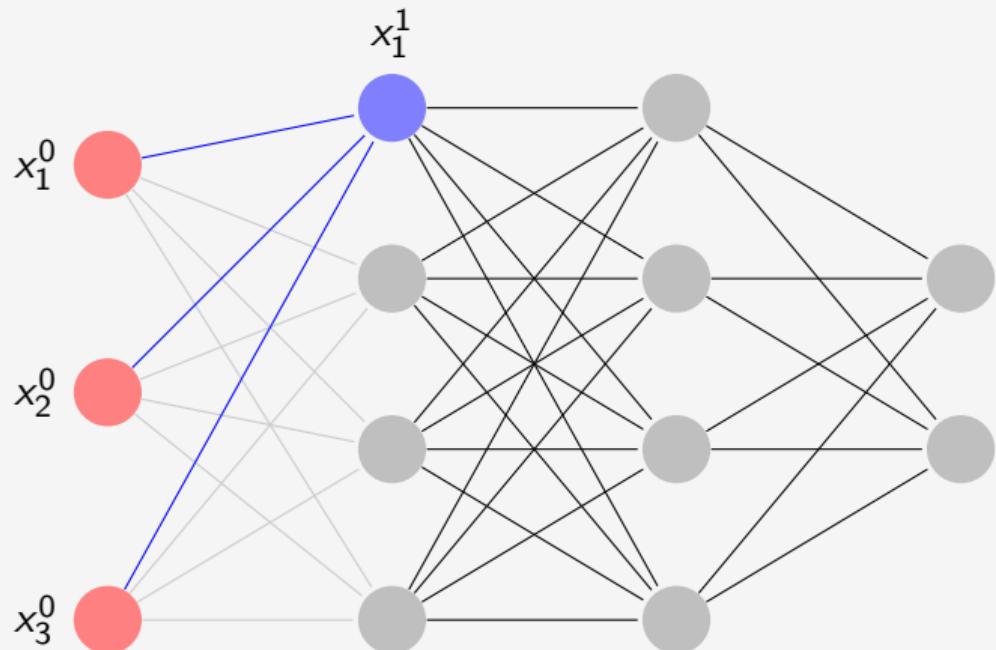
1 Deep Learning



Vogliamo definire il valore del neurone x_1^1 .

Multi-layer Perceptron

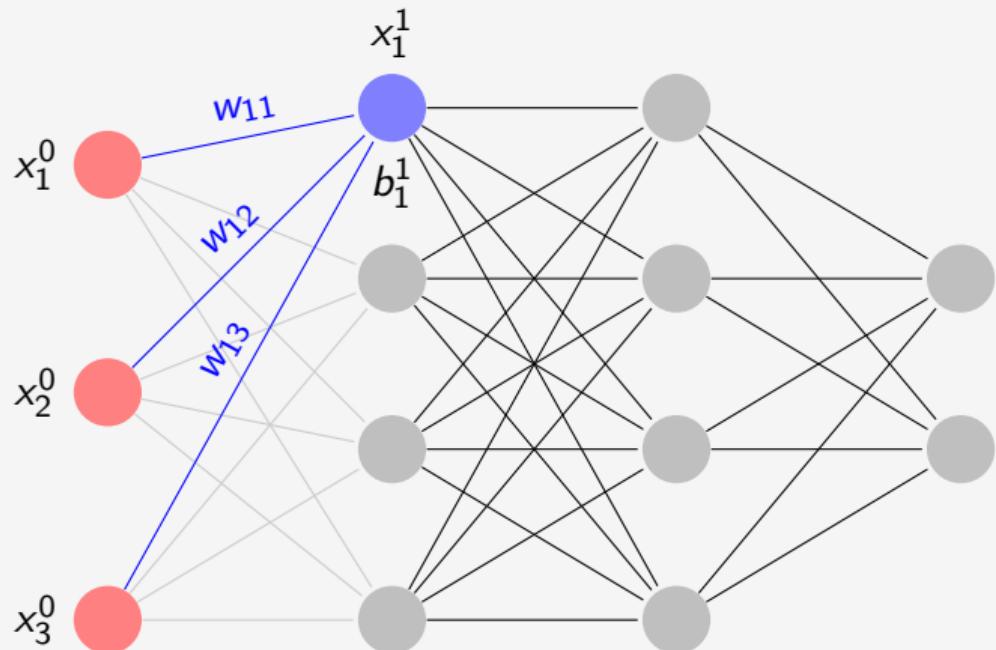
1 Deep Learning



Evidenziamo le connessioni con
i neuroni nel layer precedente.

Multi-layer Perceptron

1 Deep Learning

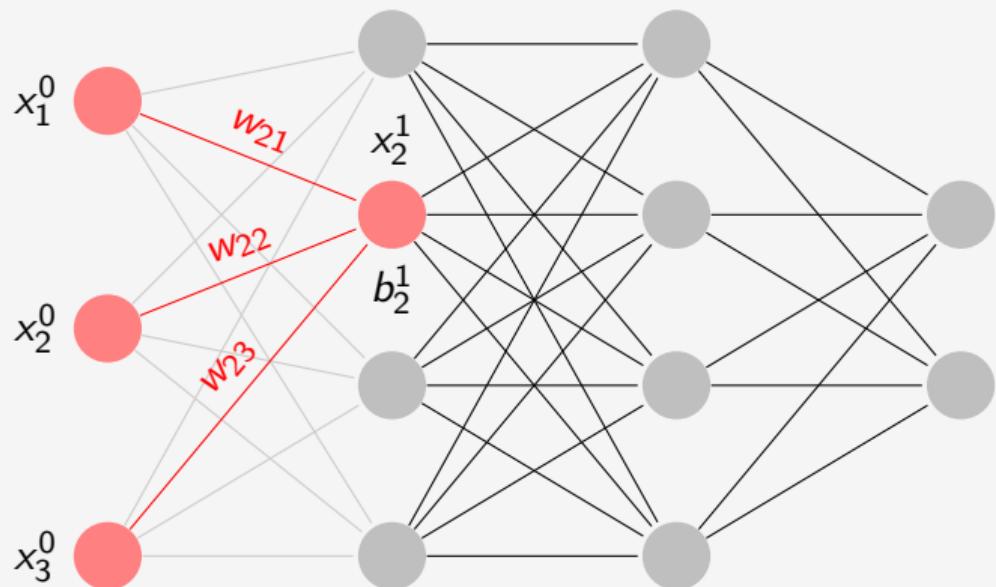


$$x_1^1 = \sigma \left(b_1^1 + \sum_{i=1}^3 w_{1i} x_i^0 \right)$$

Dove σ è una funzione di attivazione non lineare.

Multi-layer Perceptron

1 Deep Learning

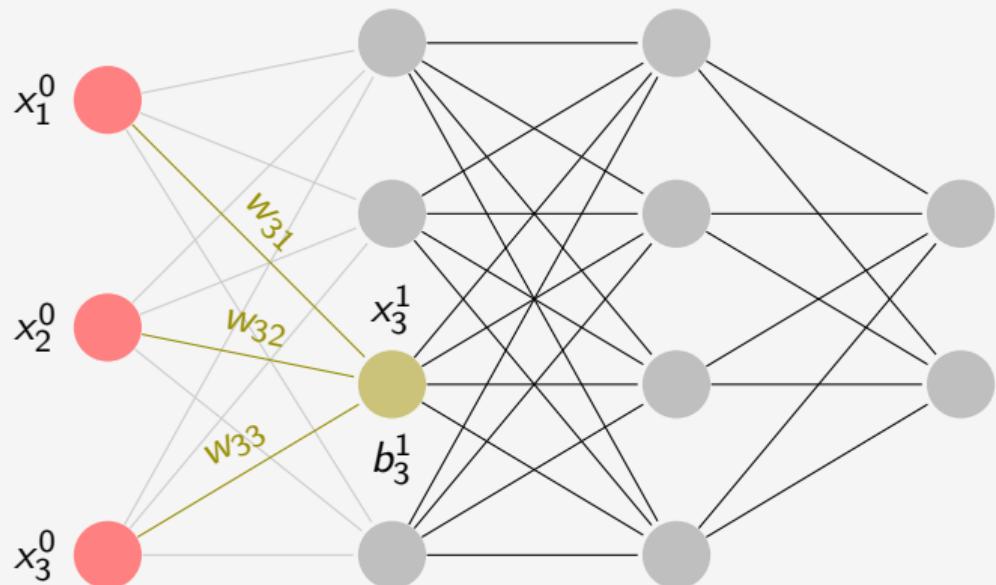


$$x_1^1 = \sigma \left(b_1^1 + \sum_{i=1}^3 w_{1i} x_i^0 \right)$$

$$x_2^1 = \sigma \left(b_2^1 + \sum_{i=1}^4 w_{2i} x_i^0 \right)$$

Multi-layer Perceptron

1 Deep Learning



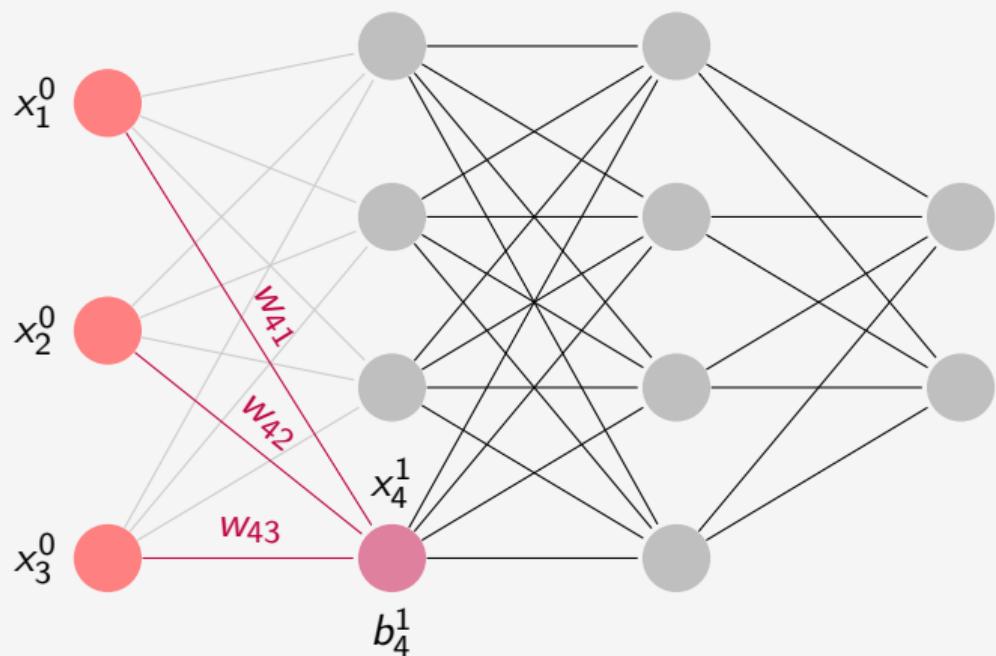
$$x_1^1 = \sigma \left(b_1^1 + \sum_{i=1}^3 w_{1i} x_i^0 \right)$$

$$x_2^1 = \sigma \left(b_2^1 + \sum_{i=1}^4 w_{2i} x_i^0 \right)$$

$$x_3^1 = \sigma \left(b_3^1 + \sum_{i=1}^4 w_{3i} x_i^0 \right)$$

Multi-layer Perceptron

1 Deep Learning



$$x_1^1 = \sigma \left(b_1^1 + \sum_{i=1}^3 w_{1i} x_i^0 \right)$$

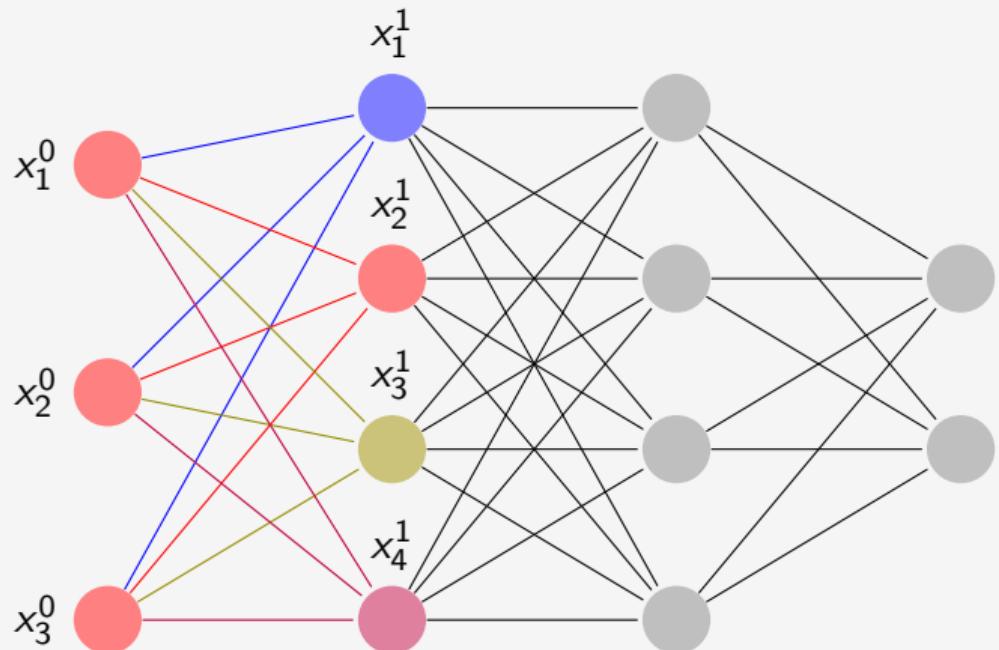
$$x_2^1 = \sigma \left(b_2^1 + \sum_{i=1}^4 w_{2i} x_i^0 \right)$$

$$x_3^1 = \sigma \left(b_3^1 + \sum_{i=1}^4 w_{3i} x_i^0 \right)$$

$$x_4^1 = \sigma \left(b_4^1 + \sum_{i=1}^2 w_{4i} x_i^0 \right)$$

Multi-layer Perceptron

1 Deep Learning



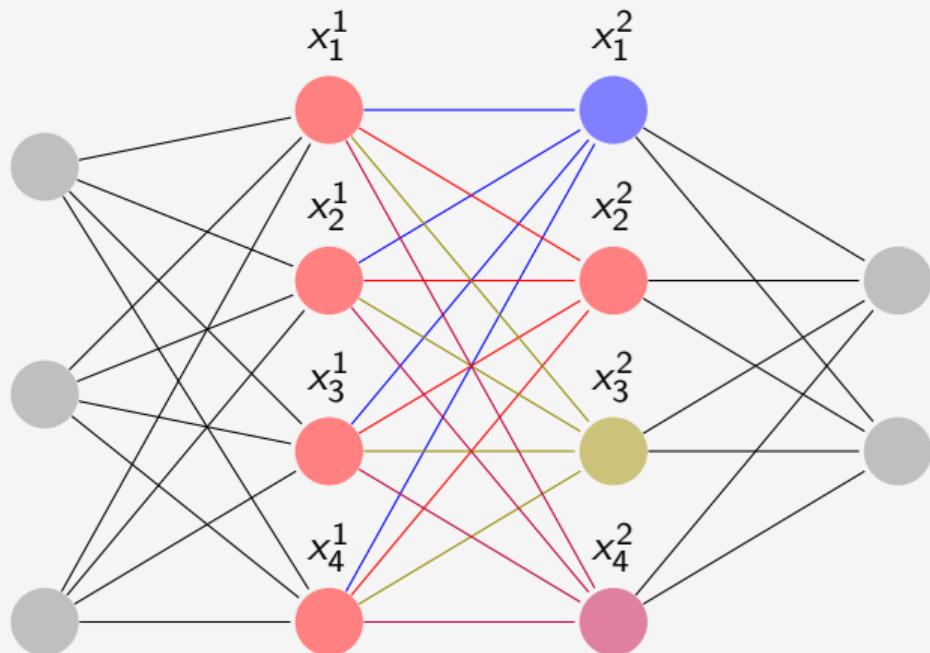
Dato $\mathbf{x}^0 \in \mathbb{R}^3$

$$\mathbf{x}^1 := \sigma(\mathbf{W}_1 \mathbf{x}^0 + \mathbf{b}_1) \in \mathbb{R}^4$$

con $\mathbf{W} \in \mathbb{R}^{4 \times 3}$, $\mathbf{b} \in \mathbb{R}^4$ e σ funzione di attivazione non lineare applicata componente per componente.

Multi-layer Perceptron

1 Deep Learning



Dato $\mathbf{x}^0 \in \mathbb{R}^3$

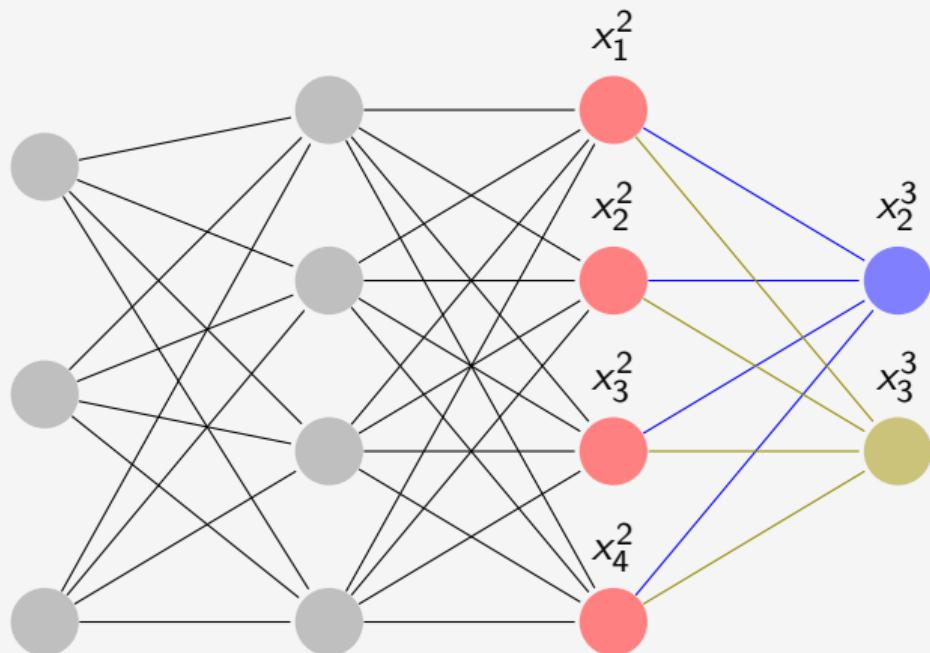
$$\mathbf{x}^1 := \sigma(\mathbf{W}_1 \mathbf{x}^0 + \mathbf{b}_1) \in \mathbb{R}^4$$

$$\mathbf{x}^2 := \sigma(\mathbf{W}_2 \mathbf{x}^1 + \mathbf{b}_2) \in \mathbb{R}^4$$

con $\mathbf{W}_2 \in \mathbb{R}^{4 \times 4}$ e $\mathbf{b} \in \mathbb{R}^4$.

Multi-layer Perceptron

1 Deep Learning



Dato $\mathbf{x}^0 \in \mathbb{R}^3$

$$\mathbf{x}^1 := \sigma(\mathbf{W}_1 \mathbf{x}^0 + \mathbf{b}_1) \in \mathbb{R}^4$$

$$\mathbf{x}^2 := \sigma(\mathbf{W}_2 \mathbf{x}^1 + \mathbf{b}_2) \in \mathbb{R}^4$$

$$\mathbf{x}^3 := \sigma(\mathbf{W}_3 \mathbf{x}^2 + \mathbf{b}_3) \in \mathbb{R}^2$$

con $\mathbf{W}_3 \in \mathbb{R}^{2 \times 4}$ e $\mathbf{b} \in \mathbb{R}^2$.

Definition

Un Multi-layer Perceptron (MLP) con input $\mathbf{x}^0 \in \mathbb{R}^{d_0}$, L hidden layers con d_1, \dots, d_L neuroni e output $\mathbf{x}^L \in \mathbb{R}^{d_L}$ è definito come

$$\mathbf{x}^1 := \sigma(\mathbf{W}_1 \mathbf{x}^0 + \mathbf{b}_1) \in \mathbb{R}^{d_1}$$

⋮

$$\mathbf{x}^L := \sigma(\mathbf{W}_L \mathbf{x}^{L-1} + \mathbf{b}_L) \in \mathbb{R}^{d_L}$$

dove $\mathbf{W}_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ e $\mathbf{b}_\ell \in \mathbb{R}^{d_\ell}$ per $\ell = 1, \dots, L$ e σ è una funzione di attivazione non lineare applicata componente per componente.

Definito un MLP e scelta una funzione di costo $C(\mathbf{x}, \mathbf{y})$ con \mathbf{y} target, si vuole minimizzare C rispetto ai parametri \mathbf{W}_ℓ e \mathbf{b}_ℓ .

Definition

Il metodo del gradiente è un metodo iterativo per minimizzare una funzione $MLP(\mathbf{p})$ definita su uno spazio \mathbb{R}^d . Dato un punto iniziale $\mathbf{p}^{(0)}$, la sequenza $\mathbf{p}^{(t)}$ è definita come

$$\mathbf{p}^{(t+1)} = \mathbf{p}^{(t)} - \eta \nabla f(\mathbf{p}^{(t)})$$

dove η è il learning rate.

Definito un MLP e scelta una funzione di costo $C(\mathbf{x}, \mathbf{y})$ con \mathbf{y} target, si vuole minimizzare C rispetto ai parametri \mathbf{W}_ℓ e \mathbf{b}_ℓ .

Definition

Il metodo del gradiente è un metodo iterativo per minimizzare una funzione $MLP(\mathbf{p})$ definita su uno spazio \mathbb{R}^d . Dato un punto iniziale $\mathbf{p}^{(0)}$, la sequenza $\mathbf{p}^{(t)}$ è definita come

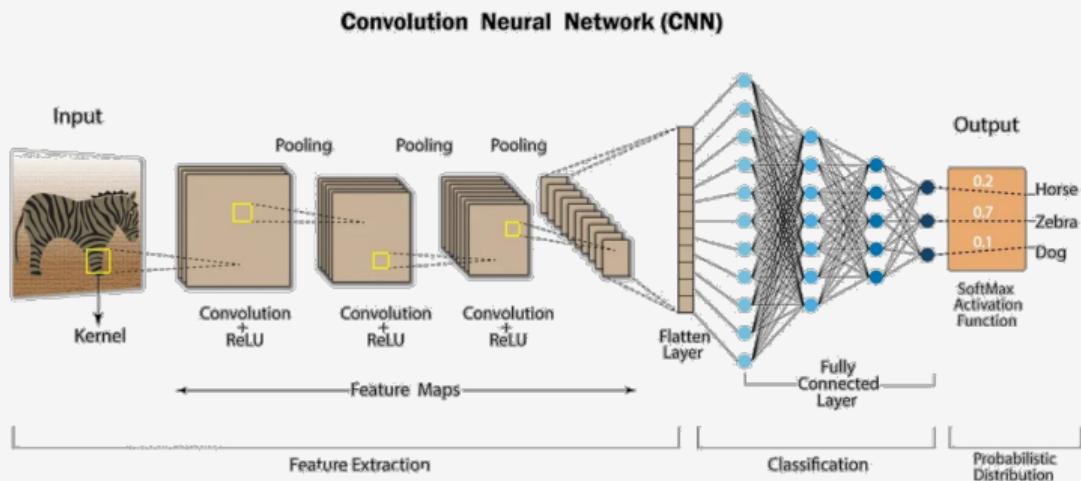
$$\mathbf{p}^{(t+1)} = \mathbf{p}^{(t)} - \eta \nabla f(\mathbf{p}^{(t)})$$

dove η è il learning rate.



Convolutional Neural Network (CNN)

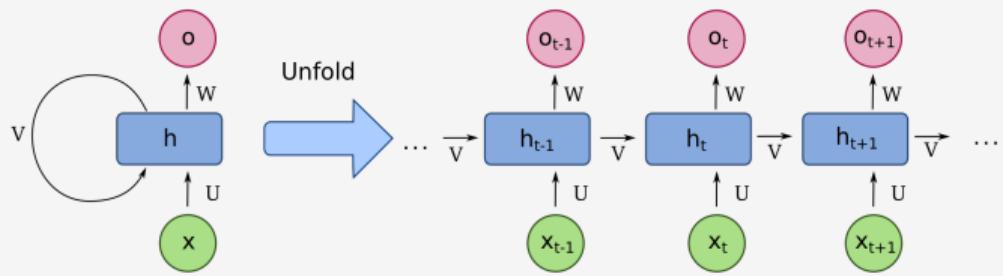
1 Deep Learning



Reti neurali convoluzionali
per l'applicazione alle
immagini.

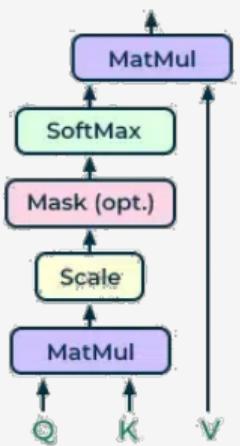
Recurrent Neural Network (RNN)

1 Deep Learning



Reti neurali ricorrenti per l'applicazione al linguaggio naturale ed ai segnali, o più in generale alle quantità dipendenti dal tempo.

Scaled Dot-Product Attention



Multi-Head Attention

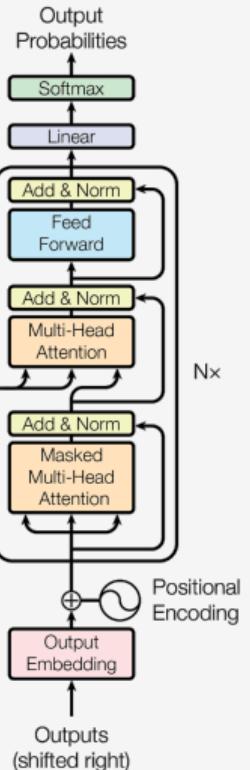
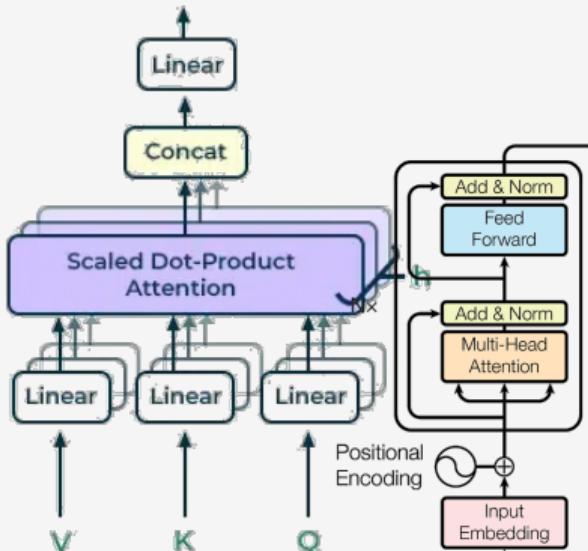


Table of Contents

- ▶ Deep Learning
- ▶ Applicazioni interessanti
- ▶ Risultati teorici
- ▶ Operatori neurali



UNIVERSITÀ DI PAVIA

Section 2

Applicazioni interessanti

Massimiliano
Ghiotto



UNIVERSITÀ
DI PAVIA

Applicazioni alle immagini

2 Applicazioni interessanti

Siti per generazione d'immagini: [DALL-E3](#)  [DALL-E](#), [Midjourney](#)  e [Ideogram](#) .



Applicazioni ai video

2 Applicazioni interessanti

Massimiliano

Ghiotto



UNIVERSITÀ
DI PAVIA

Deep learning per la generazione di video [SORA](#)  o per modificare video anche in tempo reale [NVIDIA Broadcast](#) .





- Modelli per la generazione di testo come [GPT-4](#) GPT-4 o [Copilot](#) .
- Esistono modelli open-source come [LLaMA-3](#)  scaricabile gratuitamente dal loro sito.
- Potete trovare modelli pre-addestrati e datasets su [Hugging Face](#) oppure su [kaggle](#) .

- Modelli per la generazione di testo come [GPT-4](#)  GPT-4 o [Copilot](#) .
- Esistono modelli open-source come [LLaMA-3](#)  scaricabile gratuitamente dal loro sito.
- Potete trovare modelli pre-addestrati e datasets su [Hugging Face](#)  oppure su [kaggle](#) .

Per allenare Chat-GPT3, formata da 175 bilioni di parametri, hanno impiegato 1023 Nvidia A100 per 34 giorni spendendo 5 milioni di dollari in corrente.



- Modelli per la generazione di testo come [GPT-4](#) GPT-4 o [Copilot](#) .
- Esistono modelli open-source come [LLaMA-3](#) scaricabile gratuitamente dal loro sito.
- Potete trovare modelli pre-addestrati e datasets su [Hugging Face](#) oppure su [kaggle](#) .

Per allenare Chat-GPT3, formata da 175 bilioni di parametri, hanno impiegato 1023 Nvidia A100 per 34 giorni spendendo 5 milioni di dollari in corrente.

Anche per scaricare il modello più piccolo di LLaMA 3 8B sono necessari 20 GB di VRAM e per il modello LLaMA 3 70B sono necessari 140GB di spazio e 160GB di VRAM in FP16.

Table of Contents

- ▶ Deep Learning
- ▶ Applicazioni interessanti
- ▶ Risultati teorici
- ▶ Operatori neurali



UNIVERSITÀ DI PAVIA

Section 3

Risultati teorici

Massimiliano
Ghiotto



UNIVERSITÀ
DI PAVIA

Definition

Sia $K \subset \mathbb{R}^d$ compatto e con $d \in \mathbb{N}$. Una funzione continua $f : \mathbb{R} \rightarrow \mathbb{R}$ si dice discriminante se

$$\int_K f(\mathbf{a} \cdot \mathbf{x} + b) \, d\mu(x) = 0 \quad \forall \mathbf{a} \in \mathbb{R}^d, \quad \forall b \in \mathbb{R} \quad \Rightarrow \quad \mu \equiv 0.$$

Definition

Sia $K \subset \mathbb{R}^d$ compatto e con $d \in \mathbb{N}$. Una funzione continua $f : \mathbb{R} \rightarrow \mathbb{R}$ si dice discriminante se

$$\int_K f(\mathbf{a} \cdot \mathbf{x} + b) d\mu(x) = 0 \quad \forall \mathbf{a} \in \mathbb{R}^d, \quad \forall b \in \mathbb{R} \quad \Rightarrow \quad \mu \equiv 0.$$

Definition

Una funzione continua $f : \mathbb{R} \rightarrow \mathbb{R}$ si dice sigmoidale se $\lim_{x \rightarrow -\infty} f(x) = 0$ e $\lim_{x \rightarrow +\infty} f(x) = 1$.

Definition

Sia $K \subset \mathbb{R}^d$ compatto e con $d \in \mathbb{N}$. Una funzione continua $f : \mathbb{R} \rightarrow \mathbb{R}$ si dice discriminante se

$$\int_K f(\mathbf{a} \cdot \mathbf{x} + b) d\mu(x) = 0 \quad \forall \mathbf{a} \in \mathbb{R}^d, \quad \forall b \in \mathbb{R} \quad \Rightarrow \quad \mu \equiv 0.$$

Definition

Una funzione continua $f : \mathbb{R} \rightarrow \mathbb{R}$ si dice sigmoidale se $\lim_{x \rightarrow -\infty} f(x) = 0$ e $\lim_{x \rightarrow +\infty} f(x) = 1$.

Tutte le funzioni sigmoidali sono discriminanti.

Theorem

Sia $K \subset \mathbb{R}^d$ compatto e con $d \in \mathbb{N}$ e $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ funzione discriminante allora $MLP(\sigma, d, 2)$ è denso in $C(K)$.

Ovvero per ogni $f \in C(K)$ e $\varepsilon > 0$ esiste una rete neurale $f_\theta \in MLP(\sigma, d, 2)$ tale che $\|f - f_\theta\|_\infty < \varepsilon$.

Theorem

Sia σ non polinomiale, X spazio di Banach, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$ compatti e V compatto in $C(K_1)$. Sia $G : V \rightarrow C(K_2)$ un operatore continuo non lineare, allora per ogni $\varepsilon > 0$ esistono $n, p, m \in \mathbb{N}$, $c_i^k, \theta_i^k, \xi_{ij}^k, \zeta^k \in \mathbb{R}$, $w^k \in \mathbb{R}^d$ e $x_j \in K_1$ tali che

$$\left| G(u)(y) - \sum_{k=1}^p \sum_{i=1}^n c_i^k \sigma \left(\sum_{j=1}^m \xi_{ij}^k u(x_j) + \theta_i^k \right) \sigma(w^k \cdot y + \zeta^k) \right| < \varepsilon$$

vale per ogni $u \in V$ e $y \in K_2$.

Approssimazione universale per operatori

3 Risultati teorici

Abbiamo $n, p, m \in \mathbb{N}$, $c_i^k, \theta_i^k, \xi_{ij}^k, \zeta^k \in \mathbb{R}$, $w^k \in \mathbb{R}^d$, $x_j \in K_1$, $y \in K_2 \subset \mathbb{R}^d$ e $u \in V$ tali che

$$\underbrace{\sum_{k=1}^p c_i^k}_{\text{scalar product}} \underbrace{\sum_{i=1}^n \sigma \left(\sum_{j=1}^m \xi_{ij}^k u(x_j) + \theta_i^k \right)}_{\text{branch net}} \underbrace{\sigma(w^k \cdot y + \zeta^k)}_{\text{trunk net}}. \quad (1)$$

Approssimazione universale per operatori

3 Risultati teorici

Abbiamo $n, p, m \in \mathbb{N}$, $c_i^k, \theta_i^k, \xi_{ij}^k, \zeta^k \in \mathbb{R}$, $w^k \in \mathbb{R}^d$, $x_j \in K_1$, $y \in K_2 \subset \mathbb{R}^d$ e $u \in V$ tali che

$$\underbrace{\sum_{k=1}^p}_{\text{scalar product}} \underbrace{\sum_{i=1}^n c_i^k \sigma \left(\sum_{j=1}^m \xi_{ij}^k u(x_j) + \theta_i^k \right)}_{\text{branch net}} \underbrace{\sigma(w^k \cdot y + \zeta^k)}_{\text{trunk net}}. \quad (1)$$

Dove notiamo che la branch net è una rete neurale con 2 layers ed m neuroni in input e p neuroni di output e la trunk net è una rete neurale con 1 layer con d layer di input e p .

Deep Operator Network

3 Risultati teorici

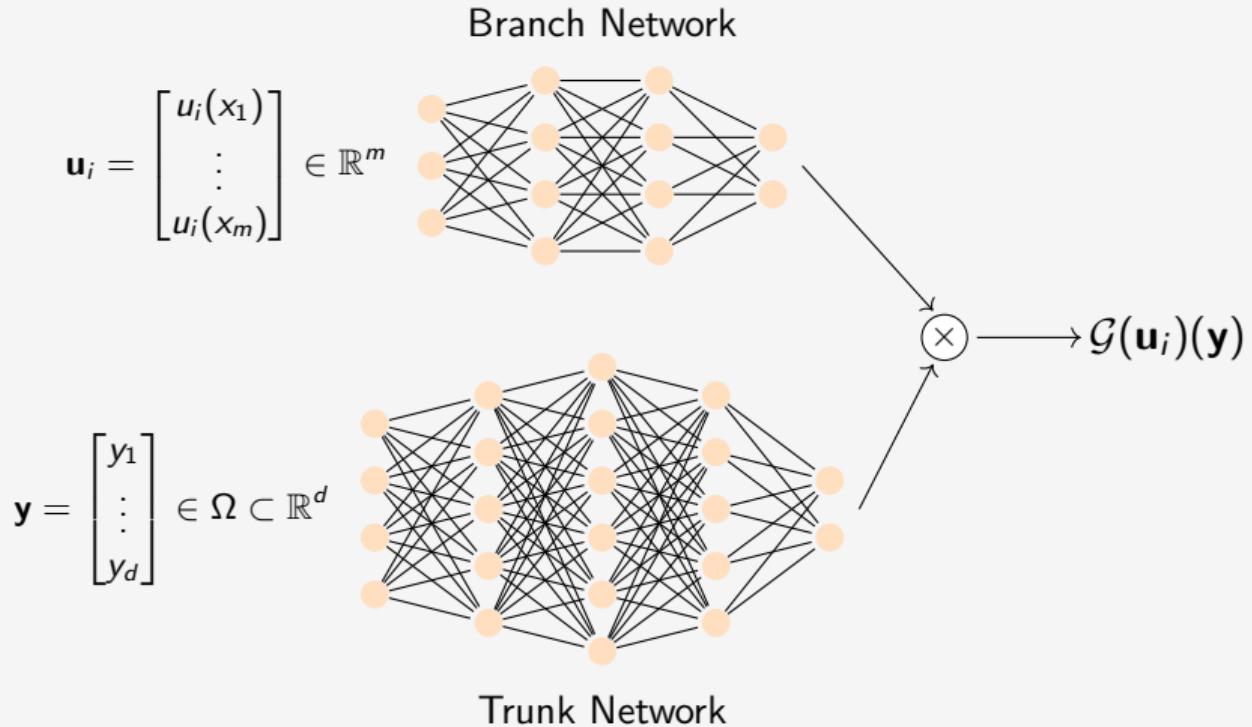


Table of Contents

- ▶ Deep Learning
- ▶ Applicazioni interessanti
- ▶ Risultati teorici
- ▶ Operatori neurali



UNIVERSITÀ DI PAVIA

Section 4

Operatori neurali

Massimiliano
Ghiotto



UNIVERSITÀ
DI PAVIA

Consideriamo il problema di Darcy in due dimensioni, sottoposto a condizioni al bordo di Dirichlet omogenee. L'equazione di flusso di Darcy, è un'equazione ellittica del secondo ordine, definita da

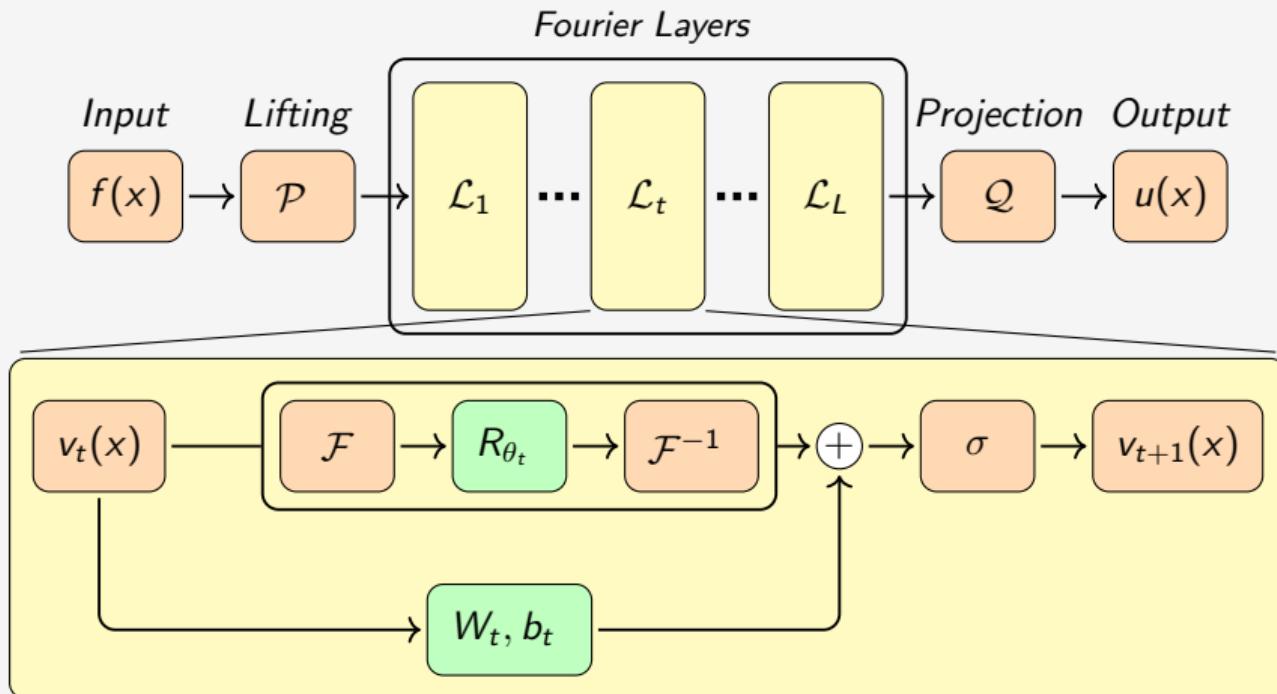
$$\begin{cases} -\nabla(a \cdot \nabla u) = f, & \text{in } D \\ u = 0, & \text{su } \partial D \end{cases} \quad (2)$$

dove $D = (0, 1)^2$. Per questo problema $\mathcal{A} = L^\infty(D, \mathbb{R}^+)$, $\mathcal{U} = H_0^1(D, \mathbb{R})$. Fissiamo come r.h.s $f \equiv 1$ e considerando la formulazione debole del problema si ha che la soluzione operatore è

$$\mathcal{G} : L^\infty(D, \mathbb{R}^+) \rightarrow H_0^1(D, \mathbb{R}), \quad \mathcal{G} : a \mapsto u.$$

Fourier Neural Operator

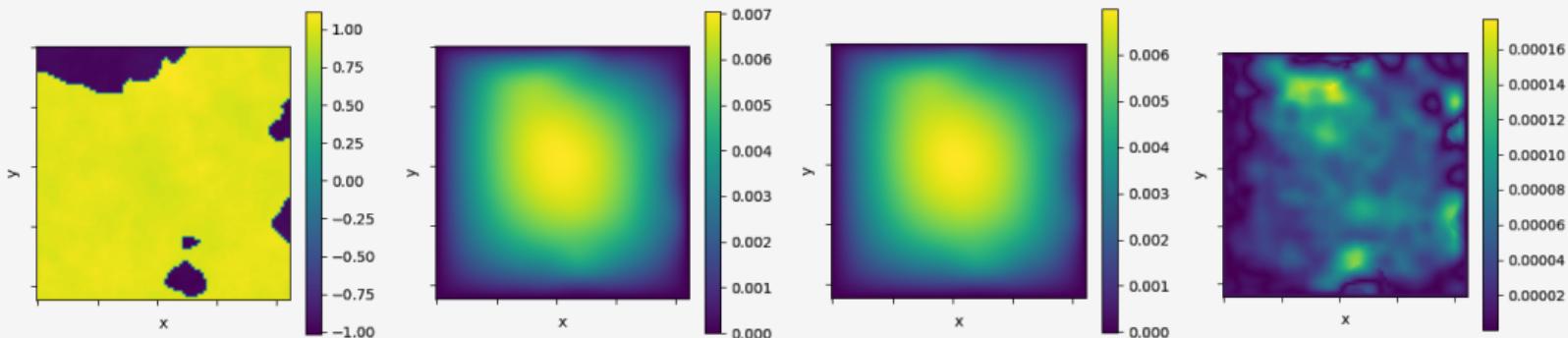
4 Operatori neurali



Darcy problem

4 Operatori neurali

Volgiamo approssimare l'operatore soluzione $\mathcal{G} : a \mapsto u$ associato al problema di Darcy.



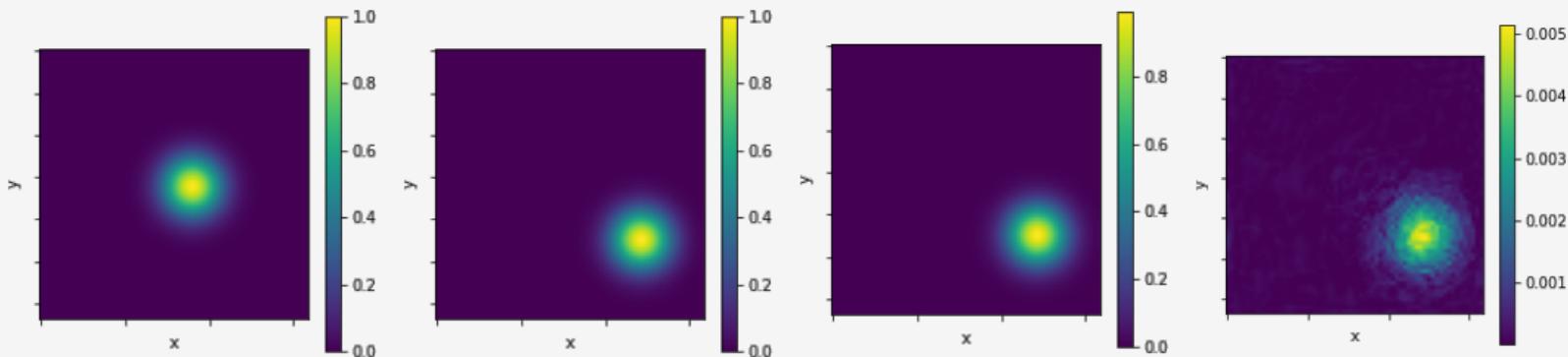
La FNO ha 2,3M parametri, tempo di training di 2 ore, errore di test relativo in norma L^2 pari a 0.008.

Transport continuous equation

4 Operatori neurali

$$u_t + v \cdot \nabla u = 0 \quad \text{in } \Omega \times (0, T), \quad u_0(x, y) = f(x, y),$$

con f funzione regolare. Si fissa $v = (0.2, 0.2)$ e si approssima l'operatore $\mathcal{G} : f \mapsto u_{T=1}$.



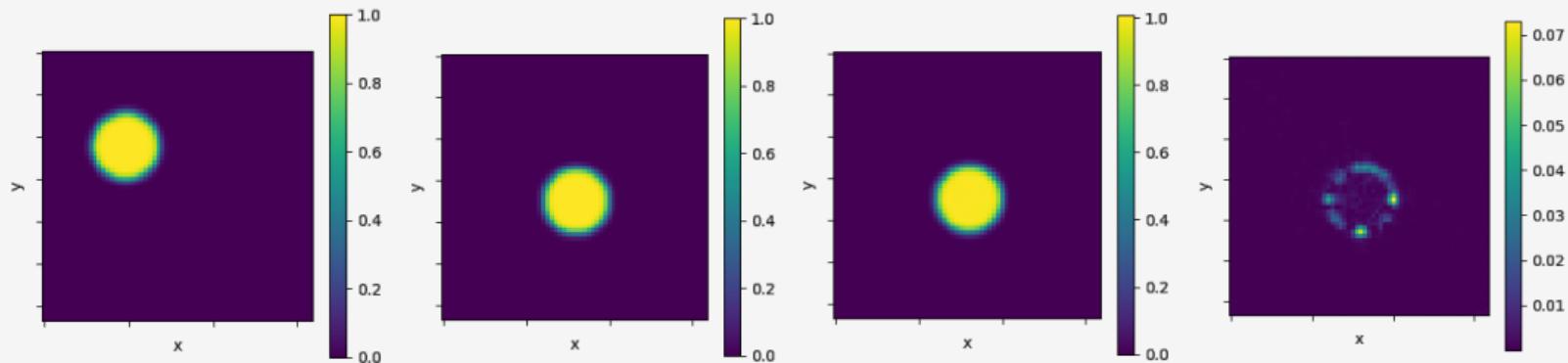
La FNO ha 21M parametri, tempo di training di 50 minuti, errore di test relativo in norma L^2 pari a 0.007.

Transport discontinuous equation

4 Operatori neurali

$$u_t + v \cdot \nabla u = 0 \quad \text{in } \Omega \times (0, T), \quad u_0(x, y) = f(x, y),$$

con f funzione discontinua. Si fissa $v = (0.2, 0.2)$ e si approssima l'operatore $\mathcal{G} : f \mapsto u_{T=1}$.



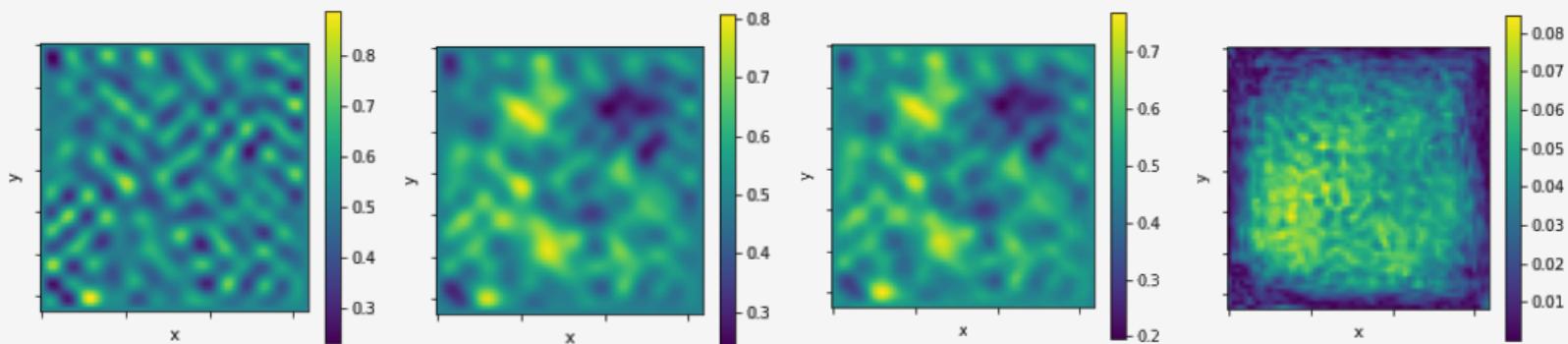
La FNO ha 16M parametri, tempo di training di 21 ore, errore di test relativo in norma L^2 pari a 0.013.

Poisson Equation

4 Operatori neurali

$$-\Delta u = f, \quad \text{in } \Omega = [0, 1]^2, \quad u|_{\partial\Omega} = 0.$$

Si approssima l'operatore $\mathcal{G} : f \mapsto u$.



La FNO ha 1M parametri, tempo di training di 20 minuti, errore di test relativo in norma L^2 pari a 0.04.

Navier-Stokes Equation

4 Operatori neurali

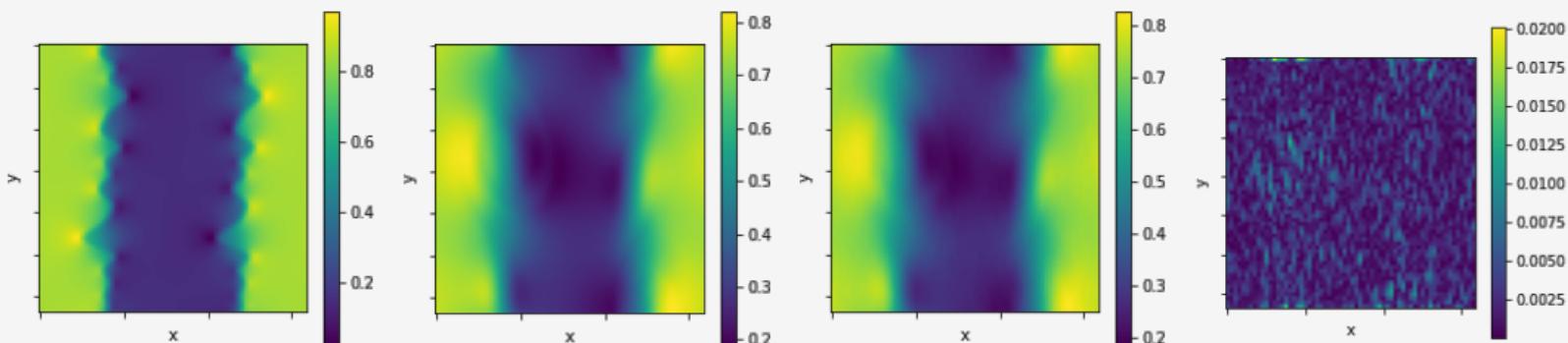
Massimiliano

Ghiotto



$$u_t + (u \cdot \nabla) u + \nabla p = \nu \Delta u, \quad \operatorname{div} u = 0, \quad u_0(x, y) = f(x, y).$$

Dove $\nu = 4 \cdot 10^{-4}$ e si approssima l'operatore soluzione $\mathcal{G} : f \mapsto u_{T=1}$.



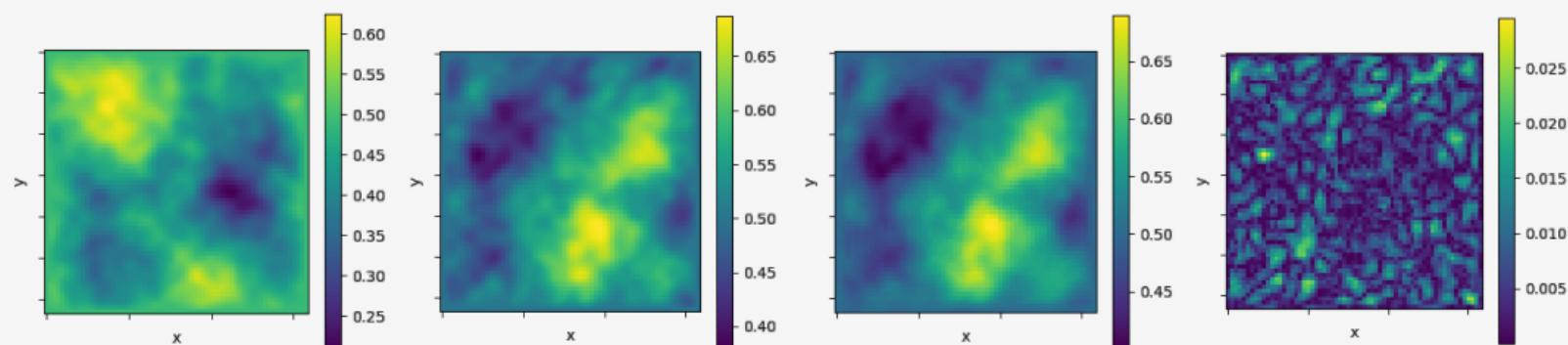
La FNO ha 32,8M parametri, tempo di training di 30 minuti, errore di test relativo in norma L^2 pari a 0.05.

Wave Equation

4 Operatori neurali

$$u_{tt} - c^2 \Delta u = 0, \quad \text{in } \Omega \times (0, T), \quad u_0(x, y) = f(x, y).$$

Dove $c = 0.1$ e si approssima l'operatore soluzione $\mathcal{G} : f \mapsto u_{T=5}$.



La FNO ha 4,9M parametri, tempo di training di 7 ore, errore di test relativo in norma L^2 pari a 0.014.

Allen-Cahn equation

4 Operatori neurali

Massimiliano

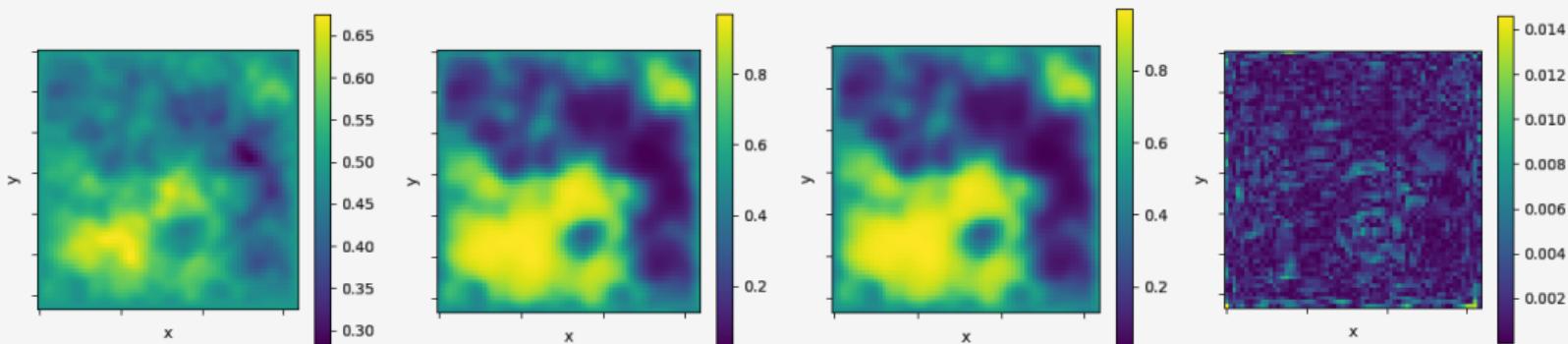
Ghiotto



UNIVERSITÀ
DI PAVIA

$$u_t = \Delta u - \varepsilon^2 u(u^2 - 1).$$

Dove $\varepsilon = 220$ e si approssima l'operatore soluzione $\mathcal{G} : f \mapsto u_T$ con $T = 0.0002$.



La FNO ha 4,9M parametri, tempo di training di 2 ore, errore di test relativo in norma L^2 pari a 0.004.

Grazie per

