



CAFFÈ BELTRAMI

UNIVERSITÀ DEGLI STUDI DI PAVIA

Caffè Beltrami presenta...

# Massimiliano Ghiotto



```
ro il vettore u
= [3 2; 1 3; 2 1]; X mi serve per costruire i vettori lato di un tr
ngth(P); X numero di vertici, cioè di funzioni base
alloc(n, n, 7^n); % per creare la matrice in modo sparso
ir0s(n, 1);
ir0s(n, 1);
= 1:length(T)
rea_t = polyarea( P(T(t,:));
for t = 1:n
    P(T(t,Index(1,1),:); % sono vett
    F con la formu
    + F(T(t,1)) + f( sum(P(T(t,:),:), 1)./3 );
    )
    = P(T(t,Index(j, 1),:); % sono vett
    + lato_1*lato_j)/(a^2)
```

Massimiliano Ghiotto è dottorando del XXXIX ciclo presso il Dipartimento di Matematica dell'Università di Pavia. La sua ricerca si concentra nell'approssimazione di PDEs tramite operatori neurali, con particolare attenzione allo sviluppo di operatori neurali applicabili a problemi definiti su geometrie multipatch.



CAFFÈ BELTRAMI

UNIVERSITÀ DEGLI STUDI DI PAVIA

# Operatori neurali

Deep Learning per matematici

Deep Learning

Applicazioni interessanti

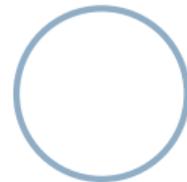
Risultati teorici

Operatori neurali



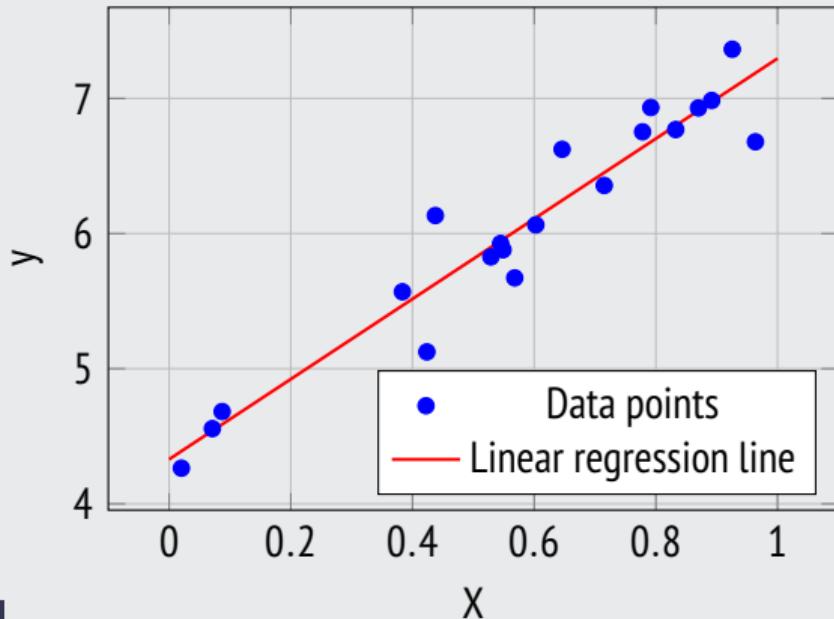
CAFFÈ BELTRAMI

# Deep Learning



# Regessione lineare

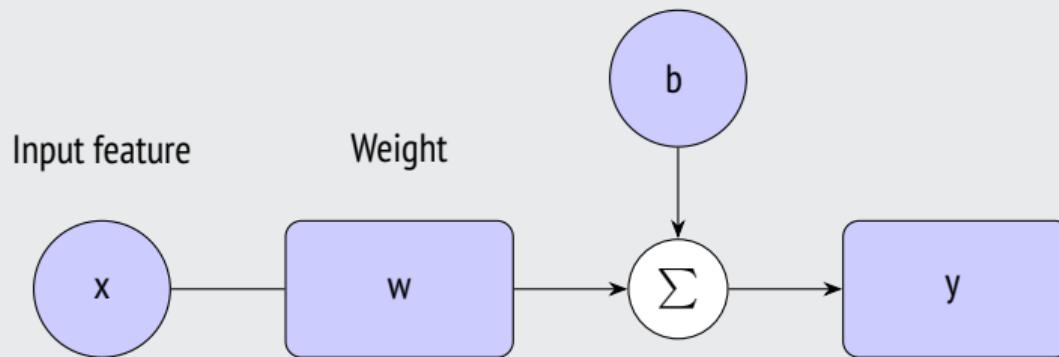
## Linear Regression Example



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.



# Regessione lineare



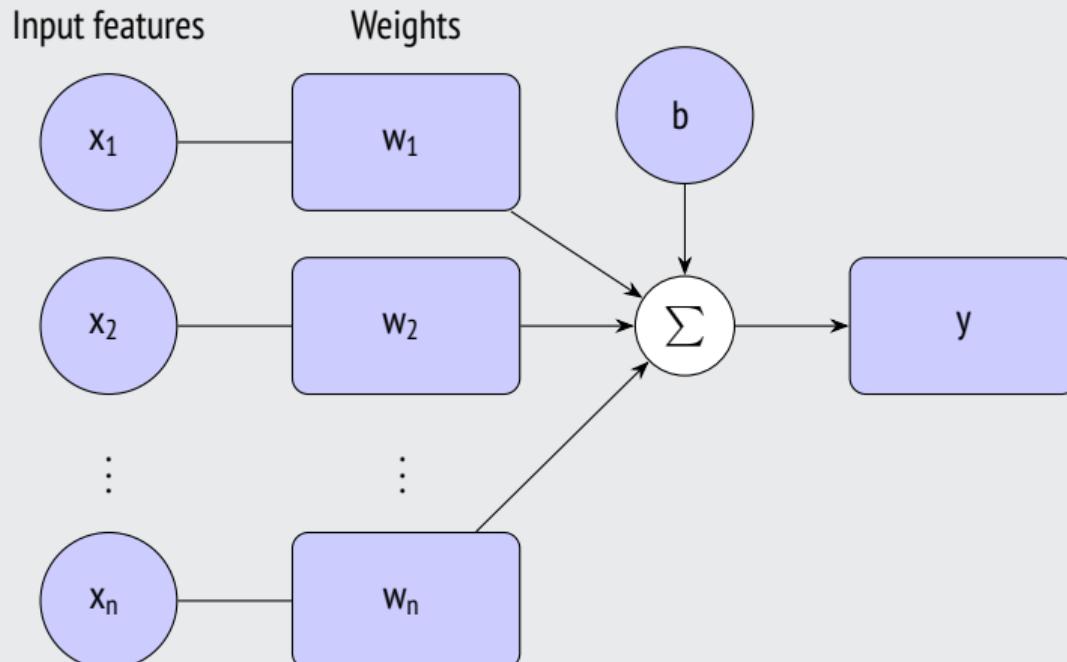
$$y = w \cdot x + b$$

- ☕  $x \in \mathbb{R}$  input,
- ☕  $w \in \mathbb{R}$  weight,
- ☕  $b \in \mathbb{R}$  bias.



CAFFÈ BELTRAMI

# Regressione lineare



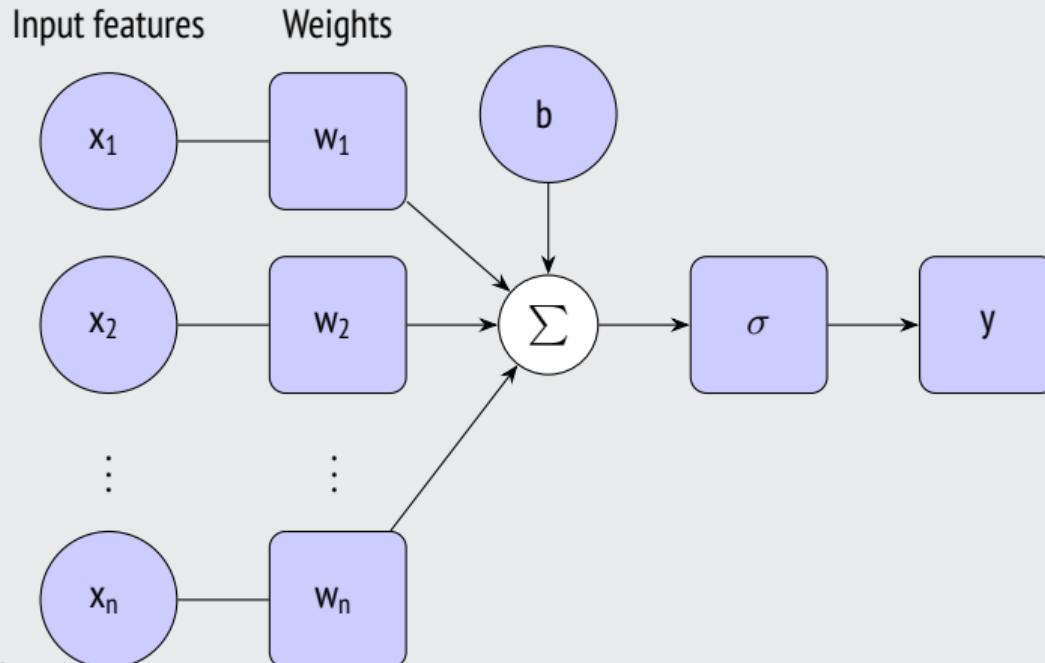
$$y = \mathbf{w} \cdot \mathbf{x} + b$$

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



CAFFÈ BELTRAMI

# Percettrone



Problema di classificazione.

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

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



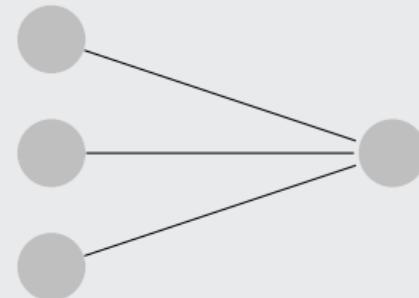
Chiamiamo  $\hat{y}(\mathbf{x})$  la label corretta associata ad  $\mathbf{x}$ ,  
l'algoritmo per trovare  $\mathbf{w}$  e  $b$  è:

- ─ 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.

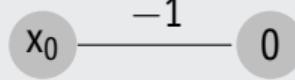


Figura: Negazione

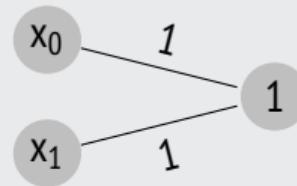


Figura: Congiunzione

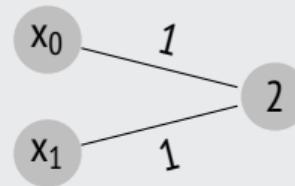


Figura: Disgiunzione



CAFFÈ BELTRAMI

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.

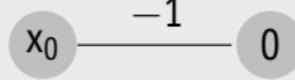


Figura: Negazione

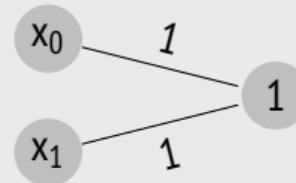


Figura: Congiunzione

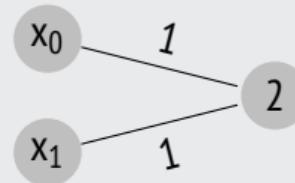


Figura: Disgiunzione

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



CAFFÈ BELTRAMI

Indicando lo XOR con  $\times$  vale che

$$x_1 \times x_2 = (x_1 \vee x_2) \wedge \neg(x_1 \wedge x_2) = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2).$$

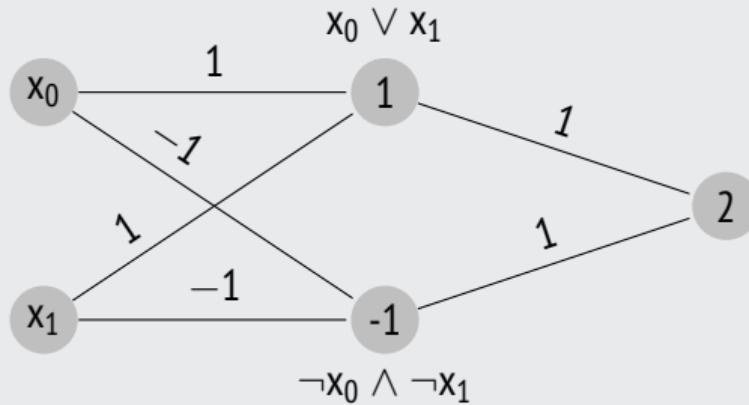


CAFFÈ BELTRAMI

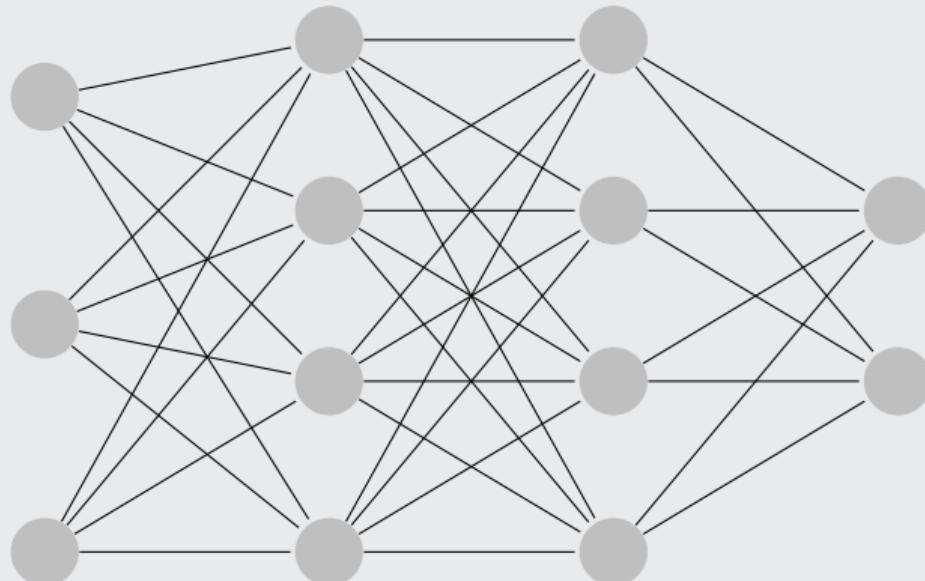
Indicando lo XOR con  $\times$  vale che

$$x_1 \times x_2 = (x_1 \vee x_2) \wedge \neg(x_1 \wedge x_2) = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2).$$

Dunque lo possiamo rappresentare con



## Multi-layer Perceptron

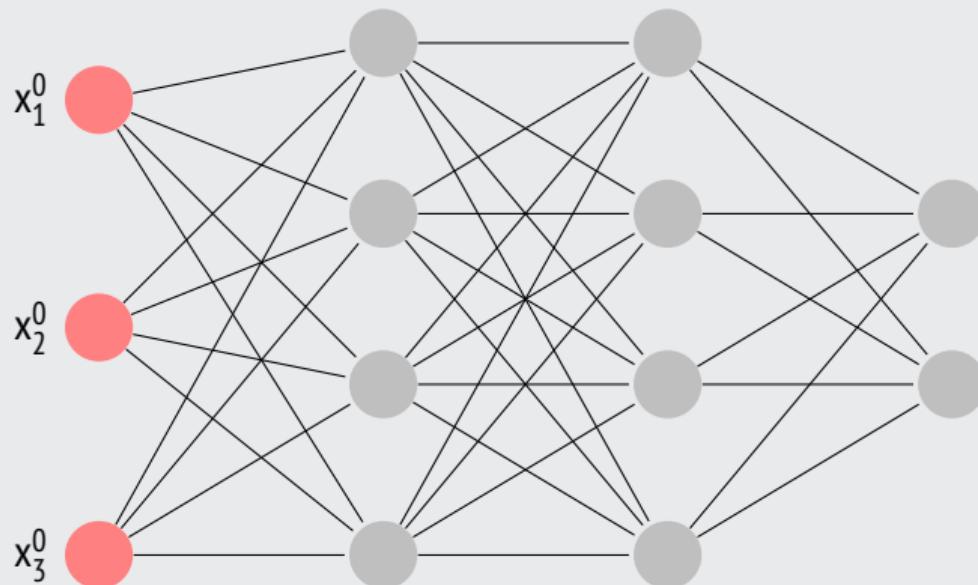


Esempio di architettura di un Multi-layer Perceptron con 3 neuroni di input, 4 neuroni nel primo hidden layer, 4 neuroni nel secondo hidden layer e 2 neuroni di output.



CAFFÈ BELTRAMI

# Multi-layer Perceptron

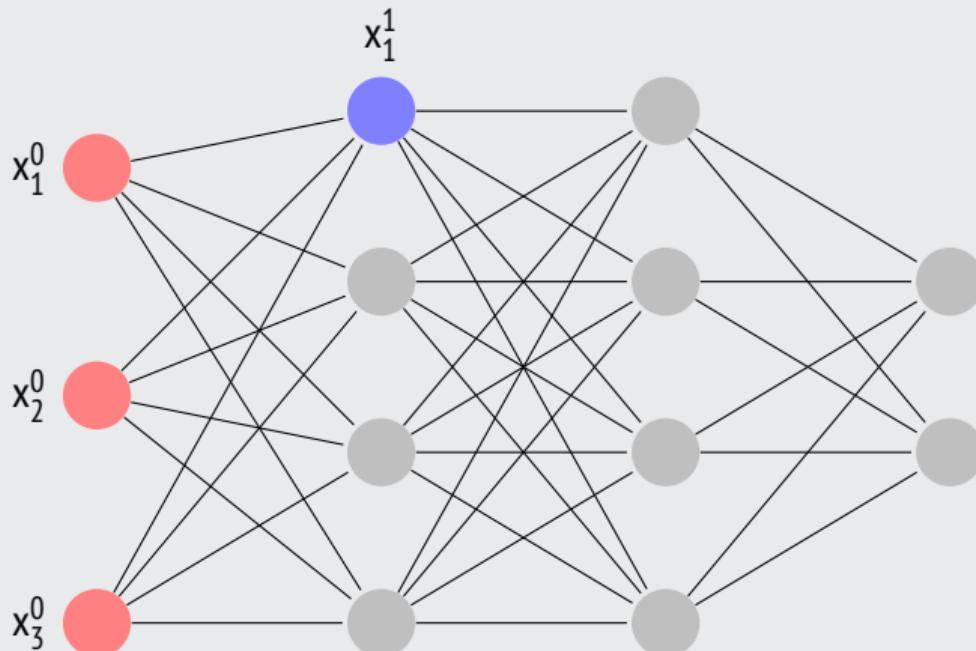


Layer di input con 3 neuroni.



CAFFÈ BELTRAMI

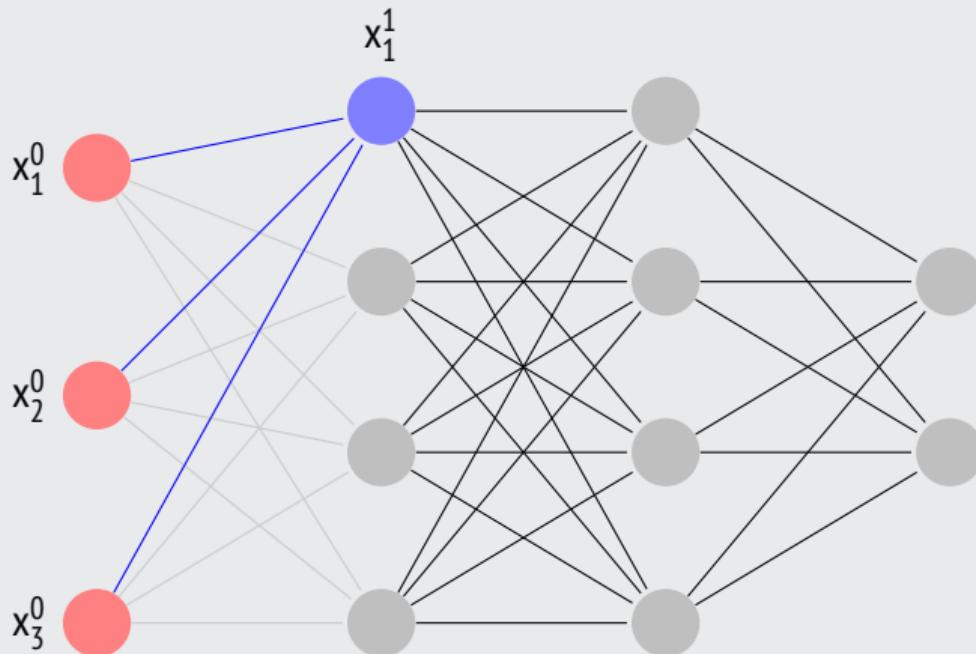
# Multi-layer Perceptron



Vogliamo definire il valore del neurone  $x_1^1$ .



# Multi-layer Perceptron

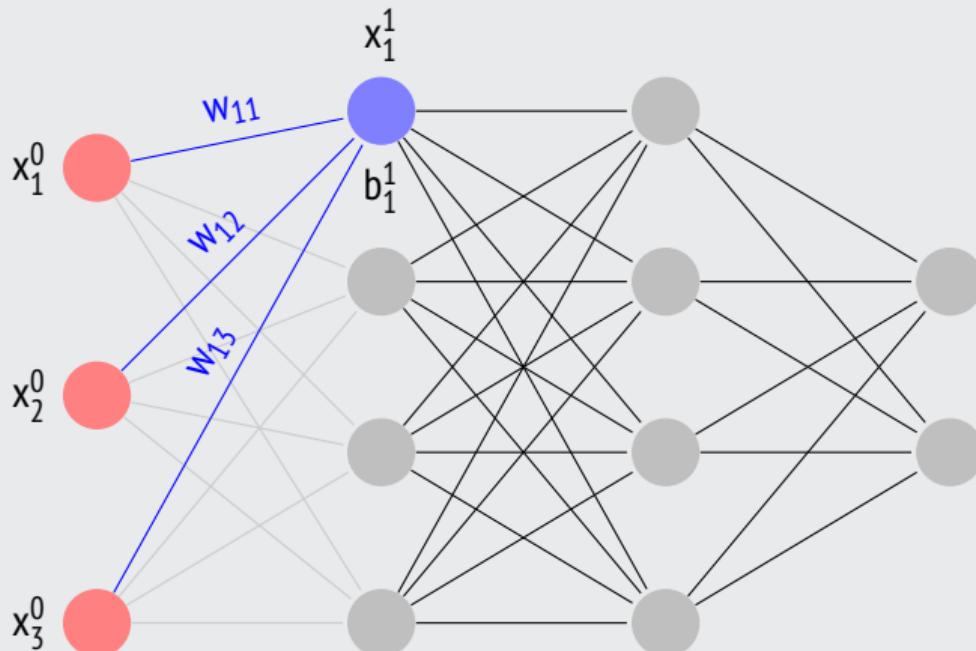


Evidenziamo le connesioni con i neuroni nel layer precedente.



CAFFÈ BELTRAMI

# Multi-layer Perceptron

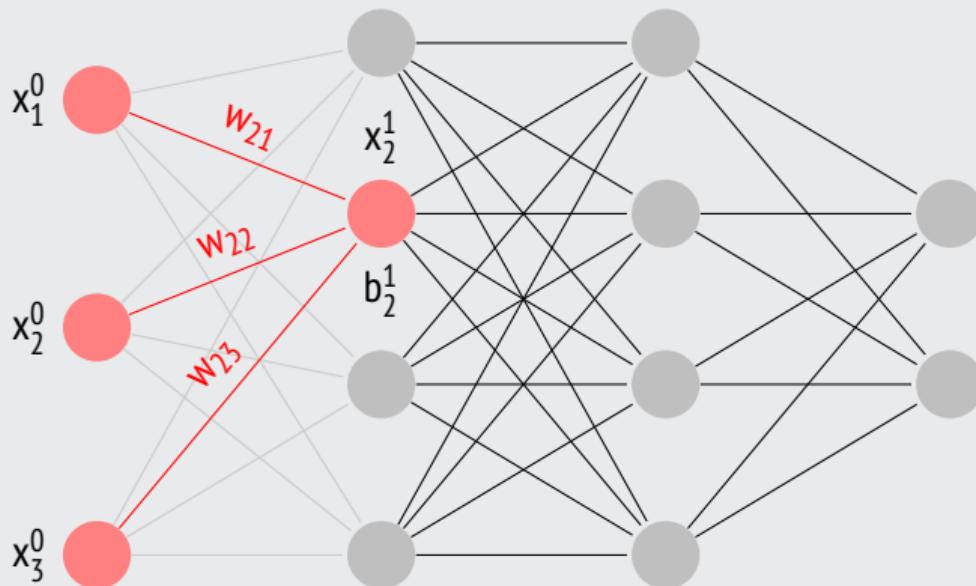


$$x_1^1 = \sigma \left( b_1^1 + \sum_{i=1}^3 w_{1i} x_i^0 \right)$$

Dove  $\sigma$  è una funzione di attivazione non lineare.



# Multi-layer Perceptron

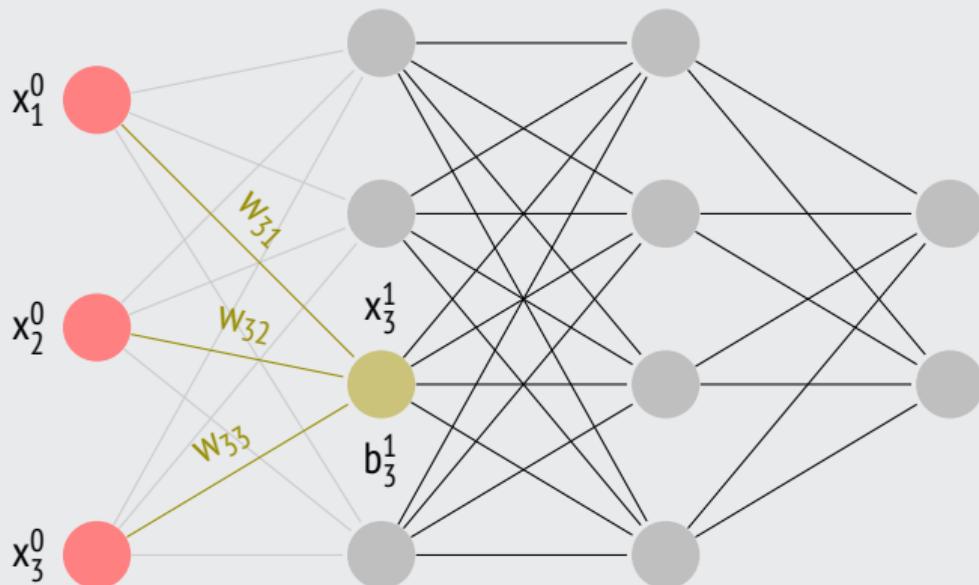


$$x_1^1 = \sigma \left( b_1^1 + \sum_{i=1}^3 w_{1i} x_i^0 \right)$$

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



# Multi-layer Perceptron



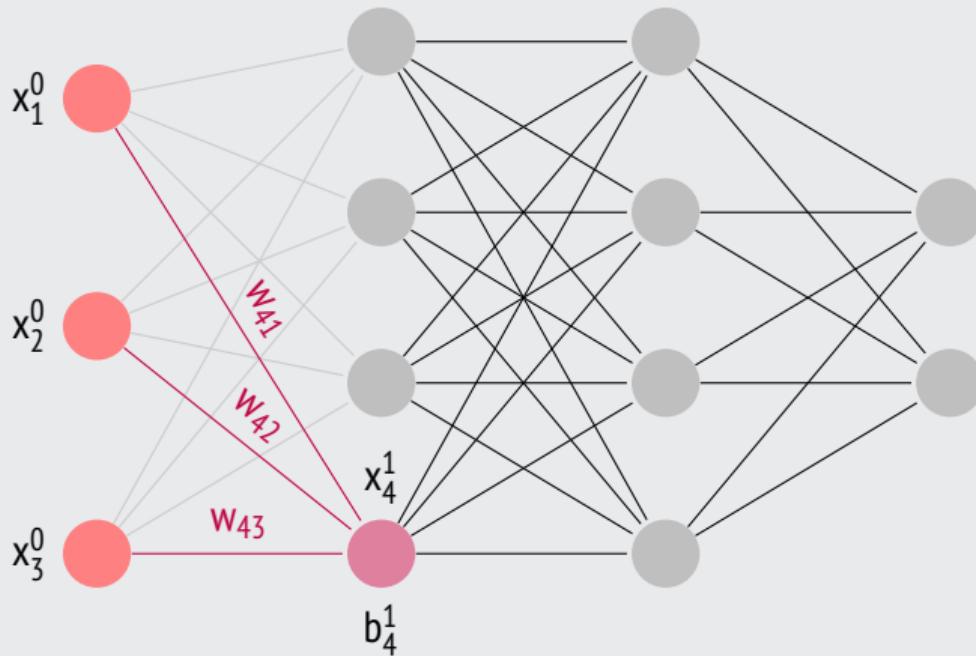
$$x_1^1 = \sigma \left( b_1^1 + \sum_{i=1}^3 w_{1i} x_i^0 \right)$$

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

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



# Multi-layer Perceptron



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

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

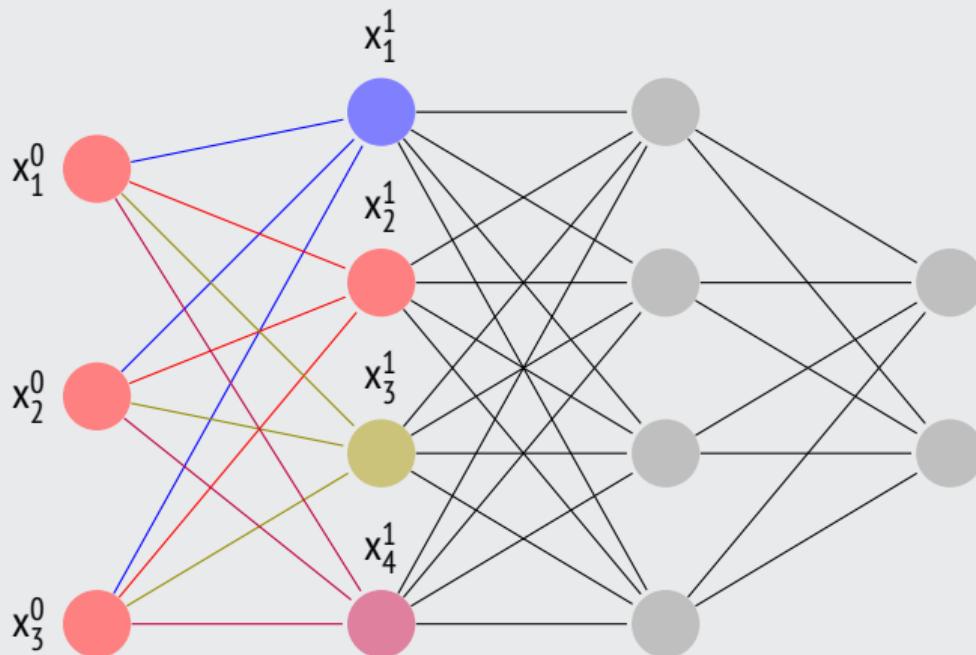
$$x_3^1 = \sigma \left( b_3^1 + \sum_{i=1}^4 w_{3i} x_i^0 \right)$$

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



CAFFÈ BELTRAMI

# Multi-layer Perceptron



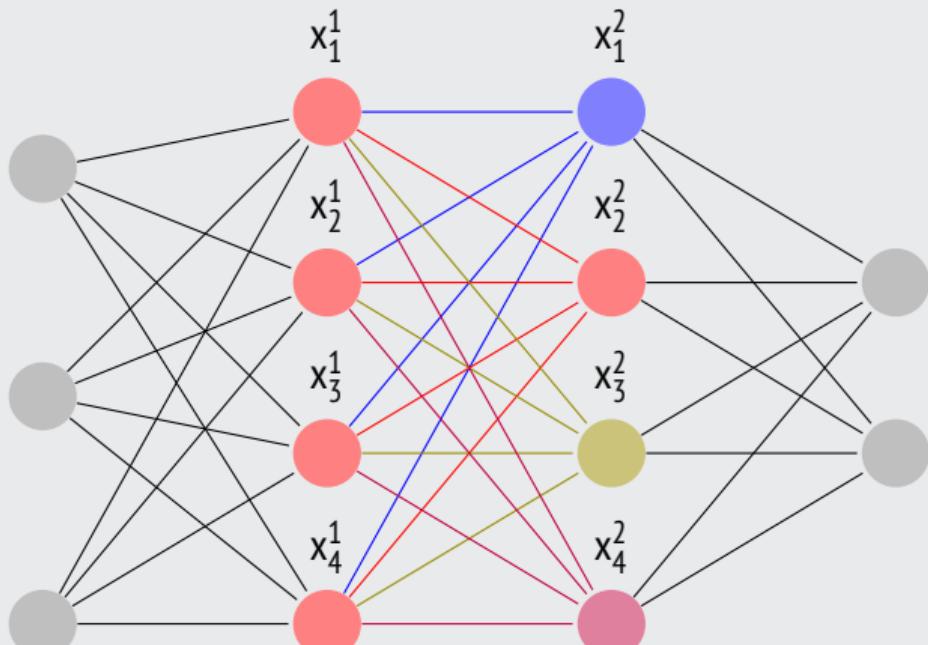
Dato  $\mathbf{x}^0 \in \mathbb{R}^3$

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

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



# Multi-layer Perceptron



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

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

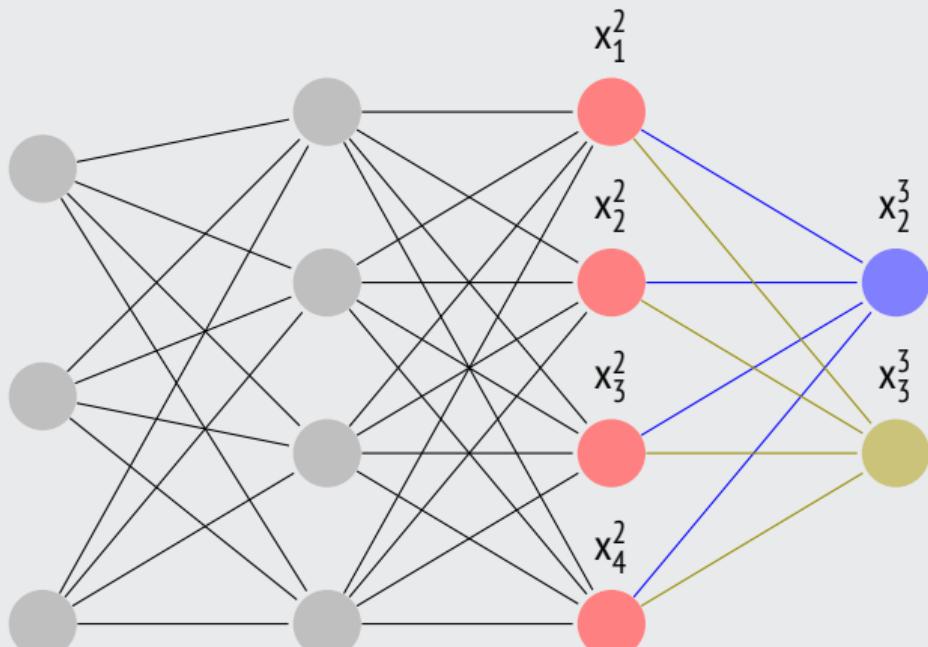
$$\mathbf{x}^2 := \sigma(\mathbf{W}_2 \mathbf{x}^1 + \mathbf{b}_2) \in \mathbb{R}^4$$

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



CAFFÈ BELTRAMI

# Multi-layer Perceptron



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  $\mathbf{b} \in \mathbb{R}^2$ .



## Definizione

Multi-layer Perceptron 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

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

$$\vdots$$

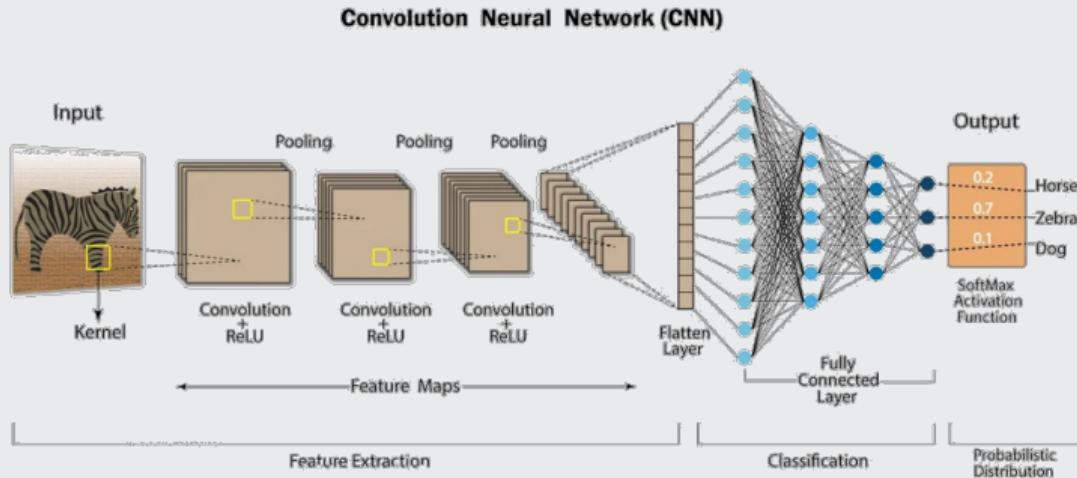
$$\mathbf{x}^L := \sigma \left( \mathbf{W}_L \mathbf{x}^{L-1} + \mathbf{b}_L \right) \in \mathbb{R}^{d_L}$$

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  $\sigma$  è una funzione di attivazione non lineare applicata componente per componente.





# Convolutional Neural Network (CNN)

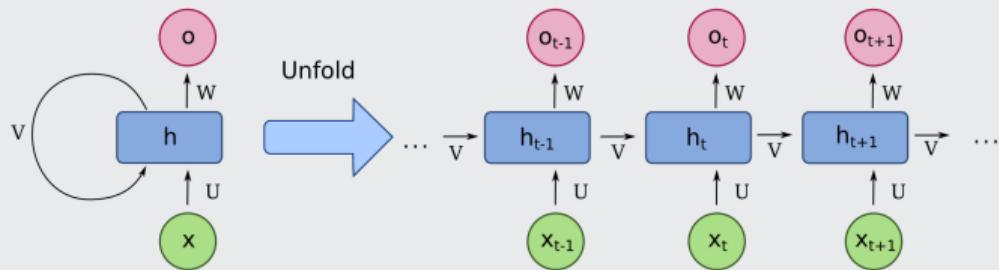


Reti neurali convoluzionali per l'applicazione alle immagini.



CAFFÈ BELTRAMI

# Recurrent Neural Network (RNN)



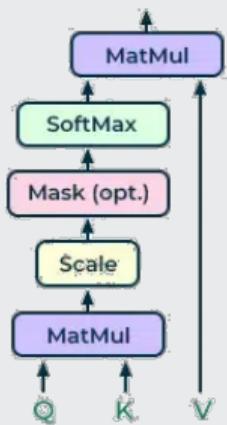
Reti neurali ricorrenti per l'applicazione al linguaggio naturale ed ai segnali, o più in generale alle quantità dipendenti dal tempo.



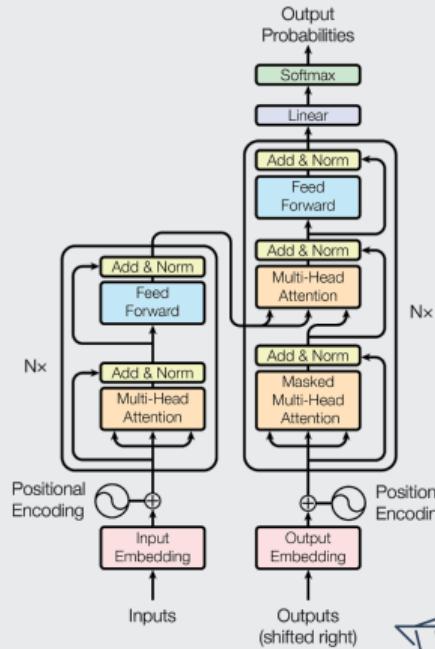
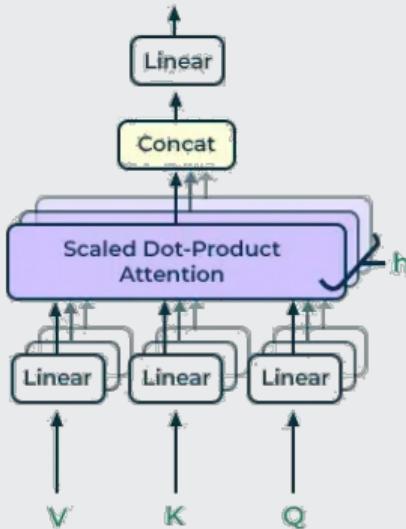
CAFFÈ BELTRAMI

# Transformer

## Scaled Dot-Product Attention

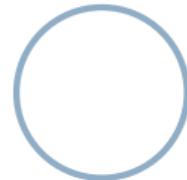


## Multi-Head Attention



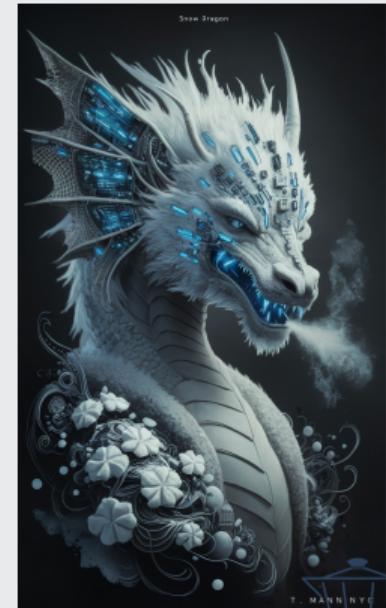
CAFFÈ BELTRAMI

# Applicazioni interessanti



# Applicazioni alle immagini

Siti per generazione d'immagini: DALL-E3  DALL-E, Midjourney  e Ideogram .



CAFFÈ BELTRAMI

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.



CAFFÈ BELTRAMI

- ☕ 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.

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.



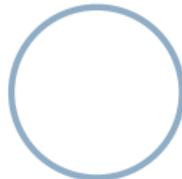
- ☕ Modelli per la generazione di testo come GPT-4  GPT-4 o Copilot .
- ☕ Esistono modelli open-source come LLaMA-3  scaricabile gratuitamente dal loro sito.

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.



# Risultati teorici



## Definizione

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

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



CAFFÈ BELTRAMI

## Definizione

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

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

## Definizione

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



CAFFÈ BELTRAMI

## Definizione

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

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

## Definizione

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.



### Teorema

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

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



## Teorema

Sia  $\sigma$  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  $y \in K_2$ .



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

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



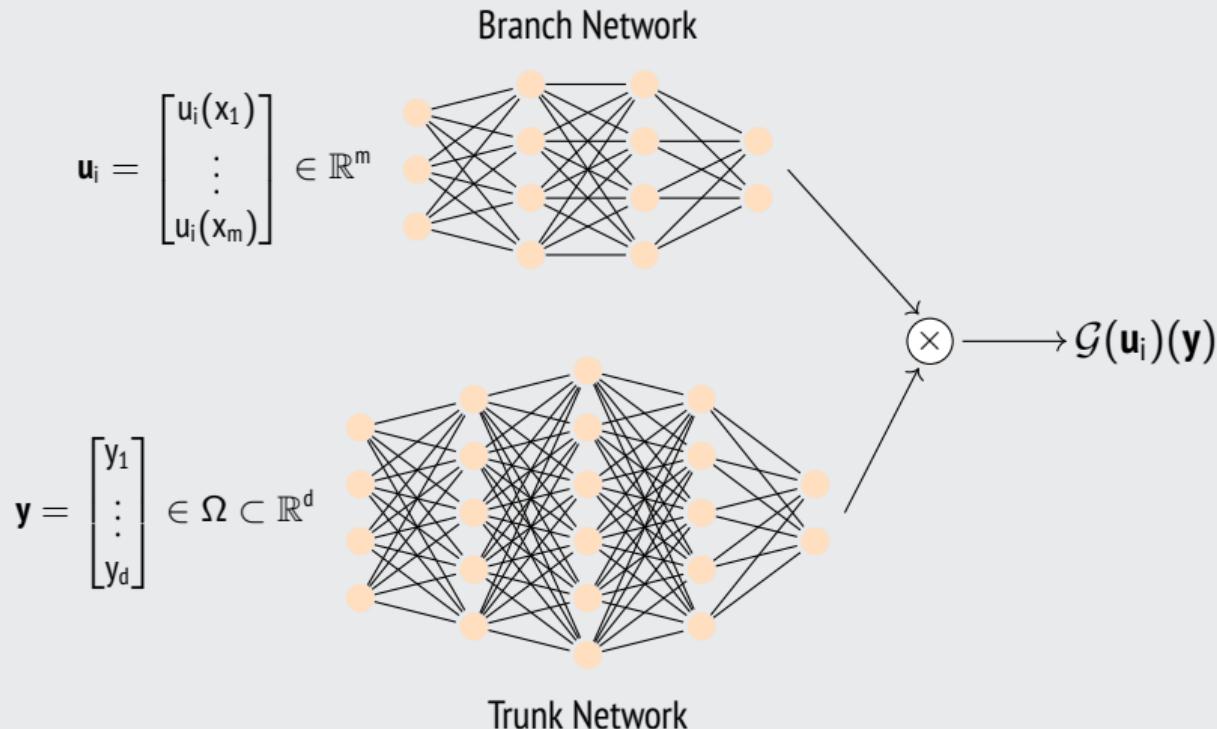
Abbiamo  $n, p, m \in \mathbb{N}$ ,  $c_i^k, \theta_i^k, \xi_{ij}^k, \zeta^k \in \mathbb{R}$ ,  $w^k \in \mathbb{R}^d$ ,  $x_j \in K_1$ ,  $y \in K_2 \subset \mathbb{R}^d$  e  $u \in V$  tali che

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

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  $p$ .

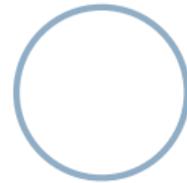


# Deep Operator Network

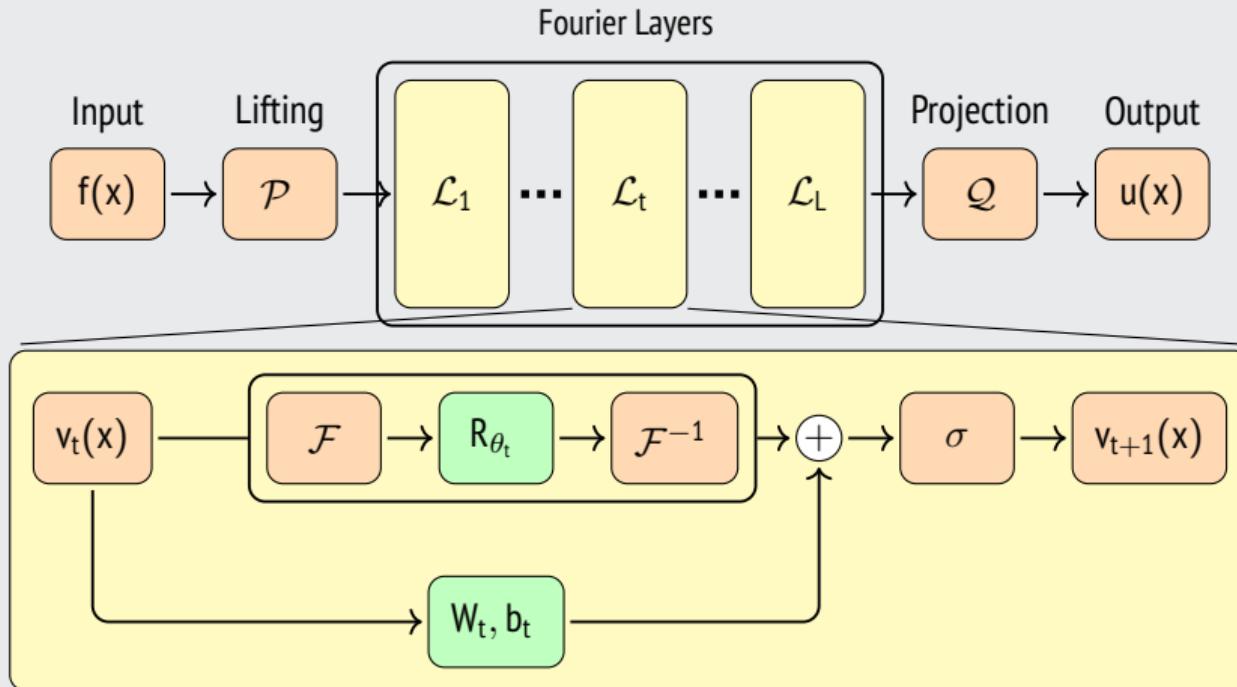


CAFFÈ BELTRAMI

# Operatori neurali



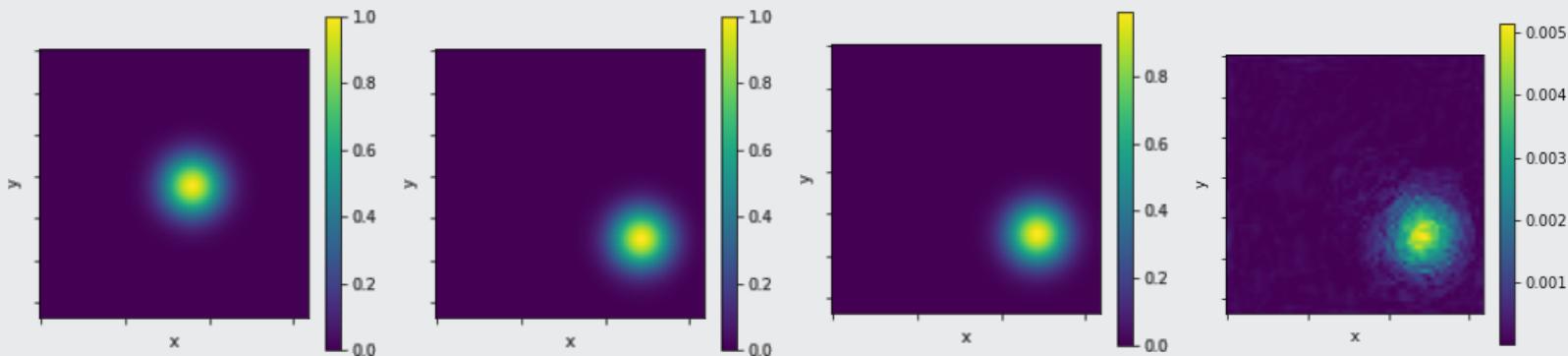
# Fourier Neural Operator



## Transport continuous equation

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

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



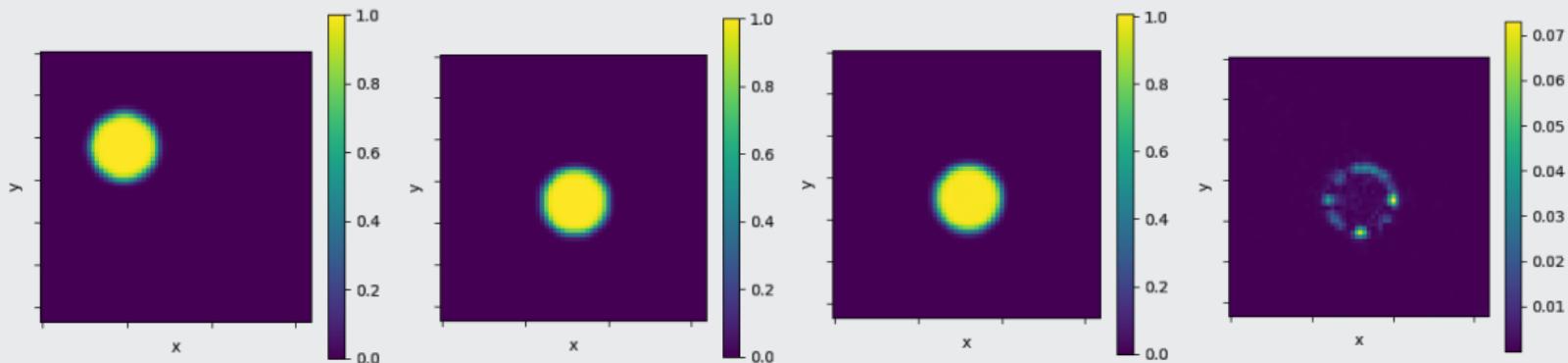
La FNO ha 21M parametri, tempo di training di 50 minuti, errore di test relativo in norma  $L^2$  pari a 0.007.



## Transport discontinuous equation

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

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



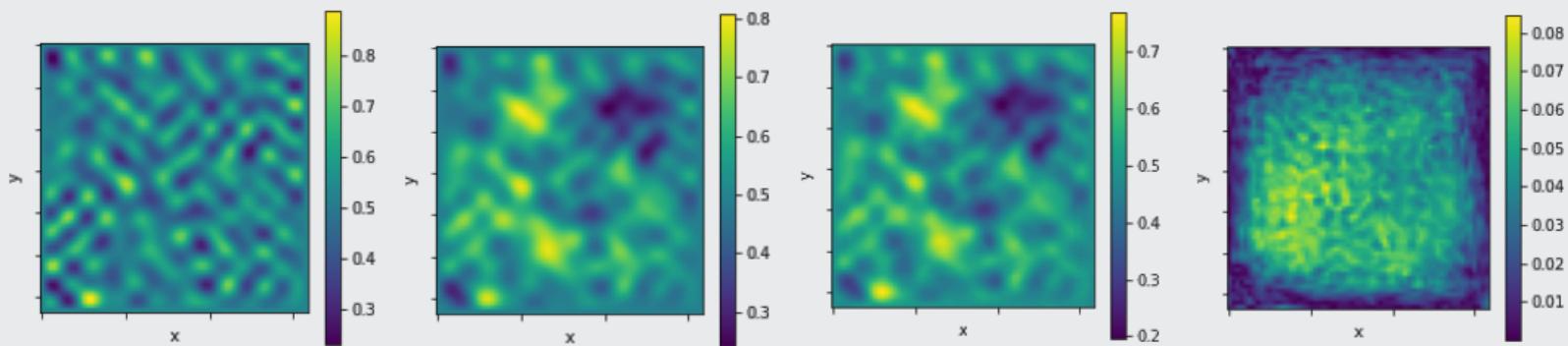
La FNO ha 16M parametri, tempo di training di 21 ore, errore di test relativo in norma  $L^2$  pari a 0.013.



# Poisson Equation

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

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



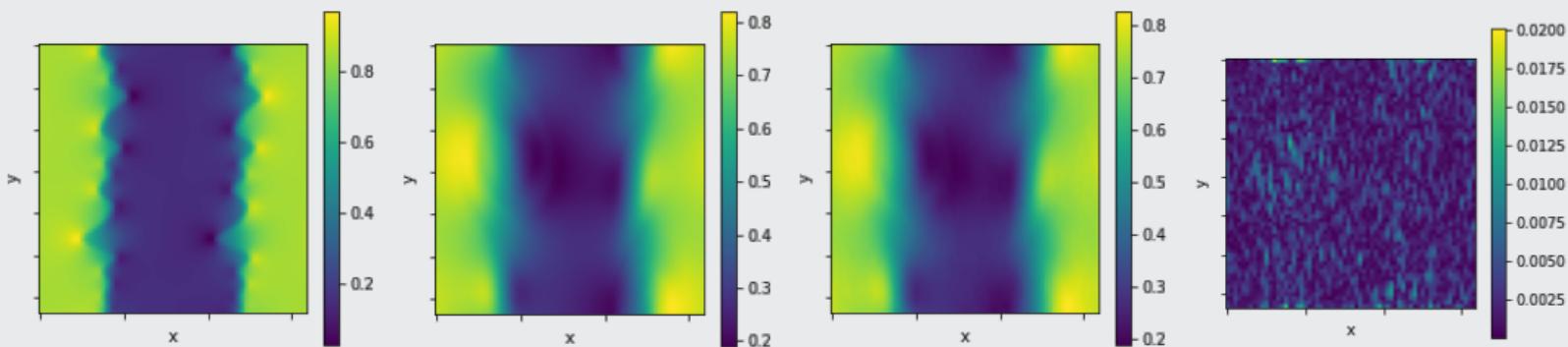
La FNO ha 1M parametri, tempo di training di 20 minuti, errore di test relativo in norma  $L^2$  pari a 0.04.



# Navier-Stokes Equation

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

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

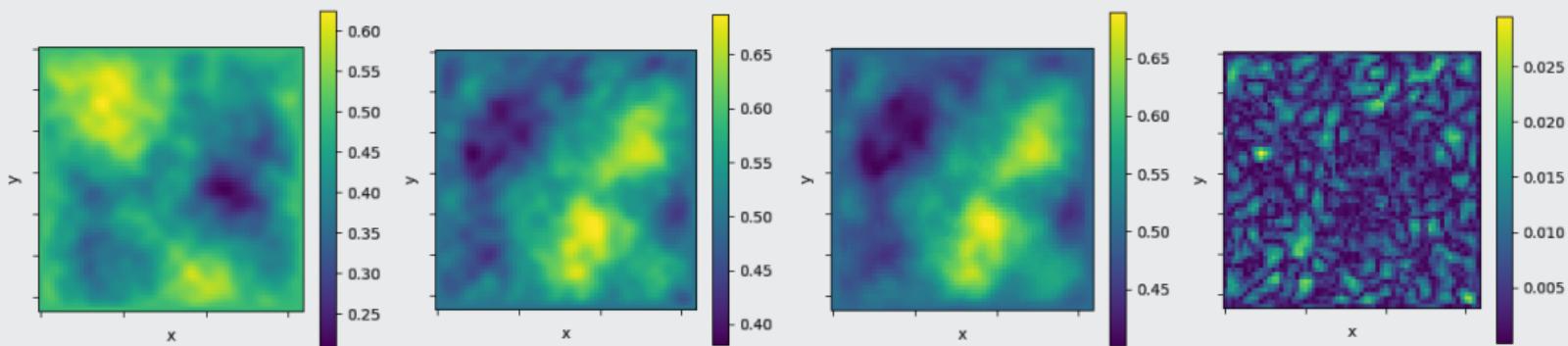


La FNO ha 32, 8M parametri, tempo di training di 30 minuti, errore di test relativo in norma  $L^2$  pari a 0.05.



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

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

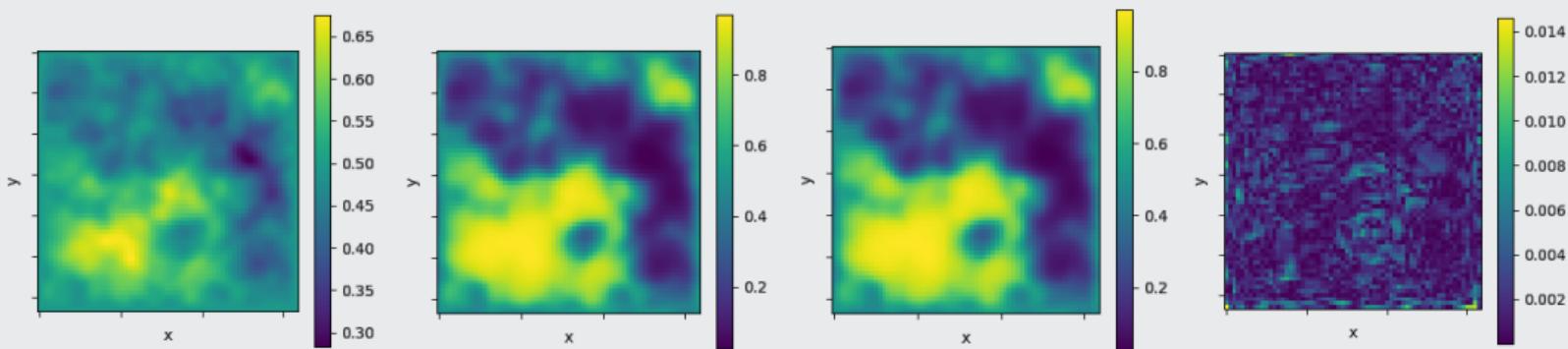


La FNO ha 4, 9M parametri, tempo di training di 7 ore, errore di test relativo in norma  $L^2$  pari a 0.014.



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

Dove  $\varepsilon = 220$  e si approssima l'operatore soluzione  $\mathcal{G} : f \rightarrow u_T$  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.

