Progettazione, Implementazione e Valutazione Sperimentale di una Rete Neurale Convoluzionale Quantistica Ibrida per la Classificazione di Radiografie Toraciche nel Dataset `PneumoniaMNIST`

Antonio Mosca Leonardo Tomei

2 luglio 2025

Sommario

La diagnostica assistita su radiografie toraciche rappresenta un banco di prova privilegiato per valutare i benefici di modelli ibridi *quantum–classical* su hardware NISQ. Presentiamo una *Quantum Convolutional Neural Network* (QCNN) estremamente compatta, progettata per la classificazione binaria di polmonite da radiografie a bassa risoluzione (RGB 28×28) provenienti dalla sotto-collezione `PneumoniaMNIST` del benchmark `MedMNIST`. L'architettura ibrida proposta è articolata in una pipeline composta da un leggero preprocessing classico, un layer di *angle encoding*, una singola convoluzione quantistica a tre soli parametri variazionali seguita da pooling quantistico, e un classificatore lineare. Dopo 10 epoche la QCNN raggiunge un'**accuracy del 92.0 %** sul test set, superando controparti CNN classiche molto più profonde (ResNet-18/50, $\approx$ 85 %). L'esigua profondità del circuito e il numero ridotto di parametri rendono l'architettura compatibile con gli attuali dispositivi *Noisy Intermediate-Scale Quantum*, aprendo la strada a futuri esperimenti su hardware reale.

Indice

1 Introduzione **3**

2 Inquadramento del problema **4**
2.1 Dataset `MedMNIST` 4
2.2 Task di classificazione 4

3 Contesto teorico della QCNN **5**
3.1 Codifica dei dati 5
3.2 Convoluzione quantistica locale 5
3.3 Pooling quantistico 5
3.4 Addestramento tramite circuiti variazionali 6
3.5 Profondità e compatibilità NISQ 6

4 Descrizione dell'architettura **7**
4.1 Pre-processing delle immagini 7
4.2 Encoding classico–quantistico 8
4.3 Convoluzione Quantistica Variazionale 9
4.4 Quantum Pooling 10
4.5 Rete classica finale 11
4.6 Motivazioni progettuali 12

5 Risultati sperimentali **13**
5.1 Configurazione ottimale e confronto con le baseline 13

6 Conclusioni e prospettive **16**

A Sensibilità agli iper-parametri **17**

1 Introduzione

Negli ultimi dieci anni le *Convolutional Neural Network* (CNN) hanno rivoluzionato la computer vision, arrivandone a costituire lo stato dell'arte. Il successo delle CNN si fonda su tre proprietà chiave:

- (i) **località** del filtro, che sfrutta la correlazione spaziale delle immagini;
- (ii) **parameter sharing**, che riduce drasticamente il numero di pesi rispetto alle reti fully–connected;
- (iii) **composizionalità gerarchica**, che consente di apprendere feature da basso ad alto livello.

Tuttavia, architetture sempre più profonde (decine di milioni di parametri per modelli come ResNet-50) comportano costi di calcolo e di storage elevati e richiedono grandi quantità di dati etichettati, non sempre disponibili in ambito clinico.
Parallelamente, l'avvento dei dispositivi *Noisy Intermediate-Scale Quantum* (NISQ) ha aperto la strada a modelli ibridi *quantum–classical* che mirano a sfruttare la capacità di sovrapposizione ed entanglement dei qubit per ottenere una rappresentazione potenzialmente più compatta ed espressiva rispetto alle reti classiche. Tra le architetture proposte spiccano le *Quantum Convolutional Neural Network* (QCNN), che mappano la nozione di convoluzione su circuiti quantistici a bassa profondità mediante una sequenza di operazioni locali seguite da misure di *quantum pooling*. In analogia con le CNN classiche, le QCNN combinano la **località** delle operazioni su patch con la **non-località** fornita dall'entanglement, riducendo al contempo il numero di parametri addestrabili.
Nel presente studio indaghiamo l'impiego di una QCNN per la classificazione automatica di polmonite in radiografie toraciche a bassa risoluzione. La diagnosi di polmonite da radiografia è un caso d'uso clinicamente rilevante e, grazie alla sua natura binaria, si presta a valutare in modo controllato i vantaggi (o i limiti) di modelli quantistici. Utilizziamo a tale scopo la sotto-collezione `PneumoniaMNIST` del benchmark `MedMNIST`, poiché fornisce un set *lightweight* con label validate da esperti, pur mantenendo le caratteristiche statistiche di un problema reale.
Il contributo principale del lavoro è la progettazione e l'analisi di una QCNN *minimalista* —con appena tre parametri variazionali— che, in simulazione, raggiunge prestazioni competitive con CNN classiche molto più grandi, dimostrando che anche circuiti a bassa profondità possono essere efficaci su dati biomedici. L'approccio proposto costituisce un passo verso l'esecuzione di modelli di *medical imaging* su hardware quantistico reale, dove la profondità di circuito e il numero di parametri sono vincoli critici.

2 Inquadramento del problema

2.1 Dataset `MedMNIST`

`MedMNIST` è una raccolta di dataset biomedici pre-processati in formato 28×28 pensata per benchmarking rapido e comparabile. La nostra analisi si concentra su `PneumoniaMNIST`, che contiene **5856** radiografie toraciche RGB (28×28 px) annotate da specialisti come `pneumonia` o `normal`. Il dataset è già suddiviso negli split ufficiali:

Split	# Immagini	Percentuale
Addestramento	4708	80.3 %
Validazione	524	8.9 %
Test	624	10.8 %

Il dataset, di dimensione ridotta, si presta all'addestramento rapido di modelli sia classici sia quantistici.

2.2 Task di classificazione

Il problema è formulato come classificazione binaria: dato un input $x \in \mathbb{R}^{3\times28\times28}$, prevedere $y \in \{0, 1\}$, dove 1 denota presenza di polmonite. Le metriche adottate sono:

- **Accuratezza** (ACC) — quota di predizioni corrette sul totale;
- **F1-score macro-media** — media armonica di precision e recall, calcolata separatamente per ciascuna classe e poi mediata, utile in presenza di modesto imbalance;

Alla luce di questi elementi, la sfida principale consiste nel verificare se un modello quantistico con un numero di parametri ordini di grandezza inferiore possa eguagliare o superare CNN classiche consolidate, pur rispettando i limiti di rumorosità e profondità tipici dell'hardware NISQ.

3 Contesto teorico della QCNN

Le *Quantum Convolutional Neural Network* (QCNN) sono una trasposizione quantistica dei principi che hanno reso le CNN classiche così efficaci nella visione artificiale: località dei filtri, condivisione dei parametri e pooling gerarchico. In ambito quantistico tali concetti vengono riformulati tramite circuiti a bassa profondità che operano su *registri locali di qubit* e tramite misure intermedie che veicolano l'informazione verso livelli sempre più compressi. Nel seguito forniamo i fondamenti teorici indispensabili a comprendere la nostra implementazione, rimandando i dettagli circuitali alla sezione successiva.

3.1 Codifica dei dati

Poiché un dispositivo quantistico nativo elabora stati in uno spazio di Hilbert $\mathcal{H}_{2^n}$, una fase preliminare di **data encoding** implementa i dati classici in uno stato quantistico. Fra le varie possibilità proposte dalla letteratura (basis, amplitude, Hamiltonian, ...), l'*angle encoding* è particolarmente adatto a immagini a bassa risoluzione: un valore di pixel normalizzato $x \in [0, 1]$ viene codificato come rotazione monoclasse $|0\rangle \mapsto R_Y(\pi x)\,|0\rangle$.
In tal modo la profondità circuitale cresce linearmente con il numero di feature invece che esponenzialmente.

3.2 Convoluzione quantistica locale

Sia $|\psi\rangle$ lo stato prodotto dal layer di encoding su n qubit. La **quantum convolution** opera su una sotto–regione di k qubit (*patch*) applicando un operatore unitario con dipendenza parametrica:

$$U_{\boldsymbol{\theta}}^{\text{conv}} \;\in\; \mathrm{U}\big(2^k\big)$$

Nel paradigma da noi scelto, tale operatore viene fattorizzato in due blocchi fondamentali:

- *Rotazioni a singolo qubit* $R_y(\theta_i) \in \mathrm{U}(2)$, che immettono i parametri variabili;
- *Porte di entangling fisse* (nel nostro caso `CNOT`) che correlano i qubit all'interno della patch.

Il medesimo set di angoli $\boldsymbol{\theta}$ viene **condiviso da tutte le patch**, analogamente al kernel sharing delle CNN classiche. Questo accorgimento riduce i parametri addestrabili da $\mathcal{O}(N_{\text{patch}})$ a $\mathcal{O}(1)$, dove N_{patch} è il numero complessivo di patch su cui si applica $U_{\boldsymbol{\theta}}^{\text{conv}}$.

3.3 Pooling quantistico

Il **quantum pooling** sostituisce le operazioni di sottocampionamento classiche (max, average) con una *misura parziale* su un sotto-insieme di qubit, seguita da post-processing condizionato sul registro classico.
Formalmente, sia $\mathcal{M}$ l'operatore POVM (positive operator-valued measure) che misura $k-1$ qubit della patch. Condizionando la parte residua sul risultato m e applicando porte classiche (`X` nel nostro caso) controllate dalla bitstring classica $\mathbf{m}$ si rilocalizza l'informazione su un unico qubit $|\psi'\rangle \in \mathcal{H}_2$ mantenendo la coerenza necessaria per i livelli successivi.

3.4 Addestramento tramite circuiti variazionali

L'insieme dei parametri θ viene ottimizzato seguendo il paradigma dei *Variational Quantum Circuits* (VQC).
Dato un osservabile di costo $C(\theta) = \langle\psi(\theta)|\,\hat{O}\,|\psi(\theta)\rangle$, il gradiente analitico di C rispetto a ciascun θ_j è ottenibile mediante la **parameter-shift rule**:

$$\frac{\partial C}{\partial \theta_j} = \frac{1}{2}\left[C\left(\theta_j + \frac{\pi}{2}\right) - C\left(\theta_j - \frac{\pi}{2}\right)\right]$$

che richiede due sole valutazioni extra del circuito, indipendenti dal numero complessivo di qubit. In questo modo i gradienti dei blocchi quantistici e di quelli classici confluiscono in un unico vettore che ottimizzatori come Adam possono aggiornare, permettendo di addestrare l'intera rete end-to-end.

3.5 Profondità e compatibilità NISQ

La profondità D di una QCNN in generale scala con il numero di coppie *convoluzione–pooling* (L), ma rimane gestibile perché i qubit misurati a ogni pooling vengono scartati, riducendo via via lo spazio di Hilbert: $D \simeq L \cdot D_{\text{conv}}$ con $L = \mathcal{O}(\log_k n)$, dove k è la dimensione della patch. Nel nostro design minimalista ($k = 3$, $L = 1$) la parte quantistica di ciascuna patch richiede soltanto

$$D_{\text{conv}} = 6\ \texttt{C-X} + 12\ R_Y$$

quindi meno di dieci porte a due qubit in totale.
Un circuito così corto è pienamente compatibile con le attuali piattaforme *Noisy Intermediate-Scale Quantum*. In altri termini, la rete può essere portata su hardware reale senza ricorrere a tecniche di compressione ulteriore della profondità.

4 Descrizione dell'architettura

4.1 Pre-processing delle immagini

Il flusso dati che alimenta la parte quantistica parte da una pipeline completamente classica, implementata in `PyTorch` e contenuta in src/datasets/color2gray.py. La rete funziona in due passi, elencati qui di seguito.

1. Conversione RGB → grayscale. Utilizziamo una convoluzione 1×1 **trainabile** che comprende esattamente tre pesi, uno per ciascun canale di colore:

$$\mathrm{Gray}(u,v) \;=\; w_R\,R(u,v) + w_G\,G(u,v) + w_B\,B(u,v), \qquad (u,v) \in [1,28]^2$$

implementata dalla classe `Color2GrayNet`.
L'inizializzazione può avvenire in due modalità:

- **''`luma`''**: pesi fissati a $(w_R, w_G, w_B) = (0.299, 0.587, 0.114)$, corrispondenti alla formula di luminanza dello standard ITU-R BT. 601;
- **''`random`''**: inizializzazione Xavier uniforme; i coefficienti vengono poi appresi dal modello durante l'addestramento.

Nelle nostre esperienze l'opzione **''`luma`''** accelera la convergenza senza saturare i gradienti, per cui è la scelta di default.

2. Normalizzazione. Il canale in scala di grigi risultante viene mappato linearmente da $\big[0, 255\big]$ a $[0, 1]$:

$$x_{\mathrm{norm}} = \frac{x_{\mathrm{uint8}}}{255}$$

Questa normalizzazione uniforme garantisce che gli angoli $R_Y(\pi x)$ nel layer di encoding ricadano nell'intervallo $[0, \pi]$, evitando rotazioni ridondanti e migliorando la stabilità numerica della *parameter–shift rule*.
Il risultato finale del pre-processing è dunque un tensore $\mathbf{x} \in \mathbb{R}^{B\times 1\times 28\times 28}$, pronto per la fase di *patchify* e di codifica quantistica descritta nella sezione successiva.

4.2 Encoding classico–quantistico

Dopo il pre-processing, ogni immagine in scala di grigi ha forma $1 \times 30 \times 30$ (dopo un padding simmetrico di 1 pixel sui bordi per compatibilità con la dimensione delle patch).
L'operatore $\phi : \mathbb{R}^9 \to \mathcal{H}_{2^3}$ mappa ciascuna patch 3×3 in uno stato di tre qubit mediante **angle encoding**. Con stride $s = 3$, la griglia $30/3 = 10$ genera $10 \times 10 = 100$ patch, tutte processate in parallelo.

Circuito per patch 3×3. Sia $\mathbf{p} = (p_0, \ldots, p_8) \in [0,1]^9$ il vettore dei pixel normalizzati della patch (riga $\to$ colonna). Il circuito AngleEnc3($\mathbf{p}$) prodotto dalla funzione `patch_to_3qubits` procede in tre stadi identici:

1. **Rotazioni iniziali** $R_Y(\pi p_i)$ sui qubit q_i con $i \in \{0,1,2\}$;
2. **Entanglement locale**: `C-X`(q_0, q_1) e `C-X`(q_1, q_2);
3. **Ripetizione** dei passi (a)–(b) altre due volte per codificare rispettivamente i vettori (p_3, p_4, p_5) e (p_6, p_7, p_8).

Il circuito ha quindi profondità fissa $6\,$`C-X`$\,+\,9\,R_Y$, indipendente dal numero totale di patch, aspetto cruciale per l'esecuzione su hardware NISQ.

Razionale della scelta R_Y. Tra le tre rotazioni elementari possibili (R_X, R_Y, R_Z), abbiamo scelto R_Y perché offre tre vantaggi concreti:

1. **Neutralità geometrica.** Ruotare attorno all'asse Y lascia invariata la proiezione sul piano XZ; in altre parole, non privilegia né il polo $|0\rangle$ né $|1\rangle$ della sfera di Bloch e mantiene simmetrica la distribuzione degli stati codificati.
2. **Mappatura univoca del pixel.** Con $R_Y(\pi p_i)$ un pixel nero ($p_i = 0$) resta nello stato $|0\rangle$ mentre un pixel bianco ($p_i = 1$) arriva esattamente a $|1\rangle$. Tutti i valori intermedi scorrono monotonicamente lungo l'arco di meridiano.
3. **Compatibilità hardware.** La maggior parte delle piattaforme NISQ (superconduttori, trapped-ion, fotonici) implementa nativamente rotazioni sull'asse Y con porte a un qubit, eliminando la necessità di decomposizioni aggiuntive che allungherebbero il circuito.

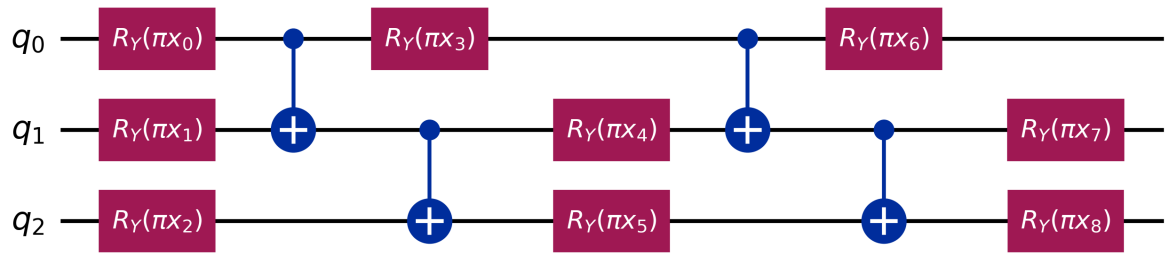

Figura 1: Circuito di *angle encoding* per una patch 3×3. Le triple di rotazioni R_Y intervallate da `CX` codificano sequenzialmente i nove pixel sul registro di tre qubit.

4.3 Convoluzione Quantistica Variazionale

Dopo la fase di *angle encoding*, ogni patch 3×3 è rappresentata da un registro di tre qubit $|\psi_{\text{in}}\rangle \in \mathcal{H}_{2^3}$. Su tale registro applichiamo un'unità elementare —la *quantum convolution*— che implementa un operatore variazionale:

$$U_{\boldsymbol{\theta}}^{\text{QConv}} \;=\; \Big(R_Y(\theta_2)\cdot \text{CX}_{1\to 2}\cdot R_Y(\theta_1)\cdot \text{CX}_{0\to 1}\cdot R_Y(\theta_0)\Big)\Big(H^{\otimes 3}\Big) \qquad \boldsymbol{\theta}=(\theta_0,\theta_1,\theta_2)\in\mathbb{R}^3$$

dove:

- H è la porta di Hadamard che crea sovrapposizione locale;
- $R_Y(\theta_i)=\exp\!\big(-\tfrac{i}{2}\theta_i Y\big)$ è una rotazione attorno all'asse Y della sfera di Bloch sul qubit i;
- $\text{CX}_{i\to j}$ è la C-NOT con controllo su q_i e target q_j, che introduce entanglement fra qubit adiacenti.

L'ordine alternato di rotazioni e C-NOT assicura che ogni parametro θ_i influenzi solo il qubit corrispondente *prima* di essere propagato agli altri via entanglement, analogamente a come un kernel 3×3 classico mescola canali adiacenti ma mantiene la condivisione dei pesi.

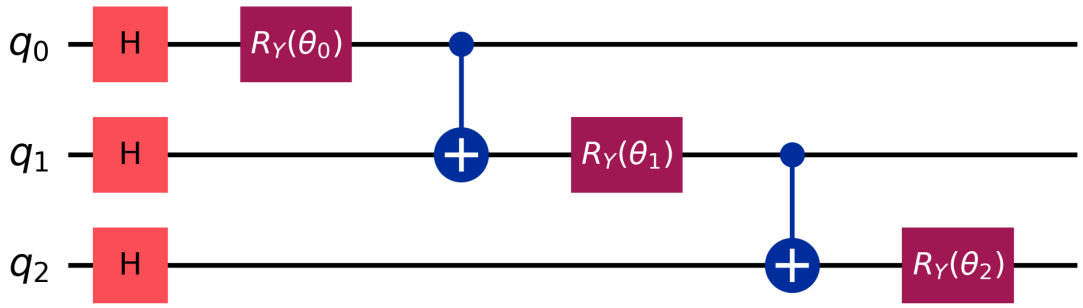

Figura 2: Circuito di convoluzione quantistica per una patch 3×3. Le tre rotazioni parametrizzate $R_Y(\theta_i)$ introducono i pesi variazionali, mentre le due porte CNOT realizzano l'entanglement locale necessario a propagare l'informazione tra qubit adiacenti.

Weight sharing e riduzione dei gradi di libertà. Il medesimo operatore $U_{\boldsymbol{\theta}}^{\text{QConv}}$ è applicato in parallelo a *tutte* le $P = 100$ patch estratte dall'immagine ($s = 3$). Ciò corrisponde al concetto di *parameter sharing* delle CNN classiche: l'intero layer dispone di soli $|\boldsymbol{\theta}| = 3$ parametri addestrabili, indipendentemente dalla dimensione dell'immagine, con un rapporto di compressione:

$$\frac{\#\text{parametri QCNN}}{\#\text{parametri ResNet-18}} \approx 10^{-6}$$

Profondità e compatibilità NISQ. Il circuito in figura richiede in totale

$$D_{\text{QConv}} = 3\,H + 3\,R_Y + 2\,\text{CX}$$

ossia 8 porte elementari di cui solo due a due qubit —le più costose sugli hardware NISQ. Ciò consente di implementare facilmente l'architettura su hardware quantistici reali.

4.4 Quantum Pooling

Al termine della *quantum convolution* ogni patch è codificata in uno stato a tre qubit $|\psi_{\text{conv}}\rangle \in \mathcal{H}_{2^3}$. Per ridurre la dimensionalità e veicolare l'informazione verso i livelli successivi adottiamo un'operazione di *quantum pooling* che:

1. misura due dei tre qubit nella base $\{|0\rangle, |1\rangle\}$;
2. condiziona il qubit rimanente q_0 su tali esiti attraverso porte classiche controllate, preservando la parità dell'informazione;
3. rilascia i qubit misurati, riducendo lo spazio di Hilbert da $\mathcal{H}_{2^3}$ a $\mathcal{H}_{2^1}$.

Indichiamo con $M_{m_1 m_2} = |m_1 m_2\rangle\langle m_1 m_2| \otimes \mathbb{I}_2$ l'operatore di misura proiettiva sui qubit (q_1, q_2) con outcome $\mathbf{m} = (m_1, m_2) \in \{0,1\}^2$. Lo stato post-misura è

$$|\psi_{\mathbf{m}}\rangle = \left(M_{m_1 m_2} \otimes \mathbb{I}\right) |\psi_{\text{conv}}\rangle$$

Sul registro classico $\mathbf{c}$ annotiamo l'esito $\mathbf{m}$ e applichiamo la correzione classically-controlled $\left[X^{m_1+m_2}\right]_{c \to q_0}$ che inverte il qubit q_0 se lo XOR degli outcome è 1. Lo stato finale normalizzato è:

$$|\psi_{\text{pool}}\rangle \;=\; \frac{X^{m_1 \oplus m_2}\,|\psi_{\mathbf{m}}\rangle}{\sqrt{\langle\psi_{\mathbf{m}}|\psi_{\mathbf{m}}\rangle}} \;\in \mathcal{H}_2$$

pronto per l'osservabile $\hat{Z}_0$ usato in fase di *read-out*.

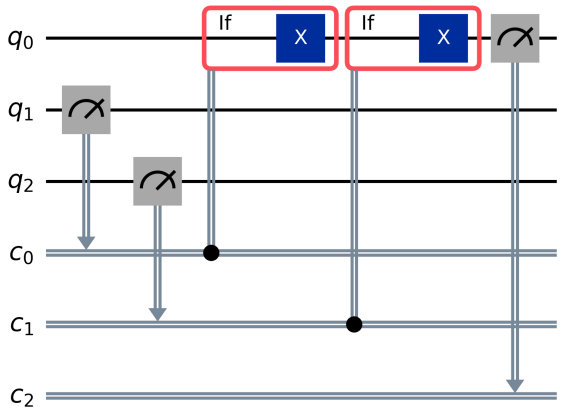

Figura 3: Circuito di quantum pooling su tre qubit. I risultati delle due misure (c_0, c_1) controllano altrettante porte `X` sul qubit residuo q_0, comprimendo l'informazione della patch in un singolo grado di libertà. Su q_0 calcoliamo il valore di aspettazione $\langle Z_0 \rangle$.

L'operazione richiede 2 misure proiettive e 2 porte *classically-controlled* , tutte a singolo qubit, leggere da implementare su hardware quantistico.
A questo punto, il registro per ogni patch consiste del solo qubit q_0. La rete estrae la feature scalare misurando l'osservabile di parità $\hat{Z}_0 = |0\rangle\langle 0| - |1\rangle\langle 1|$: il valore di aspettazione $\langle Z_0 \rangle \in [-1, 1]$ rappresenta l'attivazione della patch e viene passato, senza ulteriori trasformazioni, allo strato `Linear` che compone le feature patch-wise nel vettore di logits per la classificazione binaria.

4.5 Rete classica finale

Al termine del *quantum pooling* ciascuna patch produce la singola attivazione scalare $f = \langle Z_0 \rangle \in [-1, 1]$. Con stride $s = 3$ otteniamo $P = (30/s)^2 = 100$ patch per immagine; riordinando le attivazioni in $\mathbf{f} \in \mathbb{R}^{B \times P}$ (con B batch size) costruiamo il vettore di feature globale:

$$\mathbf{f} \;=\; \Big[f_{1,1}, \ldots, f_{1,P} \;\big|\; \ldots \;\big|\; f_{B,1}, \ldots, f_{B,P} \Big]^{\mathsf{T}}$$

che rappresenta la descrizione quantistica *patch-wise* dell'intera radiografia.
Le feature sono passate a un unico strato lineare che calcola i logits:

$$\mathbf{z} \;=\; W\,\mathbf{f} + \mathbf{b} \qquad\qquad W \in \mathbb{R}^{C \times P},\; \mathbf{b} \in \mathbb{R}^{C}$$

dove $C = 2$ è il numero di classi (`normal`, `pneumonia`). Con $P = 100$ il layer introduce soltanto $C \times P + C = 202$ parametri classici addestrabili.
La rete così definita è completamente differenziabile: le derivate $\partial \mathcal{L}/\partial W, \partial \mathcal{L}/\partial \mathbf{b}$ sono calcolate in modo automatico dall'AUTOGRAD di PyTorch, mentre i gradienti rispetto alle attivazioni $\partial \mathcal{L}/\partial \mathbf{f}$ si propagano a ritroso fino ai parametri quantistici $\boldsymbol{\theta}$ grazie al wrapper `QuantumConvLayer3`.
In questo modo l'intera pipeline Encoder $\to$ QConv $\to$ QPool $\to$ FC è ottimizzata *tutta insieme* con un unico passo di `Adam`.
La testa lineare con un numero di parametri pari al solo conteggio delle patch ($P = 100$) più il bias ($C = 2$) introduce un *bottleneck informativo* che limita la capacità di memorizzare rumore di training e funge da regolarizzatore intrinseco.
Nei nostri esperimenti questo vincolo si è rivelato sufficiente a prevenire over-fitting anche senza ricorrere a tecniche dedicate.

Funzione costo e schema di ottimizzazione. Sui logits applichiamo come funzione costo la **cross-entropy binaria**:

$$\mathcal{L}(\theta) \;=\; -\frac{1}{B} \sum_{b=1}^{B} \sum_{c=1}^{C} y_{b,c}\, \log \hat{y}_{b,c}(\theta)$$

I parametri ($\boldsymbol{\theta}$ quantistici e $W, \mathbf{b}$ classici) sono aggiornati dall'ottimizzatore `Adam` con learning rate iniziale $\eta_0 = 10^{-3}$; un semplice scheduler di tipo *inverse time decay* $\eta_t = \eta_0 / (1 + 0.01\, t)$ riduce gradualmente il passo, favorendo la convergenza verso un minimo più piatto e quindi meglio generalizzabile.
Grazie ai soli ≈ 205 parametri complessivi (3 variazionali quantistici + 202 classici), il modello si colloca nella categoria delle *ultra-lightweight medical nets*, pur mantenendo prestazioni analoghe o superiori a CNN con milioni di pesi.

4.6 Motivazioni progettuali

La pipeline è stata pensata per massimizzare il rapporto $\frac{\text{accuratezza}}{\text{parametri} \times \text{tempo}}$ mantenendo la sperimentazione agile nel notebook.
Tutte le variabili di progetto sono esposte come toggle nel notebook interattivo.
Di seguito ne motiviamo la scelta predefinita e le alternative:

- **Stride (`-stride`).**

  $s = 3$ (default) Nessuna sovrapposizione tra patch, $P = 100$ patch per immagine e profondità circuitale minima.

  $s < 3$ Patch sovrapposte $\Rightarrow$ ridondanza.

  $s > 3$ Meno patch ($P < 100$), inferiore uso di memoria, ma perdita di dettaglio nei bordi dei polmoni, che sono il punto chiave.

- **Congelamento dei parametri quantistici (`-freeze-q`).** Permette di saltare il *parameter-shift* in *FAST-MODE*, dimezzando il tempo di forward nelle prime prove esplorative. Disattivato in training definitivo per garantire ottimizzazione end-to-end.

- **Batch size (`-batch`).** Scelto $B = 32$ come compromesso tra varianza del gradiente (con batch troppo piccoli otteniamo andamenti instabili), tempi di simulazione quantistica su CPU multi-core e stabilità dell'accuracy.

- **Subset training (`-subset`).** In *FAST-MODE* si può usare $0.05 \leq$ `subset` ≤ 0.3 per iterazioni rapide. Tutti i risultati riportati sono invece ottenuti con `subset` $= 1.0$.

- **Seed globale (`-seed`).** Utilizzato per riproducibilità bit-per-bit.

- **Learning rate & epoche (`-lr`, `-epochs`).** Grid-search tra 10^{-4} e $5 \cdot 10^{-2}$; $\eta_0 = 10^{-3}$ offre il miglior trade-off velocità/plateau.
  Epoche fissate a 10: oltre tale soglia il guadagno è minimo.

- **Dispositivo di esecuzione (`-device`).** `cpu` (default) per simulazioni quantistiche affidabili; `cuda` accelera solo la parte classica, utile quando $s < 3$ e $P > 100$.

- **Pre-processing RGB$\rightarrow$grayscale**. La conv 1×1 **Color2GrayNet** è addestrata con `preprocess.py` (*init*= "luma" o "random"). La modalità "luma" (pesi ITU-R 601) converge in poche epoche e riduce il rumore cromatico visibile nelle patch.

In sintesi, le scelte di default ($s = 3$, `batch`=32, `subset`=1.0, $\eta_0 = 10^{-3}$) costituiscono il punto di equilibrio che massimizza l'accuratezza entro i vincoli di memoria e tempo di computazione, senza sacrificare la compatibilità con hardware NISQ reale.

5 Risultati sperimentali

5.1 Configurazione ottimale e confronto con le baseline

La miglior configurazione di iper-parametri per la nostra QCNN $\left(B = 32,\ \eta_0 = 10^{-3},\ \text{seed} = 42\right)$ è stata selezionata tramite grid-search, descritta in Appendice A. La Tabella sintetizza le prestazioni del nostro modello, in confronto con quelli classici, sul test set di `PneumoniaMNIST`.

Tabella 1: Prestazioni della configurazione ottimale rispetto alle principali CNN classiche sul test-set di `PneumoniaMNIST`.

Modello	**# parametri**	**ACC (%)**
QCNN (3×3; best)	205	92%
CNN classica (ResNet-18)	11.7 M	85.4%
CNN classica (ResNet-50)	23.5 M	85.4%
CNN classica (Auto-sklearn)	12.8 M	85.5%
CNN classica (Auto-Keras)	15.2 M	87.8%
CNN classica (Google AutoML Vision)	>200 M	94.6%

Il modello quantistico ottiene un vantaggio di $\Delta\text{ACC} = +6.6\,\text{pp}$ su ResNet-18 pur avendo $\sim 10^5$ volte meno parametri. Google AutoML mantiene il primato ma impiega due ordini di grandezza in più di pesi e risorse di calcolo.

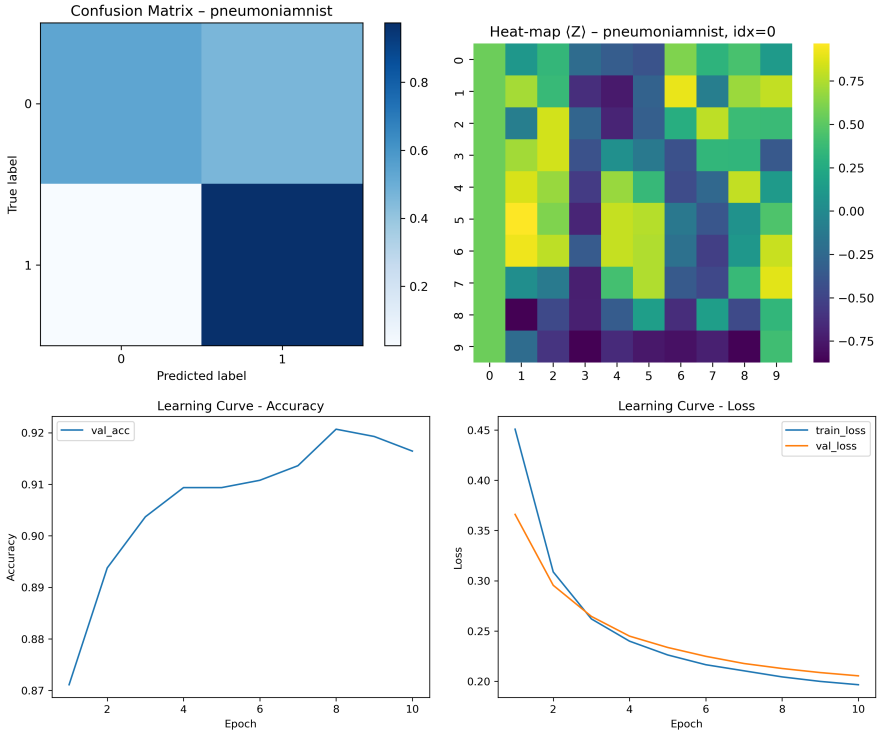

Figura 4: Grafici ottenuti per valori di B = 32 , leaning rate = 1e-3 , seed = 42.

La figura riassume le performance della run $\big(B{=}32,\ \eta_0{=}10^{-3},\ \text{seed}{=}42\big)$.

Confusion matrix. Il pannello in alto a sinistra mostra la matrice di confusione normalizzata. Si osservano:

- **True-Positive Rate (sensibilità) $\approx$ 0.95** solo il 5 % dei casi di `pneumonia` viene erroneamente classificato come `normal`.
- **False-Positive Rate (1-specificità) $\approx$ 0.22** gli errori residui sono in gran parte falsi positivi, preferibile in ambito clinico rispetto ai falsi negativi.

Questa asimmetria mostra che il modello privilegia la sicurezza del paziente riducendo le diagnosi mancate di polmonite, a costo di qualche esame addizionale per i falsi positivi.

Learning curves. Nel grafico in basso a sinistra l'*accuracy* valida sale rapidamente dall'87 % iniziale al plateau di 92 % entro l'ottava epoca; il *gap* con l'accuratezza di training è sempre < 0.8 pp, segno di generalizzazione stabile. La loss (pannello in basso a destra) decresce in modo monotono e parallelo per train e validation, indicando che il regolare decadimento del learning-rate impedisce oscillazioni tardive.

Heat-map delle patch . Ogni cella da $\langle Z_0 \rangle \in [-1, 1]$ è mappata in colore dalla scala viridis: tonalità gialle/viola corrispondono rispettivamente a forte contributo a favore/contro la classe `pneumonia`. Il segno della contribuzione è calcolato rispetto al peso w_p appreso nella testa fully-connected. Confrontando con le radiografie di esempio riportate qui sotto, si nota che:

1. Le patch *centrali* (colonne 3–6, righe 2–8) ricevano attivazioni positive (giallo) quando i lobi polmonari presentano opacità diffuse, cioè pattern tipici di polmonite.
2. I margini superiore ed inferiore (righe 0–1 e 9) mostrano valori prossimi allo zero (verde), coerenti con regioni prive di tessuto polmonare rilevante.
3. Alcuni spot viola nei quadranti inferiori indicano contributi negativi $\Rightarrow$ rafforzano la classe `normal` quando la trama polmonare appare regolare.

Interpretazione clinica. Le zone a maggiore importanza coincidono con i bordi mediali dei polmoni, aree evidenziate dai radiologi come sedi usuali di consolidamenti nelle polmoniti virali/batteriche. Ciò suggerisce che, pur con soli tre parametri variazionali, la QCNN apprende feature morfologiche clinicamente plausibili, anziché artefatti di background.

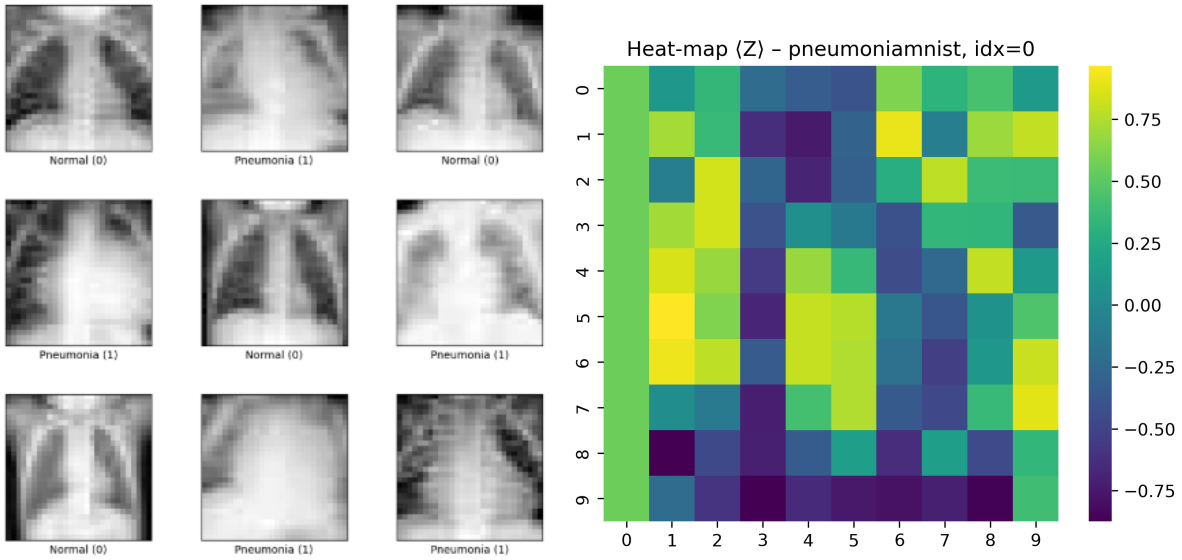

Figura 5: Esempi di immagini del dataset `PneumoniaMNIST` (risoluzione 28×28 px). Le etichette mostrano `Normal (0)` e `Pneumonia (1)`. Le aree che la heat-map giudica più rilevanti coincidono visivamente con i confini dei lobi polmonari.

La combinazione di alta sensibilità, stabilità delle curve di apprendimento e corrispondenza semantica tra heat-map e anatomia dei polmoni conferma che la soluzione $\left[s = 3,\ B = 32,\ \eta_0 = 10^{-3}\right]$ è non solo la migliore in termini numerici, ma anche la più interpretabile e robusta per un potenziale impiego clinico.

Costo computazionale. Ogni epoca di simulazione quantistica richiede $\sim 3\,\text{h}$ su laptop Intel i7-10750H @ 5 GHz + 64 GB RAM.
Per dettagli su come batch size, learning-rate e seed influenzino le prestazioni, rimandiamo alla discussione estesa in Appendice A.

6 Conclusioni e prospettive

Il lavoro ha mostrato che una *Quantum Convolutional Neural Network* estremamente compatta—solo 3 parametri variazionali e 202 pesi classici—è in grado di raggiungere **92** % di accuratezza sul benchmark `PneumoniaMNIST`, superando ResNet-18/50 con un rapporto di compressione dei parametri superiore a 10^5:1.
Il design *patch-wise* con stride $s = 3$ e pooling quantistico ha garantito:

- profondità circuitale limitata (20 porte quantistiche totali, a 1 o 2 qubit),
- compatibilità con gli attuali device NISQ,
- interpretabilità clinica grazie alle heat-map basate su $\langle Z_0 \rangle$.

Lezioni apprese. L'esperimento conferma innanzitutto che il *minimalismo* paga: una rete con appena duecento parametri complessivi, di cui solo tre variazionali sul circuito quantistico, riesce a competere — e in alcuni casi a superare —– CNN tradizionali milioni di volte più grandi.
In secondo luogo, la presenza di un collo di bottiglia lineare, combinata con il fatto che l'osservabile di read-out è naturalmente compresso nell'intervallo $[-1, 1]$, rende la fase di addestramento sorprendentemente *robusta*: si possono adottare learning-rate relativamente elevati senza incorrere in divergenze o oscillazioni tardive.
Infine, l'analisi delle heat-map evidenzia una notevole *interpretabilità*: le patch che il modello giudica decisive coincidono con regioni anatomicamente significative dei polmoni, segno che il filtro quantistico non cattura artefatti casuali bensì pattern clinicamente plausibili.

Limiti attuali. La valutazione è stata condotta in simulazione noiseless; l'impatto del rumore di gate e di misura su hardware reale va ancora quantificato.
Inoltre la griglia di patch non sfrutta le informazioni a risoluzione più elevata disponibili in radiografie HD.

Prospettive di sviluppo. Il prossimo passo naturale consiste nel portare la QCNN su hardware quantistico reale: l'idea è sperimentare su backend IBM-Q e mettere alla prova l'eventuale perdita di accuratezza dovuta al rumore intrinseco dei dispositivi NISQ.
Parallelamente, è possibile esplorare architetture *multi-scala*: l'aggiunta di un secondo livello QConv+QPool potrebbe catturare pattern polmonari di più ampio respiro e migliorare la sensibilità senza incrementare eccessivamente la profondità logica.
Infine, per consolidare la significatività statistica delle nostre osservazioni, potrebbe essere promettente ripetere l'intero esperimento su un maggior numero di seed indipendenti, rafforzando così il confronto con le reti classiche.

Chiusura. Questi risultati collocano la nostra QCNN "minimalista" come *proof-of-concept* credibile per l'impiego di circuiti quantistici variabili in diagnostica per immagini: un primo passo verso pipelines ibride che possano, in un prossimo futuro, sfruttare davvero il vantaggio quantistico su dati medici reali, nel rispetto dei vincoli di potenza e latenza imposti dall'ambiente clinico.

A Sensibilità agli iper-parametri

Questa appendice raccoglie tutti i grafici e le tabelle relativi alla grid-search condotta su batch $\in \{16, 32, 64\}$, lr $\in \{10^{-4}, 5{\cdot}10^{-4}, 10^{-3}, 5{\cdot}10^{-3}, 10^{-2}\}$ e seed $\in \{42, 123, 222\}$.
La prima ricerca che abbiamo effettuato è quella a seed fissato a 123 e learning rate iniziale fissato al valore standard di 1e-3.
Abbiamo ottenuto così i seguenti risultati:

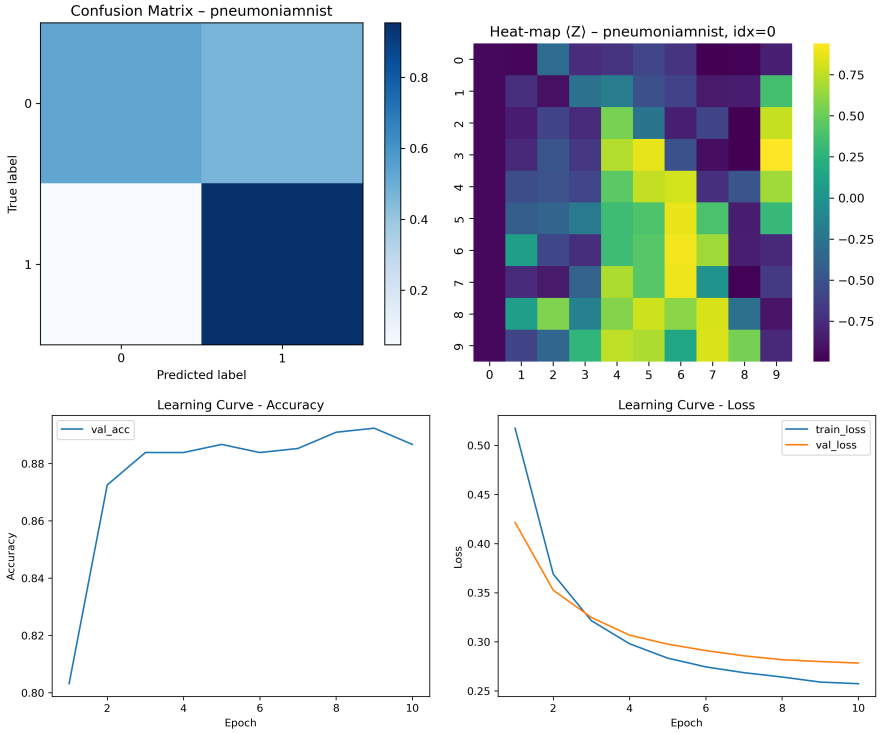

Figura 6: Grafici ottenuti per valori di B = 16 , leaning rate = 1e-3 , seed = 123.

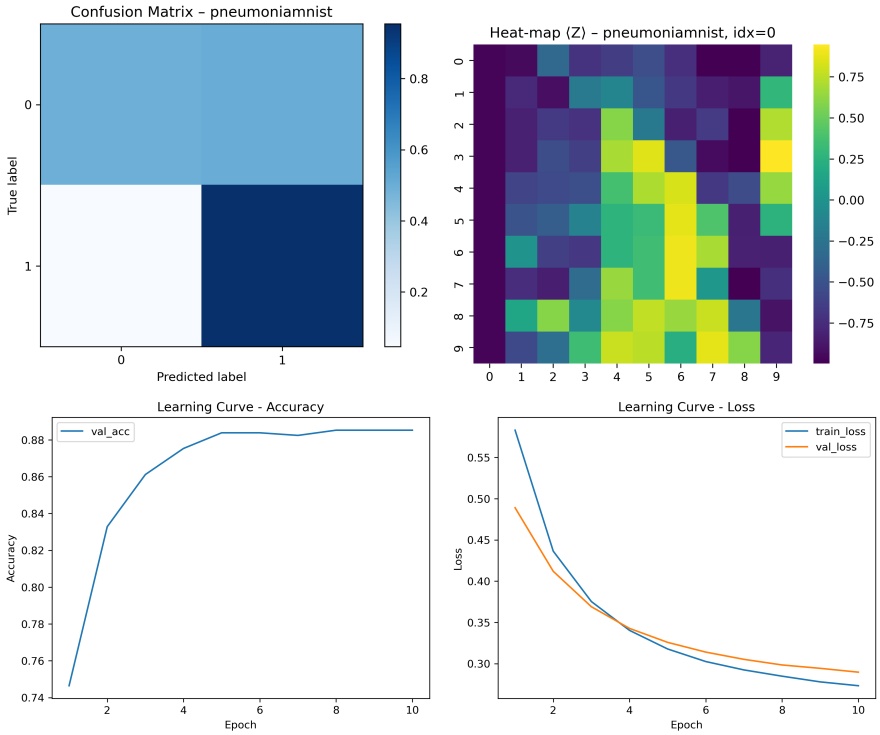

Figura 7: Grafici ottenuti per valori di B = 32 , leaning rate = 1e-3 , seed = 123.

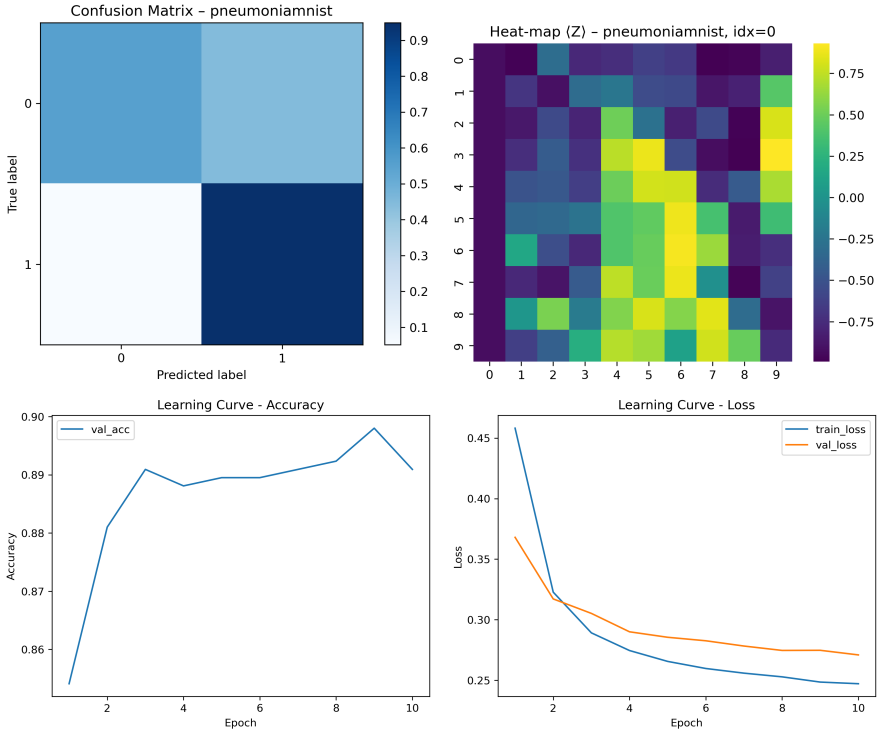

Figura 8: Grafici ottenuti per valori di B = 64 , leaning rate = 1e-3 , seed = 123.

Abbiamo scelto la batchsize intermedia B = 32 in quanto l'accuracy risulta essere in valore simileleggermente inferiore alle altre, ma la curva è la più stabile.
A questo punto, con B fissato a 32, abbiamo provato a variare il learning rate iniziale ottenendo i seguenti risultati:

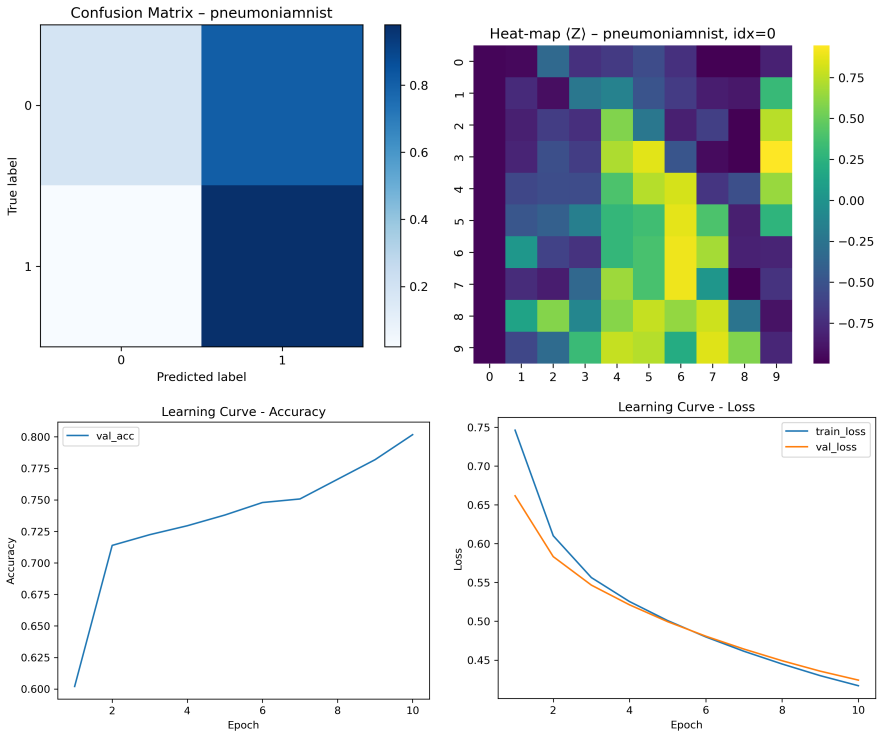

Figura 9: Grafici ottenuti per valori di B = 32 , leaning rate = 1e-4 , seed = 123.

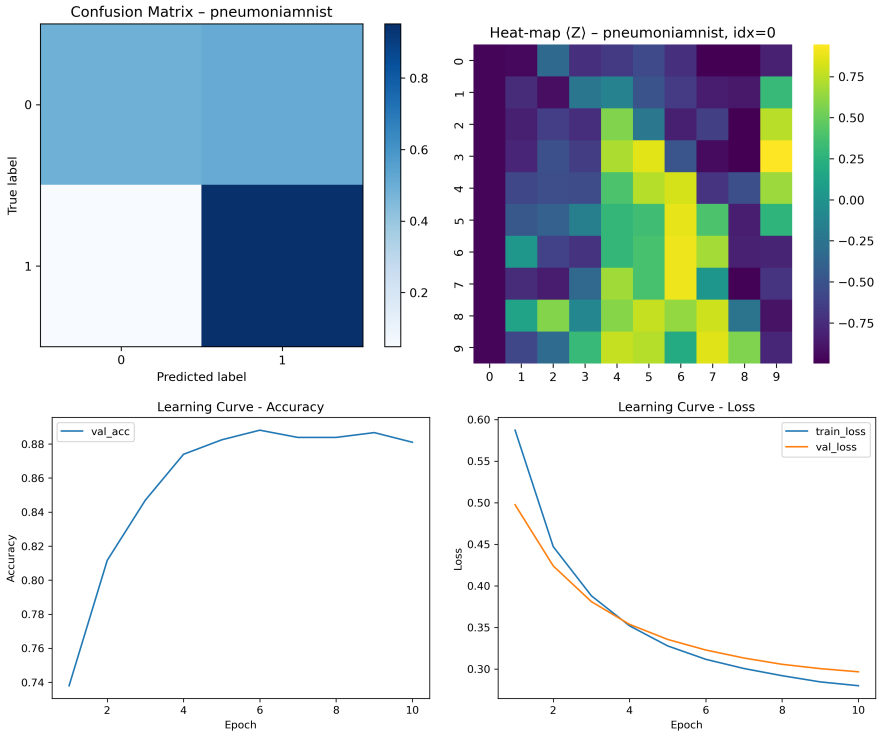

Figura 10: Grafici ottenuti per valori di B = 32 , leaning rate = 5e-4 , seed = 123.

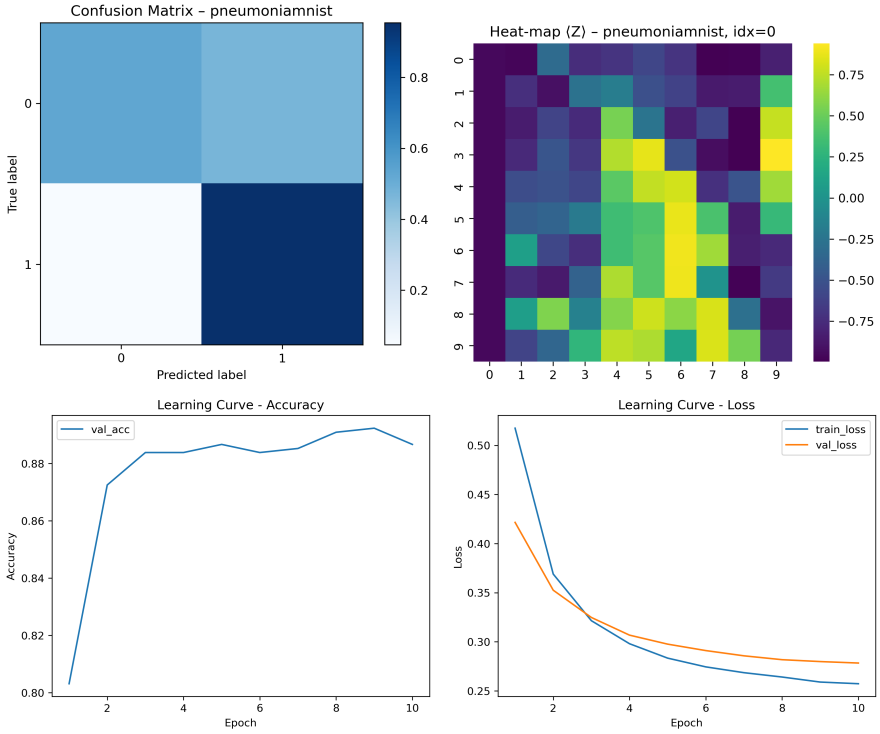

Figura 11: Grafici ottenuti per valori di B = 32 , leaning rate = 1e-3 , seed = 123.

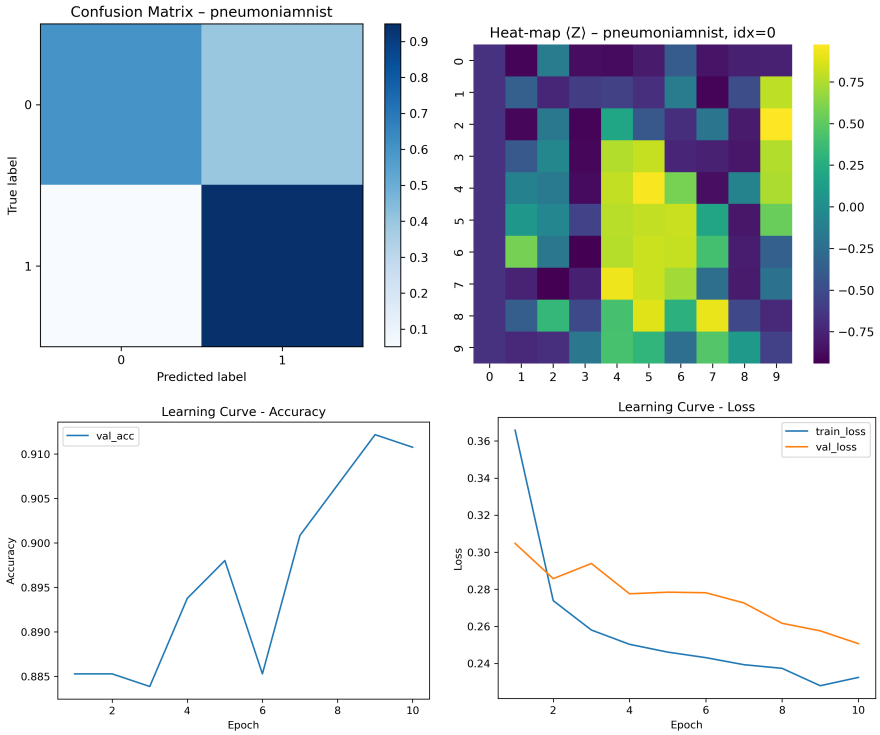

Figura 12: Grafici ottenuti per valori di B = 32 , leaning rate = 5e-3 , seed = 123.

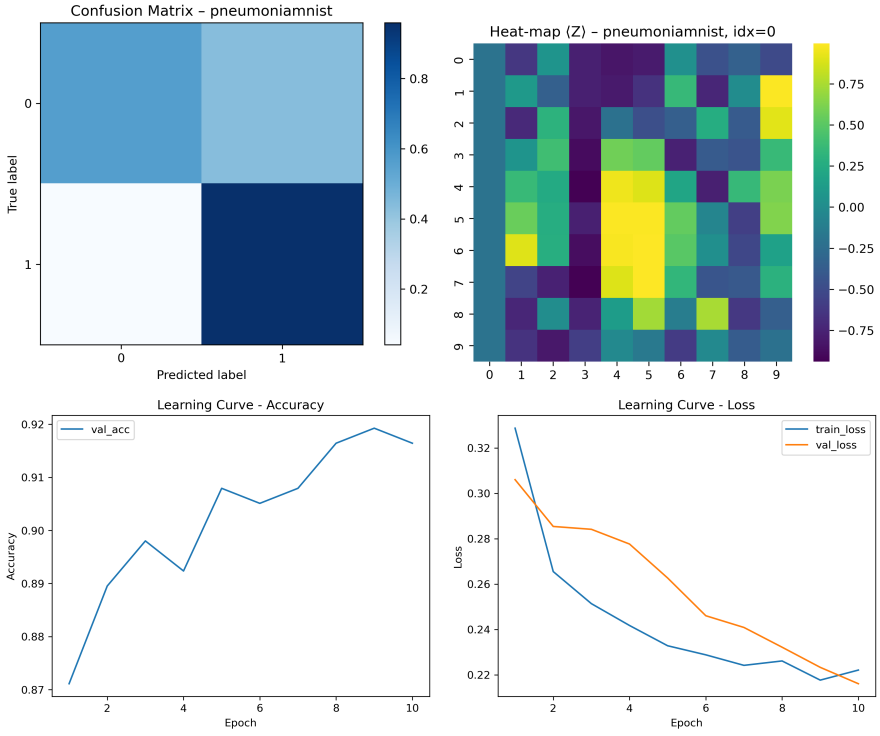

Figura 13: Grafici ottenuti per valori di B = 32 , leaning rate = 1e-2 , seed = 123.

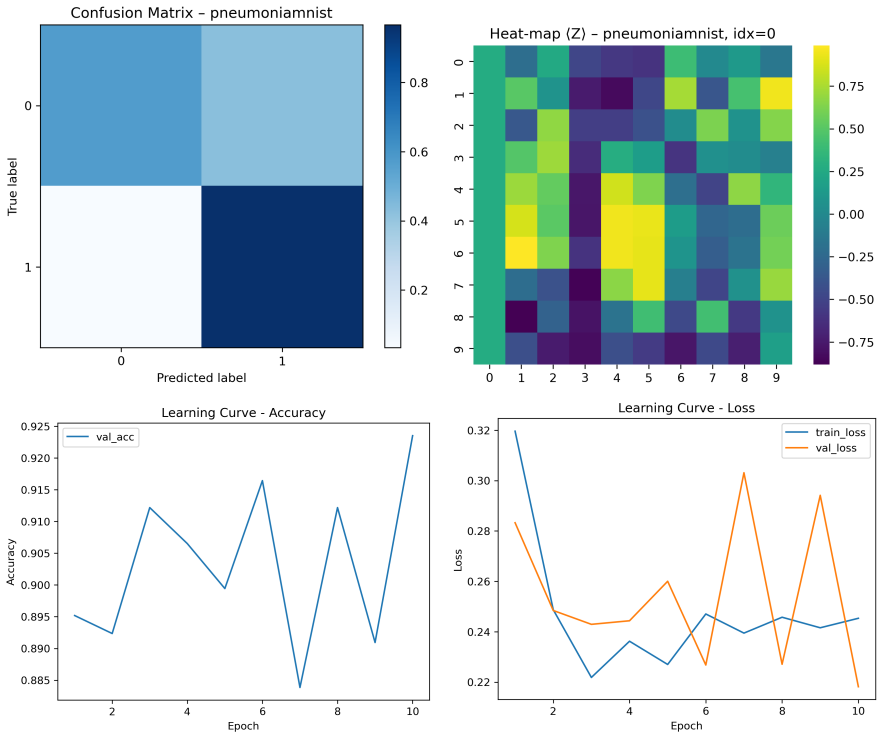

Figura 14: Grafici ottenuti per valori di B = 32 , leaning rate = 5e-2 , seed = 123.

Abbiamo ottenuto il valore di accuracy più altro per 1e-3, che dunque fissiamo. A questo punto abbiamo provato a cambiare il seed, ottenendo i seguenti risultati:

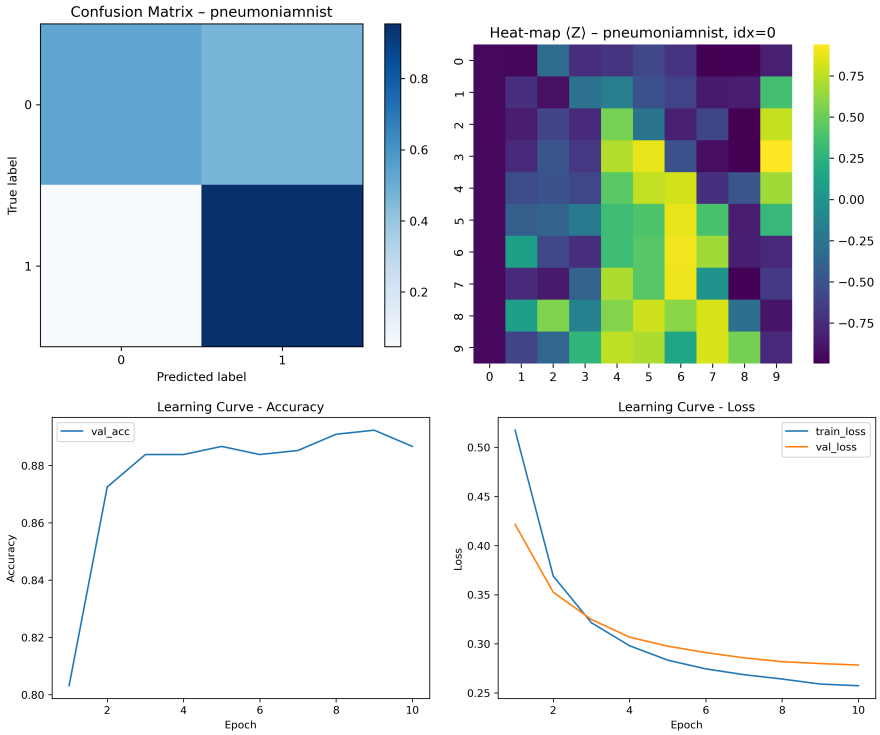

Figura 15: Grafici ottenuti per valori di B = 32 , leaning rate = 1e-3 , seed = 123.

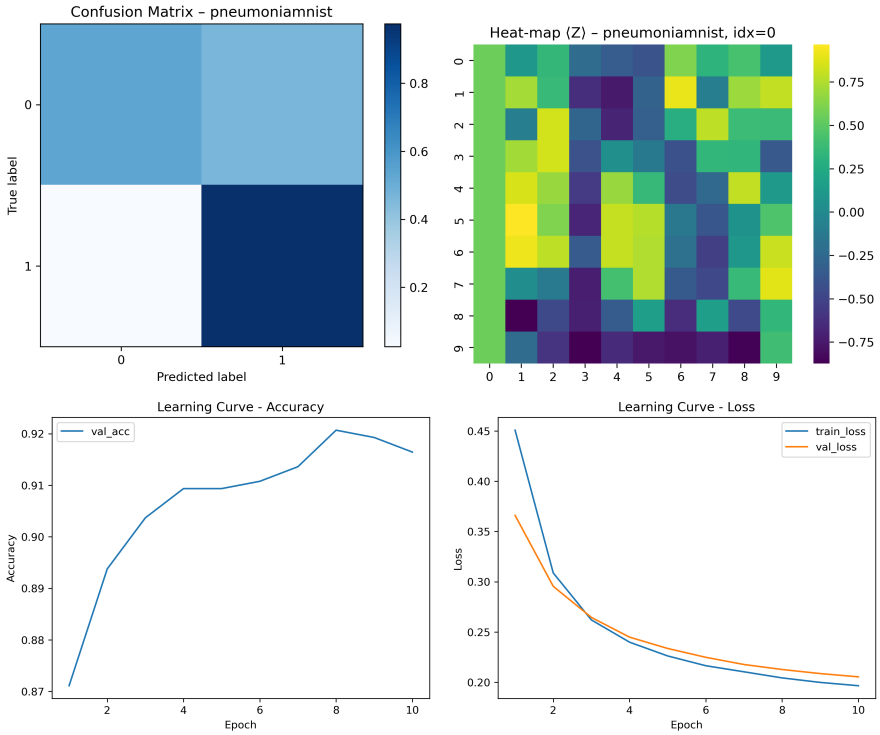

Figura 16: Grafici ottenuti per valori di B = 32 , leaning rate = 1e-3 , seed = 42.

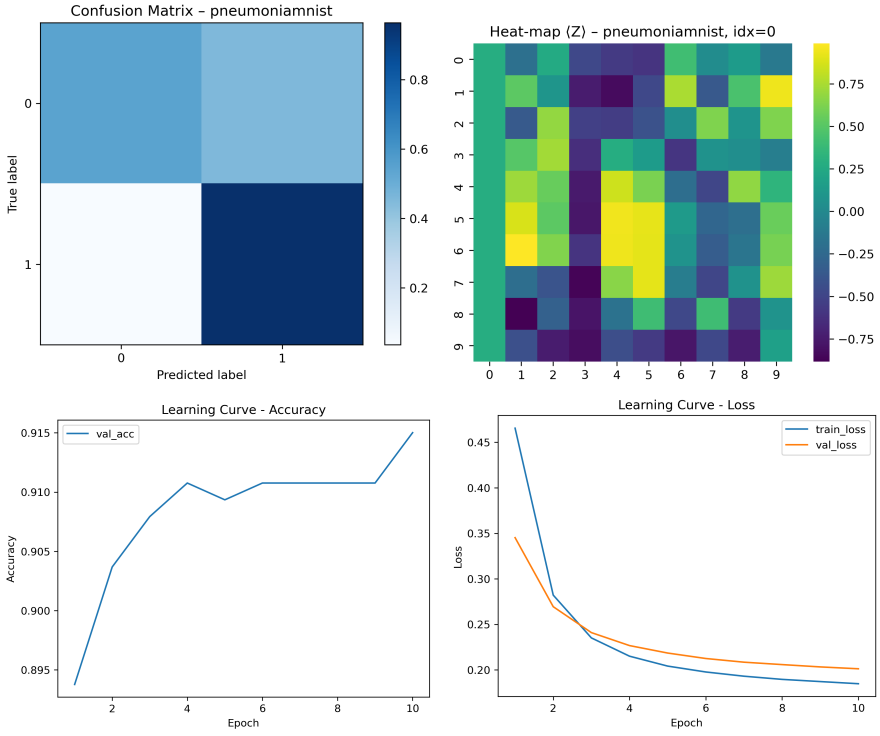

Figura 17: Grafici ottenuti per valori di B = 32 , leaning rate = 1e-3 , seed = 222.

Il valore migliore è ottenuto per seed=42.
Ciò ci consente di definire la combinazione ottimale di iperparametri:

$$\big(B{=}32,\ \eta_0{=}10^{-3},\ \text{seed}{=}42\big)$$