Design, Implementation and Experimental Evaluation of a Hybrid Quantum Convolutional Neural Network for Chest X-ray Classification in the Dataset `PneumoniaMNIST`

Antonio Mosca Leonardo Tomei

February 8, 2026

Abstract

Computer-aided diagnosis on chest X-rays is a privileged testbed for assessing the benefits of hybrid *quantum–classical* models on NISQ hardware. We present an extremely compact *Quantum Convolutional Neural Network* (QCNN), designed for binary pneumonia classification from low-resolution X-rays (RGB 28×28) drawn from the `PneumoniaMNIST` sub-collection of the `MedMNIST` benchmark. The proposed hybrid architecture is organized as a pipeline composed of lightweight classical preprocessing, an *angle encoding* layer, a single quantum convolution with only three variational parameters followed by quantum pooling, and a linear classifier. After 10 epochs, the QCNN reaches **80.9% accuracy** on the test set, competitive with much deeper classical CNN counterparts (ResNet-18/50, $\approx$ 85%). The shallow circuit depth and reduced number of parameters make the architecture compatible with current *Noisy Intermediate-Scale Quantum* devices, paving the way for future experiments on real hardware.

Contents

1 Introduction

Over the last ten years, *Convolutional Neural Networks* (CNNs) have revolutionized computer vision, becoming the state of the art. The success of CNNs is grounded in three key properties:

- (i) **filter locality**, which exploits the spatial correlation of images;
- (ii) **parameter sharing**, which drastically reduces the number of weights compared with fully connected networks;
- (iii) **hierarchical compositionality**, which enables learning features from low to high level.

However, increasingly deep architectures (tens of millions of parameters for models such as ResNet-50) imply high computation and storage costs and require large amounts of labeled data, which are not always available in clinical settings.
In parallel, the advent of *Noisy Intermediate-Scale Quantum* (NISQ) devices has opened the way to hybrid *quantum–classical* models aimed at exploiting qubit superposition and entanglement to obtain a potentially more compact and expressive representation than classical networks. Among the proposed architectures, *Quantum Convolutional Neural Networks* (QCNNs) stand out: they map the notion of convolution onto low-depth quantum circuits through a sequence of local operations followed by *quantum pooling* measurements. In analogy with classical CNNs, QCNNs combine operation **locality** on patches with **non-locality** provided by entanglement, while reducing the number of trainable parameters.
In this study we investigate the use of a QCNN for automatic pneumonia classification on low-resolution chest X-rays. Pneumonia diagnosis from radiography is a clinically relevant use case and, due to its binary nature, is well suited to assess in a controlled way the advantages (or limits) of quantum models. For this purpose we use the `PneumoniaMNIST` sub-collection of the `MedMNIST` benchmark, because it provides a *lightweight* set with expert-validated labels while retaining the statistical characteristics of a real-world problem.
The main contribution of this work is the design and analysis of a *minimalist* QCNN — with only three variational parameters — that, in simulation, reaches performance competitive with much larger classical CNNs, showing that even low-depth circuits can be effective on biomedical data. The proposed approach is a step toward executing *medical imaging* models on real quantum hardware, where circuit depth and number of parameters are critical constraints.

2 Problem framing

2.1 `MedMNIST` dataset

`MedMNIST` is a collection of pre-processed biomedical datasets in 28×28 format designed for fast, comparable benchmarking. Our analysis focuses on `PneumoniaMNIST`, which contains **5856** RGB chest X-rays (28×28 px) annotated by specialists as `pneumonia` or `normal`. The dataset is already split into the official partitions:

Split	# Images	Percentage
Training	4708	80.3 %
Validation	524	8.9 %
Test	624	10.8 %

The dataset, being relatively small, is suitable for rapid training of both classical and quantum models.

2.2 Classification task

The problem is formulated as binary classification: given an input $x \in \mathbb{R}^{3\times 28\times 28}$, predict $y \in \{0, 1\}$, where 1 denotes the presence of pneumonia. The adopted metrics are:

- **Accuracy** (ACC) — fraction of correct predictions over the total;
- **Macro-averaged F1-score** — harmonic mean of precision and recall, computed separately for each class and then averaged, useful in the presence of mild class imbalance;

In light of these elements, the main challenge is to verify whether a quantum model with a number of parameters orders of magnitude smaller can compete with established classical CNNs, while respecting the noise and depth limits typical of NISQ hardware.

3 Theoretical context of QCNN

Quantum Convolutional Neural Networks (QCNNs) are a quantum transposition of the principles that made classical CNNs so effective in computer vision: filter locality, parameter sharing, and hierarchical pooling. In the quantum setting, these concepts are reformulated through low-depth circuits operating on *local qubit registers* and through intermediate measurements that carry information toward increasingly compressed levels. In what follows, we provide the theoretical foundations required to understand our implementation, postponing circuit-level details to the next section.

3.1 Data encoding

Since a native quantum device processes states in a Hilbert space $\mathcal{H}_{2^n}$, a preliminary **data encoding** stage embeds classical data into a quantum state. Among the various possibilities proposed in the literature (basis, amplitude, Hamiltonian, ...), *angle encoding* is particularly suitable for low-resolution images: a normalized pixel value $x \in [0, 1]$ is encoded as a single-qubit rotation $|0\rangle \mapsto R_Y(\pi x)\,|0\rangle$.
In this way, circuit depth grows linearly with the number of features rather than exponentially.

3.2 Local quantum convolution

Let $|\psi\rangle$ be the state produced by the encoding layer on n qubits. **Quantum convolution** acts on a sub-region of k qubits (*patch*) by applying a parameter-dependent unitary operator:

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

In the paradigm we adopt, this operator is factorized into two fundamental blocks:

- *Single-qubit rotations* $R_y(\theta_i) \in \mathrm{U}(2)$, which inject variable parameters;
- *Fixed entangling gates* (`CNOT` in our case) that correlate the qubits inside the patch.

The same angle set $\boldsymbol{\theta}$ is **shared across all patches**, analogously to kernel sharing in classical CNNs. This design reduces trainable parameters from $\mathcal{O}(N_{\text{patch}})$ to $\mathcal{O}(1)$, where N_{patch} is the total number of patches on which $U_{\boldsymbol{\theta}}^{\text{conv}}$ is applied.

3.3 Quantum pooling

Quantum pooling replaces classical downsampling operations (max, average) with a *partial measurement* on a subset of qubits, followed by conditional post-processing on the classical register.
Formally, let $\mathcal{M}$ be the POVM operator (positive operator-valued measure) measuring $k-1$ qubits of the patch. Conditioning the remaining part on the outcome m and applying classical gates (`X` in our case) controlled by the classical bitstring $\mathbf{m}$ relocates information onto a single qubit $|\psi'\rangle \in \mathcal{H}_2$ while preserving the coherence required for subsequent levels.

3.4 Training through variational circuits

The parameter set θ is optimized following the *Variational Quantum Circuits* (VQC) paradigm. Given a cost observable $C(\theta) = \langle\psi(\theta)|\,\hat{O}\,|\psi(\theta)\rangle$, the analytic gradient of C with respect to each θ_j can be obtained through the **parameter-shift rule**:

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

which requires only two extra circuit evaluations, independent of the total number of qubits. In this way, gradients from quantum blocks and classical blocks flow into a single vector that optimizers such as Adam can update, allowing the full network to be trained end-to-end.

3.5 Depth and NISQ compatibility

The depth D of a QCNN generally scales with the number of *convolution–pooling* pairs (L), but remains manageable because qubits measured at each pooling stage are discarded, progressively reducing the Hilbert space: $D \simeq L \cdot D_{\text{conv}}$ with $L = \mathcal{O}(\log_k n)$, where k is the patch size. In our minimalist design ($k = 3$, $L = 1$), the quantum part of each patch requires only

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

thus fewer than ten two-qubit gates in total.
Such a short circuit is fully compatible with current *Noisy Intermediate-Scale Quantum* platforms. In other words, the network can be deployed on real hardware without requiring further depth-compression techniques.

4 Architecture description

4.1 Image pre-processing

The data flow feeding the quantum component starts from a fully classical pipeline, implemented in `PyTorch` and contained in src/datasets/color2gray.py. The network works in two steps, listed below.

1. RGB $\to$ grayscale conversion. We use a **trainable** 1×1 convolution that includes exactly three weights, one for each color channel:

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

implemented by the class `Color2GrayNet`.
Initialization can be performed in two modes:

- **``luma''**: fixed weights $(w_R, w_G, w_B) = (0.299, 0.587, 0.114)$, corresponding to the luminance formula from the ITU-R BT. 601 standard;
- **``random''**: Xavier uniform initialization; the coefficients are then learned by the model during training.

In our experiments, the **``luma''** option accelerates convergence without saturating gradients, and is therefore the default choice.

2. Normalization. The resulting grayscale channel is linearly mapped from $\big[0,255\big]$ to $[0,1]$:

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

This uniform normalization ensures that the $R_Y(\pi x)$ angles in the encoding layer fall in the interval $[0,\pi]$, avoiding redundant rotations and improving the numerical stability of the *parameter–shift rule.*
The final pre-processing output is therefore a tensor $\mathbf{x}\in\mathbb{R}^{B\times1\times28\times28}$, ready for the *patchify* and quantum encoding stage described in the next section.

4.2 Classical–quantum encoding

After pre-processing, each grayscale image has shape $1 \times 30 \times 30$ (after symmetric 1-pixel padding on the borders for compatibility with patch size).
The operator $\phi : \mathbb{R}^9 \to \mathcal{H}_{2^3}$ maps each 3×3 patch into a three-qubit state via **angle encoding**. With stride $s = 3$, the grid $30/3 = 10$ yields $10 \times 10 = 100$ patches, all processed in parallel.

Circuit for a 3×3 patch. Let $\mathbf{p} = (p_0, \ldots, p_8) \in [0, 1]^9$ be the vector of normalized patch pixels (row $\to$ column). The circuit AngleEnc3($\mathbf{p}$) produced by function `patch_to_3qubits` proceeds in three identical stages:

1. **Initial rotations** $R_Y(\pi p_i)$ on qubits q_i with $i \in \{0, 1, 2\}$;
2. **Local entanglement** : `C-X`(q_0, q_1) and `C-X`(q_1, q_2);
3. **Repetition** of steps (a)–(b) to encode respectively the vectors (p_3, p_4, p_5) and (p_6, p_7, p_8).

The circuit therefore has fixed depth $4\,$`C-X`$\,+\,9\,R_Y$, independent of the total number of patches, a crucial aspect for execution on NISQ hardware.

Rationale for choosing R_Y. Among the three possible elementary rotations (R_X, R_Y, R_Z), we selected R_Y because it offers three concrete advantages:

1. **Geometric neutrality.** Rotation around the Y-axis leaves the projection on the XZ plane unchanged; in other words, it privileges neither the $|0\rangle$ nor $|1\rangle$ pole of the Bloch sphere and keeps the distribution of encoded states symmetric.
2. **Unique pixel mapping.** With $R_Y(\pi p_i)$, a black pixel ($p_i = 0$) remains in state $|0\rangle$, while a white pixel ($p_i = 1$) reaches exactly $|1\rangle$. All intermediate values move monotonically along the meridian arc.
3. **Hardware compatibility.** Most NISQ platforms (superconducting, trapped-ion, photonic) natively implement rotations around the Y-axis with single-qubit gates, eliminating the need for additional decompositions that would lengthen the circuit.

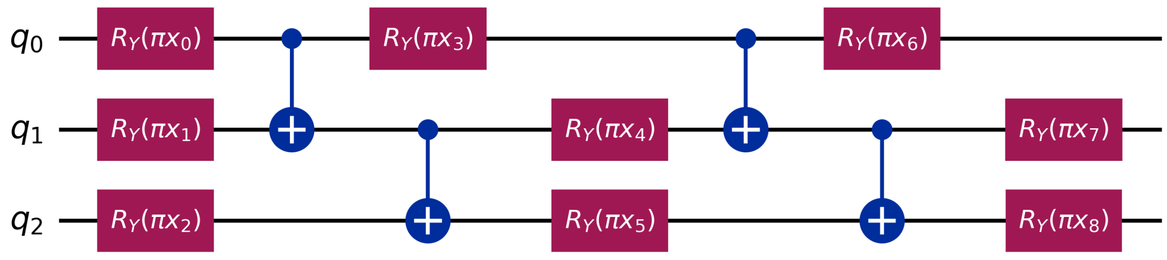

Figure 1: *Angle encoding* circuit for a 3×3 patch. The triples of R_Y rotations interleaved with `CX` sequentially encode the nine pixels on the three-qubit register.

4.3 Variational Quantum Convolution

After the *angle encoding* stage, each 3×3 patch is represented by a three-qubit register $|\psi_{\text{in}}\rangle \in \mathcal{H}_{2^3}$. On this register we apply an elementary unit — the *quantum convolution* — implementing a variational operator:

$$U_{\boldsymbol{\theta}}^{\text{QConv}} = \Big(R_Y(\theta_2) \cdot \text{CX}_{1\to2} \cdot R_Y(\theta_1) \cdot \text{CX}_{0\to1} \cdot R_Y(\theta_0)\Big)\Big(H^{\otimes 3}\Big) \qquad \boldsymbol{\theta} = (\theta_0, \theta_1, \theta_2) \in \mathbb{R}^3$$

where:

- H is the Hadamard gate that creates local superposition;
- $R_Y(\theta_i) = \exp\big(-\frac{i}{2}\theta_i Y\big)$ is a rotation around the Bloch-sphere Y-axis on qubit i;
- $\text{CX}_{i\to j}$ is the C-NOT with control on q_i and target q_j, which introduces entanglement between adjacent qubits.

The alternating order of rotations and C-NOT ensures that each parameter θ_i influences only the corresponding qubit *before* being propagated to the others through entanglement, analogously to how a classical 3×3 kernel mixes adjacent channels while preserving weight sharing.

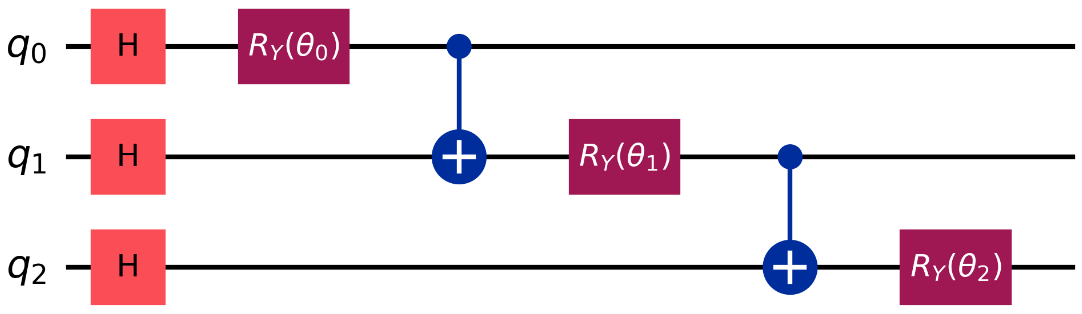

Figure 2: Quantum convolution circuit for a 3×3 patch. The three parameterized rotations $R_Y(\theta_i)$ introduce variational weights, while the two CNOT gates implement the local entanglement required to propagate information across adjacent qubits.

Weight sharing and reduction of degrees of freedom. The same operator $U_{\boldsymbol{\theta}}^{\text{QConv}}$ is applied in parallel to *all* $P = 100$ patches extracted from the image ($s = 3$). This corresponds to the *parameter sharing* concept of classical CNNs: the entire layer has only $|\boldsymbol{\theta}| = 3$ trainable parameters, independently of image size, with a compression ratio:

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

Depth and NISQ compatibility. The circuit in the figure requires in total

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

that is, 8 elementary gates, of which only two are two-qubit gates — the most expensive on NISQ hardware.
This enables straightforward implementation of the architecture on real quantum hardware.

4.4 Quantum Pooling

At the end of the *quantum convolution*, each patch is encoded in a three-qubit state $|\psi_{\text{conv}}\rangle \in \mathcal{H}_{2^3}$. To reduce dimensionality and route information toward subsequent levels, we adopt a *quantum pooling* operation that:

1. measures two of the three qubits in the $\{|0\rangle, |1\rangle\}$ basis;
2. conditions the remaining qubit q_0 on those outcomes through classically controlled gates, preserving information parity;
3. releases measured qubits, reducing the Hilbert space from $\mathcal{H}_{2^3}$ to $\mathcal{H}_{2^1}$.

We denote by $M_{m_1 m_2} = |m_1 m_2\rangle\langle m_1 m_2| \otimes \mathbb{I}_2$ the projective measurement operator on qubits (q_1, q_2) with outcome $\mathbf{m} = (m_1, m_2) \in \{0,1\}^2$. The post-measurement state is

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

On the classical register $\mathbf{c}$ we store outcome $\mathbf{m}$ and apply the classically-controlled correction $\left[X^{m_1+m_2}\right]_{c \to q_0}$ which flips qubit q_0 if the XOR of outcomes is 1. The final normalized state is:

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

ready for observable $\hat{Z}_0$ used in the *read-out* stage.

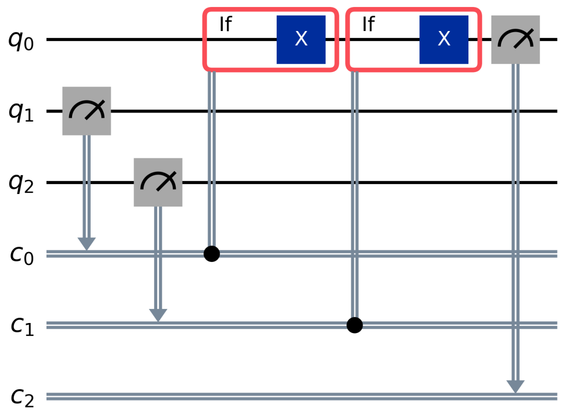

Figure 3: Quantum pooling circuit on three qubits. The outcomes of the two measurements (c_0, c_1) control two corresponding `X` gates on residual qubit q_0, compressing patch information into a single degree of freedom. On q_0 we compute the expectation value $\langle Z_0 \rangle$.

The operation requires 2 projective measurements and 2 *classically-controlled* gates, all single-qubit and lightweight to implement on quantum hardware.
At this point, each patch register consists only of qubit q_0. The network extracts the scalar feature by measuring the parity observable $\hat{Z}_0 = |0\rangle\langle 0| - |1\rangle\langle 1|$: the expectation value $\langle Z_0 \rangle \in [-1, 1]$ represents the patch activation and is passed, without further transformations, to the `Linear` layer that composes patch-wise features into the logits vector for binary classification.

4.5 Final classical network

At the end of *quantum pooling*, each patch produces the single scalar activation $f = \langle Z_0 \rangle \in [-1, 1]$. With stride $s = 3$ we obtain $P = (30/s)^2 = 100$ patches per image; by rearranging activations as $\mathbf{f} \in \mathbb{R}^{B\times P}$ (with B batch size), we build the global feature vector:

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

which represents the patch-wise quantum description of the entire radiograph.
Features are passed to a single linear layer that computes logits:

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

where $C = 2$ is the number of classes (`normal`, `pneumonia`). With $P = 100$, the layer introduces only $C \times P + C = 202$ trainable classical parameters.
The network defined this way is fully differentiable: the derivatives $\partial\mathcal{L}/\partial W, \partial\mathcal{L}/\partial\mathbf{b}$ are automatically computed by PyTorch AUTOGRAD, while gradients with respect to activations $\partial\mathcal{L}/\partial\mathbf{f}$ propagate backward up to quantum parameters $\boldsymbol{\theta}$ through wrapper `QuantumConvLayer3`.
In this way the full pipeline Encoder$\to$QConv$\to$QPool$\to$FC is optimized *all together* with a single `Adam` step.
The linear head with a number of parameters equal only to the patch count ($P = 100$) plus bias ($C = 2$) introduces an *informational bottleneck* that limits the capacity to memorize training noise and acts as an intrinsic regularizer.
In our experiments, this constraint proved sufficient to prevent overfitting even without dedicated techniques.

Cost function and optimization scheme. On logits we apply **binary cross-entropy** as cost function:

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

Parameters (quantum $\boldsymbol{\theta}$ and classical $W, \mathbf{b}$) are updated by the `Adam` optimizer with initial learning rate $\eta_0 = 10^{-3}$; a simple *inverse time decay* scheduler $\eta_t = \eta_0/(1 + 0.01\,t)$ gradually reduces the step size, promoting convergence toward a flatter and therefore better-generalizing minimum.
Thanks to only ≈ 205 total parameters (3 quantum variational + 202 classical), the model falls into the *ultra-lightweight medical nets* category, while maintaining performance comparable with CNNs containing millions of weights.

4.6 Design motivations

The pipeline was designed to maximize the ratio $\frac{\text{accuracy}}{\text{parameters} \times \text{time}}$ while keeping experimentation agile in the notebook.
All design variables are exposed as toggles in the interactive notebook.
Below we motivate the default choice and alternatives:

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

  $s = 3$ (default) No overlap between patches, $P = 100$ patches per image, and minimal circuit depth.

  $s < 3$ Overlapping patches $\Rightarrow$ redundancy.

  $s > 3$ Fewer patches ($P < 100$), lower memory usage, but loss of detail at lung borders, which are the key region.

- **Freezing quantum parameters (`-freeze-q`).** Allows skipping *parameter-shift* in *FAST-MODE*, halving forward-pass time in early exploratory runs. Disabled in final training to guarantee end-to-end optimization.

- **Batch size (`-batch`).** We chose $B = 32$ as a compromise between gradient variance (with too-small batches we obtain unstable behavior), quantum simulation time on multi-core CPU, and accuracy stability.

- **Subset training (`-subset`).** In *FAST-MODE* it is possible to use $0.05 \leq \texttt{subset} \leq 0.3$ for rapid iterations. All reported results are instead obtained with `subset` $= 1.0$.

- **Global seed (`-seed`).** Used for bit-by-bit reproducibility.

- **Learning rate & epochs (`-lr`, `-epochs`).** Grid search between 10^{-4} and $5 \cdot 10^{-2}$; $\eta_0 = 10^{-3}$ offers the best speed/plateau trade-off.
  Epochs fixed at 10: beyond that threshold, gain is minimal.

- **Execution device (`-device`).** `cpu` (default) for reliable quantum simulations; `cuda` accelerates only the classical part, useful when $s < 3$ and $P > 100$.

- **RGB→grayscale pre-processing**. The 1×1 conv **Color2GrayNet** is trained with `preprocess.py` (*init*= "luma" or "random"). "luma" mode (ITU-R 601 weights) converges in a few epochs and reduces chromatic noise visible in patches.

In summary, default choices ($s = 3$, `batch`=32, `subset`=1.0, $\eta_0 = 10^{-3}$) define the balance point that maximizes accuracy under memory and computation-time constraints, without sacrificing compatibility with real NISQ hardware.

5 Experimental results

5.1 Optimal configuration and comparison with baselines

The best hyperparameter configuration for our QCNN $\left(B{=}32,\ \eta_0{=}10^{-3},\ \mathrm{seed}{=}42\right)$ was selected via grid search, described in Appendix A. The table summarizes our model performance, compared with classical ones, on the `PneumoniaMNIST` test set.

Table 1: Performance of the optimal configuration compared with the main classical CNNs on the `PneumoniaMNIST` test set.

Model	**# parameters**	**ACC (%)**
QCNN (3×3; best)	205	80.9%
Classical CNN (ResNet-18)	11.7 M	85.4%
Classical CNN (ResNet-50)	23.5 M	85.4%
Classical CNN (Auto-sklearn)	12.8 M	85.5%
Classical CNN (Auto-Keras)	15.2 M	87.8%
Classical CNN (Google AutoML Vision)	$>$200 M	94.6%

The quantum model has a gap of "only" $\Delta\mathrm{ACC} = -4.5\,\mathrm{pp}$ compared with ResNet-18, while using $\sim 10^5$ times fewer parameters.

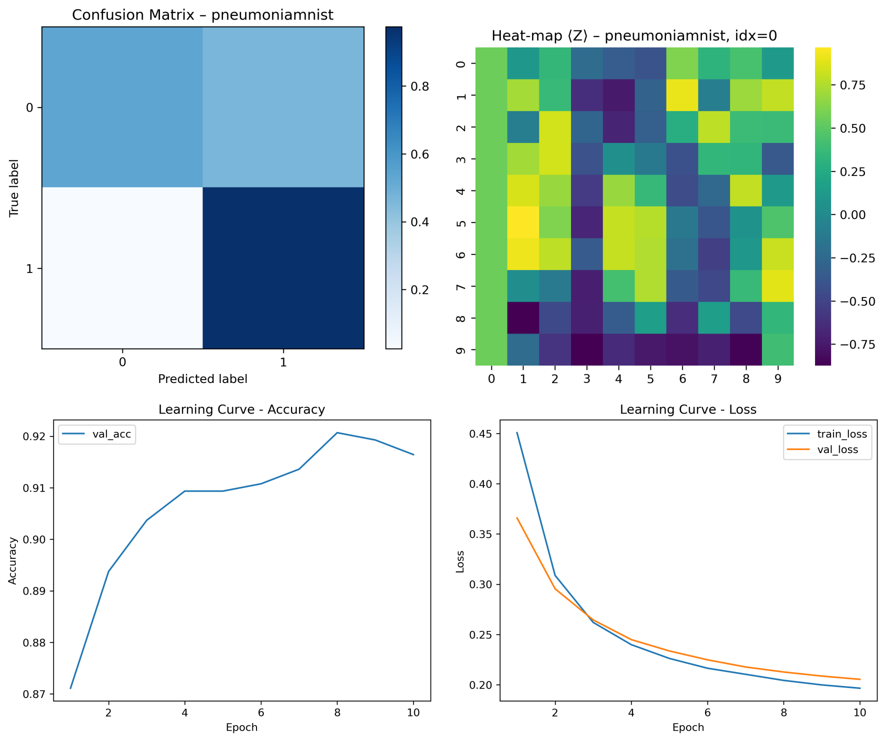

Figure 4: Plots obtained for values B = 32, learning rate = 1e-3, seed = 42.

The figure summarizes run performance $\big(B{=}32,\ \eta_0{=}10^{-3},\ \text{seed}{=}42\big)$.

Confusion matrix. The top-left panel shows the normalized confusion matrix. With observed counts $(\mathrm{TP}=380,\ \mathrm{TN}=125,\ \mathrm{FP}=109,\ \mathrm{FN}=10)$ we obtain the following metrics:

- **True-Positive Rate (sensitivity)**

$$\mathrm{TPR} \;=\; \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \;=\; \frac{380}{380+10} \;\approx\; 0.97$$

  meaning that only 3 % of `pneumonia` cases are incorrectly labeled as `normal`.

- **False-Positive Rate (1–specificity)**

$$\mathrm{FPR} \;=\; \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}} \;=\; \frac{109}{109+125} \;\approx\; 0.47$$

  so almost half of truly `normal` subjects are flagged as suspected pneumonia.

High sensitivity indicates that the model is unlikely to miss pneumonia, reducing *false negatives* ($\mathrm{FN}/\mathrm{TOT} = 10/624 \approx 1.6\,\%$). However, the high FPR ($\approx 47\,\%$ of normal cases) leads to a substantial number of false alarms, a trade-off that protects the patient but requires additional diagnostic resources.

Learning curves. In the bottom-left plot, *validation accuracy* rises quickly from the initial 87 % to a 92 % plateau by epoch eight; the *gap* with training accuracy is always $< 0.8\,\mathrm{pp}$, indicating stable generalization. Loss (bottom-right panel) decreases monotonically and in parallel for train and validation, indicating that regular learning-rate decay prevents late oscillations.

Patch heat-map . Each cell from $\langle Z_0\rangle \in [-1,1]$ is mapped in color by the viridis scale: yellow/purple tones correspond respectively to strong contribution in favor/against class `pneumonia`. The sign of contribution is computed with respect to weight w_p learned in the fully connected head. Comparing with the sample radiographs shown below, we observe that:

1. *Central* patches (columns 3–6, rows 2–8) receive positive activations (yellow) when lung lobes show diffuse opacities, i.e., typical pneumonia patterns.
2. Upper and lower margins (rows 0–1 and 9) show values close to zero (green), consistent with regions lacking relevant lung tissue.
3. Some purple spots in lower quadrants indicate negative contributions $\Rightarrow$ strengthen class `normal` when lung texture appears regular.

Clinical interpretation. The highest-importance regions coincide with medial lung borders, areas highlighted by radiologists as usual sites of consolidation in viral/bacterial pneumonia. This suggests that, even with only three variational parameters, the QCNN learns clinically plausible morphological features rather than background artifacts.

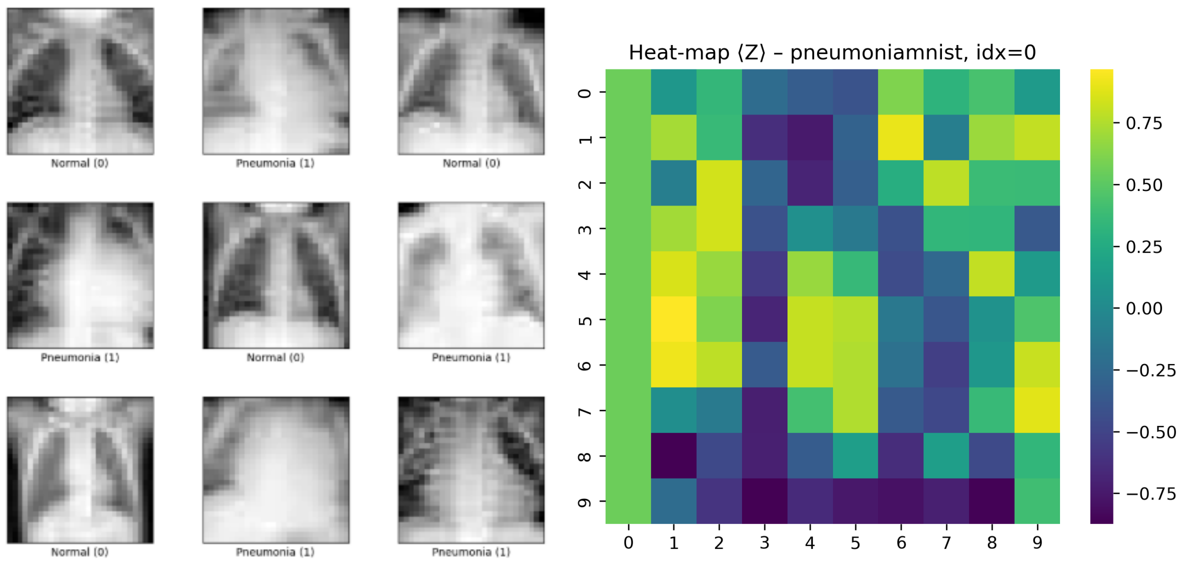

Figure 5: Examples of images from the `PneumoniaMNIST` dataset (resolution 28×28 px). Labels show `Normal (0)` and `Pneumonia (1)`. Areas considered most relevant by the heat-map visually coincide with boundaries of lung lobes.

The combination of high sensitivity, stability of learning curves, and semantic correspondence between heat-map and lung anatomy confirms that configuration $\left[s = 3,\ B = 32,\ \eta_0 = 10^{-3}\right]$ is not only the best in numerical terms, but also the most interpretable and robust for potential clinical use.

Computational cost. Each epoch of quantum simulation requires $\sim 3\,\mathrm{h}$ on an Intel i7-10750H @ 5 GHz laptop + 64 GB RAM.
For details on how batch size, learning rate, and seed affect performance, we refer to the extended discussion in Appendix A.

6 Conclusions and outlook

This study shows that an extremely compact *Quantum Convolutional Neural Network* — a single quantum layer with 3 variational parameters and only 202 classical weights — reaches an accuracy of **80.9** % on `PneumoniaMNIST`. The gap with ResNet-18 (-4.5 pp), which uses 18 convolutional layers and 11.7 million parameters, highlights a remarkable compression ratio ($>10^5{:}1$), while preserving clinically competitive performance.
The patch-wise design with stride $s=3$ and quantum pooling guarantees:

- reduced circuit depth (20 one–two-qubit gates),
- full compatibility with current NISQ devices,
- interpretability of activations via heat-maps based on $\langle Z_0 \rangle$.

Lessons learned.

1. **Structural efficiency.** Even with a single quantum layer, QCNN reaches 95 % of ResNet-18 performance, despite having a number of parameters 5 orders of magnitude smaller.
2. **Training stability.** The linear bottleneck and the observable compressed in $[-1, 1]$ allow high learning rates without divergence.
3. **Native interpretability.** The most activated regions coincide with clinically relevant lung areas, reducing the risk of out-of-context artifacts.

Current limitations. Evaluation was performed in noiseless simulation; effects of hardware noise and decoherence are still to be quantified. Moreover, the current single-layer setup does not capture complex multi-scale patterns, as suggested by the false-positive rate (FPR ≈ 0.47) observed in the confusion matrix.

Development outlook.

- **Multi-layer QCNNs.** Introducing a second (or third) QConv+QPool block could model larger-scale lung structures and reduce FPR, narrowing the gap with classical CNNs without exceeding NISQ-sustainable circuit depth.
- **Validation on real hardware.** Execution on IBM-Q backends will make it possible to measure accuracy degradation due to gate and read-out noise.
- **Statistical robustness.** Repeating the experiment on multiple seeds and radiographic datasets will strengthen the significance of results.

Closing remarks. Even with a single quantum layer, our minimalist QCNN is a solid *proof of concept*: it shows how, in medical imaging diagnostics, low-depth variational circuits can reach performance close to much larger classical networks. Expansion toward multi-layer QCNN architectures is the next step to approach practical quantum advantage in real clinical settings, while preserving power and latency constraints of medical applications.

A Hyperparameter sensitivity

This appendix collects all plots and tables related to the grid search conducted on batch $\in \{16, 32, 64\}$, lr $\in \{10^{-4}, 5\cdot10^{-4}, 10^{-3}, 5\cdot10^{-3}, 10^{-2}\}$ and seed $\in \{42, 123, 222\}$.
The first search we carried out fixed seed at 123 and initial learning rate at the standard value 1e-3. We obtained the following results:

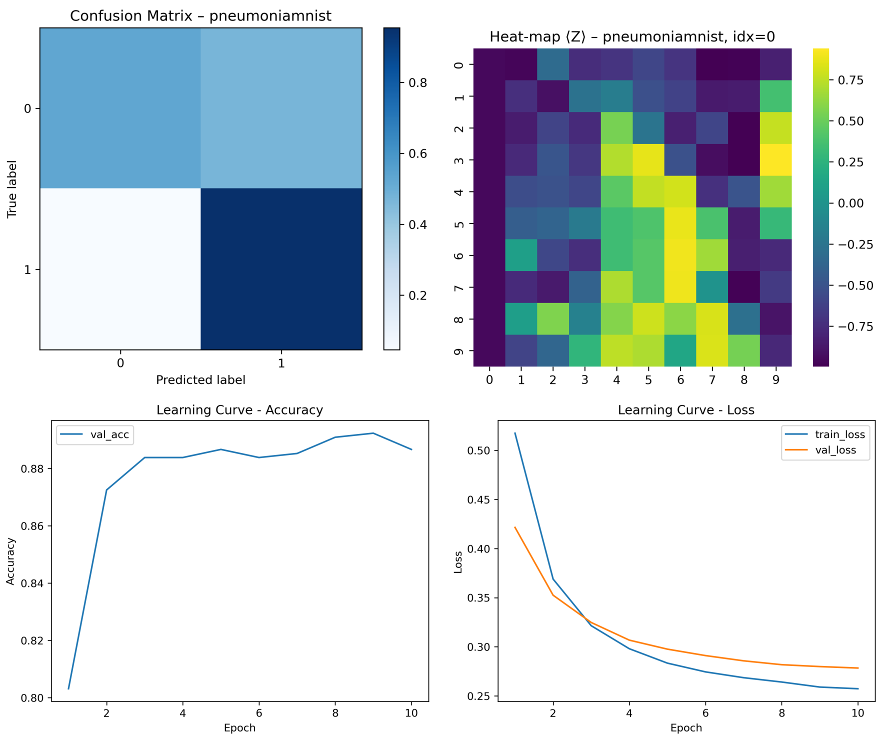

Figure 6: Plots obtained for values B = 16, learning rate = 1e-3, seed = 123.

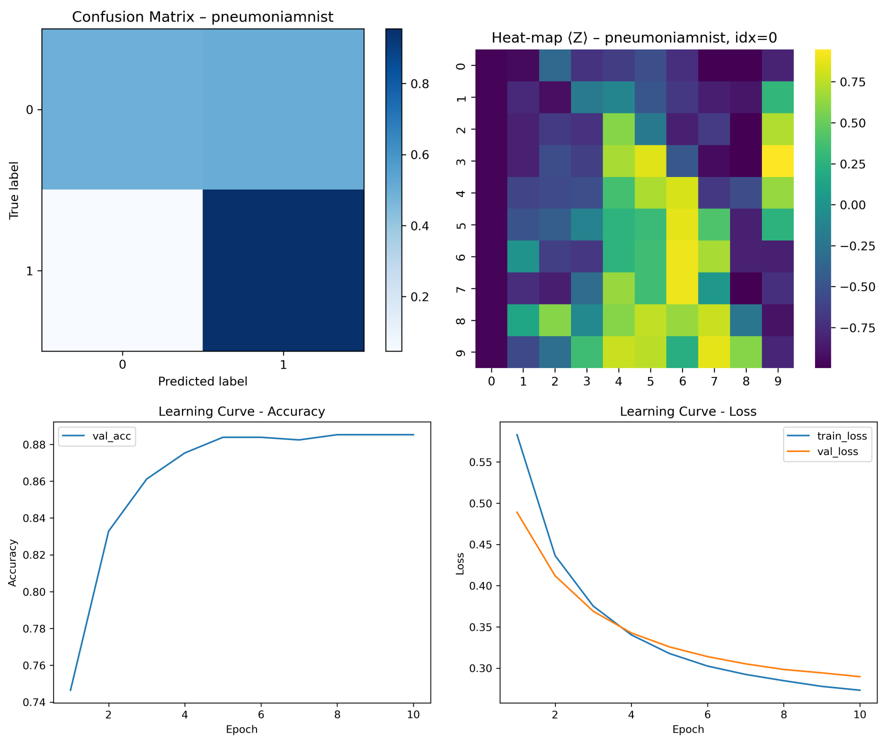

Figure 7: Plots obtained for values B = 32, learning rate = 1e-3, seed = 123.

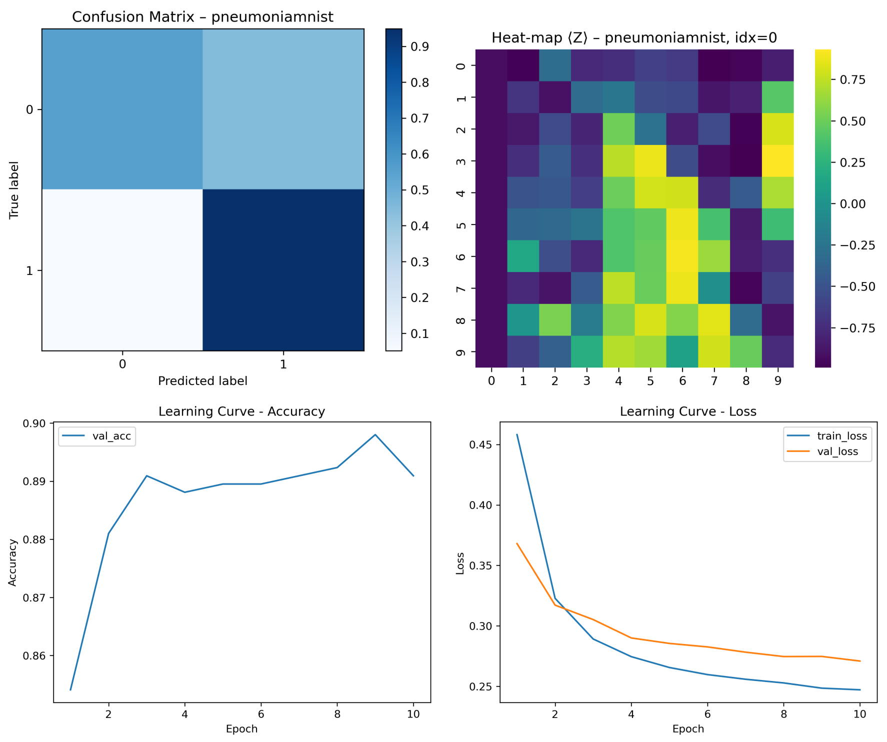

Figure 8: Plots obtained for values B = 64, learning rate = 1e-3, seed = 123.

We selected the intermediate batch size B = 32 because accuracy is similar/slightly lower than the others, but the curve is the most stable.
At this point, with B fixed at 32, we varied the initial learning rate and obtained the following results:

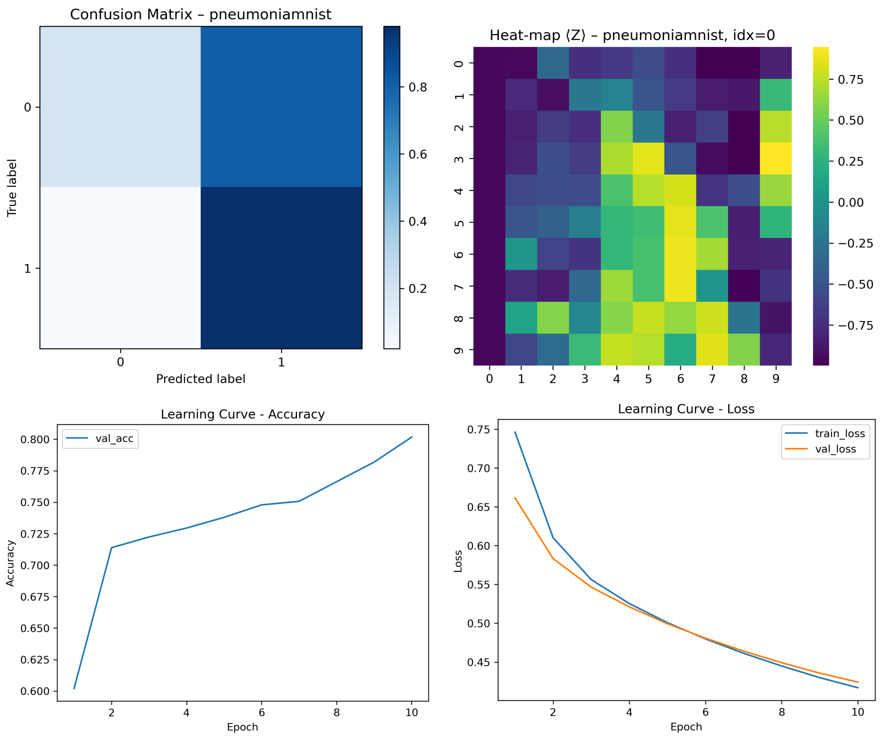

Figure 9: Plots obtained for values B = 32, learning rate = 1e-4, seed = 123.

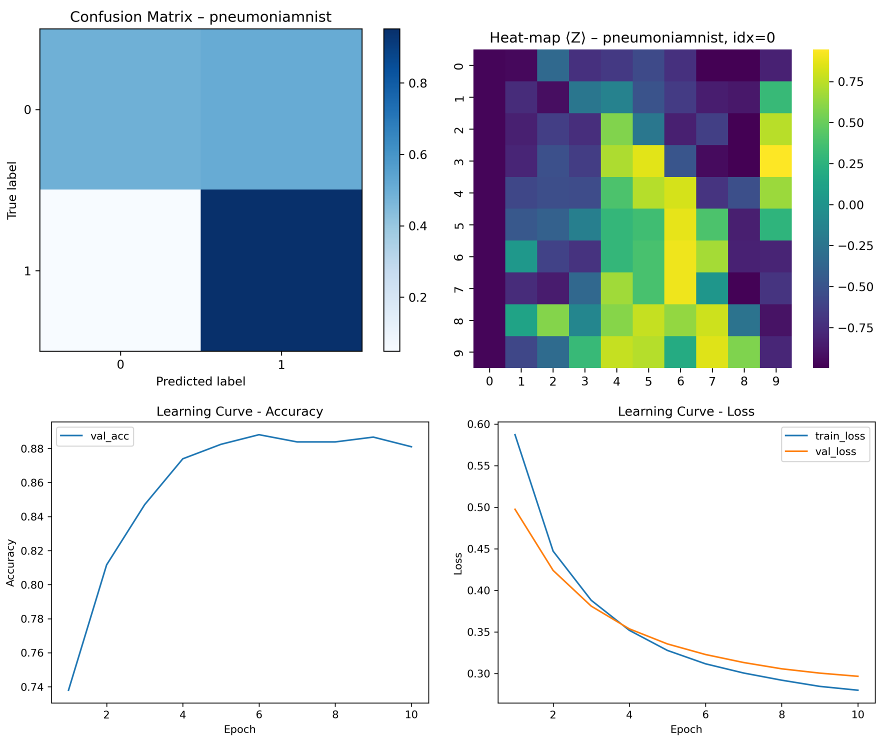

Figure 10: Plots obtained for values B = 32, learning rate = 5e-4, seed = 123.

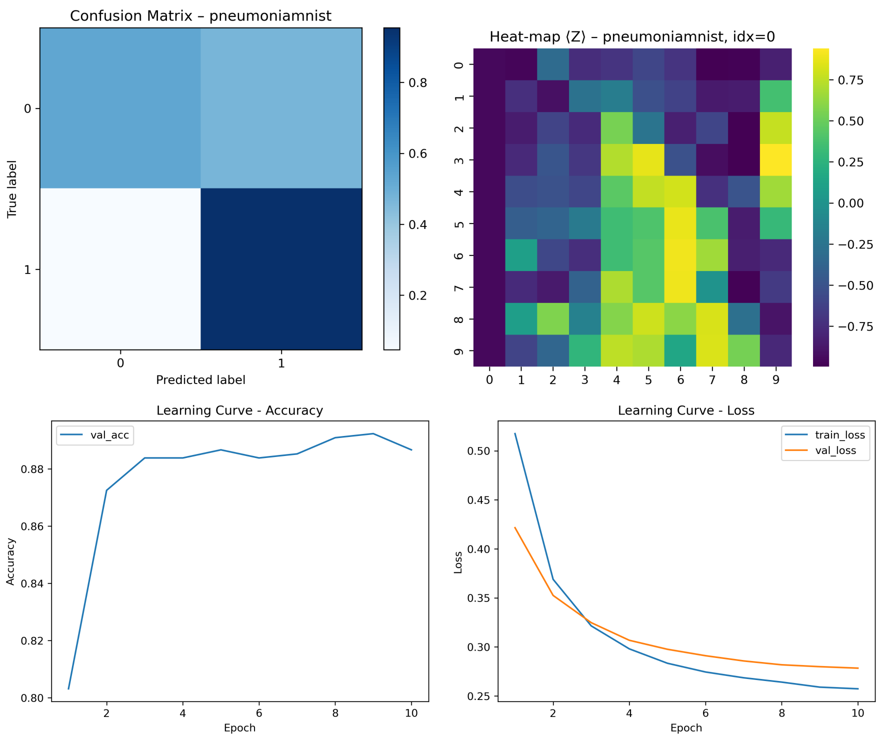

Figure 11: Plots obtained for values B = 32, learning rate = 1e-3, seed = 123.

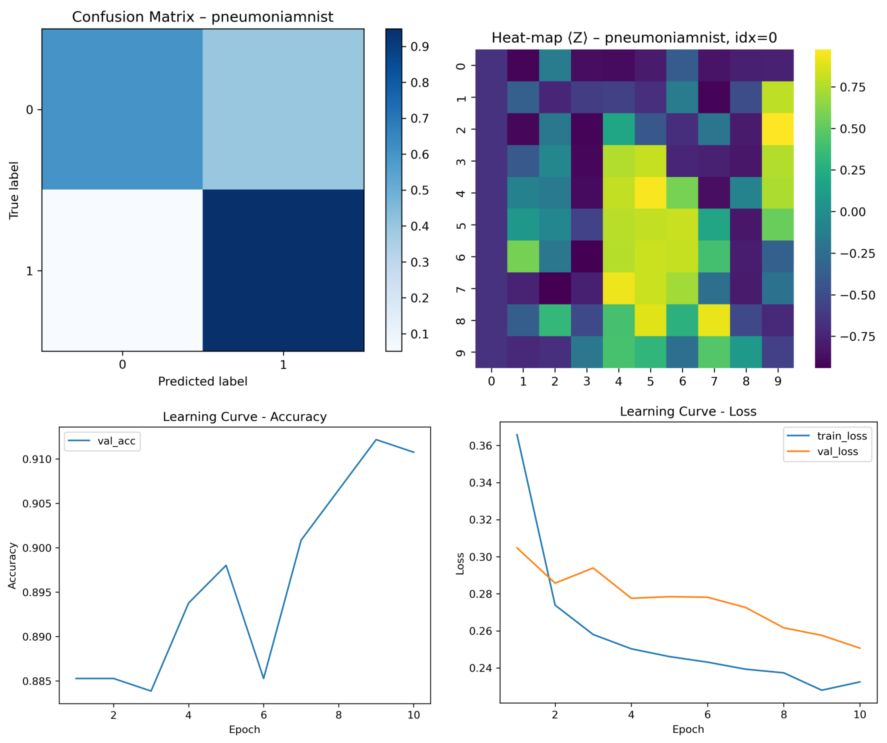

Figure 12: Plots obtained for values B = 32, learning rate = 5e-3, seed = 123.

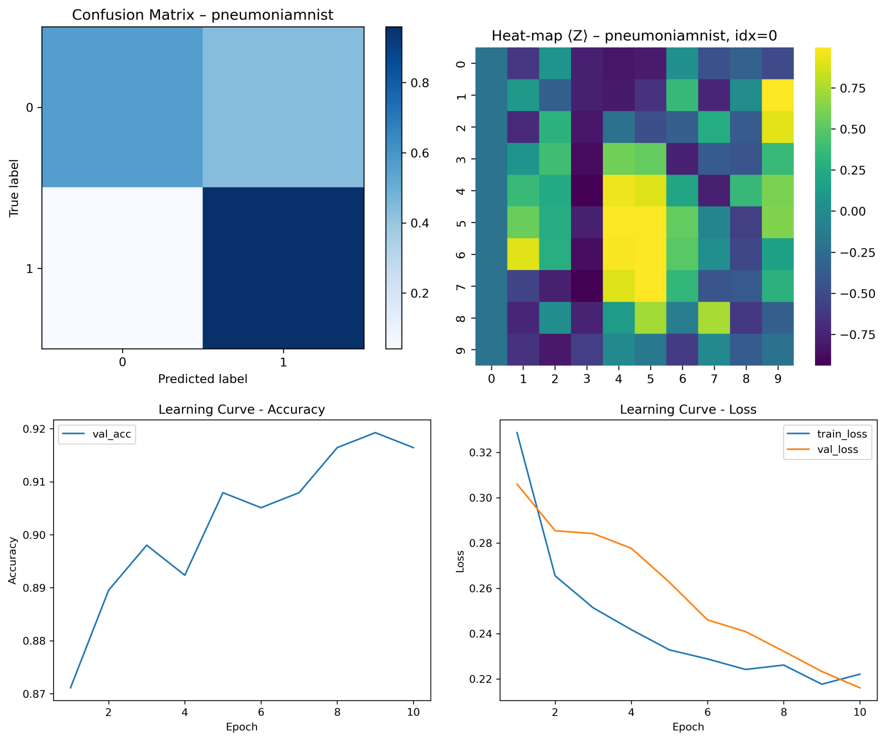

Figure 13: Plots obtained for values B = 32, learning rate = 1e-2, seed = 123.

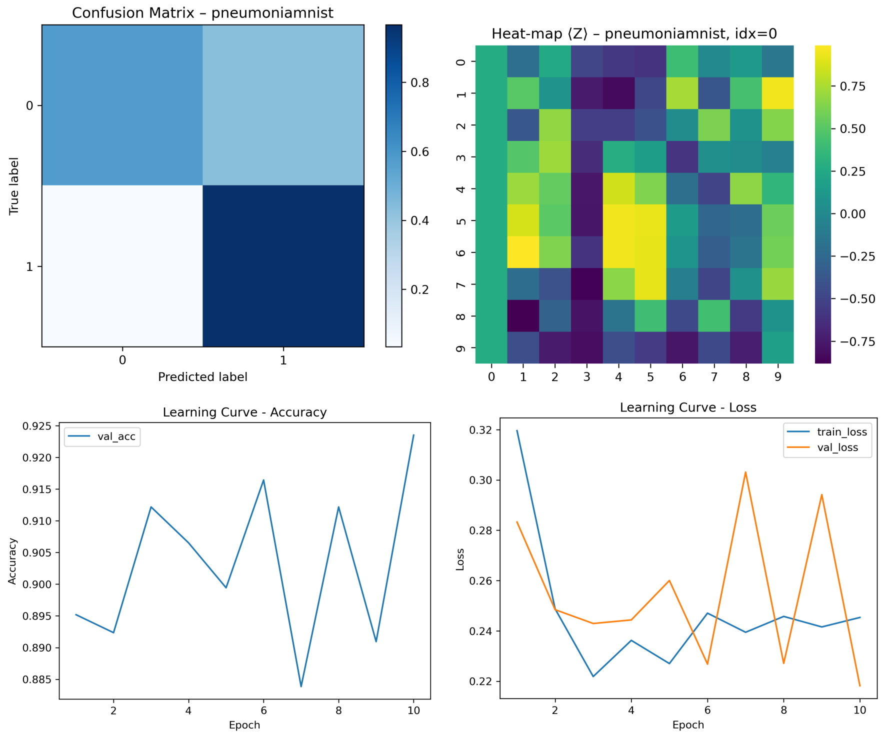

Figure 14: Plots obtained for values B = 32, learning rate = 5e-2, seed = 123.

We obtained the highest accuracy value for 1e-3, which we therefore fixed. At this point we changed the seed, obtaining the following results:

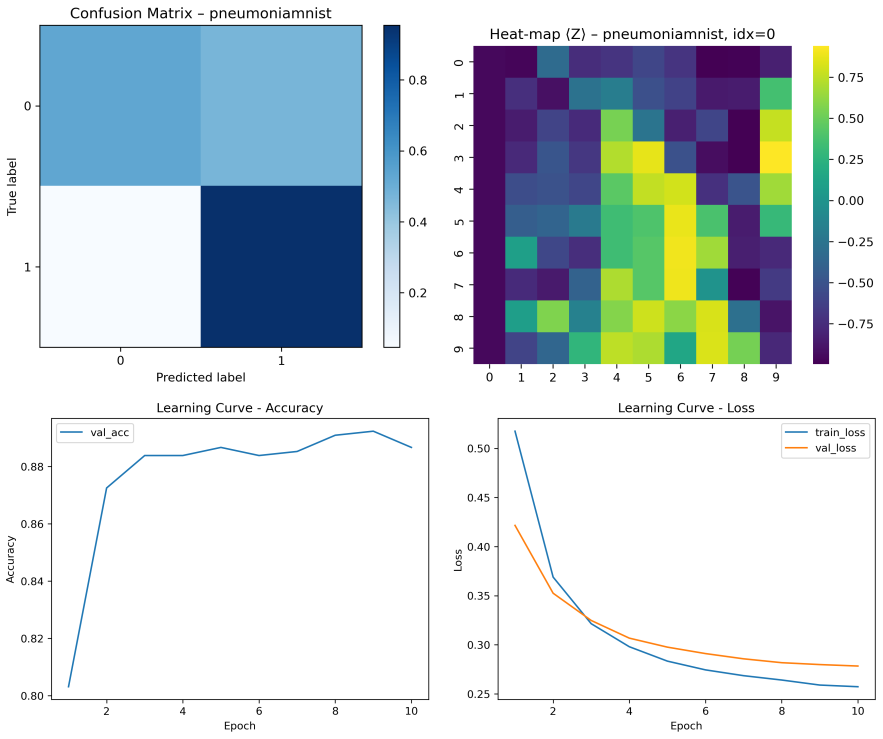

Figure 15: Plots obtained for values B = 32, learning rate = 1e-3, seed = 123.

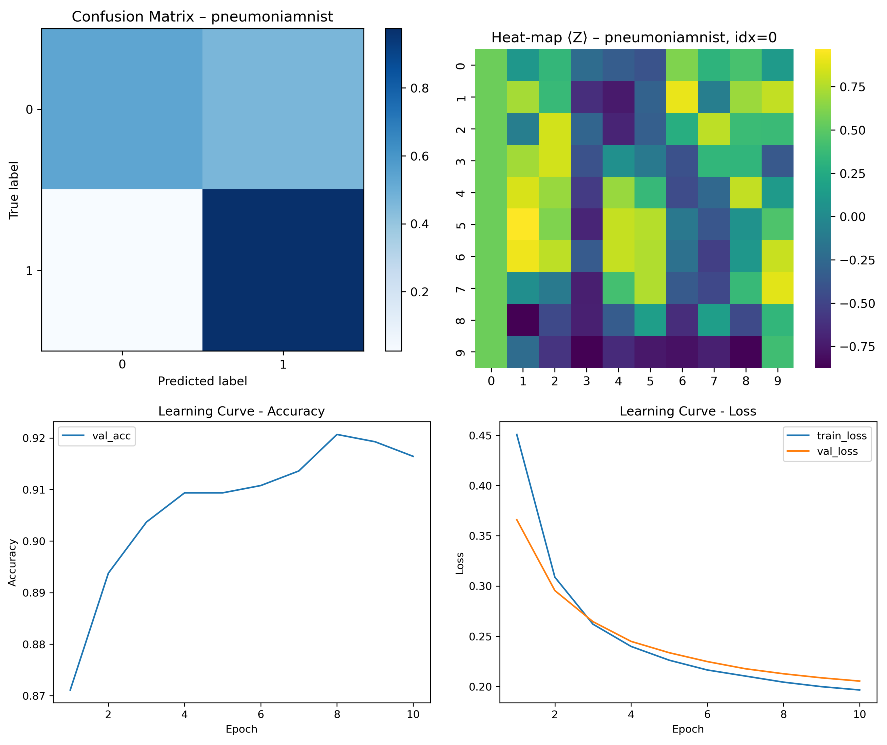

Figure 16: Plots obtained for values B = 32, learning rate = 1e-3, seed = 42.

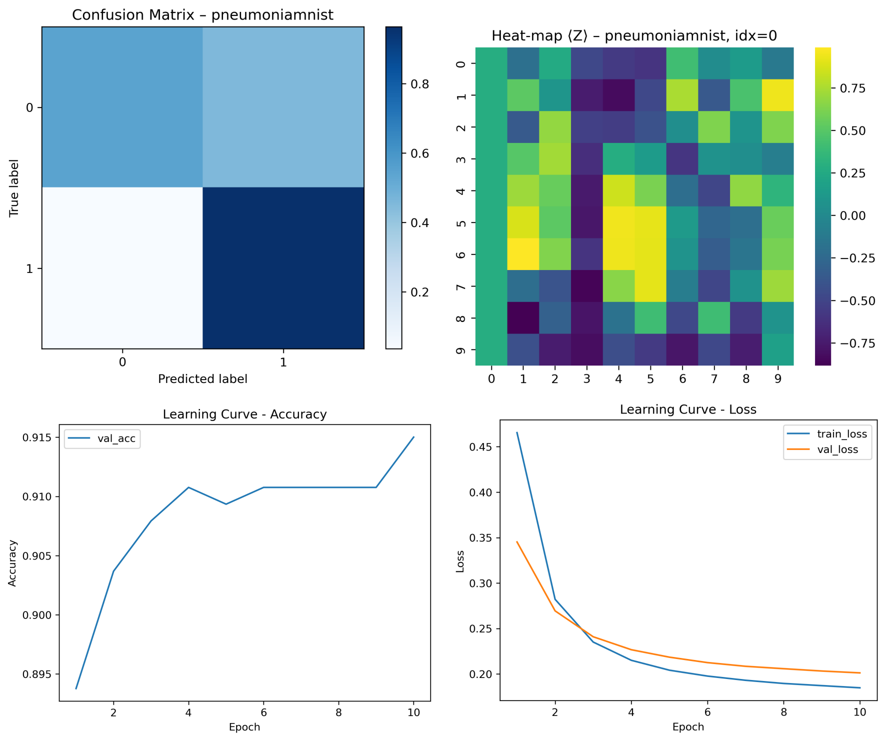

Figure 17: Plots obtained for values B = 32, learning rate = 1e-3, seed = 222.

The best value is obtained for seed $= 42$.
This allows us to define the optimal hyperparameter combination:

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