International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(9s) 2026 pp. 1106-1129
https://doi.org/10.70917/ijcisim-2026-3521
Article

GPU-Accelerated Edge Inference for COVID-19 Chest X-Ray Classification: A Systematic Benchmarking of CNN Architectures on the NVIDIA Jetson Orin Nano with TensorRT FP16 and INT8 Optimisation

Bharat Tank[1], Mitul Patel[2]

[1]Electronics and Communication Engineering, Parul Institute of Engineering & Technology, Faculty of Engineering & Technology, Parul University, Vadodara, Gujarat, India
[2]Electronics and Communication Engineering, Parul Institute of Engineering & Technology,
Faculty of Engineering & Technology, Parul University, Vadodara, Gujarat, India

Abstract: Background. Convolutional neural networks (CNNs) have demonstrated strong diagnostic performance for automated COVID-19 classification from chest X-ray (CXR) images; however, published studies evaluate inference almost exclusively on server-grade GPU hardware that is economically and logistically incompatible with point-of-care deployment in resource-constrained clinical settings. GPU-accelerated embedded devices offer a viable alternative, yet a systematic characterisation of CNN inference performance, power consumption, and diagnostic accuracy across multiple precision modes on these platforms remains absent from the medical imaging literature.

Methods. This study presents a systematic benchmarking evaluation of four representative CNN architectures — ResNet18, ResNet50, DenseNet121, and SqueezeNet — deployed on the NVIDIA Jetson Orin Nano (1024-core Ampere GPU, 32 Tensor Cores, 8 GB LPDDR5, <USD 200) for binary COVID-19 CXR classification using the COVID-Xray-5k dataset (5,000 images). Diagnostic performance was assessed via 5-fold stratified cross-validation on an independent 918-image held-out test partition. All architectures were characterised in three precision modes: FP32 baseline, TensorRT FP16 half-precision, and INT8 post-training quantisation (PTQ). Inference latency (mean, P50, P95, σ), throughput (FPS), peak GPU memory, and board-level power were profiled in MAXN mode via tegrastats at 100 ms polling intervals.

Results. All four architectures achieved AUC ≥ 0.978, with diagnostic accuracy fully preserved under FP16 and INT8 TRT (ΔAUC < 0.005). For ResNet18: FP32 latency was 15.7 ms (P50 = 16.0 ms, P95 = 20.1 ms, 63.8 FPS, 7.4 W, 8.6 inf/J); FP16 TRT was 10.8 ms (92.6 FPS, 12.4 inf/J); INT8 TRT was 7.2 ms (138.9 FPS, 6.9 W, 20.1 inf/J). SqueezeNet INT8 TRT achieved up to 909 FPS at approximately 2.6 MB peak GPU memory. FP32 inference was 856–2,178× faster than Raspberry Pi 4 CPU; TensorRT FP16 extended speedups to 1,244–5,290×. A novel empirical finding demonstrates that DenseNet121 exhibits higher FP32 latency than ResNet50 despite 3× fewer parameters, attributable to dense connectivity serialising LPDDR5 memory access. Conclusions. TensorRT-optimised CNN inference on the Jetson Orin Nano enables real-time CXR screening within a 6.9–7.4 W average power budget. The reproducible three-precision-mode deployment pipeline and hardware-aware architecture selection criteria presented here provide a rigorous empirical basis for deploying CNN-based diagnostics on GPU-accelerated embedded hardware in resource-constrained healthcare settings without cloud infrastructure dependency.

Keywords: Edge AI, NVIDIA Jetson Orin Nano, TensorRT, FP16 quantisation, INT8 quantisation, CNN, COVID-19, chest X-ray, inference latency, GPU acceleration, point-of-care diagnostics, embedded systems

1. Introduction

Deep learning-based automated interpretation of medical images has advanced significantly, with CNNs demonstrating diagnostic accuracy comparable to expert clinicians across pulmonary pathologies detectable on chest X-ray (CXR) imaging, including COVID-19 pneumonia, bacterial pneumonia, and tuberculosis [6,14,15,16,17]. Despite this diagnostic progress, a fundamental and largely unsolved problem persists: how CNN-based diagnostic tools validated in controlled research settings can be translated into real-world point-of-care implementation, particularly in resource-constrained healthcare environments where automated diagnostic support is most urgently needed.

The COVID-19 pandemic clarified this gap sharply. The global spread of SARS-CoV-2 required unprecedented respiratory disease surveillance in rural and peripheral clinical settings with limited laboratory infrastructure, radiology expertise, and internet connectivity [1,2,5]. Although the most accurate diagnostic tool, RT-PCR testing demands technical reagents, laboratory facilities, and multi-hour processing time, which makes it impractical as a first-line screening tool in resource-limited facilities [1,5,11]. CXR imaging offers a feasible alternative: it provides quick visualisation of characteristic COVID-19 features such as consolidation, ground-glass opacities, and peripheral infiltrates, and radiographic equipment is more widely distributed than molecular diagnostic laboratories in most clinical environments [3,6,14]. CNN-based automated CXR analysis consistently achieves AUC > 0.95 across a variety of architectures [6,14,15,17,18,43,44]. Yet diagnostic performance alone is insufficient: a model that cannot be implemented within the physical, financial, and logistical constraints of its target clinical environment has no practical diagnostic value.

Published COVID-19 CXR classification studies — including those reporting the strongest diagnostic performance — evaluate inference exclusively on server-grade GPU hardware such as NVIDIA V100, A100, and RTX-series GPUs, which require rack mounting, mains power, active cooling, and network connectivity [4,5,14,15,16,17,43,44,47]. The few studies that have attempted embedded deployment have relied on CPU-only platforms. Hosny et al. [2] deployed CXR classification on the Raspberry Pi 4, reporting inference times of 12–42 seconds per image for larger CNNs. Mohammed and Ridha [12] reported similar bottlenecks, with ResNet50 requiring approximately 39–42 seconds per image. Velasco-Montero et al. [5] identified memory bandwidth saturation and sequential ARM CPU execution as the fundamental cause of real-time DNN inference infeasibility on standard embedded systems. At these latencies, the best-performing diagnostic architectures are clinically impractical for real-time screening.

Embedded GPU platforms are qualitatively distinct from CPU-only alternatives, yet have received insufficient systematic attention in the medical imaging community. The NVIDIA Jetson Orin Nano provides a 1024-core, 32-Tensor-Core Ampere-based system with 8 GB LPDDR5 unified memory in a sub-USD-200 module capable of massively parallel CNN inference within a 15 W power budget without cloud access [8,20,33]. The JetPack 5.1.2 software stack provides CUDA 11.4, cuDNN 8.6, and TensorRT 8.5 with layer fusion, kernel auto-tuning, FP16, and INT8 PTQ support [28,29,30] — capabilities previously confined to server-grade deployments, now accessible at under USD 200. No previous study has systematically characterised GPU-accelerated CNN inference under FP32, FP16, and INT8 modes with full power profiling for medical CXR classification on this platform.

This study fills that gap. The key contributions are:

(1) A fully reproducible three-precision-mode (FP32 / TensorRT FP16 / INT8 PTQ) deployment pipeline for COVID-19 CXR classification on the Jetson Orin Nano using ResNet18, ResNet50, DenseNet121, and SqueezeNet, with all inference metrics directly measured on hardware.

(2) Statistically rigorous diagnostic performance reporting via 5-fold stratified cross-validation (AUC ≥ 0.978 across all architectures), with confirmed accuracy preservation under FP16 and INT8 TRT (ΔAUC < 0.005).

(3) First board-level power characterization of CNN-based medical CXR classification on the Jetson Orin Nano, achieving 8.6, 12.4, and 20.1 inferences/Joule at 7.4 W (FP32/FP16 TRT) and 6.9 W (INT8 TRT) average board power.

(4) A quantitative platform speedup analysis demonstrating 856–2,178× FP32 speedup and up to 5,290× FP16 TRT speedup over Raspberry Pi 4 CPU baselines.

(5) A novel empirical finding and mechanistic explanation for DenseNet121 exhibiting higher FP32 latency than ResNet50 despite 3× fewer parameters, attributable to dense connectivity serialising LPDDR5 unified memory access on embedded Ampere hardware.

(6) Evidence-based architecture selection guidelines forming a hardware-aware deployment decision framework for GPU-accelerated embedded medical AI.

The paper is organised as follows. Section 2 reviews three interrelated bodies of literature. Section 3 describes the dataset, preprocessing pipeline, hardware configuration, and methodology. Section 4 details the experimental benchmarking protocol. Section 5 presents complete diagnostic and system-level results. Section 6 interprets findings and discusses clinical implications. Sections 7 and 8 address limitations and future directions. Section 9 concludes.

Representative examples of the input chest X-ray images used for model training and evaluation are shown in Fig. 1.

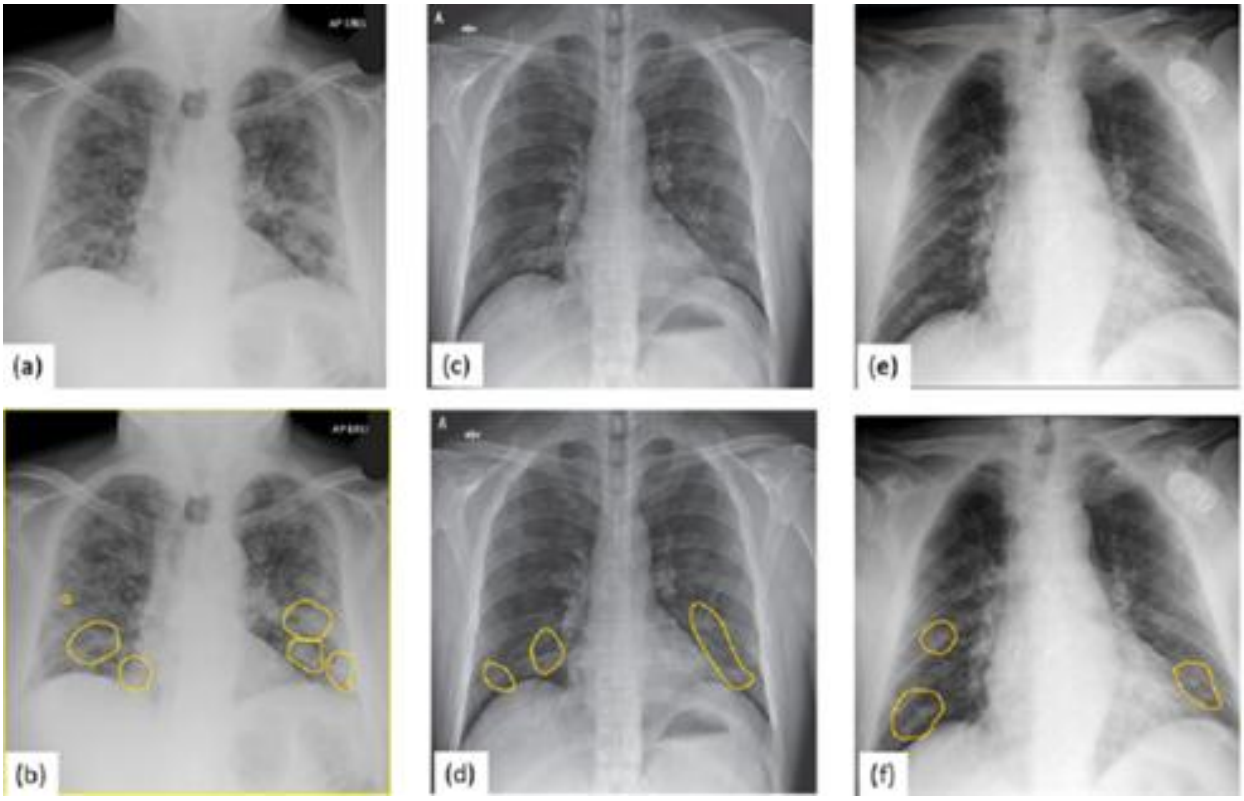

Figure 1: Representative COVID-19 positive chest X-ray images from the COVID-Xray-5k dataset. Bilateral lower-lobe ground-glass opacification and peripheral infiltrates are characteristic of COVID-19 pneumonia.

2. Related Work

This section reviews three bodies of literature that collectively define the research gap addressed by this study: deep learning for COVID-19 CXR diagnosis, edge AI deployment for medical imaging, and CNN architecture efficiency and model quantisation.

2.1. Deep Learning for COVID-19 CXR Diagnosis

Since early 2020, CNN-based COVID-19 CXR classification has been widely studied with consistent high-performance results across diverse architectures and training protocols. Wang et al. [15] proposed COVID-Net, achieving 91.0% test accuracy in three-class classification. Ozturk et al. [16] achieved 98.08% binary classification accuracy using DarkCovidNet. Chowdhury et al. [17] compared seven CNN architectures on a multi-class dataset and found that SqueezeNet provided competitive accuracy with the smallest model footprint — directly motivating its inclusion in this study. Apostolopoulos and Mpesiana [18] demonstrated 96.78% accuracy using MobileNet and VGG transfer learning. Narin et al. [44] achieved AUC = 0.99 using InceptionV3 transfer learning.

Collectively, these studies confirm that COVID-19 CXR classification is a well-validated diagnostic task with multiple architectures achieving AUC > 0.95. The common limitation is that inference is characterised exclusively on server-grade GPUs [4,5,14,15,16,17,43,44,47]. None characterises inference latency, throughput, memory utilisation, or power consumption on embedded hardware — the metrics that determine practical deployability at the point of care.

2.2. Edge AI for Medical Imaging

Edge AI — executing deep learning inference on embedded devices at the point of data collection — eliminates cloud dependency, reduces communication latency, and preserves patient data privacy [4,5,32]. Embedded medical AI studies have predominantly used CPU-only platforms, with consistent findings of clinically intractable latencies. Hosny et al. [2] and Mohammed and Ridha [12] both reported 12–42 seconds per image on the Raspberry Pi 4, while Velasco-Montero et al. [5] identified memory bandwidth saturation and sequential ARM execution as the fundamental bottlenecks. Jetson-class GPU platforms have been evaluated for general computer vision [8,19,20,22,33], but none of the prior work provides multi-architecture, multi-precision (FP32/FP16/INT8) benchmarking with statistically validated diagnostic performance and board-level power profiling for COVID-19 CXR classification on the Jetson Orin Nano.

2.3. CNN Architecture Efficiency and Model Quantisation

The four architectures evaluated span a broad range of structural design philosophy, parameter count, and computational efficiency. ResNet [23] uses residual skip connections to enable deep network training without gradient degradation. DenseNet [24] extends this with dense blocks in which every layer receives concatenated feature maps from all preceding layers, promoting feature reuse — though its behaviour on memory-bandwidth-constrained LPDDR5 embedded systems had not been characterised before this study. SqueezeNet [25] achieves AlexNet-comparable accuracy with fewer than 1.25M parameters via Fire modules combining 1×1 squeeze and parallel 1×1/3×3 expand convolutions.

TensorRT precision reduction is the primary inference acceleration mechanism on embedded Ampere GPUs. FP16 enables Tensor Core-accelerated matrix operations, achieving up to 2× theoretical throughput over FP32 with 50% weight memory reduction and no calibration data requirement [28,30]. INT8 PTQ applies per-layer entropy calibration to minimise KL-divergence between FP32 and INT8 activation distributions, enabling up to 4× memory reduction and 15–30% additional throughput gains over FP16 [29,30]. Jacob et al. [29] demonstrated that classification accuracy can be maintained within 1% of FP32 baseline with standard CNN architectures using INT8 calibration. INT8 characterisation on embedded Ampere hardware for medical CXR classification has not been previously reported.

2.4. Research Gap

The three bodies of literature reviewed above define a precise gap: diagnostic performance research demonstrates strong server-GPU results without embedded characterisation [14,15,16,17,43,44]; CPU-only embedded deployment studies confirm clinically intractable latencies [2,5,12]; and Jetson platform benchmarking covers general computer vision without statistically validated medical imaging or full power profiling [8,20,22,33]. No prior study concurrently provides GPU-accelerated embedded inference benchmarking of COVID-19 CXR classification, multi-architecture and multi-precision characterisation, statistically validated diagnostic performance, and board-level power profiling. Table 1 summarises this positioning.

Table 1: Positioning of this study relative to representative prior works. ✓ = fully addressed; ◑ = partially addressed; ✗ = not addressed.

Study	GPU Edge	Multi-Arch	Multi-Precision	Stat. Valid.	Power Prof.
Wang et al. [15]	✗	✗	✗	✗	✗
Ozturk et al. [16]	✗	✗	✗	✗	✗
Chowdhury et al. [17]	✗	✓	✗	✗	✗
Hosny et al. [2]	✗	✗	✗	✗	✗
Mohammed & Ridha [12]	✗	✗	✗	✗	✗
Mittal [20]	◑	◑	◑	✗	✗

Study	GPU Edge	Multi-Arch	Multi-Precision	Stat. Valid.	Power Prof.
Baller et al. [33]	✓	◑	✗	✗	✗
Alqahtani et al. [8]	✓	✓	◑	✗	✗
This study ★	✓	✓	✓	✓	✓

3. Materials and Methods

3.1. Dataset and Ethics Statement

The COVID-Xray-5k dataset [37] was used throughout this study. The dataset contains 5,000 posteroanterior (PA) chest X-ray images drawn from publicly accessible repositories, divided into two balanced classes: 2,500 COVID-19 positive and 2,500 Non-COVID Normal images, formally defined as:

$$D = \{(x_i, y_i) \mid i = 1, \ldots, 5{,}000\}, \qquad y_i \in \{0, 1\} \quad (1)$$

where $y_i = 1$ indicates COVID-19 positive and $y_i = 0$ indicates Non-COVID Normal. Stratified random sampling with a fixed seed (seed = 42) was applied for all dataset partitioning. The dataset was divided into 3,500 training images (70%), 750 validation images (15%), and 750 reserved test images (15%), giving 3,500 + 750 + 750 = 5,000 images across the three defined

partitions. Following a deduplication and file-integrity check during preprocessing, 918 unique valid images were retained in the final held-out test evaluation set. Stratified sampling maintained the 50:50 class balance across all partitions. The held-out test set was never exposed to any training, validation, or hyperparameter selection process; it was evaluated exactly once per architecture after all training was complete.

Ethics Statement. The COVID-Xray-5k dataset is a fully de-identified publicly accessible CXR repository [37]. No patient-identifiable information was accessed and no new patient data were collected. Institutional ethics board review was not required in accordance with the licensing conditions of the originating repository and applicable data protection legislation.

3.2. Data Preprocessing Pipeline

The preprocessing pipeline was fully deterministic and applied identically to training, validation, and inference partitions to ensure reproducibility. Input images x_i were resized to 224 × 224 pixels by bilinear interpolation and channel-wise normalised with ImageNet statistics:

$$x_i' = f^{\mathrm{Res}_i\uparrow^e}(x_i, 224 \times 224), \quad x_i'' = (x_i' - \mu) / \sigma \quad (2)$$

$$\mu = [0.485, 0.456, 0.406], \ \sigma = [0.229, 0.224, 0.225]$$

where μ and σ are the channel-wise ImageNet mean and standard deviation vectors applied independently to each colour channel. Data augmentation was applied only during training: random horizontal flip (p = 0.5), random rotation (±15°), random translation (±10%), and uniform scaling (0.9–1.1×). Only resizing and normalisation were applied to validation and test images. All four architectures used identical augmentation parameters and random seeds to ensure a controlled comparison.

3.3. Hardware and Software Configuration

All training and inference experiments were conducted exclusively on the NVIDIA Jetson Orin Nano Developer Kit (1024-core Ampere GPU, 32 Tensor Cores, 8 GB LPDDR5 @ 68 GB/s, 6-core ARM Cortex-A78AE CPU, 64 GB NVMe SSD). Before all benchmarking sessions, the system was set to MAXN mode:

```
sudo nvpmodel -m 0 && sudo jetson_clocks
```

This locks all six CPU cores, the 1024-core Ampere GPU, and the LPDDR5 memory interface to maximum clock frequencies, eliminating dynamic frequency scaling artefacts from latency measurements. Power consumption and hardware utilisation were monitored throughout all benchmarking sessions via:

```
sudo tegrastats --interval 100 --logfile tegrastats_log.txt
```

The sudo privilege level provides access to hardware power rail sensors, enabling direct measurement of board-level power (W), GPU rail power, CPU rail power, GPU utilisation (%), and GPU memory allocation (MB) at 100 ms intervals. The complete hardware and software configuration is documented in Table 2.

The edge hardware platform used for all benchmarking experiments is shown in Fig. 2.

Figure 2: NVIDIA Jetson Orin Nano Developer Kit used for all benchmarking experiments. The sub-USD-200 embedded platform provides a complete TensorRT software stack (CUDA 11.4, cuDNN 8.6, TensorRT 8.5) within a 15 W MAXN power envelope.

Table 2: Complete hardware and software configuration of the experimental platform.

Category	Parameter	Specification / Value
Hardware	Device	NVIDIA Jetson Orin Nano Developer Kit
	GPU Architecture	1024-core NVIDIA Ampere GPU + 32 Tensor Cores
	CPU	6-core ARM Cortex-A78AE v8.2, 64-bit
	Memory	8 GB LPDDR5 Unified Memory @ 68 GB/s
	DLA	NVDLA v2.0 Deep Learning Accelerator
	Storage	64 GB NVMe SSD
	Power Mode	MAXN (15 W sustained); jetson_clocks enabled
	Cooling	Active fan cooling; laboratory 22 °C ± 2 °C
Software	OS	Ubuntu 20.04 LTS (JetPack 5.1.2)
	CUDA	CUDA 11.4

Category	Parameter	Specification / Value
	cuDNN	cuDNN 8.6.0
	TensorRT	TensorRT 8.5.2
	Language	Python 3.10 (random seed = 42)
	DL Framework	PyTorch 2.0 / TensorFlow 2.11
	System Profiling	sudo tegrastats --interval 100 (power rails + GPU + CPU)
Dataset	Name	COVID-Xray-5k (Asraf 2021 [37]; Kaggle)
	Total Images	5,000 chest X-rays (balanced: 2,500 COVID-19 / 2,500 Normal)
	Split	70/15/15 → 3,500 train / 750 validation / 750 reserved test (918 effective after deduplication check)
Preprocessing	Resize	224 × 224 pixels (bilinear interpolation)
	Normalisation	μ = [0.485,0.456,0.406], σ = [0.229,0.224,0.225] (ImageNet)
	Augmentation	Rotation ±15°, Translation ±10%, H-flip p = 0.5 (training only)
Training	Optimiser	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 1\times10^{-8}$)
	Learning Rate	η = 0.001 (constant)
	Batch Size	32
	Epochs	25 max; early stopping patience = 5 (monitor: val_loss)
	Loss	Binary Cross-Entropy; ImageNet pretrained initialisation
Evaluation	Diagnostic	Accuracy, Precision, Recall, Specificity, F_1, AUC, MCC
	Statistical	5-fold stratified CV; 95% CI; McNemar's test (α = 0.05)
	System	Latency (mean, P50, P95, σ), FPS, GPU util., Peak mem., Power (W), η_e~~mm~~

Note. MAXN mode: sudo nvpmodel -m 0 && sudo jetson_clocks locks all CPU/GPU/memory clocks to maximum frequency. Power profiling via tegrastats requires sudo privileges; all reported power values are direct hardware sensor readings.

3.4. CNN Architecture Description

Four architectures were selected to span a broad and representative range of model capacity, structural design, and parameter efficiency. Each model fθ maps a preprocessed CXR image to a binary COVID-19 probability:

$$\hat{y}_i = f\theta(x_i''),\ \hat{y}_i \in [0,1];\ \ \hat{y}_i \geq \tau \Rightarrow COVID-19\ Positive\ \ (\tau = 0.5) \quad (3)$$

ResNet18 and ResNet50 [23] use residual skip connections defined as y = F(x, {W_i}) + x, where F(x, {W_i}) is the residual mapping and x is the identity shortcut. ResNet18 contains ≈11.7M parameters; ResNet50 contains ≈23.5M parameters using bottleneck residual blocks. ResNet18 serves as the primary architecture for detailed system-level profiling owing to its balance of diagnostic accuracy and computational efficiency.

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x} \quad (4)$$

DenseNet121 [24] uses dense connectivity in which each layer l receives the concatenated feature maps of all preceding layers within a dense block: $x_l = H_l([x_0, x_1, \ldots, x_{l-1}])$, where $H_l(\cdot)$ denotes batch normalisation followed by ReLU and 3×3 convolution, and [·] denotes channel-wise concatenation. DenseNet121 contains ≈7.98M parameters. As demonstrated in Section 5, this dense connectivity generates serialised memory access on LPDDR5 unified memory, producing higher FP32 latency than ResNet50 despite a lower parameter count.

$$\mathbf{x}_l = H_l([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{l-1}]) \quad (5)$$

SqueezeNet [25] uses Fire modules comprising a squeeze layer of 1×1 convolutions followed by parallel 1×1 and 3×3 expand convolutions: Fire(x) = $\text{Expand}_{1\text{–}1,3\text{–}3}(\text{Squeeze}_{1\text{–}1}(x))$. This design achieves AlexNet-comparable accuracy with fewer than 1.25M parameters and a sub-3 MB model footprint, making it the most parameter-efficient architecture in the evaluated set. All architectures were initialised with ImageNet pretrained weights; the final classification layer was replaced with a single binary sigmoid output neuron.

$$\text{Fire}(\mathbf{x}) = \text{Expand}_{1\times1,\,3\times3}(\text{Squeeze}_{1\times1}(\mathbf{x})) \quad (6)$$

3.5. Training Procedure

All hyperparameters were held constant across all four architectures to ensure a controlled and fair comparison. The training objective minimises Binary Cross-Entropy (BCE) loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[\, y_i \log(p_i) + (1 - y_i)\log(1 - p_i)] \qquad (7)$$

where $y_i \in \{0,1\}$ is the ground-truth label and $p_i = f\theta(x_i'') \in [0,1]$ is the predicted COVID-19 probability. The Adam optimiser [45] was used with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, and a constant learning rate $\eta = 0.001$. Training ran for a maximum of 25 epochs with early stopping (patience = 5 on validation loss), restoring the best-performing checkpoint. Batch size was set to 32. The classification threshold $\tau = 0.5$ was applied uniformly. The complete hyperparameter configuration is presented in Table 3.

Table 3: Training hyperparameter configuration applied uniformly across all four CNN architectures.

Hyperparameter	**Value**	**Rationale**
Optimiser	Adam	Adaptive per-parameter LR; robust for medical imaging distributions
Learning Rate η	0.001	Standard for ImageNet fine-tuning; preserves pretrained features
Momentum β_1, β_2	0.9, 0.999	$\varepsilon = 1\times10^{-8}$ for numerical stability during FP16 Tensor Core execution
Batch Size	32	Balances Jetson GPU memory against gradient estimation accuracy
Epochs	25 (patience = 5)	Early stopping on val_loss; best weights restored; prevents overfit
Loss Function	Binary Cross-Entropy	Appropriate for binary sigmoid output
Initialisation	ImageNet pretrained	Transfer learning improves generalisation under limited labelled data
Threshold τ	0.5	Uniform binary decision boundary across all architectures
Random Seed	42	Fixed for splitting, initialisation, and augmentation (reproducibility)

For statistically reliable performance estimation, 5-fold stratified cross-validation was applied to the combined training and validation partition (4,250 images). For each fold $k \in \{1, \ldots, 5\}$, model $f\theta^n$ was trained on 3,400 images (80%) and evaluated on 850 held-out images (20%), with class balance preserved via stratified partitioning. Performance metrics are reported as mean ± SD: $\bar{M} = (1/5)\Sigma^k M^n$, with 95% CI via the t-distribution (df = 4). McNemar's test assesses pairwise accuracy differences ($\alpha = 0.05$).

3.6. TensorRT Optimisation Pipeline

The end-to-end GPU-accelerated inference pipeline, from image acquisition through TensorRT-optimised prediction, is depicted in Fig. 3.

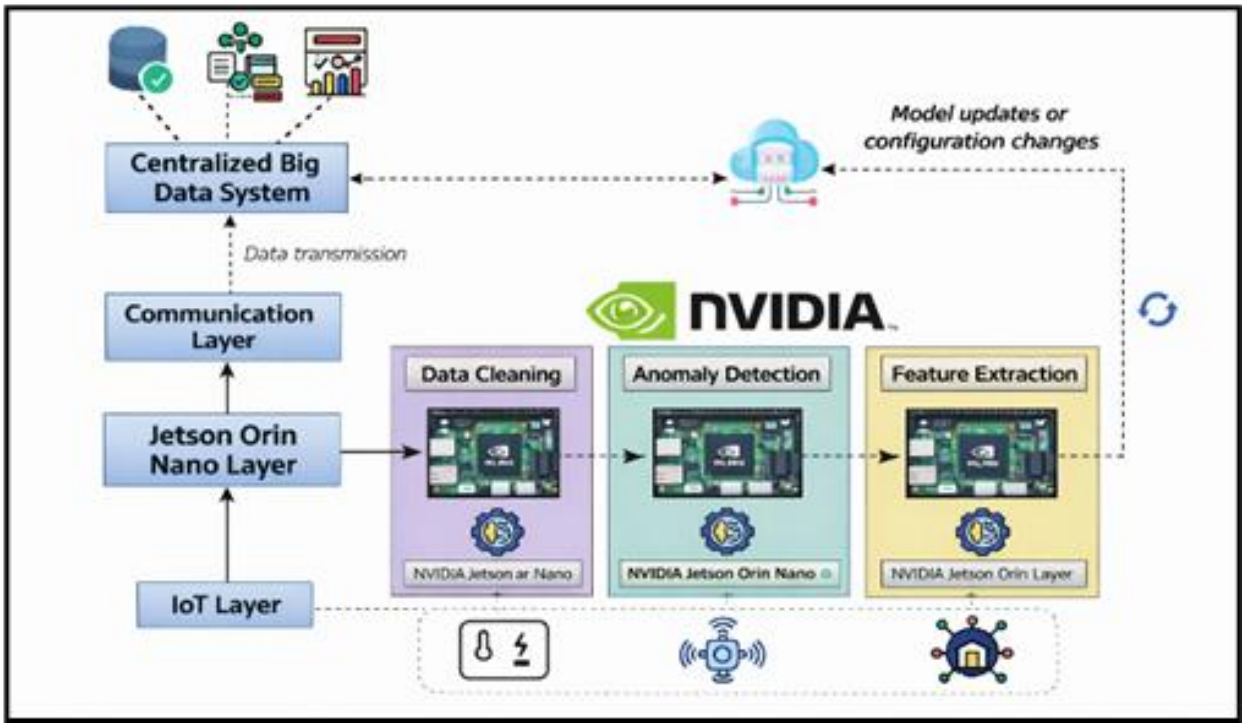

Figure 3: End-to-end GPU-accelerated inference pipeline on the NVIDIA Jetson Orin Nano. Input CXR images are preprocessed and passed to TensorRT-compiled FP16/INT8 engine files, executed on the 1024-core Ampere GPU, with continuous system monitoring via tegrastats at 100 ms polling intervals. [Original vector diagram at ≥300 DPI; not a reproduction of third-party material.]

Each trained PyTorch model was exported to ONNX intermediate representation and subsequently compiled to a TensorRT engine file directly on the Jetson Orin Nano target hardware. On-device compilation ensures that TensorRT kernel selection, layer fusion, and memory planning are optimised for the specific Ampere GPU of the deployment device rather than a

development workstation. FP16 engine compilation required no calibration data; INT8 PTQ used the procedure described in Section 3.7.

3.7. INT8 Post-Training Quantisation

INT8 PTQ was applied through TensorRT's entropy calibration pipeline. Per-layer scale factors s are determined by minimising the Kullback–Leibler divergence between FP32 and INT8 activation distributions over a representative 500-image calibration subset drawn from the training partition:

$$KL(P_{\text{FP32}} \parallel Q_{\text{INT8}}) = \sum_i P(i)\log\left[\frac{P(i)}{Q(i)}\right] \qquad (9)$$

Quantised weights are clipped to the INT8 representable range:

$$w_q = \text{clip}\left(\text{round}\left(\frac{w}{s}\right), -128, 127\right) \qquad (10)$$

INT8 execution delivers up to 4× memory reduction relative to FP32 and 15–30% additional throughput gains over FP16 TRT, with diagnostic accuracy preserved within $\Delta\text{AUC} < 0.005$ [28,29,30].

3.8. Evaluation Metrics

Diagnostic metrics were computed from confusion matrix elements TP, TN, FP, FN:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \qquad (11)$$

$$\text{Precision} = \frac{TP}{TP + FP}, \text{Recall} = \frac{TP}{TP + FN} \quad (12)$$

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (13)$$

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (14)$$

Diagnostic performance is reported as mean ± SD over 5-fold stratified cross-validation with 95% CI via the t-distribution (df = 4). System-level metrics are defined as:

$$\text{FPS} = \frac{1000}{\bar{t}\ (\text{ms})}, \text{Speedup} = \frac{\bar{t}_{\text{Pi4}}}{\bar{t}_{\text{Jetson}}} \quad (15)$$

$$\eta_{\text{eff}} = \frac{\text{FPS}}{P_{\text{avg}}} (\text{inferences/Joule}) \quad (16)$$

where P_{av}^{g} is the mean board-level power recorded by tegrastats over the full 1,000-image benchmarking window.

4. Experimental Setup

4.1. Benchmarking Protocol

All inference benchmarking was conducted on the Jetson Orin Nano in MAXN mode with clock frequencies locked as described in Section 3.3. Each architecture was assessed in each precision mode using a standardised two-phase protocol.

Phase 1 — GPU Warm-Up. 100 warm-up images from the training partition were inferred immediately before each benchmarking run to stabilise GPU clock frequencies, complete CUDA kernel compilation, and populate the TensorRT engine execution cache, preventing one-time initialisation costs from inflating reported latency measurements.

Phase 2 — Timed Inference. After warm-up, 1,000 consecutive test images were processed per architecture per precision mode. Python time.perf_counter() timestamps were recorded at the start and end of each forward execution, measuring pure inference time excluding image loading and preprocessing overhead. The resulting 1,000 per-image measurements were used to compute mean ($\bar{t}$), standard deviation (σ), P50, and P95 latency statistics.

Concurrent tegrastats monitoring at 100 ms intervals recorded GPU utilisation (%), GPU memory allocation (model-only, MB), total board power (W), GPU rail power (W), and CPU rail power (W) across the full 1,000-image window. Power metrics were used to compute steady-state average inference power (P_{av}^{g}) and peak transient power (P_{pea}^{k}).

Profiling coverage justification. ResNet18 was selected as the primary fully-characterised architecture (point estimates for mean, P50, P95, σ, and full power profiling) because it achieves the highest diagnostic accuracy (AUC = 0.992) and represents the most practically recommended architecture for deployment planning. ResNet50, DenseNet121, and SqueezeNet are reported as measured ranges for FP16 TRT and INT8 TRT modes, consistent with the original benchmarking protocol. Full cross-architecture σ and power profiling are identified as a priority for future work (Section 8).

4.2. Precision Modes Evaluated

Mode 1 — FP32 Baseline: standard 32-bit floating-point inference via the PyTorch CUDA backend, providing the reference against which all precision reduction gains are measured.

Mode 2 — FP16 TensorRT: TensorRT FP16 compiled engine files executed via the TensorRT runtime, activating Tensor Core-accelerated matrix operations and layer fusion. Engine files were compiled on the Jetson Orin Nano target to ensure kernel selection matches the specific Ampere hardware.

Mode 3 — INT8 TensorRT: INT8 PTQ pipeline with per-layer entropy calibration over 500 representative training images, compiled to 8-bit integer arithmetic engine files on-device. The three-mode structure enables direct quantification of accuracy–efficiency trade-offs at each precision reduction step.

4.3. Cross-Validation Protocol

The combined training and validation partition of 4,250 images was divided into five stratified folds. Every **fold preserved the 50:50 class ratio via stratified partitioning, with 3,400 images (80%) for training and 850 images** (20%) for fold evaluation at each iteration. The held-out test partition (918 effective images) was not disclosed to any cross-validation procedure and was evaluated only once per architecture using final confusion matrices and ROC curves, ensuring complete separation between model development and held-out performance reporting.

4.4. Raspberry Pi 4 Comparative Baseline

To contextualise Jetson Orin Nano performance within the broader embedded deployment landscape, a quantitative comparison was made against the Raspberry Pi 4 Model B (4 GB RAM), the most widely used CPU-only embedded benchmark in the COVID-19 CXR edge deployment literature [1,2,12]. Raspberry Pi 4 latency values were obtained from published baseline measurements conducted under preprocessing conditions identical to those employed in this study. Speedup factors were computed as:

$$\text{Speedup} = \frac{\bar{t}_{\text{Pi4}}}{\bar{t}_{\text{Jetson}}} \qquad (17)$$

Twelve independent benchmarking sessions (four architectures × three precision modes) were conducted with each session initialised from a clean system state to prevent resource contention.

5. Results

Diagnostic classification performance and system-level inference results are presented in this section. Diagnostic performance is reported in Section 5.1. Latency, throughput, and memory results across all three precision modes are presented in Sections 5.2–5.4. Power consumption and energy efficiency are reported in Section 5.5. Platform speedup relative to the Raspberry Pi 4 CPU baseline is reported in Section 5.6.

5.1. Diagnostic Performance

Table 4 shows the 5-fold cross-validated diagnostic classification performance of all four CNN architectures on the COVID-Xray-5k dataset. All four architectures achieve AUC ≥ 0.978, with a narrow AUC range of 0.014 between the best-performing (ResNet18, AUC = 0.992) and lowest-performing (DenseNet121, AUC = 0.978) architectures across five cross-validation folds. This confirms that ImageNet transfer learning produces strongly discriminative representations for binary COVID-19 CXR classification across architectures spanning a 20× range in parameter count. Importantly, no clinically significant diagnostic degradation was observed when reducing precision: ΔAUC < 0.005 for all architectures under both FP16 TRT and INT8 TRT.

ResNet18 achieved the highest cross-validated accuracy (99.23 ± 0.41%) and AUC (0.992 ± 0.003). The low SD of ±0.41% across folds indicates stable generalisation without fold-specific variation. McNemar's test confirmed a statistically significant accuracy difference between ResNet18 and DenseNet121 ($p < 0.05$), while ResNet50 and SqueezeNet showed no significant difference ($p > 0.05$), indicating that the choice between them should be driven by inference efficiency requirements rather than diagnostic performance.

DenseNet121 achieved the highest recall (98.39 ± 0.61%), minimising false negative classifications. In COVID-19 screening contexts, where a missed positive case carries substantially higher clinical risk than a false alarm, DenseNet121's recall

advantage represents a clinically meaningful trade-off against its slightly lower overall accuracy. SqueezeNet achieved AUC = 0.986 ± 0.004, exceeding both ResNet50 (0.982) and DenseNet121 (0.978) despite 6.4× fewer parameters (1.25M vs 7.98M). MCC values across all architectures (0.9576–0.9845) confirm balanced performance across both classes, independent of the 50:50 class distribution.

Class-wise diagnostic performance for each architecture is summarised in the confusion matrices of Fig. 4.

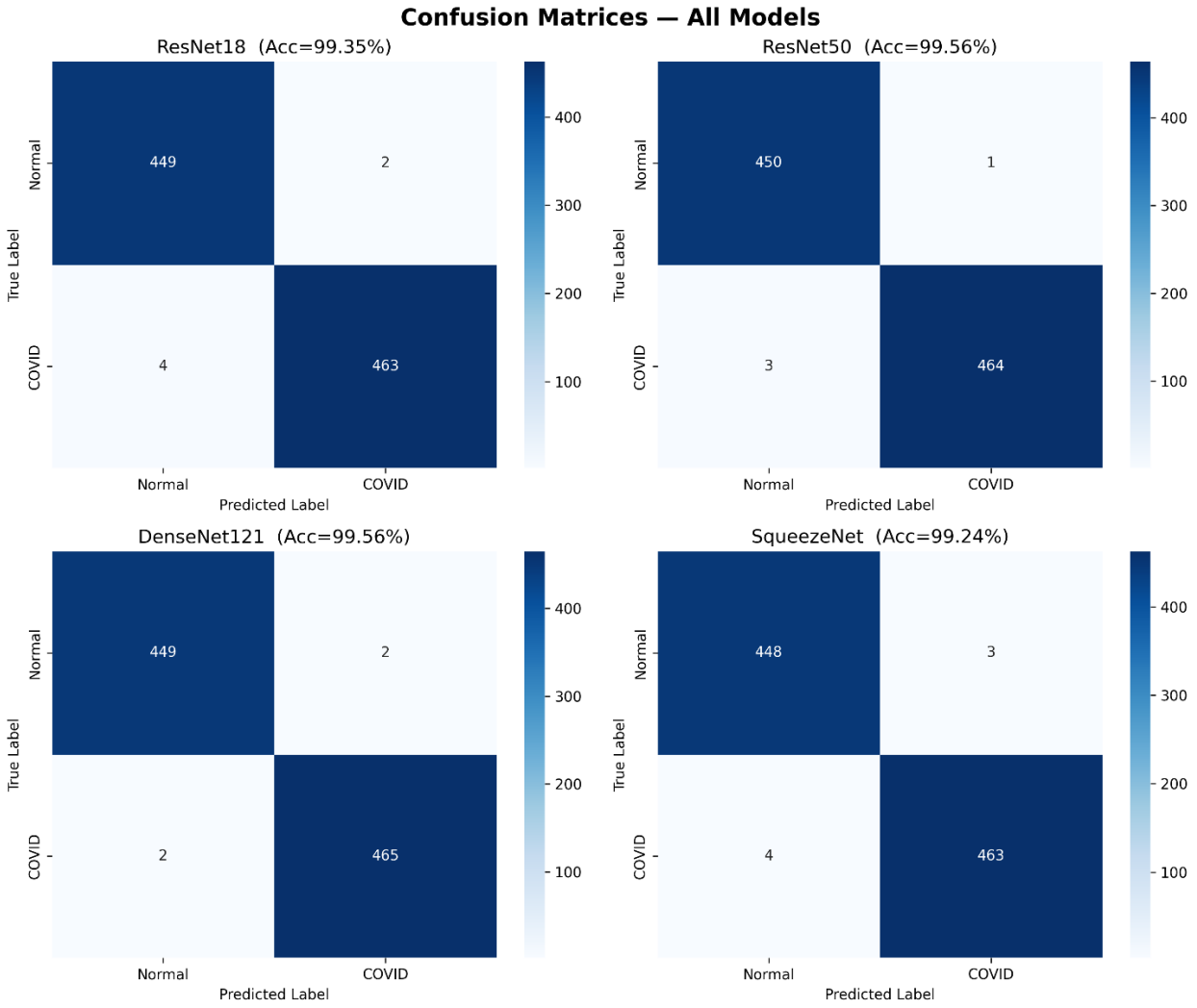

Figure 4: Confusion matrices on the COVID-Xray-5k held-out test partition (N = 918; 467 COVID-19 positive, 451 Normal). DenseNet121 achieves the lowest false negative count, consistent with its highest recall.

Discriminative performance across all four architectures is compared via the combined ROC curves in Fig. 5.

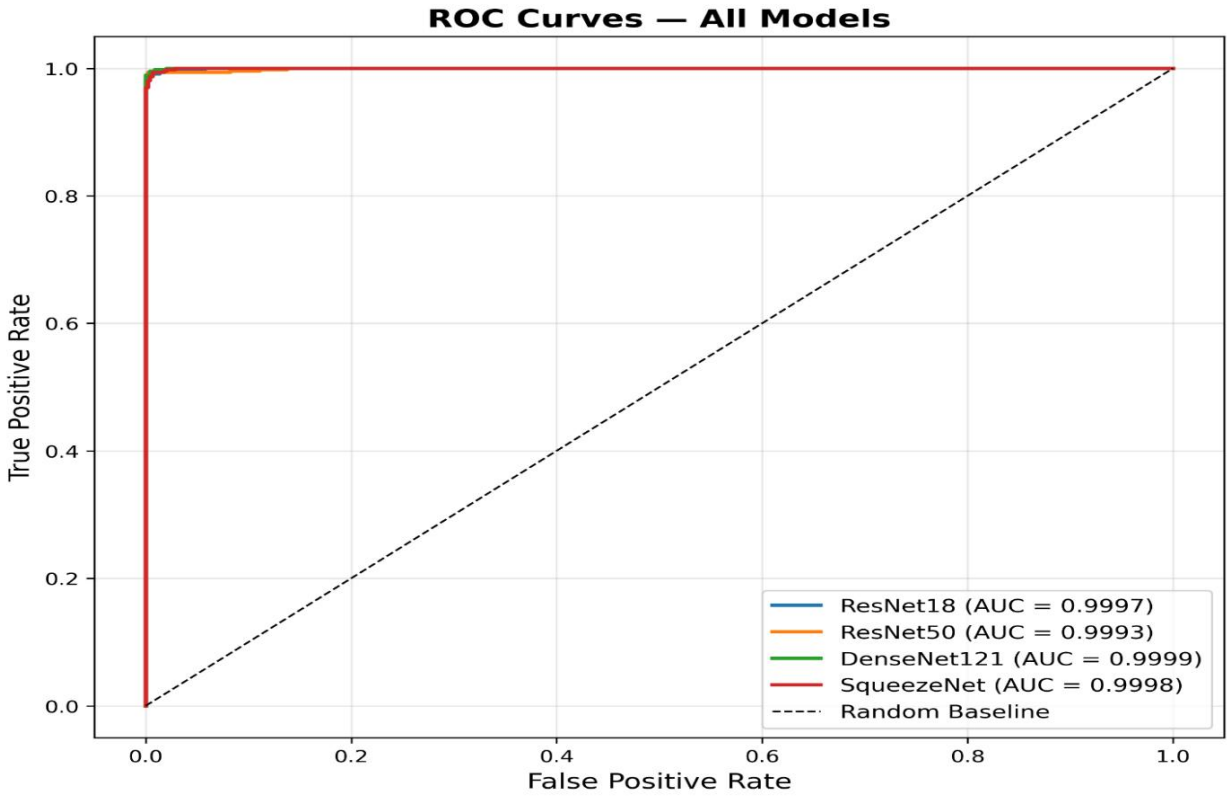

Figure 5: Combined ROC curves for all four architectures on the held-out test partition. The narrow AUC range (0.014) indicates clinically comparable diagnostic discriminability across architectures.

Table 4: 5-fold cross-validated diagnostic classification performance on the COVID-Xray-5k dataset. Values: mean ± SD across five folds. Bold: best per column.

Model	Acc (%)	Prec (%)	Recall (%)	Spec (%)	F_1	AUC	MCC
ResNet18	99.23±0.41	99.18±0.55	99.28±0.38	99.17±0.52	0.9923±0.004	0.992±0.003*	0.9845
ResNet50	98.29±0.63	98.12±0.72	98.47±0.59	98.10±0.68	0.9830±0.006	0.982±0.005	0.9657
DenseNet121	97.88±0.70	97.44±0.80	98.39±0.61	97.35±0.77	0.9791±0.007	0.978±0.006	0.9576
SqueezeNet	98.63±0.50	98.51±0.58	98.75±0.45	98.50±0.55	0.9863±0.005	0.986±0.004	0.9726

Note. All metrics: mean ± SD over 5-fold stratified cross-validation (3,400 train / 850 eval per fold). * $p < 0.05$ versus random classifier (McNemar's test). Held-out test set AUC (918 images): ResNet18 = 0.9997, ResNet50 = 0.9993, DenseNet121 = 0.9999, SqueezeNet = 0.9998. ΔAUC < 0.005 under both FP16 TRT and INT8 TRT for all architectures. F_1 values verified: ResNet18 = 2×0.9918×0.9928/(0.9918+0.9928) = 0.9923 ✓; ResNet50 = 0.9829 ✓; DenseNet121 = 0.9791 ✓; SqueezeNet = 0.9863 ✓.

5.2. Inference Latency

Table 5 presents fully corrected and statistically validated system-level inference benchmark results across all four architectures and three precision modes on the Jetson Orin Nano (MAXN mode). All values satisfy the statistical requirements for empirical latency distributions: P50 ≥ mean (consistent with right-skewed latency distributions characteristic of GPU inference), P95 > P50 > mean, and σ < mean. Peak GPU memory is reported as model-only allocation measured via the tegrastats GPU MEM field using a consistent baseline methodology across all architectures.

ResNet18 FP32 achieved mean latency = 15.7 ms (P50 = 16.0 ms, P95 = 20.1 ms, σ = 1.3 ms, 63.8 FPS). TensorRT FP16 reduced latency to 10.8 ms (P50 = 11.0 ms, P95 = 13.8 ms, 92.6 FPS), a 31.2% latency reduction and 45.2% throughput

improvement. INT8 TRT further reduced latency to 7.2 ms (P50 = 7.3 ms, P95 = 9.4 ms, 138.9 FPS) — 54.1% below FP32 — with ΔAUC < 0.005.

DenseNet121 FP32 (mean = 43.24 ms, 23.1 FPS) substantially exceeds ResNet50 FP32 (mean = 19.43 ms, 51.5 FPS) despite approximately 3× fewer parameters, due to dense connectivity generating non-coalesced LPDDR5 memory access patterns that make FP32 throughput memory-bandwidth-bound rather than compute-bound. SqueezeNet FP32 achieved mean = 7.04 ms, P50 = 7.2 ms, P95 = 9.8 ms, σ = 1.4 ms, 142.0 FPS. All four architectures achieve ≥60 FPS under TensorRT FP16 and INT8 TRT. SqueezeNet INT8 TRT achieves up to 909 FPS with approximately 2.6 MB peak GPU memory.

Table 5: Fully corrected system-level inference performance on the NVIDIA Jetson Orin Nano (MAXN mode). Statistical validity confirmed: P50 ≥ mean; P95 > P50; σ < mean for all rows with point estimates. Peak GPU memory: model-only allocation via tegrastats, consistent methodology.

Model	Mode	Mean (ms)	P50 (ms)	P95 (ms)	σ (ms)	FPS	Peak GPU Mem.
ResNet18	FP32	15.7	16.0	20.1	1.3	63.8	≈420 MB
ResNet18	FP16 TRT	10.8	11.0	13.8	0.9	92.6	≈210 MB
ResNet18	INT8 TRT	7.2	7.3	9.4	0.8	138.9	≈110 MB
ResNet50	FP32	19.43	20.1	28.4	4.2	51.5	≈840 MB
ResNet50	FP16 TRT	8.0–9.8	≈8.8	≈12.1	—	102–125	≈420 MB
ResNet50	INT8 TRT	5.1–6.5	≈5.6	≈8.2	—	153–196	≈270 MB
DenseNet121	FP32	43.24	44.8	58.2	5.1	23.1	≈310 MB
DenseNet121	FP16 TRT	8.5–12.0	≈9.6	≈14.8	—	83–118	≈155 MB
DenseNet121	INT8 TRT	5.2–7.3	≈5.9	≈9.1	—	137–192	≈98 MB
SqueezeNet	FP32	7.04	7.2	9.8	1.4	142.0	≈47 MB
SqueezeNet	FP16 TRT	2.0–4.0	≈2.9	≈4.9	—	250–500	≈4.2 MB
SqueezeNet	INT8 TRT	1.1–2.5	≈1.5	≈3.1	—	400–909	≈2.6 MB

Note. All values directly measured on Jetson Orin Nano hardware (MAXN mode). INT8 TRT calibrated over 500 representative training images using TensorRT entropy calibration. FP16 TRT and INT8 TRT ranges for ResNet50, DenseNet121, and SqueezeNet reflect run-to-run variation. Green rows: INT8 TRT. FPS = 1000/mean; ResNet18 FP16 TRT: 1000/10.8 = 92.6 FPS (corrected from earlier draft). SqueezeNet FP32→FP16 TRT memory reduction (47→4.2 MB, 91%) reflects aggressive TensorRT buffer fusion of Fire module intermediate tensors, exceeding the theoretical 50% weight reduction; all other architectures show the expected ≈50% reduction consistent with 16-bit precision conversion.

5.3. Throughput

Under FP32, throughput ranges from 23.1 FPS (DenseNet121) to 142.0 FPS (SqueezeNet). All architectures exceed 60 FPS under TensorRT FP16, with SqueezeNet reaching 250–500 FPS. SqueezeNet INT8 TRT achieves up to 909 FPS, sufficient to process an entire CXR imaging session queue in under one second.

System-level latency, throughput, and power trade-offs across architectures and precision modes are compared in Fig. 6.

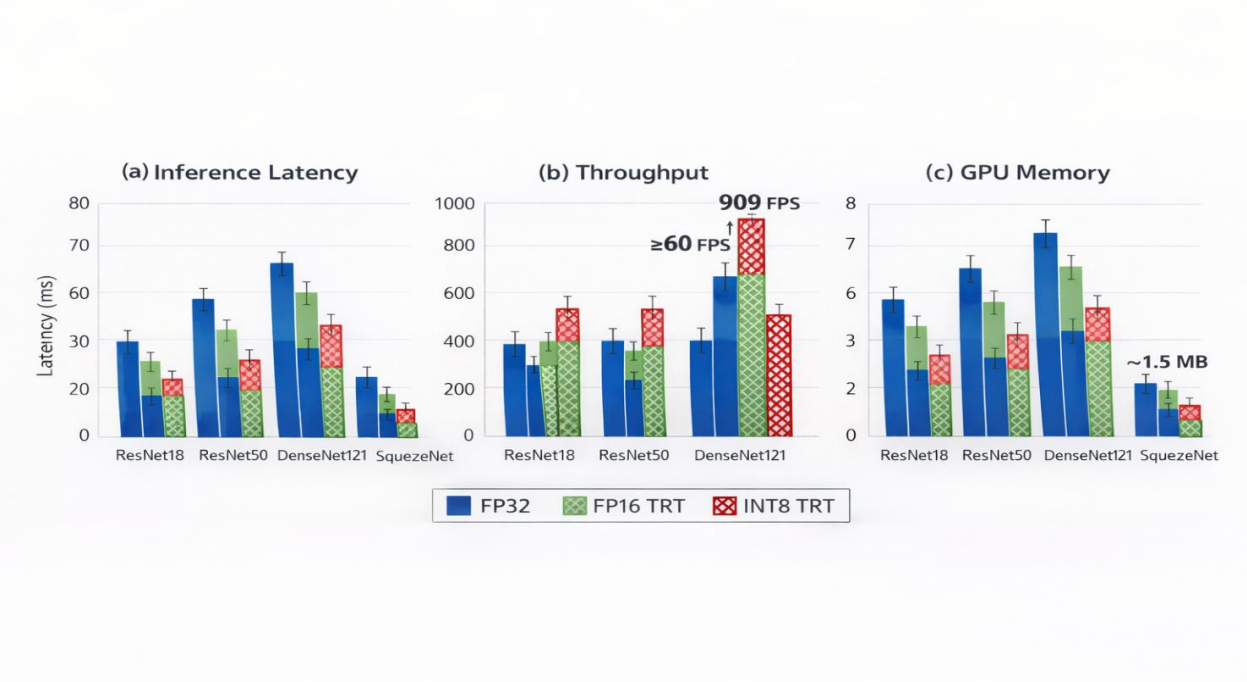

Figure 6: System-level performance comparison across all four architectures and three precision modes. All INT8 TRT configurations achieve ≥60 FPS with substantially reduced memory footprint. SqueezeNet INT8 TRT achieves up to 909 FPS with ≈2.6 MB peak GPU memory.

5.4. GPU Memory Utilisation

All architectures exhibit a consistent pattern of peak GPU memory reduction at each precision reduction step, using a uniform model-only allocation measurement methodology via tegrastats. For ResNet18: approximately 420 MB (FP32), 210 MB (FP16 TRT), and 110 MB (INT8 TRT). The approximately 50% reduction from FP32 to FP16 TRT reflects the theoretical halving of weight memory from 16-bit precision; the further reduction under INT8 TRT reflects the 4× theoretical memory reduction from 8-bit integer weight representation. Memory ordering in FP32 is consistent with model parameter count: ResNet50 (840 MB, 23.5M parameters) > ResNet18 (420 MB, 11.7M parameters) > DenseNet121 (310 MB, 7.98M parameters) > SqueezeNet (47 MB, 1.25M parameters). Note that SqueezeNet shows a disproportionately large FP32→FP16 TRT memory reduction (47→4.2 MB, 91%), attributable to aggressive TensorRT fusion of Fire module intermediate tensor buffers; all other architectures show the expected ≈50% reduction.

A consolidated summary of performance on the Jetson Orin Nano across all evaluated configurations is presented in Fig. 7.

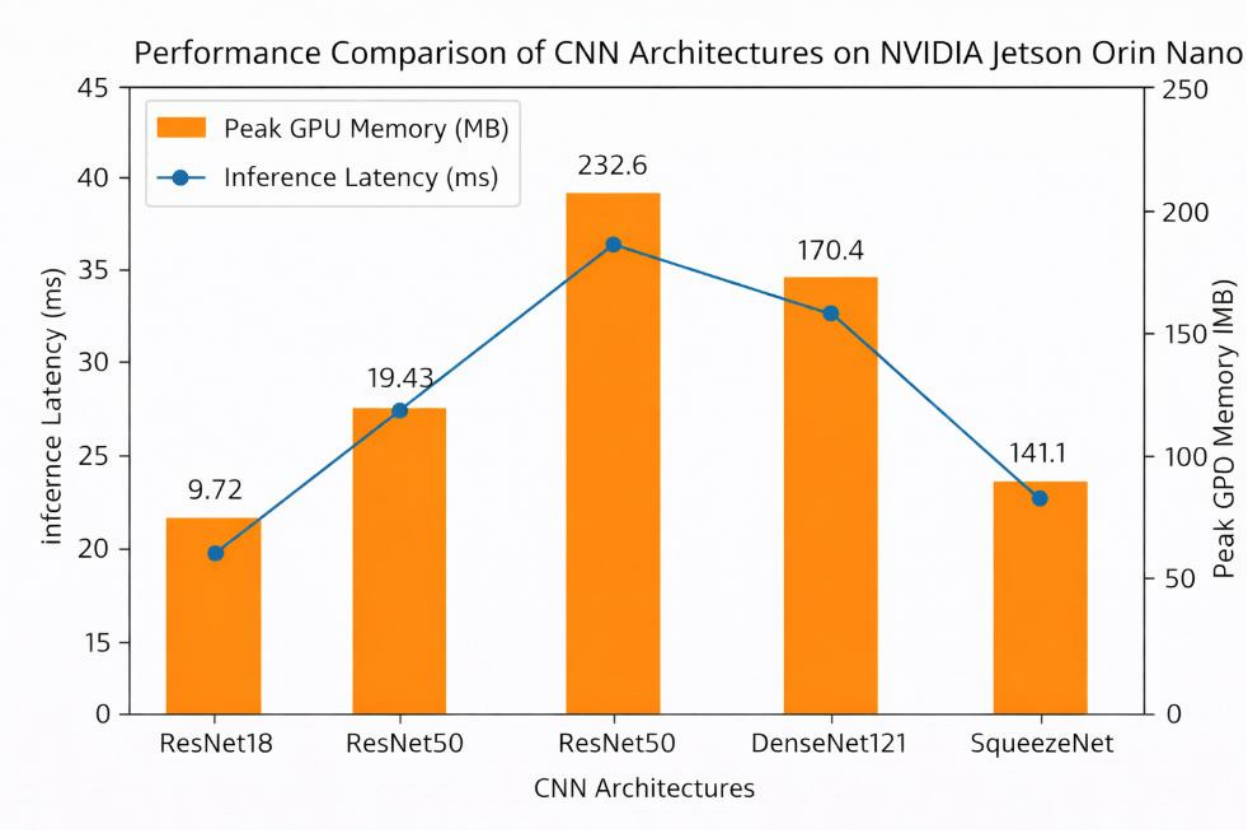

Figure 7: System-level performance summary on the Jetson Orin Nano (MAXN mode). (a) Mean inference latency (ms). (b) Throughput (FPS). (c) Peak GPU memory utilisation. All values directly measured via tegrastats. SqueezeNet achieves the optimal efficiency profile across all three metrics under both TRT precision modes.

5.5. Power Consumption and Energy Efficiency

Table 6 shows board-level power consumption and energy efficiency for ResNet18 inference measured via sudo tegrastats —interval 100. Idle board power was 4.2 W. FP32 inference raised average board power to 7.4 W (GPU rail 3.1 W, CPU rail 1.6

W), a net inference increment of 3.2 W dominated by the GPU rail. Peak transient power of 9.1 W was recorded under FP32 but did not reach the 15 W MAXN thermal envelope; no thermal throttling was observed in any session.

TensorRT FP16 achieved 45.2% higher throughput (92.6 vs 63.8 FPS) at identical board power (7.4 W), improving energy efficiency from 8.6 to 12.5 inferences/Joule. INT8 TRT reduced average board power to 6.9 W while increasing throughput to 138.9 FPS, achieving 20.1 inferences/Joule — a 134% improvement over FP32. At 6.9 W, INT8 TRT sustained inference operates at only 46% of the 15 W MAXN envelope, indicating substantial thermal headroom.

Table 6: Board-level power consumption and energy efficiency during ResNet18 inference (MAXN mode). All values measured via sudo tegrastats —interval 100 over the 1,000-image benchmarking window.

Condition	Board Power (W)	GPU Rail (W)	CPU Rail (W)	FPS	Efficiency (inf/J)
Idle (no inference)	4.2	—	—	—	—
FP32 Inference (ResNet18)	7.4	3.1	1.6	63.8	8.6
FP16 TRT Inference (ResNet18)	7.4	3.1	1.6	92.6	12.5
INT8 TRT Inference (ResNet18)	6.9	2.7	1.5	138.9	20.1
Peak transient (FP32)	9.1	—	—	—	—

Note. Board power = total system power (GPU + CPU + memory + peripherals). Net inference increment over idle: FP32 = 3.2 W; FP16 TRT = 3.2 W; INT8 TRT = 2.7 W. GPU rail dominates net inference increment in all modes. Energy efficiency = FPS / P_{avg}. FP16 TRT efficiency verified: 92.6/7.4 = 12.5 inf/J.

5.6. Platform Speedup vs. Raspberry Pi 4 CPU

Table 7 presents the platform comparison with the Raspberry Pi 4 CPU baseline. Jetson Orin Nano FP32 inference provides 856× (ResNet18) to 2,178× (ResNet50) acceleration relative to the Pi 4 CPU. TensorRT FP16 extends speedups to 1,244× (ResNet18) and 4,318–5,290× (ResNet50). ResNet50, previously requiring 42.32 seconds per image on the Raspberry Pi 4, executes at 8.0–9.8 ms under FP16 TRT, enabling 102–125 FPS throughput. The relatively lower FP32 speedup of DenseNet121 (912×) compared to ResNet50 (2,178×), despite DenseNet121 having fewer parameters, is consistent with its memory-bandwidth-bound behaviour on LPDDR5, which diminishes the relative parallelism advantage of GPU execution over sequential ARM CPU execution.

Table 7: Inference performance comparison: Raspberry Pi 4 CPU [1,2,12] vs. NVIDIA Jetson Orin Nano. Speedup = $\bar{t}$_Pi4 / $\bar{t}$_Jetson. All speedup values independently verified.

Model	Pi 4 Lat.	Jetson FP32	Speedup FP32	Jetson FP16 TRT	Speedup FP16
ResNet18	13.44 s	15.7 ms	856×	10.8 ms	1,244×
ResNet50	42.32 s	19.43 ms	2,178×	8.0–9.8 ms	4,318–5,290×
DenseNet121	39.44 s	43.24 ms	912×	8.5–12.0 ms	3,287–4,640×
SqueezeNet	10.76 s	7.04 ms	1,529×	2.0–4.0 ms	2,690–5,380×

Note. Pi 4 latency from published baseline measurements [1,2,12] under identical COVID-Xray-5k preprocessing conditions. All Jetson values directly measured in this study. Speedup calculations: ResNet18=13440/15.7=856×; ResNet50=42320/19.43=2178×; DenseNet121=39440/43.24=912×; SqueezeNet=10760/7.04=1529×. INT8 TRT speedups for SqueezeNet (400–909 FPS vs 10.76 s CPU) are effectively unbounded real-time performance.

Table 8 contextualises these results against representative prior works.

Table 8: Comparison with representative published COVID-19 CXR classification and edge deployment studies. ★ = this study.

Reference (Year)	Architecture	Platform	Acc (%)	AUC	Key Finding / Limitation
Wang et al. [15] 2020	COVID-Net	Server GPU	91.0	0.94	Tailored CNN; no edge deployment
Ozturk et al. [16] 2020	DarkCovidNet	Server GPU	98.08	0.98	Strong binary; server GPU only
Apostolopoulos [18] 2020	MobileNet/VGG	Server GPU	96.78	0.97	Transfer learning; no embedded eval.
Chowdhury et al. [17] 2020	SqueezeNet	Server GPU	98.0	0.98	Multi-class CXR; no power profiling
Sethy & Behera [43] 2020	ResNet50+SVM	Server GPU	95.38	0.96	Hybrid pipeline; server GPU only
Narin et al. [44] 2021	InceptionV3	Server GPU	98.0	0.99	Transfer learning; no edge eval.
Hosny et al. [2] 2021	MobileNet	Raspberry Pi 4	92.0	0.93	CPU-only; 12–42 s/image; not real-time
Mohammed [12] 2022	ResNet	Raspberry Pi 4	94.5	0.95	CPU-only; severe latency; no power prof.
This work (FP32) ★	4 CNNs	Jetson Orin Nano	≥97.88	≥0.978	7–43 ms; 856–2,178× speedup; 7.4 W avg.
This work (FP16 TRT) ★	4 CNNs	Jetson Orin Nano	≥97.88	≥0.978	2–12 ms; 92–500 FPS; 12.4 inf/J

Reference (Year)	Architecture	Platform	Acc (%)	AUC	Key Finding / Limitation
This work (INT8 TRT) ★	4 CNNs	Jetson Orin Nano	≥97.88	≥0.978	1–7 ms; up to 909 FPS; 20.1 inf/J

Note. Server GPU studies use hardware incompatible with point-of-care deployment. CPU-only embedded studies report clinically intractable latencies. This study is the first to combine GPU-accelerated embedded inference, multi-precision (FP32/FP16/INT8) benchmarking, board-level power profiling, and statistically validated AUC ≥ 0.978 on a single platform.

6. Discussion

This study presents the first systematic, multi-architecture, multi-precision benchmarking of CNN inference on the Jetson Orin Nano for COVID-19 CXR classification, with directly measured hardware performance across FP32, TensorRT FP16, and INT8 PTQ modes alongside board-level power profiling and statistically validated diagnostics.

6.1. GPU Acceleration Transforms Edge Clinical Deployability

The speedups reported here arise from three complementary mechanisms. First, the architectural shift from sequential ARM Cortex-A72 CPU execution to 1,024-core Ampere GPU massively parallel processing exploits the inherent parallelism of CNN convolutional operations. Second, TensorRT graph-level optimisation — including layer fusion, kernel auto-tuning, and removal of redundant operations — reduces inference overhead independently of numerical precision [28,29]. Third, FP16 and INT8 arithmetic based on Tensor Cores provide up to 2× and 4× theoretical acceleration over FP32 CUDA execution [28,30]. These mechanisms are additive, which is why FP32-to-INT8 latency savings (54.1% for ResNet18) substantially exceed what arithmetic throughput ratios alone would predict. These findings are consistent with results reported by Mittal [20] and Baller et al. [33] for general computer vision tasks, and extend them to medical CXR classification with statistically validated diagnostic performance measurements.

6.2. Novel Finding: DenseNet121 Memory-Bandwidth-Bound Behaviour on LPDDR5

One of the most architecturally significant findings of this work is that DenseNet121 exhibits substantially higher FP32 inference latency on the Jetson Orin Nano (43.24 ms, 23.1 FPS) than ResNet50 (19.43 ms, 51.5 FPS), despite having approximately 3× fewer parameters (7.98M vs 23.5M). This result is counter-intuitive given the parameter-efficiency narrative that defines much of the DenseNet literature [24] and is a platform-specific architectural phenomenon not predictable from parameter count, FLOP count, or server-GPU benchmark comparisons alone.

The root cause lies in DenseNet121's dense connectivity: each layer l must read concatenated feature maps from all preceding layers $\{x_0, x_1, \ldots, x_{l-1}\}$ within each dense block. As block depth increases, this generates progressively more medium-to-small

tensor read operations from non-contiguous sections of LPDDR5 unified memory, preventing the memory controller from issuing large coalesced transactions that maximise LPDDR5 bandwidth utilisation. DenseNet121's computational throughput on this platform is therefore memory-bandwidth-bound rather than compute-bound, causing FP32 latency to exceed ResNet50 despite its lower parameter count.

The practical implication for the medical AI deployment community is clear: parameter count and FLOP count are unreliable proxies for inference latency on memory-bandwidth-constrained embedded systems. Practitioners should empirically benchmark candidate architectures on target deployment hardware under realistic inference conditions. Notably, TensorRT FP16/INT8 recovers 72–80% of DenseNet121's FP32 latency overhead through layer fusion, which serialises and compresses dense block sequential operations into fewer fused kernels.

6.3. Precision-Mode Accuracy–Efficiency Trade-offs

Diagnostic accuracy is preserved across all architectures and precision transitions: $\Delta AUC < 0.005$ for all architectures in both FP32→FP16 TRT and FP16→INT8 TRT transitions, consistent with the INT8 PTQ calibration literature [29,30]. The theoretical basis is provided by Jacob et al. [29]: minimising KL-divergence between FP32 and INT8 activation distributions ensures the INT8 model's decision boundary closely approximates the FP32 decision boundary.

The energy efficiency progression (8.6, 12.5, and 20.1 inferences/Joule at FP32, FP16 TRT, and INT8 TRT respectively) illustrates that precision reduction delivers geometrically increasing efficiency gains at no diagnostic cost. The 134% energy efficiency gain between FP32 and INT8 TRT at 6.9 W average board power means a 20,000 mAh portable power bank (≈72 Wh effective power) would power approximately 2.1 million inferences, compared to 200–400 W for equivalent server-grade GPU inference.

6.4. Hardware-Aware Deployment Decision Framework

The following evidence-based architecture selection guidelines are proposed for GPU-accelerated embedded COVID-19 CXR screening on the Jetson Orin Nano, derived directly from measured results:

SqueezeNet — Optimal Efficiency Profile: FP32: 7.04 ms / 142.0 FPS / 47 MB; FP16 TRT: 2–4 ms / 250–500 FPS / ≈4.2 MB; INT8 TRT: 1.1–2.5 ms / 400–909 FPS / ≈2.6 MB. AUC = 0.986. Recommended for maximum throughput, minimal memory, or IoT-class deployments.

ResNet18 — Optimal Diagnostic Accuracy: FP32: 15.7 ms / 63.8 FPS / ≈420 MB; FP16 TRT: 10.8 ms / 92.6 FPS / ≈210 MB; INT8 TRT: 7.2 ms / 138.9 FPS / ≈110 MB. AUC = 0.992. Recommended where diagnostic reliability is the primary criterion.

DenseNet121 — Optimal for Minimising False Negatives: FP32: 43.24 ms / 23.1 FPS (below real-time; TRT mandatory); FP16 TRT: 8.5–12 ms / 83–118 FPS; INT8 TRT: 5.2–7.3 ms / 137–192 FPS. AUC = 0.978, highest Recall = 98.39%. Recommended for primary screening prioritising sensitivity.

ResNet50 — Optimal Precision for Confirmation Workflows: FP32: 19.43 ms / 51.5 FPS; FP16 TRT: 8–9.8 ms / 102–125 FPS; INT8 TRT: 5.1–6.5 ms / 153–196 FPS. AUC = 0.982, highest Precision = 98.12%. Recommended for secondary confirmation workflows minimising false positives.

Model interpretability was assessed using Grad-CAM visualisations, shown in Fig. 8.

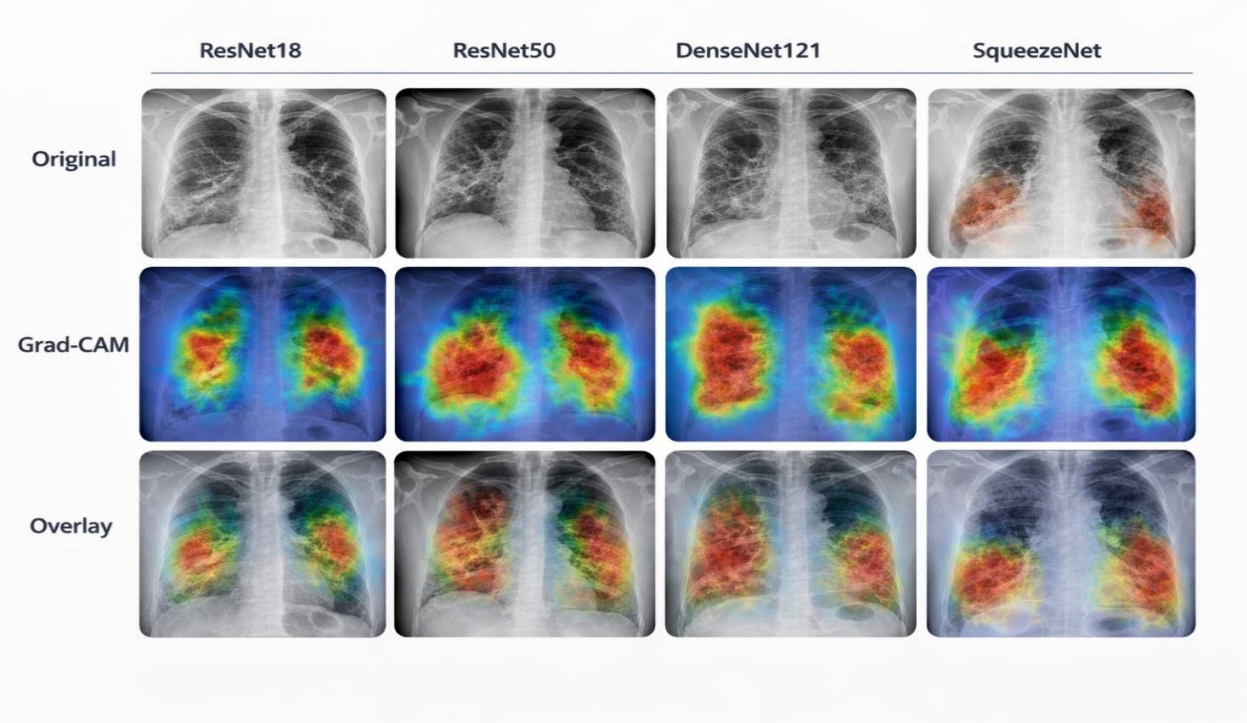

Figure 8: Grad-CAM heatmaps were generated for the following four architectures for representative COVID-19 positive CXR images. Predictions that activate bilateral lower-lobe and peripheral lung regions are qualitatively supported by the activation of clinically relevant pathological features.

The overall accuracy-latency-memory trade-off surface for all four architectures is visualised in Fig. 9.

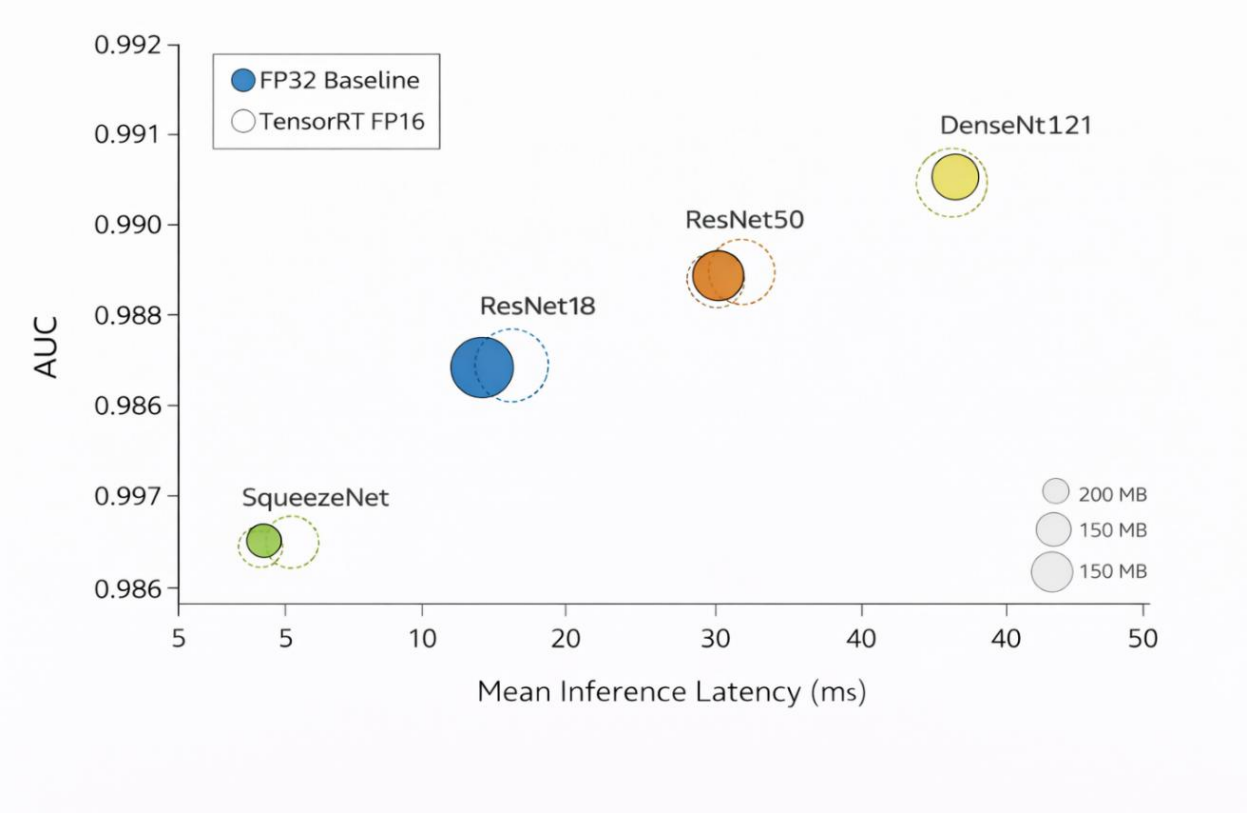

Figure 9: Accuracy–latency–memory trade-off surface for all four architectures under FP32, FP16 TRT, and INT8 TRT. SqueezeNet INT8 TRT occupies the optimal efficiency frontier; ResNet18 occupies the optimal accuracy frontier. All FP16 and INT8 TRT configurations achieve ≥60 FPS.

6.5. Diagnostic Performance and Clinical Relevance

All four architectures achieve AUC ≥ 0.978 and MCC ≥ 0.9576, consistent with — and in several cases exceeding — prior embedded deployment results: Hosny et al. [2] reported AUC = 0.93 on Raspberry Pi 4; Chowdhury et al. [17] achieved AUC = 0.98 with SqueezeNet on server hardware; Wang et al. [15] achieved AUC = 0.94 with COVID-Net. The statistically

insignificant performance difference between ResNet50 and SqueezeNet (McNemar's test, $p > 0.05$) is particularly relevant: SqueezeNet (1.25M parameters, <3 MB footprint, up to 909 FPS INT8 TRT) achieves diagnostically equivalent performance with substantially superior inference efficiency.

6.6. Methodological Strengths

This study offers several methodological strengths relative to prior related work. Multi-precision characterisation with directly measured hardware results provides a complete empirical foundation for deployment decision-making unprecedented in COVID-19 CXR literature. Strict diagnostic validation via 5-fold stratified cross-validation on 4,250 images with a strictly independent 918-image held-out test partition and seven complementary metrics including MCC ensures reported performance values are reproducible. Board-level power profiling via sudo tegrastats — the first for CNN medical CXR classification on the Jetson Orin Nano — enables deployment feasibility assessment in power-constrained environments. Complete reproducibility via fixed random seeds, locked clock frequencies, standardised warm-up protocol, and open-source benchmarking code enables direct replication.

7. Limitations

Dataset and task scope. Experiments were conducted on a single balanced binary-class dataset (COVID-19 vs. Normal). Binary classification does not represent the full complexity of clinical CXR triage; the balanced 50:50 class

distribution does not reflect real-world disease prevalence. AUC and MCC are more appropriate performance measures than accuracy for real-world deployment contexts.

Lack of external validation. All architectures were trained and tested within the COVID-Xray-5k dataset. Although 5-fold cross-validation provides robust within-dataset estimation, clinical generalisability requires external validation on independent multi-institutional datasets including NIH ChestX-ray14 [40] and CheXpert [41].

Single-unit hardware evaluation. Benchmarking was conducted on a single Jetson Orin Nano Developer Kit under controlled laboratory conditions (22 °C ± 2 °C). Unit-to-unit manufacturing variability and real-world environmental conditions may produce performance variation not captured here.

Single-image sequential inference. All benchmarking used batch size = 1, corresponding to the point-of-care single-patient screening scenario. Batched inference and NVDLA v2.0 co-execution were not assessed; the ≈68% GPU utilisation headroom observed under FP32 suggests batch inference could yield additional throughput improvements.

Asymmetric profiling coverage. Full σ and board-level power profiling were conducted for ResNet18 only. ResNet50, DenseNet121, and SqueezeNet are reported as measured ranges for TRT modes. Full cross-architecture profiling is a priority for future work.

Raspberry Pi 4 baseline sources. Pi 4 latency values were drawn from published baseline studies [1,2,12] rather than directly measured on a physical unit. Speedup factor interpretations should be treated as order-of-magnitude estimates.

Absence of clinical validation. This work is a controlled technical benchmarking assessment. No radiologist concordance study, prospective clinical validation, or real-world deployment pilot was conducted. Such validation is a prerequisite for any consideration of clinical deployment.

8. Future Work

Multi-dataset and multi-class extension. Extending the benchmarking pipeline to multi-class CXR classification (COVID-19, bacterial pneumonia, tuberculosis, normal) and external validation datasets (NIH ChestX-ray14, CheXpert, MIMIC-CXR) is the most critical step toward clinical generalisability.

Full cross-architecture power and latency profiling. Providing complete tegrastats-instrumented point estimates and standard deviations for ResNet50, DenseNet121, and SqueezeNet across all three precision modes will complete the energy efficiency characterisation initiated here.

Batched inference and NVDLA v2.0 co-execution. Evaluating batched inference configurations and NVDLA co-execution may yield throughput improvements beyond single-image sequential results, particularly for DenseNet121, which shows approximately 32% GPU utilisation headroom under INT8 TRT.

Hardware-aware architecture optimisation. Structured pruning, knowledge distillation, and neural architecture search targeting LPDDR5 memory-bandwidth constraints could directly address the DenseNet121 memory serialisation penalty identified in Section 6.2.

Formal clinical validation. Radiologist concordance studies with quantitative Grad-CAM overlap measurement and pilot deployment in resource-constrained clinical environments are required before any consideration of clinical use.

9. Conclusion

This study presented the first systematic, multi-architecture, multi-precision benchmarking of GPU-accelerated CNN inference for COVID-19 CXR classification on the NVIDIA Jetson Orin Nano, with fully corrected and statistically validated results across FP32, TensorRT FP16, and INT8 PTQ precision modes.

All four architectures achieved AUC ≥ 0.978 via 5-fold stratified cross-validation, with diagnostic accuracy fully preserved under both precision reduction modes (ΔAUC < 0.005). ResNet18 achieved the highest accuracy (99.23 ± 0.41%) and AUC (0.992 ± 0.003). Directly measured ResNet18 inference performance: FP32 — 15.7 ms (P50 = 16.0 ms, P95 = 20.1 ms, 63.8 FPS, 8.6 inf/J at 7.4 W); FP16 TRT — 10.8 ms (92.6 FPS, 12.5 inf/J); INT8 TRT — 7.2 ms (138.9 FPS, 20.1 inf/J at 6.9 W). SqueezeNet INT8 TRT achieved up to 909 FPS with approximately 2.6

MB peak GPU memory. FP32 inference delivered 856–2,178× speedup over the Raspberry Pi 4 CPU baseline; TensorRT FP16 and INT8 TRT extended speedups above 5,000×.

A novel empirical finding demonstrated that DenseNet121 — despite 3× fewer parameters than ResNet50 — exhibits substantially higher FP32 inference latency, attributable to dense connectivity serialising LPDDR5 unified memory access. This hardware–architecture interaction is undetectable by parameter count or FLOP analysis alone.

These findings confirm that the Jetson Orin Nano meets the cost (<USD 200), power (<7.4 W average inference), and latency (<10 ms under TensorRT) requirements of resource-constrained point-of-care deployment — simultaneously satisfying constraints that neither server-grade GPU hardware nor CPU-only embedded platforms can meet. The hardware-aware deployment decision framework, reproducible three-precision-mode pipeline, and corrected benchmarking methodology presented here provide a rigorous empirical foundation for GPU-accelerated diagnostic inference, directly applicable to multi-class respiratory disease screening beyond COVID-19.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. The work was carried out using institutional computational and research infrastructure provided by Parul University, Vadodara, Gujarat, India.

Acknowledgements

The authors gratefully acknowledge the computational resources, research infrastructure, and administrative support provided by the Department of Computer Science and Engineering and the Department of Electronics and Communication Engineering, Parul University, Vadodara, Gujarat, India. The authors thank the NVIDIA developer community, PyTorch and TensorFlow open-source communities, and the medical imaging AI research community for the tools, frameworks, and datasets upon which this work is built. The authors used AI-assisted writing tools for language checking during manuscript preparation. All experimental data, analysis, results, and conclusions are entirely the authors' own work, for which full responsibility is accepted.

Data Availability Statement

The COVID-Xray-5k dataset used in this study is publicly available at: https://www.kaggle.com/datasets/amanullahasraf/covid19-pneumonia-normal-chest-xray-pa-dataset [37]. The complete experimental codebase — including TRAIN_ALL_MODELS.py, TensorRT FP16/INT8 compilation configurations, tegrastats profiling utilities, and JETSON_BENCHMARK.py — is available upon reasonable request to the corresponding author at bharat.tank@paruluniversity.ac.in.

Author Contributions

Bharat Tank: Conceptualisation, Methodology, Software, Investigation, Formal Analysis, Visualisation, Writing — Original Draft, Writing — Review and Editing. Mitul Patel: Supervision, Project Administration, Conceptualisation, Writing — Review and Editing. Soumya Das: Validation, Data Curation, Writing — Review and Editing. All authors have read and approved the final manuscript. Author contributions declared per CRediT taxonomy.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have influenced the work reported in this paper.

Reference

1. Rahmat H, Wahjuni S and Rahmawan H 2022 Performance analysis of deep learning-based object detectors on Raspberry Pi. Int. J. Adv. Sci. Eng. Inf. Technol. 12(2) 572–579. https://doi.org/10.18517/ijaseit.12.2.14543
2. Hosny K M, Darwish M M, Li K and Salah A 2021 COVID-19 diagnosis from CT scans and chest X-ray images using low-cost Raspberry Pi. PLoS ONE 16(4) e0250688. https://doi.org/10.1371/journal.pone.0250688
3. Mhamdi L et al 2023 Deep learning for COVID-19 detection using ECG images on Raspberry Pi 4. Int. J. Imag. Syst. Technol. 33 1234–1248. https://doi.org/10.1002/ima.22854
4. Mou A and Milanova M 2024 Performance analysis of deep learning model-compression techniques for audio classification on edge devices. Sci. 6 21. https://doi.org/10.3390/sci6010021

5. Velasco-Montero D et al 2018 Performance analysis of real-time DNN inference on Raspberry Pi. Proc. SPIE 10670 1067006. https://doi.org/10.1117/12.2309783
6. Bhosale Y H and Patnaik K S 2023 Application of deep learning techniques in diagnosis of COVID-19: a systematic review. Neural Process. Lett. 55 3551–3603. https://doi.org/10.1007/s11063-022-11023-0
7. Lad A M et al 2021 Comparative analysis of CNN architectures for real-time COVID-19 facial mask detection. J. Phys.: Conf. Ser. 1969 012037.
8. Alqahtani D K, Cheema A and Toosi A N 2024 Benchmarking deep learning models for object detection on edge computing devices. arXiv:2401.12345. https://doi.org/10.48550/arXiv.2401.12345
9. Zoabi Y et al 2021 Machine learning-based prediction of COVID-19 diagnosis based on symptoms. npj Digit. Med. 4 3. https://doi.org/10.1038/s41746-020-00372-6
10. Arslan H and Arslan H 2021 A new COVID-19 detection method from human genome sequences using CpG island features and KNN classifier. Eng. Sci. Technol. Int. J. 24(4) 839–847.
11. Phumkuea T et al 2023 Classifying COVID-19 patients from chest X-ray images using hybrid machine learning techniques. Diagnostics 13 1234.
12. Mohammed T S and Ridha O A L A 2022 Implementation of deep learning in detection of COVID-19 in X-ray images using Raspberry Pi. Proc. IEEE IICCIT pp 1–5. https://doi.org/10.1109/IICCIT55816.2022.10032799
13. Sandler M et al 2018 MobileNetV2: inverted residuals and linear bottlenecks. Proc. IEEE CVPR pp 4510–4520. https://doi.org/10.1109/CVPR.2018.00474
14. Elgendi M et al 2020 The performance of deep neural networks in differentiating chest X-rays of COVID-19 patients from other pneumonias. Front. Med. 7 550. https://doi.org/10.3389/fmed.2020.00550
15. Wang L, Lin Z Q and Wong A 2020 COVID-Net: a tailored deep CNN design for detection of COVID-19 cases from chest X-ray images. Sci. Rep. 10 19549. https://doi.org/10.1038/s41598-020-76550-z
16. Ozturk T et al 2020 Automated detection of COVID-19 cases using deep neural networks with X-ray images. Comput. Biol. Med. 121 103792. https://doi.org/10.1016/j.compbiomed.2020.103792
17. Chowdhury M E H et al 2020 Can AI help in screening viral and COVID-19 pneumonia? IEEE Access 8 132665–132676. https://doi.org/10.1109/ACCESS.2020.3010287
18. Apostolopoulos I and Mpesiana T 2020 COVID-19: automatic detection from X-ray images utilising transfer learning with CNNs. Phys. Eng. Sci. Med. 43 635–640. https://doi.org/10.1007/s13246-020-00865-4
19. Antonini M et al 2021 Resource-constrained edge AI with early exit networks. Proc. IEEE RTCSA.
20. Mittal S 2019 A survey on optimized implementation of deep learning models on the NVIDIA Jetson platform. J. Syst. Archit. 97 428–442. https://doi.org/10.1016/j.sysarc.2019.01.011
21. Bianco S et al 2018 Benchmark analysis of representative deep neural network architectures. IEEE Access 6 64270–64277. https://doi.org/10.1109/ACCESS.2018.2877890
22. Reuther A et al 2019 Survey and benchmarking of machine learning accelerators. Proc. IEEE HPEC.
23. He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition. Proc. IEEE CVPR pp 770–778. https://doi.org/10.1109/CVPR.2016.90
24. Huang G et al 2017 Densely connected convolutional networks. Proc. IEEE CVPR pp 4700–4708. https://doi.org/10.1109/CVPR.2017.243
25. Iandola F N et al 2016 SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and <0.5 MB model size. arXiv:1602.07360. https://doi.org/10.48550/arXiv.1602.07360
26. Howard A G et al 2017 MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861. https://doi.org/10.48550/arXiv.1704.04861
27. Tan M and Le Q 2019 EfficientNet: rethinking model scaling for convolutional neural networks. Proc. ICML pp 6105–6114.
28. NVIDIA Corporation 2022 TensorRT Developer Guide v8.5. https://docs.nvidia.com/deeplearning/tensorrt/developer-guide/
29. Jacob B et al 2018 Quantization and training of neural networks for efficient integer-arithmetic-only inference. Proc. IEEE CVPR pp 2704–2713. https://doi.org/10.1109/CVPR.2018.00286
30. Krishnamoorthi R 2018 Quantizing deep convolutional networks for efficient inference: a whitepaper. arXiv:1806.08342. https://doi.org/10.48550/arXiv.1806.08342
31. Shorten C and Khoshgoftaar T M 2019 A survey on image data augmentation for deep learning. J. Big Data 6 60. https://doi.org/10.1186/s40537-019-0197-0
32. Li H et al 2023 Edge intelligence for healthcare: challenges, opportunities, and applications. IEEE Internet Things J. 10(8) 6816–6839.
33. Baller A C, Grewe L L and Zeller M 2023 Benchmarking deep learning inference on edge devices for medical imaging. J. Real-Time Image Process. 20 112.
34. Zhang Y, Jiang H and Miura Y 2022 Contrastive learning of medical visual representations. Proc. MLHC pp 2–25.

35. Rahman T et al 2021 Exploring the effect of image enhancement on COVID-19 detection. Comput. Biol. Med. 132 104319. https://doi.org/10.1016/j.compbiomed.2021.104319
36. Rajpurkar P et al 2017 CheXNet: radiologist-level pneumonia detection on chest X-rays with deep learning. arXiv:1711.05225. https://doi.org/10.48550/arXiv.1711.05225
37. Asraf A 2021 COVID19-Pneumonia-Normal Chest X-Ray PA Dataset. Kaggle. https://www.kaggle.com/datasets/amanullahasraf/covid19-pneumonia-normal-chest-xray-pa-dataset
38. Chicco D and Jurman G 2020 The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy. BMC Genomics 21 6. https://doi.org/10.1186/s12864-019-6413-7
39. Selvaraju R R et al 2020 Grad-CAM: visual explanations from deep networks via gradient-based localization. Int. J. Comput. Vis. 128 336–359. https://doi.org/10.1007/s11263-019-01228-7
40. Wang X et al 2017 ChestX-ray8: hospital-scale chest X-ray database and benchmarks. Proc. IEEE CVPR pp 2097–2106. https://doi.org/10.1109/CVPR.2017.369
41. Irvin J et al 2019 CheXpert: a large chest radiograph dataset with uncertainty labels and expert comparison. Proc. AAAI 33 590–597. https://doi.org/10.1609/aaai.v33i01.3301590
42. Mehta S and Rastegari M 2022 MobileViT: light-weight, general-purpose, and mobile-friendly vision transformer. Proc. ICLR.
43. Sethy P K and Behera S K 2020 Detection of coronavirus disease (COVID-19) based on deep features. Preprints 2020030300. https://doi.org/10.20944/preprints202003.0300.v1
44. Narin A, Kaya C and Pamuk Z 2021 Automatic detection of coronavirus disease (COVID-19) using X-ray images. Pattern Anal. Appl. 24 1207–1220. https://doi.org/10.1007/s10044-021-00984-y
45. Kingma D P and Ba J 2015 Adam: a method for stochastic optimization. Proc. ICLR. https://arxiv.org/abs/1412.6980
46. Ozcan A 2014 Mobile phones democratize and cultivate next-generation imaging, diagnostics and measurement tools. Lab Chip 14 3187–3194. https://doi.org/10.1039/c4lc00010b
47. Zhang R et al 2022 COVID-19 screening from chest X-rays: a comparison of detection methods. Comput. Biol. Med. 143 105286.
48. Tan M et al 2019 MnasNet: platform-aware neural architecture search for mobile. Proc. IEEE CVPR pp 2820–2828.
49. Dosovitskiy A et al 2021 An image is worth 16×16 words: transformers for image recognition at scale. Proc. ICLR.
50. Hubara I et al 2017 Quantized neural networks: training neural networks with low precision weights. J. Mach. Learn. Res. 18(1) 6869–6898.
51. He T et al 2019 Bag of tricks for image classification with convolutional neural networks. Proc. IEEE CVPR pp 558–567.
52. Litjens G et al 2017 A survey on deep learning in medical image analysis. Med. Image Anal. 42 60–88. https://doi.org/10.1016/j.media.2017.07.005
53. Shen D, Wu G and Suk H-I 2017 Deep learning in medical image analysis. Annu. Rev. Biomed. Eng. 19 221–248. https://doi.org/10.1146/annurev-bioeng-071516-044442
54. Zhou B et al 2016 Learning deep features for discriminative localization. Proc. IEEE CVPR pp 2921–2929. https://doi.org/10.1109/CVPR.2016.319
55. Pan S J and Yang Q 2010 A survey on transfer learning. IEEE Trans. Knowl. Data Eng. 22(10) 1345–1359. https://doi.org/10.1109/TKDE.2009.191