Computational efficiency of the implementation of algorithms in Deep Learning applications for health in large-scale architectures

Félix Armando Mejía Cajicá

Doctoral thesis to qualify for the title of Doctor in Computer Science

Advisors:

Carlos Jaime Barrios Hernández
Habilitation à Diriger des Recherches in Computer Science

Michel Riveill
Habilitation à Diriger des Recherches in Computer Science

John Anderson García Henao
Doctor of Philosophy in Computer Science

Universidad Industrial de Santander
Facultad de Ingenierías Fisicomecánicas
Escuela de Ingeniería de Sistemas e Informática
Doctorado en Ciencias de la Computación
Bucaramanga
2026

Acknowledgements

I want to express my deepest thanks to my advisors, Carlos Jaime Barrios Hernández, Michel Riveill and John Anderson García Henao, who with their wisdom, patience, and constant support provided me with the necessary guidance to overcome every challenge on this academic journey. Thanks to their dedication and trust, this thesis has been made possible.

To my beloved mother, Tulia Cajicá de Mejía, whose unconditional love and support have been the driving force behind me. Thank you for believing in me even in the most difficult moments, and to my father, Feliciano Mejía Flórez, who guides me from heaven and inspires me to be better each day.

To my wife, Yolanda Acevedo Silva, for her infinite patience, understanding, and love. Your constant support and faith have been my refuge and motivation at every stage. This achievement is also the result of your dedication and sacrifice.

To all of you, my sincerest recognition and gratitude. This accomplishment is as much yours as it is mine.

Table of Contents

List of Tables

List of Figures

Abstract

Title: Computational efficiency of the implementation of algorithms in Deep Learning applications for health in large-scale architectures.[1]

Author: Félix Armando Mejía Cajicá.[2]

Keywords: Population Based Training (PBT), Generative Enhanced Neural Architecture Search (GenENAS), Knowledge Distillation, Energy Efficiency .

Description: The deployment of *Deep Learning* models for medical diagnosis is dominated by oversized architectures whose computational and energy costs make them impractical in resource-constrained environments. This thesis proposes, implements, and validates a three-phase framework—generative architecture search, knowledge distillation, and energy-efficiency analysis during inference—validated on the multilabel classification of chest X-rays from the CheXpert dataset. The first stage is GenENAS (*Generative Enhanced Neural Architecture Search*), which employs large language models (GPT-4 and Llama 2) as black-box optimizers: the LLM iteratively proposes architectures that are trained and evaluated, receiving feedback based on the obtained AUC-ROC. The resulting architecture (1.91 M parameters, 7.28 MB, AUC-ROC=0.869) is approximately 30 times smaller than ResNet-152 (58.15 M, 232.62 MB, AUC-ROC=0.875), with a gap of only 0.006 points, highlighting the over-parameterization of standard architectures. The second stage transfers, through knowledge distillation, the predictive capability of a teacher model (DenseNet-121) to lightweight student models (GenENAS, MobileNet V2, and EfficientNet-B0). The best distilled student model nearly matches the teacher in terms of AUC-ROC (0.884 versus 0.891), while achieving a 67.9% reduction in model size and number of parameters. The third stage quantifies the energy impact: the compact distilled models reduce energy consumption and carbon footprint by approximately 45%, while nearly doubling integrated energy efficiency (AUC-ROC per joule) compared with the teacher model. Overall, this research demonstrates that the combination of generative architecture search, knowledge distillation, and energy-aware design enables the development of robust and sustainable models that can be deployed on resource-constrained devices without reliance on cloud infrastructure, thereby promoting the democratization of medical artificial intelligence in alignment with the *Green AI* paradigm.

[1] Doctoral thesis

[2] Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática.

Advisors: Ph.D. Carlos Jaime Barrios Hernández (cbarrios@uis.edu.co), Ph.D. Michel Riveill (michel.riveill@inria.fr), Ph.D. John Anderson García Henao (John.GarciaHenao@balgrist.ch)

Resumen

Título: Eficiencia computacional de la implementación de algoritmos en aplicaciones de aprendizaje profundo para la salud en arquitecturas a gran escala.[3]

Autor: Félix Armando Mejía Cajicá.[4]

Palabras clave: Entrenamiento basado en población (PBT), búsqueda de arquitectura neuronal generativa mejorada (GenENAS), destilación de conocimiento, eficiencia energética.

Descripción: El despliegue de modelos de *Deep Learning* para el diagnóstico médico está dominado por arquitecturas sobredimensionadas cuyo costo computacional y energético las hace inviables en entornos con recursos limitados. Esta tesis propone, implementa y valida un *framework* de tres fases encadenadas —búsqueda generativa de arquitecturas, destilación de conocimiento y análisis de eficiencia energética en inferencia— validado en la clasificación multietiqueta de radiografías de tórax del conjunto de datos CheXpert. La primera etapa es GenENAS (*Generative Enhanced Neural Architecture Search*), que emplea modelos de lenguaje de gran escala como optimizadores de caja negra: el LLM propone iterativamente arquitecturas que se entrenan y evalúan, retroalimentándose con el AUC-ROC obtenido. La arquitectura resultante (1.91 M de parámetros, 7.28 MB, AUC-ROC=0.869) es aproximadamente 30 veces más pequeña que ResNet-152 (58.15 M, 232.62 MB, AUC-ROC=0.875), con una brecha de apenas 0.006 puntos, lo que evidencia el sobredimensionamiento de las arquitecturas estándar. La segunda etapa transfiere, mediante destilación de conocimiento, la capacidad predictiva de un modelo maestro (DenseNet-121) a estudiantes ligeros (GenENAS, MobileNet V2 y EfficientNet-B0). El mejor estudiante destilado iguala prácticamente al maestro en AUC-ROC (0.884 frente a 0.891) con una reducción del 67.9% en tamaño y parámetros. La tercera etapa cuantifica el impacto energético: los modelos compactos destilados reducen cerca de un 45% el consumo energético y la huella de carbono, y prácticamente duplican la eficiencia energética integrada (AUC-ROC por julio) respecto al maestro. En conjunto, la investigación demuestra que la combinación de búsqueda generativa, destilación y diseño energéticamente consciente permite construir modelos robustos y sostenibles, desplegables en dispositivos con recursos limitados y sin dependencia de la nube, promoviendo la democratización de la inteligencia artificial médica en consonancia con el paradigma de la *Green AI*.

[3]Tesis Doctoral

[4]Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática.
Directores: Ph.D. Carlos Jaime Barrios Hernández (cbarrios@uis.edu.co), Ph.D. Michel Riveill (michel.riveill@inria.fr), Ph.D. John Anderson García Henao (John.GarciaHenao@balgrist.ch)

Introduction

Russell and Norvig define Artificial Intelligence (AI) as the ability of machines to use algorithms that enable them to learn from data and apply that learning to decision-making in a manner similar to that of humans [1]. However, unlike humans, machines provide continuous operational availability, the ability to process large volumes of information, and lower error rates in certain tasks. For this reason, advancing research in this field is essential to further maximize the usefulness of these tools.

Due to the increase in computational capabilities, which have grown not only in processing speed but also in information storage capacity, AI-based technologies are widely used to help people achieve significant improvements and greater efficiency in nearly every aspect of life. The idea that machines or computer programs could learn to make decisions began with the development of expert systems, in which specialists' knowledge and decision-making rules were encoded . Many decision-making processes, such as disease diagnosis and treatment, were significantly accelerated, leading to substantial improvements in the quality of healthcare services [2].

The complexity of developing expert systems gave rise to new techniques that enabled the creation of algorithms capable of making decisions more automatically. This led to classical Machine Learning, with algorithms such as Support Vector Machines and Random Forests, which laid the foundations for predictive analytics.

The turning point came with the emergence of Deep Learning and its deep neural networks, such as Convolutional Neural Networks (CNNs), which automate representation learning directly from raw data, revolutionizing fields such as computer vision applied to medical imaging [3]. The rapid growth of deep learning models has produced significant advances, transforming AI from a

distant promise into a driving force behind modern medicine through the automated interpretation of medical images. However, these advances have been accompanied by a substantial increase in the size and complexity of model architectures, resulting in higher computational costs and making their deployment in resource-constrained clinical environments more challenging.

More recently, a paradigm shift has been identified with the emergence of Large Language Models (LLMs) and foundation models such as ChatGPT, Gemini, and Llama. These systems are not only redefining human–machine interaction but also promise to analyze and integrate vast amounts of information.

For AI to fulfill its transformative role in healthcare on a global scale, it is necessary to develop models that are accurate, efficient, compact, and deployable beyond high-performance computing infrastructures. To address this challenge, this thesis proposes a framework composed of three stages: GenENAS for generative neural architecture search, KD-MLC for knowledge distillation in multi-label classification, and a third stage dedicated to analyzing energy efficiency during inference.

The first chapter presents a review of the state of the art, beginning with the emergence of deep learning and covering advanced population-based training techniques and tools for optimizing neural network architectures across a variety of applications. It also discusses the energy consumption and carbon footprint associated with these models.

The second chapter introduces the first stage of the proposed framework: GenENAS (Generative Enhanced Neural Architecture Search). This chapter describes its theoretical foundations, design, and implementation, as well as the parallelization mechanisms employed in the methodology, using chest X-ray images as the primary use case.

The third chapter analyzes, designs, and develops the second stage of the framework, KD-MLC (Knowledge Distillation for Multilabel Image Classification), in which the Knowledge Distillation technique is applied.

Finally, the fourth chapter designs and develops the third stage of the framework, in which the models obtained from the previous stage are evaluated for energy efficiency during inference

across different data sizes. The results demonstrate that the proposed framework yields models that achieve a superior balance among predictive performance, stability, and energy efficiency.

Lastly, the fifth chapter presents the conclusions derived from this research and outlines future research directions that may further strengthen and extend this work.

Research Problem

The implementation of Deep Learning (DL) algorithms has been made possible by the increase and diversity of computational capabilities, which not only enable repetitive parallel processing but also support the processing of large volumes of data. Medicine is one of the fields that have benefited the most from this possibility of implementation and execution, posing at this time for computer science challenges that involve the identification of good computational architectures (in terms of computational efficiency, cost and relevance) as well as such as the portability of the algorithms and the possibilities of scalability (not only considering the data, but also the respective process threads).

These challenges raise questions such as what computational architectures can best process DL applications, like identifying factors of efficiency, scalability, portability, and accuracy to propose performance patterns, as well as other issues related to how algorithms and implementation mechanisms are proposed, maintaining their consistency, when a specific problem is solved by changing the scale of both data and processes.

The computational demands of modern deep learning models pose a significant barrier to their deployment in real-world healthcare settings, particularly in developing countries and resource-constrained regions, where access to high-performance computing infrastructure is limited. As a result, the adoption of these models remains restricted despite their demonstrated clinical potential.

This limitation becomes even more critical in healthcare scenarios that require rapid decision-making, such as emergency care, patient triage, disease screening, and point-of-care diagnosis. In these settings, models must operate under strict constraints regarding memory

consumption, inference latency, and computational capacity. However, most deep learning architectures are primarily designed to maximize predictive accuracy, often without considering deployment requirements related to efficiency, portability, and sustainability.

To address these challenges, several research directions have emerged. Neural Architecture Search (NAS) aims to automatically discover efficient neural network architectures, while Knowledge Distillation (KD) transfers knowledge from large, high-performing models to smaller and more efficient ones. Similarly, the Green AI paradigm promotes the development of computationally sustainable systems by considering energy consumption and carbon emissions as relevant evaluation criteria. Although each of these approaches has demonstrated significant benefits, they have generally been studied and applied independently. Consequently, there is a lack of integrated methodologies that simultaneously optimize predictive performance, model compactness, deployment efficiency, and energy sustainability.

This situation raises the following research question: How can a unified framework be designed to integrate Architecture Search, Knowledge Distillation, and Energy Analysis to generate compact deep learning models that maintain high predictive performance while reducing computational requirements, energy consumption, and carbon emissions in healthcare environments with limited computational resources?

To address this problem, this thesis proposes a three stages framework that combines automatic neural architecture generation, knowledge distillation, and energy-efficiency analysis. This integrated approach enables the development of lightweight and sustainable deep learning models without compromising diagnostic performance.

Motivation

The expansion of deep learning in the healthcare sector has been limited by a significant gap in technological infrastructure: hospitals, clinics, and healthcare centers in developing countries are unable to adopt high-performance clinical diagnostic models because the high cost of the hardware

required to implement these systems is simply beyond their reach.

Compact models enable fast inference with performance very close to that of robust models, allowing them to run on devices with limited computational resources such as desktop computers, medical tablets, portable X-ray equipment, and even mobile medical devices. This makes it possible to provide diagnostic support in centers without reliable internet access, reducing the time between image acquisition and clinical decision-making, and facilitating the adoption of AI-assisted tools, thereby helping to bridge the technological gap in healthcare without the need to invest in server infrastructure.

In critical situations, such as patient triage, sepsis detection, or emergency radiology in remote areas without network connectivity, models that run locally on available hardware become essential and are a valuable tool in these scenarios. The ability to perform inference at the point of care, without relying on external servers, transforms AI from a hospital-based technology into a truly implementable clinical instrument.

It is important to consider whether environmental care policies align with the Green AI trend, which aims to reduce global warming. Large models consume significant energy both during training and inference. By reducing model size, energy consumption and carbon footprint are considerably lowered, providing sustainability benefits essential for systems that must operate continuously.

Regarding privacy and information security, deploying small models directly on medical devices eliminates the need to send sensitive data, such as X-rays or MRI images, to external servers. This protects patient confidentiality and complies with international regulations, including the Health Insurance Portability and Accountability Act (HIPAA) and the General Data Protection Regulation (GDPR).

Finally, by eliminating the need for expensive, high-capacity computing infrastructure, access to advanced artificial intelligence technologies can be broadened and democratized. Consequently, compact Deep Learning models represent not only a technological advancement but also an ethical and social contribution, enabling equitable access to diagnostic support tools that can

positively impact patients' quality of life across diverse healthcare environments.

Research Aim

Propose a methodology that allow to reduce the cost of searching for hyper-parameters to achieve the desired accuracy (volume of messages exchanged and volume of weights) when searching for hyper-parameters, and then also to reduce the cost of training the model to build a model with a small memory footprint to be deployed on mobile devices.

Research Objectives

- Evaluate PBT techniques to identify hyperparameters and metrics needed in performance evaluation of deep neural networks.
- Analyze implementation mechanisms of PBT techniques on multiple GPUs to guarantee a balanced deployment and low energy consumption.
- Design policy networks for the PBT controller to improve performance in the neural architecture search and to minimize the memory impact.
- Define execution patterns to establish deployment strategies of PBT models in large scale architectures.
- Analyze the proposed framework by evaluating performance and the relationship of implementation patterns with precision and sensitivity.

Hypothesis

Population-based learning (PBT) techniques, such as reinforcement learning (RL) and evolutionary algorithms, require large-scale Deep Learning search models to achieve the desired accuracy with an

impact on memory consumption per node and parameter (weight) transfer in distributed systems. Therefore, the computational efficiency and accuracy of a PBT implementation in distributed computing architectures is limited by the impact on memory due to centralization.

Contributions

This thesis addresses the development and deployment of computationally efficient deep learning models for multi-label classification, investigated through the use case of pulmonary medical imaging. The central original contribution of this work is an integrated, end-to-end framework that unifies three stages —Generative Enhanced Neural Architecture Search (GenENAS stage), Knowledge Distillation for MultiLabel Image Classification (KD-MLC stage), and Energy Efficiency Analysis (EEA Stage)— under a single computational-efficiency objective, in such a way that the lightweight architecture discovered in the first stage is the one transferred in the second stage and the one whose energy efficiency is assessed in the third stage. Beyond combining established and adapted techniques, the novelty lies in their chaining into a coherent pipeline in which a single model is progressively optimized and evaluated for deployment in resource-constrained health environments. This framework comprises the following three components, each of which also constitutes a contribution in its own right:

- **Generative Enhanced Neural Architecture Search for the Medical Domain (GenENAS)**. An LLM-driven neural architecture search method, adapted from GENIUS [4], which utilizes Large Language Models (LLMs) such as GPT-4 and Llama 2 as black-box optimizers. This work extends and adapts that framework, achieving the following contributions to the state of the art:
  - *Adaptation to the multi-label medical domain:* While GENIUS operates on CIFAR-10, a standard academic multi-class dataset where each image belongs to a single category. GenENAS is applied to the CheXpert dataset. CheXpert is a large-scale clinical chest X-ray dataset where each image can represent multiple simultaneous pathologies. This

required a fundamental architectural modification: GenENAS replaces the standard Softmax output layer with independent Sigmoid functions for each of the five target pathologies (atelectasis, cardiomegaly, consolidation, edema, and pleural effusion). This shift results in a binary multi-label classification for each pathology, where each output yields an autonomous probability between 0 and 1, reflecting the clinical reality that a patient may be diagnosed with several concurrent conditions.

- *Shift in optimization metrics:* GENIUS optimizes for accuracy, which is suitable for CIFAR-10 but inadequate for medical datasets characterized by severe class imbalance. GenENAS adopts the Area Under the ROC Curve (AUC-ROC) as the primary optimization metric because it is robust to class imbalance and evaluates the model's discriminative power across all possible decision thresholds.
- *LLM Diversification:* GenENAS demonstrates that the methodology is transferable to different LLMs by utilizing GPT-4 and Meta's Llama 2 (7B). Llama 2, an open-source model with 7 billion parameters, is considerably more lightweight than GPT-4. This proves that architectural reasoning capabilities are not exclusive to large-scale commercial models, thereby increasing the accessibility and reproducibility of this methodology in environments where proprietary APIs are restricted or economically unfeasible.
- *Generative Parallelization (GenENAS Parallel):* GenENAS introduces a parallel extension not considered in the original GENIUS framework. Using the LLM, GenENAS Parallel proposes N candidate architectures per iteration and trains them simultaneously on multiple GPUs in High-Performance Computing (HPC) environments. Experiments demonstrate that GenENAS Parallel reaches an AUC-ROC of 0.87 in only 5 iterations (compared to 7 in serial mode), reducing the search time from 285.15 to 225.52 hours. Furthermore, this research quantifies the energy trade-off: the parallel version consumes 41.29% more energy (645,435 kJ vs. 456,815 kJ) because it trains 10 architectures rather than 7 in the serial version. This analysis of the compromise between

temporal acceleration and energy cost is an original contribution of this research.

  - *Comprehensive Energy Analysis:* GenENAS incorporates a holistic evaluation framework that includes the number of parameters, model footprint (MB), execution time, and energy consumption for every evaluated architecture in both serial and parallel modes. This multi-faceted assessment is a contribution absent from the original GENIUS framework.
  - *Dataset Scale and Complexity:* Unlike CIFAR-10 (60,000 images, 10 balanced classes), CheXpert contains 223,414 high-resolution radiographs from 65,540 patients with 14 clinical observations. GenENAS operates on 187,641 training images and 7,505 test images, demonstrating scalability to real-world, high-volume clinical datasets.

- **Knowledge Distillation for Multi-Label Classification (KD-MLC)**. The original method by Hinton [5] established the foundations of Knowledge Distillation (KD) for multi-class classification. The method proposed in this thesis adapts and extends this method to address multi-label medical image classification through the following contributions:

  - **Reformulation of Distillation for Multi-Label Problems:** In the context of CheXpert, it is necessary to replace Hinton's Softmax-based approach with independent Sigmoid functions for each pathology, treating each class as an autonomous binary classification problem. Consequently, the "hard loss" is reformulated using `BCEWithLogitsLoss` (Binary Cross-Entropy with logits) instead of categorical cross-entropy. The "soft loss" is redefined as a per-class Bernoulli KL divergence between the Sigmoid-smoothed probabilities (with temperature scaling) of the teacher and the student. While Hinton's KL divergence operates on categorical distributions, this research applies it to C independent Bernoulli distributions.
  - **Incorporation of Class-Wise Weight Balancing:** To address the severe imbalance in CheXpert (e.g., Consolidation at 6.63% vs. Pleural Effusion at 38.61%), this thesis incorporates a weighting mechanism within the `BCEWithLogitsLoss` function. By

assigning weights inversely proportional to pathology frequency, we ensure that minority classes contribute proportionally to the gradient, preventing the model from biasing predictions toward majority classes—a challenge not addressed in Hinton's original algorithm.

  - **Uncertainty Handling Strategies:** The CheXpert dataset includes labels with four states: positive (1), negative (0), uncertain (-1), and not mentioned (null or blank). This thesis proposes two imputation strategies: *Zeros* (treating uncertainty as absence) and *Ones* (treating uncertainty as presence). Results show that the *Ones* strategy consistently yields superior AUC-ROC and F1-Score results. This experimental dimension of clinical uncertainty management has no equivalent in the foundational KD work.

- **Energy Efficiency Analysis (EEA)**. This thesis incorporates, as the third stage of the framework, a systematic energy characterization of the entire pipeline—from architecture search to clinical inference—which is rarely reported in the NAS and Knowledge Distillation literature. Its specific contributions are the following:

  - **Per-stage energy accounting of the complete pipeline:** The execution time, energy consumption, and CO_2 emissions of every stage are measured and reported. The architecture search consumed 456,815.1 kJ in serial mode and 645,435.4 kJ in its parallel variant, while the distillation process consumed 2,106.23 kJ (0.1519 kgCO_2eq) under the Zeros strategy and 2,061.37 kJ (0.1486 kgCO_2eq) under the Ones strategy. This end-to-end accounting makes the true computational cost of producing a deployable model explicit and reproducible, in line with the reporting practices promoted by the Green AI paradigm.

  - **Multi-scale inference benchmark:** Energy consumption, CO_2-equivalent emissions, throughput, latency, and peak memory are evaluated for the models (teacher, students, and distilled student) over inference workloads ranging from 1,000 to 40,000 chest X-ray images, with the AUC-ROC and its standard deviation measured across five independent

runs. The benchmark demonstrates that energy and emissions scale linearly with the workload, enabling reliable extrapolation to large-scale clinical deployments, and that the distilled compact models reduce the energy and carbon footprint of inference while preserving most of its diagnostic performance.

– **Energy efficiency as an explicit evaluation metric:** The framework adopts the ratio between predictive performance and energy consumed (EE = AUC-ROC/E_{Total}) as a first-class model selection criterion. Under this metric, the compact models nearly double the efficiency of the teacher (9.10×10^{-5} AUC-ROC/J), with Distilled EfficientNet-B0 emerging as the most efficient configuration (1.67×10^{-4}, an 83% improvement). The analysis further reveals that parameter count alone does not determine operational cost—the GenENAS-derived models, despite having the fewest parameters, exhibit intermediate energy consumption and the largest memory footprint—evidencing the need to measure deployment metrics empirically rather than inferring them from model size.

The originality of the proposed framework resides not only in each of the three components individually, but in their integration and in the complementary roles they play: GenENAS yields a compact architecture, reducing the number of parameters and the memory footprint; KDT-MLC then transfers the knowledge of a larger teacher into that compact architecture, maximizing its diagnostic performance without increasing its size; and the energy efficiency analysis quantifies the resulting efficiency. The pipeline thus links architecture discovery, knowledge transfer, and computational sustainability into a single, reproducible process that produces models which are simultaneously lightweight and accurate, suitable for deployment in environments with limited computational resources.

Publications Derived from this Thesis

The research developed throughout this doctoral thesis has been disseminated through articles, presentations at international conferences published in indexed proceedings, a poster presentation,

and publications in indexed journals that were validated through peer review. These contributions are listed below in chronological order and are directly related to the methodologies, experiments, and results presented in this document.

Journal Articles

- F. A. Mejía-Cajicá, C. J. Barrios-Hernández, M. Riveill, and J. A. García-Henao, "Optimización de redes neuronales profundas mediante la destilación de conocimiento para una clasificación eficiente y sostenible de imágenes médicas," *Revista UIS Ingenierías*, vol. 25, no. 1, pp. 49–70, 2026. (Indexed journal, Publindex Category B.) doi: 10.18273/revuin.v25n1-2026005.

- F. A. Mejía Cajicá, J. A. García Henao, C. J. Barrios Hernández, and M. Riveill, "Efficiency in the classification of chest X-ray images through generative parallelization of the neural architecture search," *ACI Avances en Ciencias e Ingenierías*, vol. 17, no. 1, 2025. (Extended version of the work presented at CARLA 2024.) doi: 10.18272/aci.v17i1.3707.

International Conference Papers (Published in Proceedings)

- F. A. Mejía Cajicá, J. A. García Henao, C. J. Barrios Hernández, and M. Riveill, "Analysis of regularization in deep learning models on testbed architectures," in *High Performance Computing (CARLA 2020)*, S. Nesmachnow, H. Castro, and A. Tchernykh, Eds., Communications in Computer and Information Science, vol. 1327. Cham: Springer, 2020. doi: 10.1007/978-3-030-68035-0_13.

- F. A. Mejía Cajicá, J. A. García Henao, C. J. Barrios Hernández, and M. Riveill, "Optimization techniques based on PBT in the search for hyperparameters," in *Workshop 7: HPC and Data Sciences meet Scientific Computing*, Porto Alegre, Brazil, 2022. Zenodo.

doi: 10.5281/zenodo.7473624.

- F. A. Mejía Cajicá, J. A. García Henao, C. J. Barrios Hernández, and M. Riveill, "Assessment of generative neural network architecture search in chest X-ray image classification," in *Bio-CARLA 2023 Workshop Proceedings*, Cartagena de Indias, Colombia, Sep. 2023. Zenodo. doi: 10.5281/zenodo.17053330.

- F. A. Mejía Cajicá, J. A. García Henao, C. J. Barrios Hernández, and M. Riveill, "Efficiency in the classification of chest X-ray images through generative parallelization of the neural architecture search," presented at the *Latin American High Performance Computing Conference (CARLA 2024)*, 2024. Santiago de Chile - CHILE. (Extended version published in *ACI Avances en Ciencias e Ingenierías*, vol. 17, no. 1, 2025; see Journal Articles above.)

Poster Presentations

- F. A. Mejía Cajicá, J. A. García Henao, C. J. Barrios Hernández, and M. Riveill, "Improving chest X-ray image classification via parallelized generative neural architecture search," Poster presented at the *Platform for Advanced Scientific Computing Conference (PASC 24)*, Zurich, Switzerland, Jun. 2024. [Online]. Available: `https://pasc24.pasc-conference.org/presentation/?id=pos163&sess=sess151`

1. Literature Review

The use of artificial intelligence (AI) in medicine is not a 21st-century phenomenon; its roots lie in the early days of computing, when AI pioneers such as Alan Turing and John McCarthy postulated that machines could simulate human thought processes, including decision-making. The initial

ambitions of creating a "General Problem Solver," a single algorithm capable of solving any task, soon gave way to the fundamental idea: high-level intelligence does not emerge from pure logic, but from the application of vast domain-specific knowledge [1].

This paradigm shift, from general logic to specific knowledge, marked the birth of knowledge-based systems, commonly known as expert systems. These represented the first successful attempt to encapsulate the knowledge and reasoning of a human specialist in software [6], and medicine, with its complex diagnosis and treatment structure, became one of its most fertile and challenging fields of application.

These expert systems demonstrated that complex clinical reasoning could be computationally modeled. However, they never achieved widespread adoption in clinical practice due to fundamental limitations. These included complex knowledge acquisition; their brittleness, as they would fail when faced with cases that did not fit perfectly within their rules; and their inability to learn, as they could not gain new knowledge through experience or discover new patterns from new patient data. Furthermore, their knowledge could only be updated manually by a knowledge engineer, making their adaptability and updates very complex.

Due to these shortcomings, the transition to the next era of AI in medicine was spurred: machine learning, in which systems learn to identify patterns directly from data rather than being programmed with explicit rules. The era of expert systems, though surpassed, laid the indispensable theoretical and conceptual foundations for the Deep Learning revolution being experienced today.

1.1 Knowledge-Based Systems

Also known as expert systems, they are computer programs designed to emulate the decision-making ability of a human expert within a very specific and restricted domain of knowledge. Instead of processing data with conventional algorithms, their purpose is to reason over a knowledge base to solve complex problems that would normally require the expertise, intuition, and experience of a specialist. The fundamental purpose was to capture, preserve, and distribute scarce expert

knowledge, enabling other professionals to consult them and receive high-quality guidance [2].

The key innovation of expert systems was the explicit separation between the domain knowledge and the reasoning mechanism. This modular architecture consisted of two core components:

1. **The Knowledge Base:** This was the repository of expert knowledge, which was divided into:
    - **Fact Base:** Factual and dynamic information about the problem at hand (e.g., a patient's symptoms, lab results).
    - **Rule Base:** Expert knowledge encoded in the form of production rules, usually with an IF-THEN structure. These rules served as heuristics, or "rules of thumb," that a specialist would use to reach a conclusion.
2. **The Inference Engine:** This acted as the logical "brain" of the system. Its function was to apply the rules to the available facts to deduce new conclusions. This engine primarily used two reasoning strategies:
    - **Forward Chaining:** A data-driven strategy. It started from the initial facts and applied all possible rules to generate new conclusions, repeating the process until no more could be deduced.
    - **Backward Chaining:** A goal-driven strategy, ideal for diagnostics. It started with a hypothesis (e.g., "the patient has pneumonia") and worked backward, looking for rules that could confirm it and prompting the user for the information needed to validate the conditions of those rules.

The canonical and most influential example in medicine is MYCIN, developed at Stanford University in the 1970s to diagnose blood infections and recommend antibiotic therapy [7]. MYCIN used backward chaining and pioneered the management of uncertainty inherent in medical reasoning through Certainty Factors. These allowed the system to weigh evidence and present diagnoses with an associated degree of confidence, rather than a simple yes-or-no answer [7]. In formal evaluations, MYCIN's performance was comparable to that of human experts, demonstrating the approach's

viability and inspiring later systems like INTERNIST-1, which focused on the vast field of internal medicine [8].

The legacy of expert systems is undeniable: they demonstrated that complex clinical reasoning could be modeled computationally and established the foundations for clinical decision support systems. However, their widespread adoption was hindered by fundamental limitations that paved the way for the next generation of AI. The main limitations include:

- **The Knowledge Acquisition Bottleneck:** The process of interviewing experts and manually coding their knowledge into rules was extremely slow, expensive, and prone to inconsistencies.
- **Brittleness:** The systems were highly competent within their niche but would fail abruptly and unpredictably when faced with situations not explicitly covered by their rules, as they lacked "common sense."
- **Static Knowledge and Inability to Learn:** This was their most critical limitation. An expert system could not learn from new cases or data. Its knowledge base was fixed and could only be updated manually. It could not discover patterns or correlations that the human expert had not previously articulated.

1.2 Machine Learning

The inherent limitations of expert systems, particularly the knowledge acquisition bottleneck and their inability to learn from experience, made it clear that a fundamentally different approach was needed. Artificial intelligence experts found that developing systems that learn patterns directly from data is more feasible and scalable for solving complex problems than manually encoding human knowledge. This realization marked the transition from the era of explicit knowledge to the era of inferred knowledge, giving rise to the field of Machine Learning (ML).

Unlike expert systems, where rules are programmed by humans, Machine Learning is defined by its ability to learn without being explicitly programmed. Arthur Samuel, a pioneer in

the field, described it in 1959 as the field of study that enables computers to learn without being explicitly programmed. A more formal and operational definition, proposed by Tom Mitchell, states that a program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance in tasks T, as measured by P, improves with experience E [9].

In essence, the paradigm shifts from RULES + DATA -> ANSWERS (classical programming) to DATA + ANSWERS -> RULES (Machine Learning). The goal of the ML algorithm is to infer a function or a model that can make predictions or decisions on new, unseen data.

1.2.1 The Paradigms of Machine Learning

The field of Machine Learning is traditionally structured into three main paradigms, each defined by the nature of the data and the type of "learning signal" available.

a) Supervised Learning

This is the most common and well-studied paradigm. In supervised learning, the algorithm learns from a training dataset that has been previously labeled by a human expert. Each data point consists of an input-output pair (X,y). The objective is to learn a mapping function f such that $y \approx f(X)$, which can generalize correctly to unlabeled data. Supervised learning problems are mainly divided into two categories:

- **Classification:** The output label y is a discrete category. The goal is to assign a class to a new observation. In medicine, this applies to tasks such as diagnosis (e.g., predicting if a tumor is "benign" or "malignant") or risk prediction (e.g., "high," "medium," "low"). Canonical algorithms include Logistic Regression, Support Vector Machines (SVM) [10], Decision Trees, and ensembles like Random Forests [11].
- **Regression:** The output label y is a continuous value. The goal is to predict a numerical value. Medical applications include predicting the length of a hospital stay, estimating

blood pressure from other variables, or predicting the value of a biomarker. Classic algorithms include Linear Regression and its regularized (Ridge, Lasso) [12].

The effectiveness of supervised models has been widely demonstrated in recent medical literature, where they are used to create more accurate risk scoring systems and diagnostic support tools than traditional statistical methods [13].

b) **Unsupervised Learning** In this paradigm, the algorithm works with unlabeled data. There is no predefined "correct answer." The objective is to discover inherent patterns, structures, or clusters within the data itself. The main tasks are:

- **Clustering:** Grouping data into clusters where the members of one group are more similar to each other than to those in other groups. In medicine, this is extremely useful for disease phenotyping—that is, identifying subgroups of patients with similar characteristics who might respond differently to treatments. Common algorithms are K-Means, Hierarchical Clustering, and DBSCAN [14].
- **Dimensionality Reduction:** implifying data by reducing the number of variables (dimensions), either to facilitate visualization (e.g., PCA, t-SNE) or to improve the performance of a subsequent supervised algorithm.

Unsupervised learning is fundamental to precision medicine, as it allows for the discovery of new disease taxonomies based on complex molecular and clinical data.

c) **Reinforcement Learning** In reinforcement learning (RL), an agent learns to make decisions by interacting with an environment. The agent performs actions that change the environment's state and, in return, receives a reward (positive or negative). The agent's goal is to learn a policy (a strategy for taking actions) that maximizes the cumulative reward over time. Unlike supervised learning, the feedback is neither instantaneous nor direct.

In medicine, RL is emerging as a powerful tool for optimizing dynamic treatment regimens, in which the dosage of a medication or the choice of therapy is continuously adjusted based

on the patient's response over time. Its application in the management of chronic diseases, the optimization of chemotherapy, or mechanical ventilation in intensive care is an active and promising area of research [15].

Classical Machine Learning successfully overcame the inability of expert systems to learn. However, its performance critically depended on a prior, often manual and laborious step: feature engineering. Although the algorithm learned automatically, it required a human expert to transform the raw data into an informative, well-structured set of features. For example, to predict heart disease, an expert might need to manually calculate ratios, interactions, or indices from the raw values of a blood test. The performance of the final model was, to a large extent, a reflection of the quality of these hand-crafted features.

This reliance on feature engineering became a new bottleneck, especially for unstructured, high-dimensional data such as medical images and biological signals. The need for systems that could learn not only the final decision but also the relevant features directly from raw data led to the next and most recent evolution in the field: Deep Learning, which will be addressed in the next section.

1.3 Deep Learning

The classical Machine Learning paradigm, despite its success, inherited a bottleneck that limited its potential, especially in domains with high-dimensional, unstructured data such as medicine. The effectiveness of models such as SVMs or Random Forests critically depended on feature engineering, a manual, laborious, and subjective process in which a human expert had to extract and select the most informative variables from the raw data. This dependency meant the model's performance was limited by the quality of the features designed. To overcome this barrier, a subfield of Machine Learning emerged that would change the field forever: Deep Learning.

Deep Learning is based on the use of Artificial Neural Networks (ANNs) with a "deep" architecture—that is, composed of multiple processing layers stacked hierarchically. The funda-

mental and disruptive innovation of Deep Learning is not merely the depth of the network, but its ability to perform automatic representation learning [16].

Unlike classical Machine Learning, where features are designed by humans, in a deep network, each layer learns to transform the data representation from the previous layer into a representation at a slightly higher, more complex level of abstraction. For example, in a medical image analysis task:

- The first layers might learn to detect simple features like edges, gradients, and textures.
- Intermediate layers might combine these features to learn to recognize more complex patterns such as shapes, organ contours, or cellular structures.
- The final layers could integrate these representations to identify high-level concepts, such as the presence of a tumor or signs of a specific pathology.

This process of feature extraction and hierarchization is completely automatic and is optimized end-to-end during training. By eliminating the need for manual feature engineering, Deep Learning has unleashed unprecedented performance, especially in interpreting perceptual data.

1.3.1 Fundamental Architectures of Deep Learning

Fundamental Architectures of Deep Learning The versatility of Deep Learning lies in the development of specialized architectures for different types of data. Three have been particularly transformative:

a) Convolutional Neural Networks (CNNs): The Domain of Vision

CNNs are the quintessential architecture for processing data with a grid-like topology, such as images. Their design is bio-inspired by the human visual cortex. Their power resides in two key operations: convolution, which uses filters (kernels) to efficiently detect local patterns (edges, shapes, textures) by sharing weights; and pooling, which reduces the dimensionality

of the representations to make them more robust to small translations. Iconic architectures such as AlexNet [17] and ResNet [18] have set the standard in computer vision and driven the greatest advances in computational radiology, pathology, and dermatology [3].

b) Recurrent Neural Networks (RNNs) and LSTMs: Handling Sequences

For sequential data, such as time series (ECG, EEG) or text (clinical notes), RNNs were the dominant approach. These networks possess "memory," as their output at time t depends on both the current input and the outputs of previous steps, allowing them to model temporal dependencies. However, simple RNNs suffered from the vanishing gradient problem, which prevented them from learning long-term dependencies. To solve this, more sophisticated variants were developed, such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), which use gating mechanisms to regulate the flow of information and preserve memory across long sequences.

c) Transformers: The Dominant Architecture of the Modern Era

The most disruptive architecture of recent years is the Transformer, initially introduced for machine translation tasks [19]. Its core innovation is the self-attention mechanism, which eliminates the need to process data sequentially as in RNNs. Attention allows the model to weigh the importance of all other parts of the sequence when processing a specific element, capturing complex long-range dependencies much more effectively and enabling massive parallelization of computation.

Although they originated in Natural Language Processing (NLP), giving rise to Large Language Models (LLMs) like BERT and GPT, Transformers have been adapted with overwhelming success to other domains, including computer vision (Vision Transformers, ViT) and medicine. Currently, they are used to analyze genomic sequences and longitudinal medical records, and as a powerful backbone for analyzing medical images, in many cases outperforming CNNs [20].

1.4 State of the Art

Most automatic hyperparameter tuning mechanisms are sequential optimization methods in which the results of each training run with a particular set of hyperparameters inform the search for subsequent hyperparameters [21].

1.4.1 PBT

In standard machine learning practice, training a model consists of adjusting its parameters θ of a model f to maximize the objective function Q, whether for classification, regression, or reconstruction, typically via gradient-based methods like stochastic gradient descent. PBT extends this conventional formulation by treating hyperparameters h not as fixed design choices set before training, but as variables that are optimized jointly with θ and directly on the target metric Q, rather than on a proxy. [21].

First, an *eval* function is defined that evaluates the objective function Q, using the current state of the model f. For simplicity, all terms except θ are ignored and *eval* is defined as a function of the trainable parameters θ alone. The process for finding the optimal set of parameters that maximizes Q is as follows.

$$\theta^{\star} = \underset{\theta \in \Theta}{argmax}\, \text{eval}(\theta) \tag{1.1}$$

A key design feature of PBT is that the evaluation function used to measure population fitness operates independently of the differentiable loss used during gradient updates. The only requirement is that both functions be statistically correlated, meaning that improvements in one tend to reflect improvements in the other [21].

If the model is a neural network, the weights θ are optimized iteratively, using stochastic gradient descent on the objective function Q. Each step of the iterative optimization procedure

updates the model's parameters and is conditioned on some hyperparameter $h \in \mathcal{H}$, where Θ denotes the model parameter space and $\mathcal{H}$ denotes the space of admissible hyperparameter schedules, defined as all sequences $(h_t)_{t=1}^{T}$ where each (h_t belongs to a fixed hyperparameter domain $\mathcal{H}$).

The iterations update the parameters at each step:

$$\theta \leftarrow \text{step}(\theta \mid h) \tag{1.2}$$

These steps are chained together to form a sequence of updates that converges to the optimal solution:

$$\begin{aligned} \theta^* &= \text{optimise}(\theta \mid h) \\ &= \text{optimise}\left(\theta \mid (h_t)_{t=1}^{T}\right) \\ &= \text{step}\Big(\text{step}\big(\ldots \text{step}(\theta \mid h_1) \ldots \mid h_{T-1}\big) \mid h_T\Big) \end{aligned} \tag{1.3}$$

The sequential nature of this optimization makes it computationally prohibitive at scale: convergence to an acceptable θ^* may require a large number of gradient steps, T, each with its own computational overhead, extending training to days or even weeks. Compounding this difficulty, the final quality of θ^* is highly dependent on the entire sequence of hyperparameter values applied during training, a sensitivity that standard fixed-schedule methods cannot adequately address.

Poorly chosen hyperparameters can steer training toward suboptimal solutions or prevent convergence altogether. Identifying adequate values typically requires extensive prior experimentation, which is itself resource-intensive. Since hyperparameter requirements shift across training stages, the search space of valid schedules expands exponentially with the number of iterations. In practice, this complexity is managed by either keeping all hyperparameters constant throughout training or by applying simple predefined schedules, such as learning-rate decay — approximations that sacrifice adaptability for tractability. [21].

$$\theta^* = \text{optimise}(\theta|h^*),\ h^* = \arg\max_{\boldsymbol{h}\in\mathcal{H}^T} \text{eval}(\text{optimise}(\theta|h)) \tag{1.4}$$

To optimize equation (1.4) quickly and in a computationally efficient manner, consider training the N models $\left\{\theta^i\right\}_{i=1}^N$ that form the population P, which are optimized with different hyperparameters $\left\{h^i\right\}_{i=1}^N$. The objective is to find an optimal model across the entire population of models. Instead of using a parallel search approach, the set of partial solutions from the population is used to perform a meta-optimization, in which the hyperparameters h and the weights θ are further adapted based on the performance of the entire population [21].

Algorithm 1 Population Based Training (PBT)

1: **procedure** TRAIN($\mathcal{P}$) ▷ initial population $\mathcal{P}$
2: **for all** $(\theta, h, p, t) \in \mathcal{P}$ **(asynchronously in parallel) do**
3: **while** no end of training **do**
4: $\theta \leftarrow \text{step}(\theta|h)$ ▷ one optimisation step using hyperparameters h
5: $p \leftarrow \text{eval}(\theta)$ ▷ current model evaluation
6: **if** ready$(p, t, \mathcal{P})$ **then**
7: $\hat{h}, \hat{\theta} \leftarrow \text{exploit}(h, \theta, p, \mathcal{P})$ ▷ use the rest of the population to find a better solution
8: **if** $\theta \neq \hat{\theta}$ **then**
9: $h, \theta \leftarrow \text{explore}(\hat{h}, \hat{\theta}, \mathcal{P})$ ▷ produce new hyperparameters
10: $p \leftarrow \text{eval}(\theta)$ ▷ new model evaluation
11: **end if**
12: **end if**
13: update $\mathcal{P}$ with new $(\theta, h, p, t+1)$ ▷ update population
14: **end while**
15: **end for**
16: **return** θ with the highest p in $\mathcal{P}$
17: **end procedure**

Algorithm 1 trains a population $\mathcal{P}$ of N models asynchronously in parallel. Each member of the population is defined by a tuple (θ, h, p, t), where θ represents the model weights, h the hyperparameters, p the current fitness value, and t the training step counter. The procedure iterates through four operations:

Step (line 4): Each member performs a standard gradient update on its weights θ using its current hyperparameters h (learning rate, batch size, regularization strength). This is the

conventional training step shared with any non-population-based method.

Eval (line 5): The model is evaluated on a held-out validation set using a task-appropriate metric — such as accuracy, AUC-ROC, or BLEU score — that is independent of the differentiable training loss. This evaluation produces the fitness value p that drives all subsequent population-level decisions.

Exploit (line 7): Periodically, once the readiness criterion is met (line 6)—after N training steps—the population is ranked by fitness p and a truncation-selection rule is applied. If the current member falls within the bottom [20%] of the ranking, it is truncated: its weights θ and hyperparameters h are overwritten with those of another member sampled uniformly at random from the top [20%]. If the member does not fall in the bottom [20%], no replacement occurs and it continues training with its own weights. This direct weight transfer is possible because all population members share the same network architecture.

Explore (line 9): If a weight replacement occurred ($\theta \neq \hat{\theta}$), the inherited hyperparameters are perturbed to introduce diversity—for example, each hyperparameter is multiplied by 0.8 or 1.2 with equal probability. The model is then re-evaluated (line 10) and resumes training from the inherited weights with the perturbed hyperparameters.

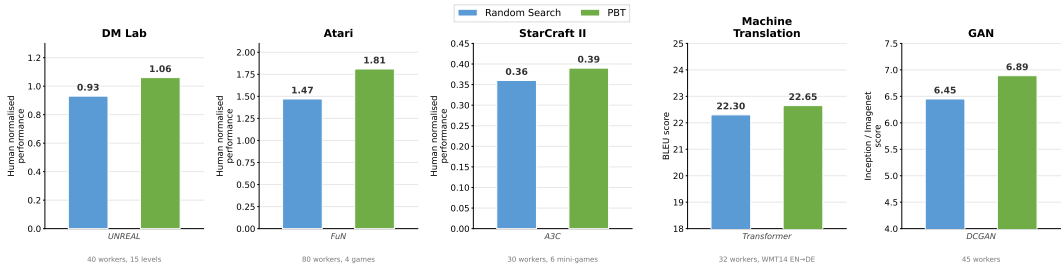
At termination, the algorithm returns the member with the highest fitness value p across the entire population (line 16). The key advantage of this scheme over conventional hyperparameter search is that computational resources are dynamically reallocated during training: underperforming configurations are abandoned early and redirected toward more promising regions of the search space, while successful configurations continue to be refined [21].

Figure 1 summarises the results reported by Jaderberg et al. [21] across five diverse domains: reinforcement learning (DM Lab, Atari, StarCraft II), sequence-to-sequence modelling (Machine Translation), and generative modelling (GAN). In every case, PBT was given the same total computational budget as the random search baseline — the same number of workers trained for the same number of steps — yet consistently achieved equal or superior final performance. The gains are most pronounced in reinforcement learning tasks, where the dynamic adaptation of

hyperparameters such as learning rate and entropy cost during training proves particularly beneficial, as these parameters require different values at different stages of the learning process.

Figure 1

Comparison of Population-Based Training (PBT) versus random search across five domains.



Each panel shows the final performance of the best member in the population (green bar, PBT) against the best result from an equivalent random hyperparameter search (blue bar). **DM Lab**: human-normalised score of UNREAL agents (40 workers, 15 levels). **Atari**: human-normalised score of Feudal Networks (80 workers, 4 games). **StarCraft II**: human-normalised score of A3C agents (30 workers, 6 mini-games). **Machine Translation**: BLEU score of Transformer networks on WMT 2014 EN→DE (32 workers). **GAN**: Inception score of DCGAN (45 workers). In all cases PBT matches or surpasses random search while using the same fixed computational budget. Reproduced from Jaderberg et al. [21].

The effectiveness of PBT is conditioned on several configuration choices. In reinforcement learning settings, parameters such as learning rate, entropy cost, and auxiliary loss weights have a pronounced effect on stability, and inadequate policy exploration can cause the optimization to collapse or converge prematurely to local optima. Population size is another critical factor: excessively small populations produce high variance in outcomes because the exploitation step has few candidates to draw from, increasing the risk of population-wide local convergence. Empirical results from [21] suggest that populations of 20 to 40 workers offer a practical balance between computational cost and result consistency.

1.4.2 EPBT

Evolutionary Population-Based Training (EPBT) augments the PBT framework by introducing a second layer of meta-optimization: in addition to adapting hyperparameters, the algorithm coevolves the loss functions used to train each member of the population. These loss functions are represented through a Taylor expansion parameterization (TaylorGLO), which EPBT can optimize using black-box genetic operators without requiring gradient information through the loss [22]. Because simultaneously adapting weights and loss functions risks degenerate solutions, EPBT incorporates two stabilizing mechanisms: a diversity-preserving selection heuristic called Novelty-Pulsation, and a population-based knowledge distillation step that transfers learned representations from high-performing members to weaker ones, preventing overfitting. Evaluated on image classification benchmarks (CIFAR-10 and SVHN), this combination yields faster convergence and higher final accuracy than both fixed-loss PBT and non-population baselines [22], because EPBT relies entirely on evolutionary operators rather than gradient-based meta-updates, it extends population-based meta-learning to settings where differentiating through the training process is computationally intractable or ill-defined, a significant practical advantage over gradient-based meta-learning methods such as MAML [22].

At the beginning of each generation g, the population $\mathcal{P}_g$ consists of individuals P_{gi}. Each individual is defined as $P_{gi} = \{\theta_{gi}, h_{gi}, p_{gi}\}$ where θ_{gi} represents the model weights (of a user-specified DNN architecture), h_{gi} is a set of hyperparameters, and p_{gi} is a real scalar fitness value.

The process unfolds in the following steps for each generation:

1. In the first step, the fitness value p_{gi} is used to select promising individuals to form a parent set $\hat{\mathcal{P}}_g \subseteq \mathcal{P}_g$, where $|\hat{\mathcal{P}}_g| \leq |\mathcal{P}_g|$.

2. In the second step, the parent set $\hat{\mathcal{P}}_g$ is used to create a set $\mathcal{N}_g$ containing new individuals N_{gi}. Each of these new individuals inherits the model weights θ_{gi} unchanged from its parent but with modified hyperparameters $\hat{h}_{gi}$. Genetic operators are used to generate $\hat{h}_{gi}$ from h_{gi}.

3. In the third step, each new individual N_{gi} is evaluated by training its model θ_{gi} on a task or dataset, thereby creating a model with updated weights $\hat{\theta}_{gi}$. The validation performance of $\hat{\theta}_{gi}$ is used to determine a new fitness value $\hat{p}_{gi}$.

Thus, at the end of generation g, the population as a whole contains the evaluated individuals for the next generation, defined as $\mathcal{P}_{g+1} = \{\hat{\theta}_{gi}, \hat{h}_{gi}, \hat{p}_{gi}\}$. This process is repeated for several generations until the best individual in the population converges.

The EPBT pipeline integrates four specialized components that work in concert within the evolutionary loop [22]. First, a set of genetic operators, mutation and crossover, is applied specifically to hyperparameter vectors, introducing controlled variation without disrupting inherited weight configurations. Second, Novelty-Pulsation guides selection by rewarding individuals that are both high-performing and sufficiently distinct from the rest of the population, preventing premature convergence to a single local optimum. Third, loss functions are encoded using the TaylorGLO representation, enabling EPBT to evolve them as continuous numeric vectors amenable to standard genetic operations. Fourth, Population-Based Distillation (PBD) transfers knowledge from the current best-performing model to lower-ranked members, accelerating their training and reducing the risk of population-wide stagnation.

Algorithm 2 Evolutionary Population Based Training (EPBT)

procedure EPBT($\mathcal{P}_0$) ▷ initial population $\mathcal{P}_0$ of N individuals
  for $g = 0$ **to** $n - 1$ **do**
    $\hat{\mathcal{P}}_g \leftarrow \tau(\mathcal{P}_g)$ ▷ tournament selection
    for all $(\theta_{gi}, h_{gi}, p_{gi}) \in \hat{\mathcal{P}}_g$ **do**
      $\hat{h}_{gi} \leftarrow \xi(\gamma(h_{gi}))$ ▷ mutation then crossover on hyperparameters
      $N_{gi} \leftarrow (\theta_{gi}, \hat{h}_{gi})$ ▷ weights inherited from parent
      $\hat{\theta}_{gi}, \hat{p}_{gi} \leftarrow \text{eval}(N_{gi})$ ▷ train and evaluate the individual
      $\hat{N}_{gi} \leftarrow (\hat{\theta}_{gi}, \hat{h}_{gi}, \hat{p}_{gi})$
    end for
    $\mathcal{P}^{\star}_g \leftarrow \text{top-}k(\mathcal{P}_g)$ ▷ elitism: keep the best k by fitness p
    $\mathcal{P}_{g+1} \leftarrow \hat{\mathcal{N}}_g \cup \mathcal{P}^{\star}_g$ ▷ update population
  end for
  return θ with the highest p in $\mathcal{P}_n$
end procedure

In this algorithm, it is evident that each individual does not depend on the other individ-

uals, so the entire EPBT process is easily parallelizable. This algorithm uses standard black-box evolutionary optimization operators [23] to fine-tune the individuals.

The EPBT algorithm first performs the initialization step, creating a population of P individuals, $\mathcal{P}_0 = \{P_1, P_2, \ldots, P_P\}$. For each P_i, a fixed DNN architecture is established, and its weights θ_i are randomly initialized, each hyperparameter variable h_i is uniformly sampled within a fixed range, and the fitness p_i is set to 0. Then, using the tournament selection operator τ, it repeatedly selects individuals at random from $\mathcal{P}_g$. Each of these individuals is compared, and the one with the highest fitness is added to $\hat{\mathcal{P}}_g$. This process is repeated several times until $|\hat{\mathcal{P}}_g| = |\mathcal{P}_g| - k$, where k is the number of elites. The next step is mutation and crossover for each $\hat{P}_{gi} \in \hat{\mathcal{P}}_g$, where a uniform mutation operator γ is applied to introduce multiplicative Gaussian noise independently to each variable of h_{gi}. The mutation operator can randomly and independently restart each variable. This approach enables exploration of new hyperparameter combinations. Once a mutation is performed, a uniform crossover operator ξ is applied, in which each variable of h_{gi} is randomly exchanged (with a 50% probability) with the same variable of another individual in $\hat{\mathcal{P}}_g$, resulting in the creation of $\hat{h}_{gi}$. The weights θ_{gi} are copied from $\hat{P}_{gi}$ and combined with $\hat{h}_{gi}$ to form the unevaluated individual N_{gi}. Subsequently, the evaluation process results in the evaluated individuals $\hat{N}_{gi}$. After evaluation, EPBT uses an elitism heuristic to preserve progress. In elitism without novelty selection, the population is sorted by fitness p_{gi}, and the individuals with better performance $\mathcal{P}_g^\star$ are kept and combined with $\hat{\mathcal{N}}_g$ to form $\mathcal{P}_{g+1}$, the population for the next generation [22].

The loss functions are represented by leveraging the TaylorGLO parameterization, which is defined as a fixed set of continuous values [24], in contrast to the original tree-based GLO parameterization [22].

To analyze the effectiveness of EPBT, the author applied the algorithm to optimize the loss function of two popular image classification datasets, CIFAR-10 and SVHN. CIFAR-10 [25] is a widely used image classification dataset consisting of 60,000 natural images classified into ten classes, in which 50,000 images are used for training and 10,000 for testing. To evaluate the individuals in EPBT, the training set was divided into a separate validation set of 1,250 images and

a training set of 48,750 images. EPBT converges quickly to maximum accuracy on the tests and outperforms all baselines.

SVHN [26] is a dataset composed of about 600,000 training images and 26,000 test images. EPBT achieves higher test accuracy, learns faster, and converges to a high test accuracy. The baseline begins to overfit and lose accuracy toward the end of training, whereas the mechanisms of EPBT prevent this and maintain performance.

Table 1 summarises final test accuracy averaged over five runs for EPBT against two baselines: standard training (no PBT) and conventional Population-Based Training. Column headers denote the dataset and the architecture used: ResNet-32 is the standard 32-layer ResNet, and WRN-*d*-*k* denotes a Wide ResNet of depth *d* and widening factor *k* as introduced by Zagoruyko and Komodakis [27]. EPBT achieves the highest test accuracy across all six configurations (boldface), with absolute improvements ranging from +0.13% (CIFAR-10 / WRN-16-8) to +1.26% (CIFAR-10 / ResNet-32, against the PBT baseline). Although these margins appear small in absolute terms, the standard deviations reported in parentheses (mostly $\leq 0.2\%$) indicate that the gains are statistically meaningful: in every configuration the EPBT mean lies more than one standard deviation above the corresponding baseline mean.

Table 1

Mean and standard deviation (over five runs) of final test accuracies on the CIFAR-10 and SVHN datasets. EPBT achieves better results (bold) compared to the baselines. Results are reported in percentage.

Algorithm	CIFAR-10, ResNet-32	CIFAR-10, WRN-10-12	CIFAR-10, WRN-16-8	CIFAR-10, WRN-22-6	CIFAR-10, WRN-28-5	SVHN, ResNet-32
Baseline 1 (no PBT)	92.42 (0.21)	94.18 (0.09)	95.66 (0.13)	95.87 (0.18)	95.95 (0.07)	97.81 (0.04)
Baseline 2 (PBT)	91.53 (0.56)	–	–	–	–	–
EPBT	**92.79 (0.15)**	**94.38 (0.14)**	**95.79 (0.08)**	**96.05 (0.09)**	**96.02 (0.12)**	**98.08 (0.08)**

The results show that EPBT achieves the best results for multiple datasets and model architectures. Among its advantages, EPBT enables models to converge significantly faster than non-population-based methods, especially with a limited number of training epochs. This is because EPBT simultaneously trains multiple models, each with different loss functions. If one of the models makes progress, its loss function or model weights are distributed to the other models,

thereby improving their performance [22].

Regarding the number of required training epochs, EPBT surpasses the fully trained performance of the baselines, demonstrating its ability not only to train the best models but also to do so faster [22]. In conclusion, EPBT is an evolutionary algorithm for regularization meta-learning that improves upon PBT through more advanced genetic operators and the evolution of TaylorGLO loss functions, and it avoids overfitting through population-based distillation [22].

1.4.3 EfficientNet

EfficientNet is a family of convolutional neural network architectures that addresses a fundamental limitation of conventional Convolutional Neural Networks (ConvNets) scaling by introducing a compound coefficient that simultaneously adjusts depth, width, and input resolution. Rather than scaling one dimension independently, the standard practice, this approach coordinates all three under a single principled constraint, allowing a base architecture to be systematically adapted to different computational budgets without degrading efficiency [28].

A ConvNet layer F_i can be defined as a function $Y_i = F_i(X_i)$, where F_i is the operator, Y_i s the output tensor, and X_i is the input tensor, of the form (H_i, W_i, C_i), where H y W are the spatial dimensions and C_i is the channel dimension. A ConvNet N can be represented by a list of composed layers.

$$N = F_k \circ \cdots \circ F_2 \circ F_1(X_1) = \circ_{j=1..k} F_j(X_1). \tag{1.5}$$

In practice, ConvNet layers are typically divided into several stages, and all layers within each stage share the same architecture [28]. For example, ResNet [18] has five stages, and all layers within each stage use the same convolutional type, except for the first layer, which performs downsampling. Therefore, each ConvNet can be defined as:

$$N = \bigcirc_{i=1..s} F_i^{L_i}(X_{(H_i,W_i,C_i)}) \tag{1.6}$$

where $F_i^{L_i}$ denotes the layer F_i repeated L_i times in stage i, (H_i, W_i, C_i) denotes the shape of the input tensor X for layer i [28].

Unlike regular ConvNet designs that focus on finding the best layer architecture F_i, the scaling model aims to increase the network length (L_i), the width of C_i, and/or the resolution (H_i, W_i) without altering the predefined $F_{(i)}$ in the base network. By fixing F_i, the scaling model simplifies the design problem for new resource constraints, though it still remains a large design space to explore different L_i, C_i, H_i, and W_i for each layer. To further reduce the design space, we constrain all layers to scale uniformly with a constant ratio. The goal is to maximize the model's accuracy for any resource constraint, which can be formulated as an optimization problem [28]:

$$\begin{aligned} &\max_{d,w,r} \quad Accuracy\left(N(d,w,r)\right) \\ &\text{s.t.} \quad N(d,w,r) = \bigcirc_{i=1..s} F_i^{d\cdot L_i}\left(X_{(r\cdot H_i, r\cdot W_i, w\cdot C_i)}\right) \\ &\qquad \text{Memory}(N) \leq \text{destination memory} \\ &\qquad \text{FLOPS}(N) \leq \text{objectives flops} \end{aligned} \tag{1.7}$$

Where w, d, and r are the coefficients for scaling the width, depth, and resolution of the network, respectively. These are predefined parameters in the reference network.

The difficulty of this problem lies in the fact that each of the optimal values of d, w, and r depends on the others, and these values change under different resource constraints. Due to this difficulty, prior to EfficientNet, the literature explored three independent scaling strategies, each with characteristic trade-offs. Increasing network depth allows the model to learn more abstract representations, but beyond a certain point the marginal accuracy gains diminish, ResNet-1000 and ResNet-101 achieve nearly identical performance on standard benchmarks despite the former being far deeper and training stability degrades due to vanishing gradients, even when mitigated by residual connections and batch normalization [28], [18], [29], [30], [31], [32]. Widening the network [27], [28], [33], by contrast, facilitates the capture of fine-grained features and generally improves training stability [27], but shallow-and-wide architectures tend to underfit on complex tasks that require hierarchical abstraction [28]. Higher input resolution offers a third avenue for

improvement by preserving spatial detail that would otherwise be lost at lower resolution; however, this benefit also saturates, with accuracy gains diminishing substantially beyond resolutions already adopted in modern pipelines [28], [31], [34]. Critically, all three dimensions are interdependent: higher resolution demands greater depth to integrate the additional spatial information and greater width to process the increased number of features per layer. Optimizing any single dimension in isolation thus fails to exploit the complementary gains available from coordinated scaling [35], [36], [37].

EfficientNet proposes a compound scaling method, which uses a compound coefficient ϕ to uniformly scale the width, depth, and resolution of a neural network in a way based on the following principles:

$$\begin{aligned} \text{depth:} \quad & d = \alpha^{\phi} \\ \text{width:} \quad & w = \beta^{\phi} \\ \text{resolution:} \quad & r = \gamma^{\phi} \\ \text{s.t.} \quad & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \quad \beta \geq 1, \quad \gamma \geq 1 \end{aligned} \tag{1.8}$$

Where α, β, and γ are constants that can be determined through a small grid search. Intuitively, ϕ is a user-specified coefficient that controls how many additional resources are available to scale the model, while α, β, and γ determine how those additional resources are allocated to depth, width, and resolution of the network, respectively.

The base network EfficientNet-B0 was identified through a multi-objective neural architecture search conducted over the same design space used by MnasNet [38], with a key distinction in the optimization criterion: whereas MnasNet targets inference latency as its efficiency proxy, EfficientNet uses floating-point operations (FLOPs) as the efficiency objective. This shift in criterion yields a baseline architecture that is comparably structured to MnasNet but substantially more

compact, operating at approximately 400M FLOPs [28].

Based on this idea, EfficientNet-B0 is used, and the compound scaling method is applied as follows:

- **Step 1:** First, $\phi = 1$ is fixed, assuming a fixed amount of available resources, and a small grid search is performed over α, β, and γ based on equations (1.8). The best values found for EfficientNet-B0 are $\alpha = 1.2$, $\beta = 1.1$, and $\gamma = 1.15$, considering the constraint $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$.
- **Step 2:** Having fixed α, β, and γ as constants, the base network is expanded using different values of ϕ with equation (1.8), thus obtaining EfficientNet-B1 through B7.

Table 2
EfficientNet Performance Results on ImageNet.

Model	**Accuracy**		**Parameters**		**FLOPs**	
	Top-1	**Top-5**	**#**	**Ratio**	**#**	**Ratio**
EfficientNet-B0	**77.1%**	**93.3%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 [18]	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 [29]	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**79.1%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 [18]	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 [29]	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 [31]	78.8%	94.4%	24M	3.0x	5.7B	8.1x

Continued on the next page

Model	Accuracy		Parameters		FLOPs	
	Top-1	**Top-5**	**#**	**Ratio**	**#**	**Ratio**
Xception [39]	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**80.1%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 [40]	80.0%	95.0%	48M	5.2x	13B	13x
Inception-ResNet-v2 [40]	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.6%**	**95.7%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 [41]	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet [42]	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.9%**	**96.4%**	**19M**	**1x**	**4.2B**	**1x**
SENet [43]	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A [34]	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A [44]	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet [45]	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.6%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**

Continued on the next page

Model	**Accuracy**		**Parameters**		**FLOPs**	
	Top-1	**Top-5**	**#**	**Ratio**	**#**	**Ratio**
AmoebaNet-C [46]	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.8%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.3%**	**97.0%**	**66M**	**1x**	**37B**	**1x**
GPipe [35]	84.3%	97.0%	557M	8.4x	–	–

Note: Tan, M., and Le, Q. V. (2019). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. arXiv preprint arXiv:1905.11946.

Table 2 compares EfficientNet (B0 - B7) with representative convolutional baselines on the ImageNet classification benchmark. Top-1 accuracy reflects the proportion of test images whose highest-probability class matches the ground truth, whereas Top-5 accuracy allows the correct class to lie anywhere among the five most probable predictions; the latter is the standard reporting convention for ImageNet given its 1,000 fine-grained classes. The columns labelled "Ratio" express the parameter count and FLOPs of each baseline relative to the EfficientNet variant in the same accuracy band: a value of, for example, 7.6× indicates that the baseline requires 7.6 times the parameters of the corresponding EfficientNet model to reach a comparable Top-1 score. Two patterns emerge, first, every EfficientNet variant either matches or exceeds its non-EfficientNet counterpart at a fraction of the parameters and FLOPs, second, the parameter ratio widens monotonically with depth: EfficientNet-B7 reaches 84.3% Top-1 accuracy with 66M parameters, matching GPipe's 84.3% with 557M parameters (8.4× ratio). This widening ratio is the empirical justification for the compound scaling strategy formalised in Equation 1.8.

This efficiency is achieved through a combination of better architecture design, improved

compound scaling, and a training pipeline specifically adapted to EfficientNet.

The inference efficiency advantages of EfficientNet are equally pronounced: measured on real hardware, EfficientNet-B1 reduces inference time to approximately one-sixth that of ResNet-152, and EfficientNet-B7 achieves a speedup of over six times relative to GPipe [35] while matching its top-1 accuracy. These results confirm that the compound scaling strategy produces gains not only in the theoretical FLOPs metric but also in practical deployment throughput.

1.4.4 DiagnoseNET

DiagnoseNET [47] integrates the deep learning lifecycle via automated orchestration, enabling efficient model scaling from local workstations to distributed clusters. At its core, the framework provides a unified programming interface — a single-expression API — through which users can define network architectures, configure hyperparameter search strategies, and specify data partitioning and batching policies without directly managing the underlying distribution logic. The execution environment translates these high-level specifications into coordinated training jobs across hardware ranging from single GPU workstations to multi-node clusters, supporting both synchronous and asynchronous gradient aggregation via gRPC and MPI communication protocols, and spanning x86 and ARM processor architectures [47].

DiagnoseNet is oriented towards the design of an eco-intelligent medical workflow that allows for the implementation of medical diagnostic tools with minimal infrastructure requirements and low energy consumption. The first application built automated the unsupervised patient phenotypic representation workflow and was trained on an NVIDIA Jetson TX2 minicluster [47].

Figure 2 shows the schematic integration of the DiagnoseNET modules and their functionalities.

The first module is the deep learning model graph generator, which has two expression languages: the Sequential Graph API, designed to automate hyperparameter search, and a Custom Graph that supports TensorFlow expression code for sophisticated neural networks. To optimize

computational resources, population-based training was implemented in the DiagnoseNet Framework.

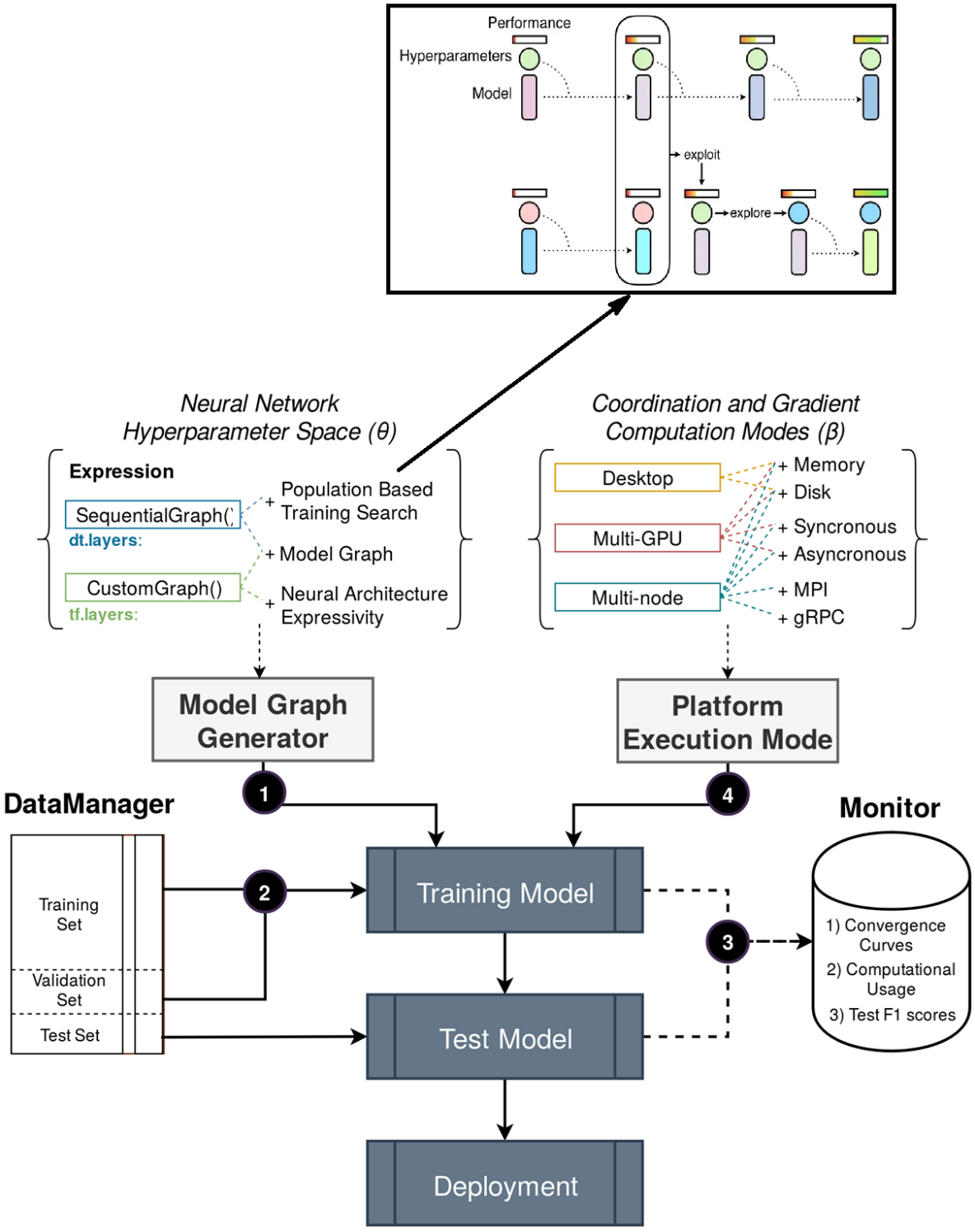
DiagnoseNET integrates PBT [21] as its hyperparameter optimization engine. In this configuration, candidate models are initialized with diverse random hyperparameter configurations and trained in parallel. At periodic checkpoints, each model's validation performance is evaluated asynchronously; models that lag behind the population median are replaced by copies of the current best-performing member — including its weights and hyperparameter state — which then undergo a perturbation step to encourage exploration of adjacent configurations. This adaptive redistribution of compute toward the most promising regions of the search space enables DiagnoseNET to discover well-performing configurations substantially faster than sequential or grid-based tuning, while using no additional hardware beyond what would be required for standard parallel training [47].

The second module is the data manager, composed of three classes for splitting, batching, and multi-tasking datasets across GPU workstations and multi-node computational platforms. The third module extends the enerGyPU monitor for workload characterization, consisting of a data capture in runtime to collect the convergence tracking logs and the computing factor metrics; and a dashboard for the experimental analysis results [48]]. The fourth module is the runtime that enables platform selection from GPU workstations to multi-node systems with different execution modes, such as synchronous and asynchronous gradient computations with gRPC or MPI communication protocols [47].

1.4.5 Energy Efficiency and Environmental Sustainability in Deep Learning

Energy Efficiency and Environmental Sustainability in Deep Learning as is well known, Deep Learning has revolutionized different fields of artificial intelligence, computer vision, natural language processing, among others, demonstrating enormous capability in very complex tasks. However, this computational revolution comes at a high environmental cost. Large-scale Deep Learning models require vast amounts of energy, which consequently increases carbon dioxide

Figure 2
DiagnoseNET framework scheme [47].



emissions throughout their life cycle, including both training and inference during production [49] [50]. This has led to environmental and economic challenges that require solutions to improve the energy efficiency of models without sacrificing accuracy or performance.

Due to the complexity and computational demands of Deep Learning models, there is high energy consumption, making energy efficiency crucial for their deployment on resource-constrained devices such as portable devices or embedded systems. Reducing energy expenditure prolongs lifespan and improves sustainability while promoting the Green AI concept, which emphasizes environmentally responsible practices.

.Energy consumption in model training Training Deep Learning models consumes exorbitant amounts of energy. A well-known example is the training of GPT-3 models, which required approximately 1287 megawatt-hours (MWh) of electricity—enough to power about 120 American households for an entire year [51] and produced around 502 tons of CO_2 emissions, equivalent to driving approximately 112 gasoline-powered cars for one year [52].

Subsequent models, such as GPT-4 and GPT-5, have significantly increased their parameter counts, leading to excessive energy consumption. For this reason, studies conducted at the University of Massachusetts Amherst reported that training large deep learning models for natural language processing can generate more than 284 metric tons of CO_2-equivalent emissions, highlighting the environmental impact associated with large-scale artificial intelligence systems [53].

The factors that determine energy consumption during training include the number of computational operations required, which increases with the number of model parameters, dataset size, number of training epochs, and neural network architecture [54]. Therefore, more complex architectures require greater computational power, and Transformer models used in natural language processing are especially demanding.

.Energy consumption in inference This refers to the energy consumed by hardware or system components (CPU, GPU, Memory) to produce predictions from a pre-trained model. This depends

on the average power during service and the processing time per request. This cost is influenced by the batch size (larger batches tend to reduce mJ/inf), the model architecture and its optimizations (compact models, distillation, pruning, fused kernels), numerical precision and format, and the hardware used.

This consumption is frequently underestimated because most energy is consumed during the training of Deep Learning models. However, as the intensive use of DL models becomes widespread, this consumption significantly increases, as in the case of modern generative models, leading inference energy demand to dominate total consumption [51]. Each query to these systems requires the model to process information, multiply matrices, and perform activation operations, consuming energy at every step. It is also important to consider the location or geography of the data center where the deployment occurs because data centers situated in regions with access to renewable energy generate significantly less CO_2 emissions than those in regions reliant on fossil fuels.

.Methodologies and Strategies for Sustainability In pursuit of energy-efficient Deep Learning models, the concept of Green AI has emerged, seeking to develop artificial intelligence systems that prioritize energy efficiency and minimize environmental impact [55]. A series of techniques and strategies have been developed, among which are:

- Architecture optimization: Designing more compact and efficient neural networks can dramatically reduce energy consumption. Techniques such as knowledge distillation, model compression, and neural architecture search (NAS) enable the creation of lighter models that maintain comparable performance [54].

- Infrastructure level: Data centers where training and inference are performed must fundamentally use renewable energy, as this directly impacts CO_2 emissions [53].

- Algorithm level: Optimizations in training processes, adjustments in batch size, learning rates, and regularization strategies help improve energy efficiency without sacrificing perfor-

mance [56].

The energy impact of Deep Learning poses significant challenges. It is estimated that energy demand in data centers will increase by 160% by 2030 due to Large Language Model technologies such as ChatGPT, Gemini, and Grok, among others [55]. The scientific community must adopt a comprehensive approach to Green AI, considering not only model performance and accuracy but also their energy costs and carbon footprints. Efforts should focus on developing more efficient models deployable on clean computational infrastructures, continuously measuring their impact, and sharing findings with the scientific community to truly contribute to a sustainable future. This chapter provided the conceptual and methodological foundations for the research presented in this thesis. Sections 1.1 through 1.3 traced the evolution from knowledge-based systems to machine learning and deep learning, establishing the theoretical background for the architectures and optimization techniques discussed in subsequent chapters. Section 1.4 reviewed the state of the art in hyperparameter optimization and architecture design: Population-Based Training (PBT) and its evolutionary extension EPBT were presented as methods for efficient hyperparameter scheduling through parallel population-based search, while EfficientNet was examined as a reference family of architectures that achieves competitive accuracy through compound scaling of depth, width, and resolution, albeit with significant increases in parameters and computational cost. The DiagnoseNET framework was analyzed for its integration of energy monitoring, data management, and parallel training orchestration in healthcare environments with limited infrastructure. Finally, Section 1.5 addressed the energy efficiency and environmental sustainability of deep learning models, examining the high energy consumption during both training and inference, and reviewing strategies aligned with the Green AI paradigm. It is evident that the growing complexity of current DL models requires new strategies to design more efficient, lightweight, and sustainable architectures without sacrificing performance.

1.5 Synthesis and Consensus of Related Work

The previous sections reviewed, as background, three lines of research that frame the problem addressed in this thesis, although none of them is directly employed in the proposed methodology. In the area of automated model search, Population-Based Training (PBT) [21] and its evolutionary extension (EPBT) [22] introduced the automated optimization of hyperparameters and model weights within a population of candidate solutions. These approaches constitute an important precedent for the automated search paradigm that, when extended to architectural design, leads to Neural Architecture Search (NAS) [57], which is employed in this work. In the area of efficient architecture design, EfficientNet [28] demonstrated that compound scaling can effectively balance predictive performance and computational cost through a principled, manually designed architecture. It represents the handcrafted approach to efficiency that automated search methods seek to systematize and is also used as one of the reference architectures in the experimental evaluation. Finally, in the field of sustainable artificial intelligence for healthcare, DiagnoseNET [47] established an early precedent for incorporating energy monitoring into the lifecycle of clinical machine learning models, anticipating the Green AI dimension that is adopted and quantified in this thesis.

From these antecedents, a consensus emerges: obtaining models that are simultaneously accurate and deployable in resource-constrained environments cannot be achieved with a single technique, but requires articulating three directions: efficient architecture optimization, knowledge transfer, and the explicit consideration of energy cost. Table 3 compares the approaches reviewed in these dimensions, together with the multi-label medical domain and the degree of integration. As the table makes evident, each prior line addresses only a subset of these aspects: PBT/EPBT [21] [22] and EfficientNet [28] focus on model optimization, Hinton's distillation on knowledge transfer [5], and DiagnoseNET [47] on energy monitoring in the clinical domain, while GENIUS [4] introduces LLM-guided architecture search but is confined to a non-medical, multi-class setting. None of them combines architecture search, knowledge transfer, and energy analysis within a single end-to-end process for multi-label medical imaging. This gap —visible in the final row of Table 3—

Table 3
Comparison of related approaches across the key dimensions of the proposed framework. (✓: addressed; ∼: partially addressed; 55: not addressed.)

Approach	Arch. Search	Knowledge Transfer	Energy Metrics	Medical MLC	End-to-End Framework
PBT/EPBT	∼	55	55	55	55
EfficientNet	✓	55	55	55	55
DiagnoseNET	55	55	✓	✓	55
GENIUS	✓	55	55	55	55
KD (Hinton)	55	✓	55	55	55
Proposed	✓	✓	✓	✓	✓

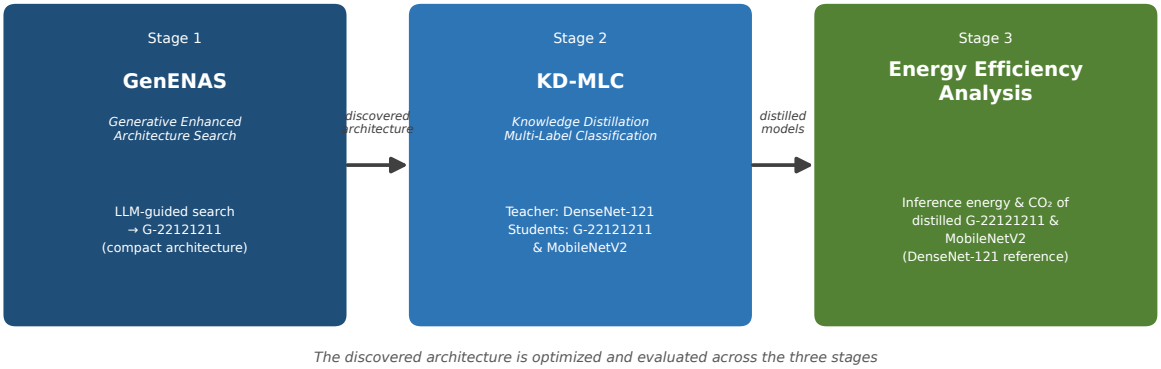
is precisely what this thesis addresses; the core approaches of the proposal are reviewed in detail in the chapters in which they are developed.

1.6 Overview of the Proposed Framework

To address the identified gap, this thesis proposes an integrated, three-stage framework that operates end-to-end on a single model. In the first stage, Generative Enhanced Neural Architectures Search (GenENAS), a large-language-model-guided generative architecture search, discovers a compact architecture adapted to the multi-label medical domain. In the second stage, Knowledge Distillation for MultiLabel Image Classification (KD-MLC) transfers the knowledge of a high-capacity teacher model into that compact architecture, maximizing its diagnostic performance without increasing its size. In the third stage, an energy-efficiency analysis quantifies the energy consumption and CO_2 emissions incurred by the resulting models *during inference* —that is, when a trained model processes new inputs to produce predictions, as opposed to the one-time cost of training—, since inference is the operation that is repeated continuously once the model is deployed and therefore dominates its long-term energy footprint. This analysis assesses the feasibility of the models for deployment in environments with limited computational resources. Figure 3 summarizes this flow. Unlike the antecedents reviewed above, the contribution lies not in each technique in isolation, but in their integration into a single, reproducible pipeline.

Figure 3

Overview of the proposed three-stage framework, chained end-to-end on a single model.



2. GenENAS: Automatic Architecture Discovery for Efficient Deep Learning

This chapter presents the first stage of the framework proposed in this thesis, which consists of automatically discovering a high-performance neural architecture through Generative Enhanced Neural Architecture Search (GenENAS). Based on the methodology introduced in GENIUS [4] and adapted to chest X-ray images from the CheXpert dataset for the specific use case addressed in this work, GenENAS leverages the capabilities of Large Language Models (LLMs), such as GPT-4 and LLaMA, by employing them as black-box optimization engines to efficiently navigate the architectural search space. Through this process, GenENAS identifies promising candidate architectures and iteratively refines them to improve their predictive performance and computational efficiency.

The design of neural network architectures remains one of the most important research areas in Deep Learning, having led to significant advances such as LeNet, AlexNet, VGGNet, GoogLeNet [30], ResNet, DenseNet, and EfficientNet, among others. These developments have demonstrated

that architectural design plays a critical role in determining both predictive performance and computational cost.

The compact architecture discovered during this stage serves as the foundation for the next stage of the proposed framework. In that stage, a substantially more robust teacher model transfers its knowledge to a smaller and lighter student model through a knowledge distillation process. Consequently, the student model is able to achieve performance levels comparable to those of the teacher while maintaining significantly lower computational requirements. The architecture identified by GenENAS serves as this student model and becomes the recipient of the knowledge transferred during the subsequent stage, thereby enabling the integration of automatic architecture discovery and Knowledge Distillation Transfer within a unified optimization pipeline.

The adaptability of deep neural networks has enabled their application in a wide range of tasks, including computer vision, robotics, data analysis, and natural language processing, among others. This has extended their application across various sectors, especially in medicine and industry. With the advent of very large models, an extremely effective path for solving all types of tasks has emerged. The drawback, however, is that these massive models require ever-increasing computational resources to train, which limits their use.

The architecture of a neural network is a configuration of components that organizes and directs the flow of information in a machine learning system. This architecture is inspired by the conceptual understanding of the human brain's functioning, where neurons form connections that transmit signals and enable the integration of information.

These artificial neural networks are composed of processing units divided into a sequence of layers. The input layer receives data, followed by a series of intermediate layers that perform complex calculations, and culminates in the output layer, which generates the final response. The developer is responsible for defining the neural network's structure based on the complexity of the problem, as the number and distribution of processing units directly influence the system's ability to identify patterns and extract relationships in the data.

The arrangement of layers and the number of neurons are directly related to the system's

capacity to represent the inherent complexity of the data; integrating multiple intermediate layers enables the capture of non-linear relationships in the data. The principles of differential calculus and regularization techniques are employed to adjust the parameters, which prevent overfitting and enhance the model's ability to generalize when presented with new data.

To accelerate and improve neural architecture design, NAS (Neural Architecture Search) has emerged as a highly effective tool for automating the process. Its main advantage lies in using search strategies to discover architectures that outperform those created manually [58].

2.1 Neural Architecture Search (NAS)

The Neural Architecture Search (NAS) initially [59] [34] employed reinforcement learning to explore the search space of potential architectures and subsequent approaches used evolutionary strategies [60] and Bayesian optimization [61]. Recent years have focused on reducing the computational burden of search, with proposals such as DARTS [62], which leverages gradient-based search, and EfficientNAS [63], which employs subnet sampling to increase efficiency. GENIUS employs a straightforward process that prompts GPT-4 to propose designs from a given search space with a handful of examples.

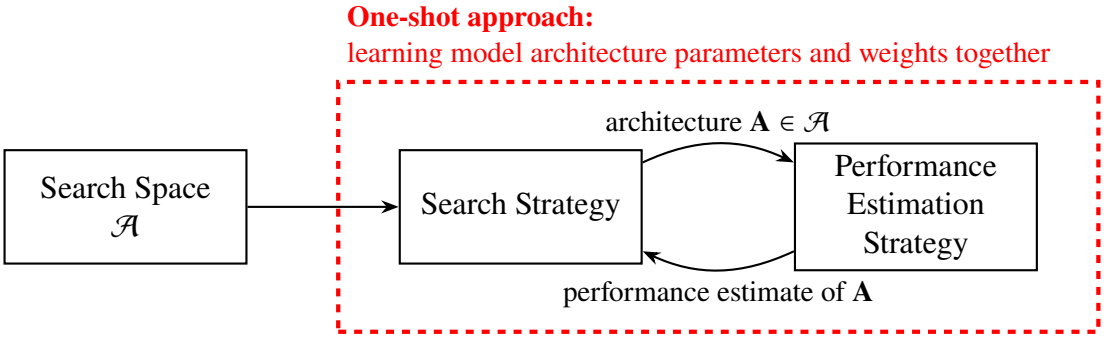
Neural Architecture Search automates the design of deep learning models by systematically exploring a wide range of candidate network configurations to identify the architecture best suited to a given task and dataset — a process that previously required substantial expert intuition and iterative manual experimentation.

The design process and enhance the quality of neural architectures could be accelerated with the Neural Architecture Search (NAS) method was developed. It has become one of the most effective methods for the automated design of neural network architectures.

The NAS pipeline is structured around three interdependent components [57]. The **search space** delimits the universe of architectures that the algorithm can construct, typically comprising a set of operations — convolutions of varying kernel sizes, pooling, skip connections — and rules

Figure 4

Three main components of Neural Architecture Search (NAS) models [57].



governing how those operations may be combined into valid graphs. Human expertise is inevitably embedded in these choices: the decision to include or exclude certain operation types implicitly encodes prior knowledge about what tends to work for the target domain [57]. The **search strategy** determines how the algorithm navigates this space. Early NAS systems relied on computationally intensive approaches such as reinforcement learning, where a controller policy was trained to sequentially select architectural components, or evolutionary algorithms, where populations of architectures were mutated and selected across generations. Gradient-based methods such as DARTS [62] substantially reduced this cost by relaxing the discrete architecture selection into a continuous optimization problem, enabling architecture and weights to be learned jointly [57]. The **performance estimation strategy** assigns a quality score to each candidate architecture so the search strategy can rank them. Full training on the target dataset is the most faithful estimator but prohibitively expensive at scale. In practice, cost-reduction techniques are applied: training on data subsets, early stopping after a small number of epochs, or sharing weights across candidate architectures so that no model is trained entirely from scratch [57].

Neural Architecture Search (NAS) has shown strong results in creating models for many uses, often outperforming those designed by people [57]. It is widely used to build neural networks for object detection and image classification. A well-known example is the EfficientNet family, in which the network's depth, width, and resolution were carefully balanced to achieve the best results [28].

In healthcare, Neural Architecture Search (NAS) can help create models for tasks like tumor detection and cell structure segmentation [64]. NAS also enables the design of networks that run efficiently on medical device hardware, leading to faster and more accurate diagnoses. This could make medical technologies much more effective.

2.2 LLMs as Black-Box Optimizers

The neural network architectures search can be naturally formulated as a black-box optimization problem, where the objective is to maximize a target function whose internal structure is unknown or inaccessible. In this setting, it is only possible to propose an input—in this case, a candidate architecture—and observe the corresponding output, namely its performance after training.

Recent studies have shown that Large Language Models (LLMs) can serve as effective black-box optimizers [65]. Having been trained on extensive corpora that include scientific literature, source code, and web-scale information, these models can operate within a contextual optimization loop. Given a natural-language description of the problem, together with a history of previously evaluated configurations and their corresponding performance scores, the optimizer proposes a new candidate expected to outperform prior solutions. The performance score obtained by each proposal is incorporated into the subsequent prompt, enabling the LLM to iteratively refine its recommendations.

In this research, two different LLMs are employed as optimizers in order to demonstrate that the proposed methodology does not depend on a single model. The first is GPT-4, a large-scale proprietary model that has demonstrated remarkable capabilities across a wide range of language understanding and reasoning tasks. The second is Llama 2 (7B), a considerably lighter open-source model whose inclusion allows us to evaluate whether the ability to generate and optimize architectural proposals can be preserved in accessible and reproducible models, without the cost, availability, and dependency constraints associated with commercial APIs.

2.2.1 GPT-4

GPT-4 is a multimodal large language model developed by OpenAI that represents a great leap in artificial intelligence. Its release in March 2023 marked a new stage in machines' ability to understand and generate human language in a coherent, natural way with a high level of contextual understanding, in addition to enabling text generation from visual information.

The GPT acronym captures the three architectural and methodological pillars of the model family [66]. These systems are generative in that they produce novel text outputs based on patterns learned from the training data. They are pretrained through exposure to large unlabeled text corpora via self-supervised objectives, typically next token prediction, before being adapted to specific tasks through supervised fine-tuning on curated instruction-response pairs. The underlying architecture is the Transformer [19], which processes input sequences by computing attention weights that capture dependencies between tokens regardless of their distance in the sequence, enabling the model to maintain coherent context across long passages and infer which continuation is most probable given the preceding tokens [66].

The GPT lineage originates in a 2018 paper from OpenAI that demonstrated the viability of large-scale unsupervised pre-training followed by task-specific fine-tuning — a paradigm that challenged the then-dominant practice of training NLP models from scratch on labeled datasets for each individual task [66].

The processing power of GPT models increases with the number of parameters, which is why each new GPT model has more parameters than the last. GPT-1 has 117 million parameters, GPT-2 has 1.5 billion, while GPT-3 exceeds 175 billion, and GPT-4 has an estimated 1.76 trillion parameters [67].

2.2.2 LLama 2

Llama 2 is a family of open-source Large Language Models (LLMs) developed and released by Meta in July 2023 [68]. The family includes both pretrained and dialogue-finetuned models at scales of 7, 13, and 70 billion parameters, trained on two trillion tokens collected from publicly available data sources. The authors report that these models outperform most open-source chat models across the evaluated benchmarks and, according to human evaluations of helpfulness and safety, represent a viable alternative to closed-source models [68].

From an architectural perspective, Llama 2 is an autoregressive Transformer that retains most of the design choices introduced in its predecessor, Llama 1, while incorporating a longer context window and Grouped-Query Attention (GQA), a mechanism that shares key and value projections across multiple attention heads to improve inference efficiency.

In this research, the 7-billion-parameter variant (Llama 2-7B) is employed as an architecture optimizer within the GenENAS methodology. The selection of this variant serves a dual purpose. First, as an open-weight model, it can be executed locally without relying on commercial APIs, thereby eliminating per-query costs and availability constraints associated with proprietary models such as GPT-4, while improving the reproducibility of the proposed method.

2.2.3 GPT4-NAS

Researchers are now exploring how Large Language Models like GPT-4 can help automate neural architecture design. By using GPT-4's reasoning and text-generation abilities, they hope to improve the Neural Architecture Search process beyond traditional methods such as evolutionary algorithms, reinforcement learning, or differentiable search [62].

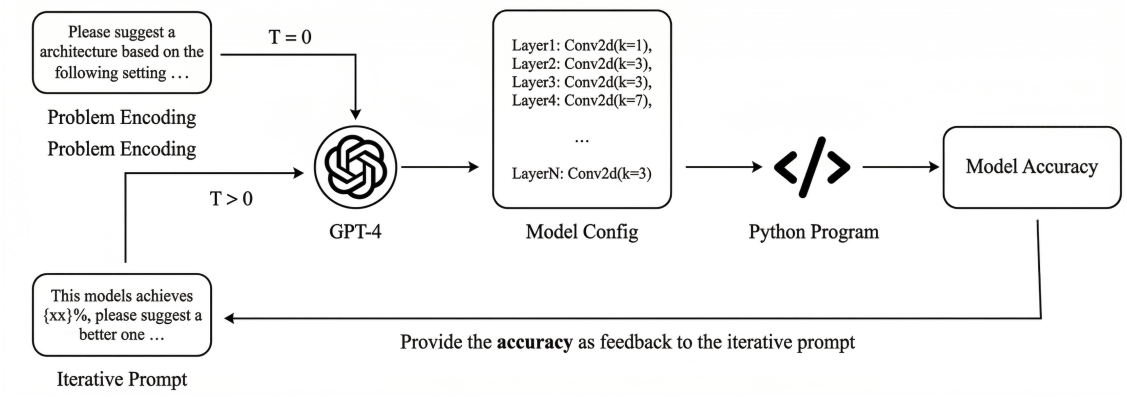
Some studies examine whether LLMs like GPT-4 can explore neural network designs on their own, a task that usually requires experts and substantial computing power. One example is the GENIUS method, where GPT-4 acts as a black-box optimizer. It uses natural language prompts to

suggest, test, and improve network designs. This approach lets users guide the process with simple instructions, even if they are not experts [4].

In the GENIUS framework [4], the NAS problem is encoded as a natural language prompt that conveys the search space constraints and any prior experimental results to GPT-4. After each training run, the measured validation performance is fed back into the next prompt, which asks the model to propose an architecture that improves upon all previously tested configurations. This feedback-driven loop allows the search to concentrate on progressively better regions of the design space without requiring the gradient-based infrastructure of conventional NAS methods.

Figure 5

An overview of the GENIUS Framework [4].



As illustrated in figure 5, the GENIUS loop begins with an initial prompt that asks GPT-4 to propose a network architecture given the search space definition and layer-level constraints. The proposed configuration is compiled, trained on the target dataset, and its validation accuracy is recorded. In every subsequent iteration, this accuracy becomes part of the new prompt: GPT-4 is asked to design an architecture that surpasses all configurations evaluated so far, along with a rationale explaining why the proposed design should outperform its predecessors. The loop terminates when accuracy stagnates across a consecutive sequence of iterations, indicating that the model has converged on its best proposal within the given search space [4].

Below is the algorithm illustrating how the GENIUS framework operates.

GENIUS is executed across multiple experiments using the CIFAR-10 dataset, achieving

Algorithm 3 GPT-4 Enhanced Neural Architecture Search (GENIUS)

Input:
    `GPT-4`: The GPT-4 API.
    `Problem_Encoding`: The human-readable text that encodes the NAS problem.
    `Run`: The training and testing codes for executing and obtaining the ground truth results.
for $T = 0$ to iteration **do**
    if $T == 0$ **then**
        model = GPT-4(Problem_Encoding)
    else
        prompt = "By using this model, we achieved an accuracy of {Accuracy}%. Please recommend a new model that outperforms prior architectures based on the abovementioned experiments. Also, Please provide a rationale explaining why the suggested model surpasses all previous architectures."
        model = GPT-4(prompt)
    end if
    Accuracy = Run(model)
end for
Output: The Best Model Configuration

highly competitive results with an accuracy of approximately 93.8 %, ranking just below optimized NAS methods such as DRNAS and DARTS [4].

2.3 Case Study: Lung diseases

Detecting lung diseases from chest X-rays is a major challenge in modern medicine. Advanced AI methods, such as Neural Architecture Search (NAS), have enabled the development of models that perform as well as, or better than, traditional approaches. More recently, new frameworks powered by large language models, such as GENIUS, have further enhanced the design of neural architectures. The CheXpert dataset has become the primary standard for evaluating algorithms that interpret lung X-rays, thanks to its complexity and clinical significance.

2.3.1 Materials and Methods

Hundreds of X-rays are taken daily around the world, and deep learning has proven to be very effective, providing radiologists with significant help in image diagnosis. However, to achieve this, it is necessary to have this set of information with reliable labels. With this in mind, the CheXpert dataset was created with the participation of certified radiologists, providing the solid database of lung pathologies necessary to train deep learning networks [69].

The CheXpert dataset [70] was assembled retrospectively from chest radiographic examinations performed at Stanford Hospital between October 2002 and July 2017, covering both inpatient and outpatient populations. The dataset contains 224,316 radiographs, of which 234 are part of the validation set and 668 are part of the test set, which are taken from 65,540 patients, each labeled across 14 clinically relevant observations — including atelectasis, cardiomegaly, edema, consolidation, and pleural effusion — with three possible label states: positive, negative, or uncertain. The explicit representation of uncertainty is a distinguishing design choice of CheXpert, motivated by the inherent ambiguity in radiological interpretation and handled through an automated rule-based labeler applied to free-text radiology reports [70].

CheXpert was created by a team of Stanford University researchers from the Departments of Computer Science, Medicine, and Radiology. [70]. Chest radiography is one of the most common diagnostic tests worldwide, used as a basis for diagnosing and treating numerous life-threatening diseases.

Irvin et al. [70] argued that automated radiograph interpretation could benefit a broad range of clinical contexts — from triage support in high-volume radiology departments to population-level screening programs in settings with limited specialist access. Constructing a dataset adequate for training and validating such systems required not only large-scale and public availability, but also a credible ground truth mechanism and radiologist-level performance benchmarks to enable meaningful human-AI comparison [70] [71].

The Figure 6 shows the label distribution for each pathology for the training set. The

positive label is shown in blue, the absent label in orange, the uncertain label in green, and the null or blank label in red. As can be seen, there are more blank labels for each pathology.

Figure 6

Label distribution in the CheXpert training dataset.

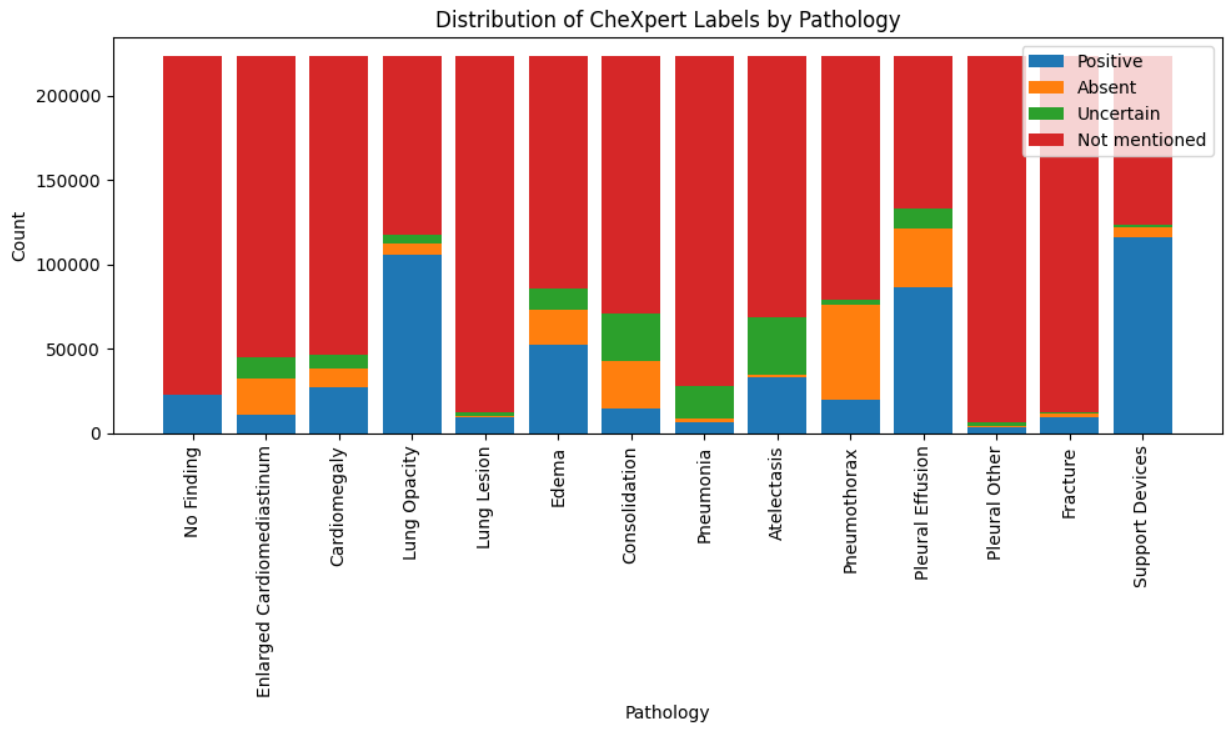


Table 4 presents the label distribution of the CheXpert training set across the 14 thoracic observations annotated in the dataset. For each pathology, the table reports the number and percentage of cases in each of the four possible label states: *Positive* (pathology present), *Absent* (pathology not present), *Uncertain* (radiologist not fully confident about the finding), and *Not Mentioned* (the pathology was not referenced in the original radiology report). The total number of records is 223,414. The distribution is markedly uneven across pathologies: some findings such as Support Devices and Lung Opacity appear as positive in nearly half of the studies, while others such as Pleural Other or Lung Lesion are positive in less than 5% of the cases. A substantial proportion of entries also fall into the Not Mentioned category, reflecting the natural variability of free-text radiology reports.

Table 4
Label distribution in the CheXpert training dataset

Pathology	Positive	Absent	Uncertain	Not Mentioned
No Finding	22,381 (10.0%)	0 (0.0%)	0 (0.0%)	201,033 (90.0%)
Enlarged Cardiomediastinum	10,798 (4.8%)	21,638 (9.7%)	12,403 (5.6%)	178,575 (79.9%)
Cardiomegaly	27,000 (12.1%)	11,116 (5.0%)	8,087 (3.6%)	177,211 (79.3%)
Lung Opacity	105,581 (47.3%)	6,599 (3.0%)	5,598 (2.5%)	105,636 (47.3%)
Lung Lesion	9,186 (4.1%)	1,270 (0.6%)	1,488 (0.7%)	211,470 (94.7%)
Edema	52,246 (23.4%)	20,726 (9.3%)	12,984 (5.8%)	137,458 (61.5%)
Consolidation	14,783 (6.6%)	28,097 (12.6%)	27,742 (12.4%)	152,792 (68.4%)
Pneumonia	6,039 (2.7%)	2,799 (1.3%)	18,770 (8.4%)	195,806 (87.6%)
Atelectasis	33,376 (14.9%)	1,328 (0.6%)	33,739 (15.1%)	154,971 (69.4%)
Pneumothorax	19,448 (8.7%)	56,341 (25.2%)	3,145 (1.4%)	144,480 (64.7%)
Pleural Effusion	86,187 (38.6%)	35,396 (15.8%)	11,628 (5.2%)	90,203 (40.4%)
Pleural Other	3,523 (1.6%)	316 (0.1%)	2,653 (1.2%)	216,922 (97.1%)
Fracture	9,040 (4.0%)	2,512 (1.1%)	642 (0.3%)	211,220 (94.5%)
Support Devices	116,001 (51.9%)	6,137 (2.7%)	1,079 (0.5%)	100,197 (44.8%)

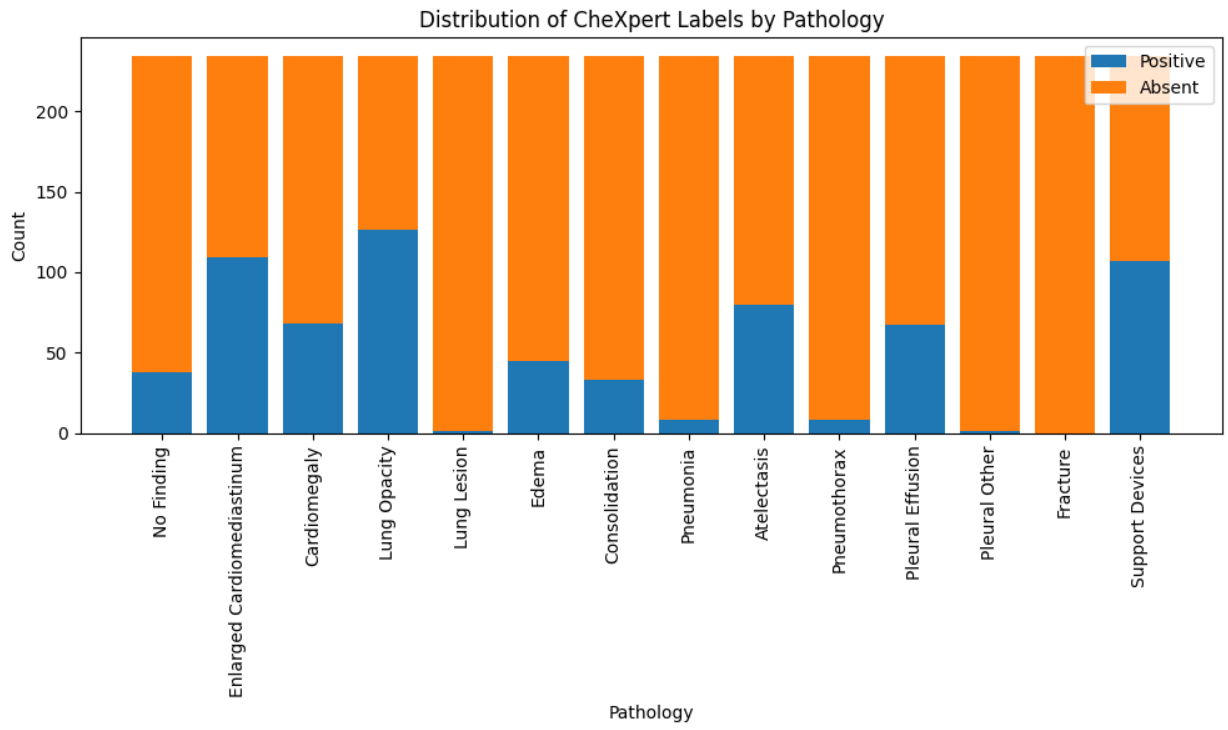
Table 5 reports the label distribution of the held-out validation set, which comprises 234 radiographic studies manually annotated by consensus among several specialists, ensuring high-quality, precise labeling. Unlike the training set, the validation labels admit only two states (positive and absent) because the consensus protocol resolved every ambiguity. Two observations are pertinent for the experimental design. First, Fracture has zero positive cases in this set and is therefore not evaluable; Pneumonia, Pneumothorax and Pleural Other each have eight or fewer positives, making per-pathology AUC-ROC estimates unreliable for these classes. Second, the five pathologies retained for the case study — Atelectasis, Cardiomegaly, Consolidation, Edema, and Pleural Effusion are precisely those whose positive count exceeds 30, allowing for stable AUC-ROC estimation, and they coincide with the panel used by the Stanford CheXpert competition. This choice prioritises evaluation reliability over coverage of the full 14-pathology taxonomy.

The distribution of labels in the validation dataset for each pathology is shown in Figure 7.

For the case study, the following five pathologies were used: Atelectasis, Cardiomegaly,

Figure 7

Label distribution in the CheXpert valid dataset.



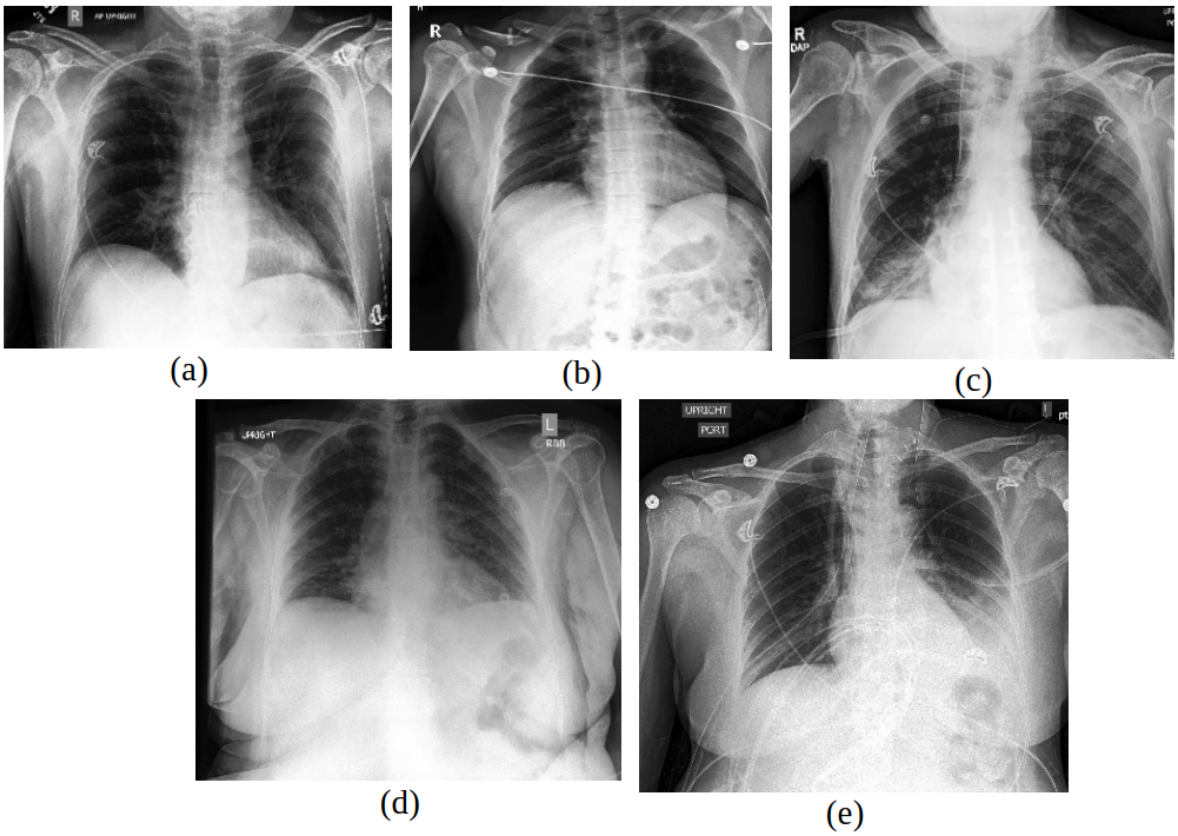
Consolidation, Edema, and Pleural Effusion. These five were selected because they are the most frequent and relevant pathologies in thoracic radiological diagnosis. Furthermore, these are the pathologies used in the competition organized by the Stanford Machine Learning Group, where model performance is primarily measured by the AUC-ROC (Area Under the ROC Curve). This competition seeks to drive the development of new artificial intelligence techniques in healthcare.

The figure 8 shows the images corresponding to each of the pathologies that were considered for this study.

Table 5
Label distribution in the CheXpert valid dataset.

Pathology	Positive	Absent
No Finding	38 (16.2%)	196 (83.8%)
Enlarged Cardiomediastinum	109 (46.6%)	125 (53.4%)
Cardiomegaly	68 (29.1%)	166 (70.9%)
Lung Opacity	126 (53.8%)	108 (46.2%)
Lung Lesion	1 (0.4%)	233 (99.6%)
Edema	45 (19.2%)	189 (80.8%)
Consolidation	33 (14.1%)	201 (85.9%)
Pneumonia	8 (3.4%)	226 (96.6%)
Atelectasis	80 (34.2%)	154 (65.8%)
Pneumothorax	8 (3.4%)	226 (96.6%)
Pleural Effusion	67 (28.6%)	167 (71.4%)
Pleural Other	1 (0.4%)	233 (99.6%)
Fracture	0 (0.0%)	234 (100.0%)
Support Devices	107 (45.7%)	127 (54.3%)

Figure 8
X-rays for each of the selected pathologies. (a) Atelectasis, (b) Cardiomegaly, (c)Consolidation, (d) Edema, (e) Pleural Effusion



.GenENAS Methodology Following the methodology employed by GENIUS, the robustness and efficacy of the automated search for optimized architectures for multi-label diagnosis of pulmonary pathologies were analyzed using the public CheXpert dataset. This led to the creation of the GenENAS methodology, which allows finding optimal architectures for the CheXpert dataset through the use of large language models (LLM). CheXpert dataset is multi-label, meaning that an output can contain one or more pathologies. Furthermore, it is a rather complex dataset due to an imbalance in its classes.

Search Space: The architecture search is restricted to a macro search space composed of eight configurable layers, each of which can assume one of three candidate operations. This results in a total of ($3^8 = 6,561$) possible architectures. Each candidate is encoded as a sequence of eight integers (e.g., ([2,2,1,2,1,2,1,1])), which the LLM interprets and generates in natural language. This compact encoding enables the model to receive, entirely through textual prompts, both the constraints of the search space and any previously acquired experimental evidence. Consequently, the architecture search process can be conducted without the gradient-based infrastructure or controller mechanisms typically required by conventional NAS methods.

Adaptation to the Multi-Label Medical Domain: Two methodological modifications distinguish GenENAS from its predecessor, GENIUS. First, the standard Softmax output layer is replaced by independent Sigmoid functions, one for each of the five target pathologies (atelectasis, cardiomegaly, consolidation, edema, and pleural effusion). In this way, each output produces an independent probability in the interval ([0,1]), reflecting the clinical reality in which multiple conditions may coexist within the same patient. Second, the optimization metric guiding the search is changed from accuracy—appropriate for a balanced dataset such as CIFAR-10, but potentially misleading under the severe class imbalance present in CheXpert—to the Area Under the Receiver Operating Characteristic Curve (AUC-ROC), which is robust to class imbalance and evaluates discriminative performance across all decision thresholds. Consequently, it is the AUC-ROC achieved by each candidate architecture, rather than its accuracy, that is fed back to the LLM to guide subsequent architectural proposals.

Table 6 defines the macro structure of the search space used in the GenENAS experiments. The column n indicates how many times each layer or block is stacked consecutively within that stage. The input column specifies the spatial and channel dimensions of the tensor entering each stage, expressed as height × width × channels; the progressive reduction in spatial resolution across stages reflects the standard downsampling pattern of efficient convolutional networks. The block column identifies the type of operation applied: the first and last convolutional stages use fixed operations (3×3 conv and 1×1 conv respectively), while the three intermediate stages use a Choice block, meaning that GenENAS selects one of three candidate operations. Select digit O for Identity connection, digit 1 for MobileNetV2 block with kernel size 3 and expansion ratio 3 (MB3_K3), or digit 2 for MobileNetV2 block with kernel size 5 and expansion ratio 6 (MB6_K5) for each of the 8 searchable layers. The channel column reports the number of output feature maps produced by each stage, increasing from 32 at the input to 1280 before the classification head. Finally, the stride column controls whether the spatial resolution is preserved (stride = 1) or halved (stride = 2); all three Choice block stages apply stride 2, producing the successive spatial downsampling from 32×32 to 4×4. Since only the Choice block layers are subject to search and each has 3 options, the total search space contains $3^8 = 6,561$ distinct candidate architectures. In the GenENAS methodology adapted for the CheXpert dataset, the final layer comprises a fully connected (FC) layer followed by a Sigmoid activation function, replacing the Softmax function traditionally employed in classification tasks. This architectural decision is necessitated by the multi-label nature of the CheXpert chest X-ray dataset, where a single radiographic image may simultaneously manifest multiple pathologies.

Unlike Softmax, which normalizes output probabilities to sum to 1—thereby forcing the model to select a single mutually exclusive class—the Sigmoid function processes each of the five outputs independently, assigning an autonomous probability between 0 and 1 to each target. Consequently, this approach enables the model to predict the presence or absence of several clinical conditions concurrently. This more accurately reflects the clinical reality of the problem, in which a patient may be diagnosed with co-occurring pathologies, such as cardiomegaly, pleural effusion,

and atelectasis, among others.

Table 6
Macro structure of search space.

n	input	block	channel	stride
1	320 x 320 x 3	3 x 3 conv	32	1
2	320 x 320 x 32	Choice block	64	2
3	160 x 160 x 64	Choice block	128	2
3	80 x 80 x 128	Choice block	256	2
1	40 x 40 x 256	1 x 1 conv	1280	1
1	40 x 40 x 512	Global avgpool	-	-
1	512	FC + Sigmoid	5	-

To assess the robustness of each architecture, AUC-ROC and Precision-Recall plots were used. The Area Under the Receiver Operating Characteristic Curve (AUC-ROC) is a performance measurement for classification problems in machine learning, and it is especially important in medical applications for several reasons. AUC-ROC is the primary performance metric in this study for two reasons specific to the clinical context. First, medical classification datasets are inherently imbalanced — pathologies such as consolidation are far less prevalent than normal findings — and accuracy inflates artificially when the majority class dominates predictions. The AUC-ROC, by measuring the model's ability to rank positive cases above negative ones across all possible decision thresholds, is insensitive to class imbalance and therefore provides a more reliable comparison across architectures [72]. Second, in clinical practice the optimal operating point on the sensitivity-specificity curve varies by clinical context: a screening tool tolerates more false positives than a confirmatory diagnostic. The ROC curve preserves this information in its entirety, allowing the threshold to be selected post hoc according to the deployment scenario, rather than committing to a fixed threshold during model development.

Precision-Recall (PR) plots are used to assess the performance of classification models, especially when the classes are imbalanced. Remember, in the context of PR curves, precision is defined as the number of true positives divided by the sum of true positives and false positives. **Recall** (also known as Sensitivity or True Positive Rate) is the number of true positives divided by

the sum of true positives and false negatives. These two metrics provide a comprehensive view of a model's performance in terms of its false-positive and false-negative rates.

2.3.2 Results

The validation of the model using a dataset of 234 radiographic studies covering five selected pathologies—atelectasis, cardiomegaly, consolidation, edema, and pleural effusion—and using the GPT-4 API as the large language model (LLM).

The AUC-ROC score is obtained for each pathology, and the average of all scores is calculated. After each iteration, the AUC-ROC score is verified, and LLM (in this case GPT-4) is asked which structure it recommends based on the results. It responds with a new configuration, which is then trained, and the results are checked again. In this case, the algorithm performs the same process over seven iterations. After the third iteration, where it achieved the highest AUC-ROC score of 0.8687, the algorithm executed four additional iterations, because patience was set to a value of 4, so the algorithm, upon detecting that it is not improving the average AUC-ROC score, stops execution and selects the architecture that obtained the best AUC-ROC. This architecture corresponds to GenENAS-22121211, which was recommended by the LLM and trained during the third iteration, but due to the Early stopping set at a value of 4, the algorithm confirmed it as the best architecture found until iteration 7. Specifically, the AUC-ROC score (i.e., the area under the ROC curve) for DL is maximized. Maximizing the AUC-ROC score has several advantages over minimizing the cross-entropy loss. First, in medical classification tasks, the AUC-ROC score is the default metric for assessing and evaluating different methods. GPT-4 is asked to propose new deep neural network architectures, where each candidate architecture proposed is identified by an 8-digit string that encodes the operation assigned to each of the 8 searchable Choice block layers, ordered from the shallowest to the deepest stage of the network. The digit 0 represents an Identity connection, the digit 1 represents a MobileNetV2 block with kernel size 3 and expansion ratio 3 (MB3_K3) and the digit 2 a MobileNetV2 block with kernel size 5 and expansion ratio 6

(MB6_K5). This compact notation allows any architecture in the 3^8 search space to be uniquely described and reproduced.

Table 7 reports the AUC-ROC scores obtained for each of the seven architectures evaluated across the five target pathologies, the per-pathology AUC-ROC ordering is stable: Pleural Effusion is the easiest target (AUC-ROC $\in [0.90, 0.92]$), followed by Edema and Consolidation, while Cardiomegaly is the most difficult (AUC-ROC $\in [0.77, 0.81]$). For instance, architecture GenENAS-22121211 applies MB6_K5 in positions 1, 2, 4, and 6, MB3_K3 in positions 3, 5, 7, and 8.

Table 7
Experiment results. AUC-ROC score values obtained.

Neural Network	**Pathology AUC-ROC Score**					
Architecture	**Atelectasis**	**Cardiomegaly**	**Consolidation**	**Edema**	**Pleural Effusion**	**Average**
11212122	0.8205	0.7945	0.8872	0.8943	0.9199	0.8633
21212121	0.8239	0.7968	0.8936	0.9044	0.8932	0.8623
22121211	**0.8368**	**0.7833**	**0.9020**	**0.9198**	**0.9017**	**0.8687**
21121211	0.8130	0.7740	0.8908	0.9071	0.9069	0.8584
12112122	0.8174	0.8074	0.8625	0.9097	0.9132	0.8620
11212112	0.8269	0.7843	0.8756	0.9091	0.9005	0.8593
22121122	0.8093	0.7730	0.8963	0.9056	0.9113	0.8591

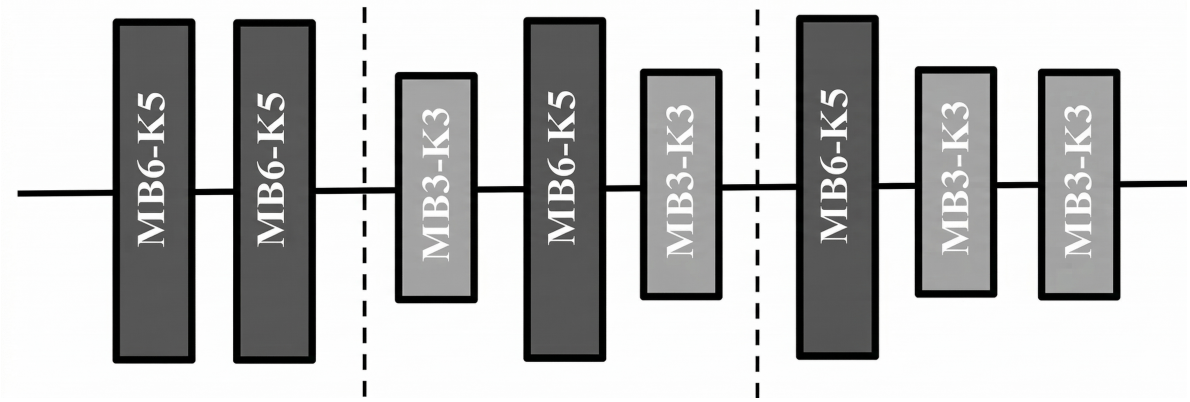
The training dataset used is 178,731 images for training and 44,683 images for testing, where pathologies identified as Uncertain or Not Mentioned were taken as Absent. Each architecture is trained for 50 epochs using a standard SGD optimizer with a learning rate of 0.0004, decay factor of 0.97, a batch size of 24, and a momentum of 0.9. Figure 9 shows the architecture that obtained the best score.

These models were trained on an PC with processor i9 9900K, memory RAM of 64 Gigabytes and NVidia RTX 3090 24 GB GPU 50 epochs were used because it was sufficient to obtain enough precision that provides information; additional epochs do not improve the final precision and instead increase the computational cost.

The experiments were carried out with recognized neural network architectures for image

Figure 9

Visualization of the best architecture GenENAS-22121211.



classification such as DenseNet, ResNet and EfficientNet and the AUC-ROC scores obtained were compared with the architecture that recommends GenENAS.

Table 8 contrasts the best GenENAS-discovered rchitecture (GenENAS-22121211) against three canonical baselines for image classification: DenseNet-121, EfficientNet-B0, and ResNet-152, all trained under identical conditions (50 epochs, SGD with learning rate 0.0004, batch size 24, NVIDIA RTX 3090). The *Overall* column reports the unweighted mean AUC-ROC across the five evaluated pathologies. Here we can observe, first, GenENAS-22121211 substantially outperforms EfficientNet-B0 on every pathology (overall +0.119 AUC-ROC), confirming that the search procedure produces architectures better adapted to multi-label chest X-ray classification than a generic image-classification baseline of comparable scale. Second, GenENAS-22121211 is competitive with the much larger DenseNet-121 (overall difference of only +0.001 AUC-ROC) and approaches the performance of ResNet-152 (overall gap of −0.006 AUC), despite using between 2.1× and 30.5× fewer parameters than the three baselines (1.91M for GenENAS-22121211 versus 4.01M for EfficientNet-B0, 6.96M for DenseNet-121, and 58.15M for ResNet-152, as detailed in Table 9). Third, the per-pathology ordering identified in Table 7 — Pleural Effusion easiest, Cardiomegaly hardest — is reproduced across all four architectures, suggesting that the task difficulty is data-driven rather than architecture-driven.

In Figures 10, 11, 12, and 13 show the ROC and PR plots in the validation set using the

Table 8
Experiment results. AUC-ROC score values obtained for DenseNet, ResNet, EfficientNet and best architecture obtained.

Neural Network	**Pathology AUC-ROC Score**					
Architecture	**Atelectasis**	**Cardiomegaly**	**Consolidation**	**Edema**	**Pleural Effusion**	**Overall**
DenseNet-121	0.8273	0.7782	0.8995	0.9055	0.9285	0.8678
EfficientNet-B0	0.7198	0.7531	0.7577	0.7532	0.7632	0.7494
ResNet-152	0.8491	0.8214	0.8958	0.8875	0.9212	0.8750
GenENAS-22121211	0.8368	0.7833	0.9020	0.9198	0.9017	0.8687

DenseNet-121, EfficientNet-B0, ResNet-152 and GenENAS-22121211 architectures respectively.

Figure 10
ROC and PR plots for DenseNet-121.

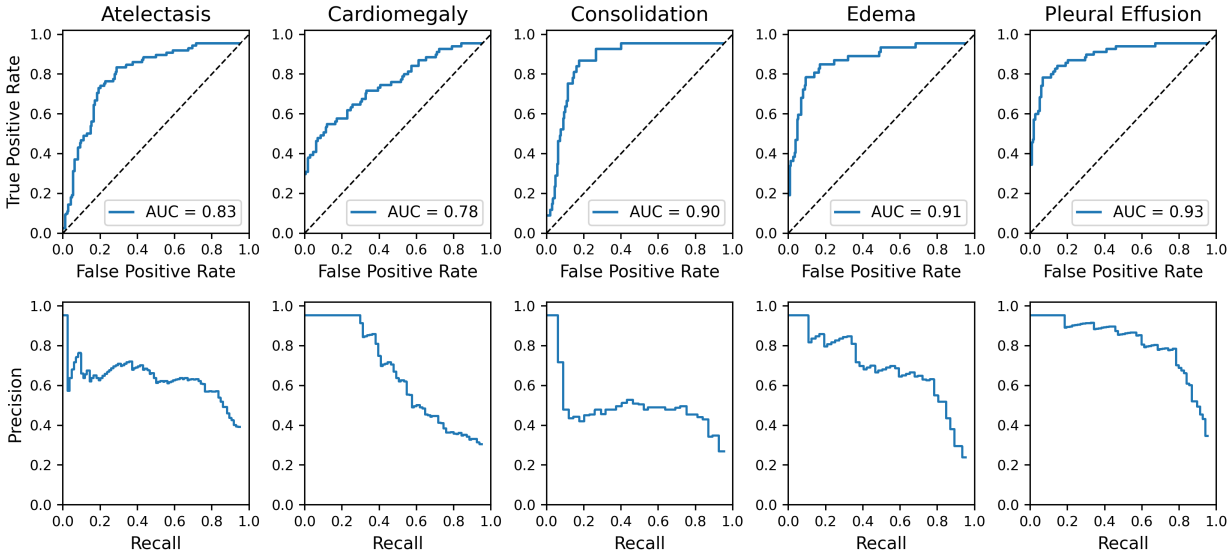


Table 9 summarizes the number of parameters, computational complexity (GFLOPs), memory footprint, individual training execution time, cumulative training time, energy consumption, cumulative energy consumption, and cumulative emissions of the architectures generated by GenENAS, as well as the reference architectures DenseNet-121, EfficientNet-B0, and ResNet-152. The inclusion of GFlops complements the parameter count by quantifying the actual computational workload of each architecture, thereby enabling a more comprehensive analysis of the relationship

Figure 11

ROC and PR plots for EfficientNet-B0.

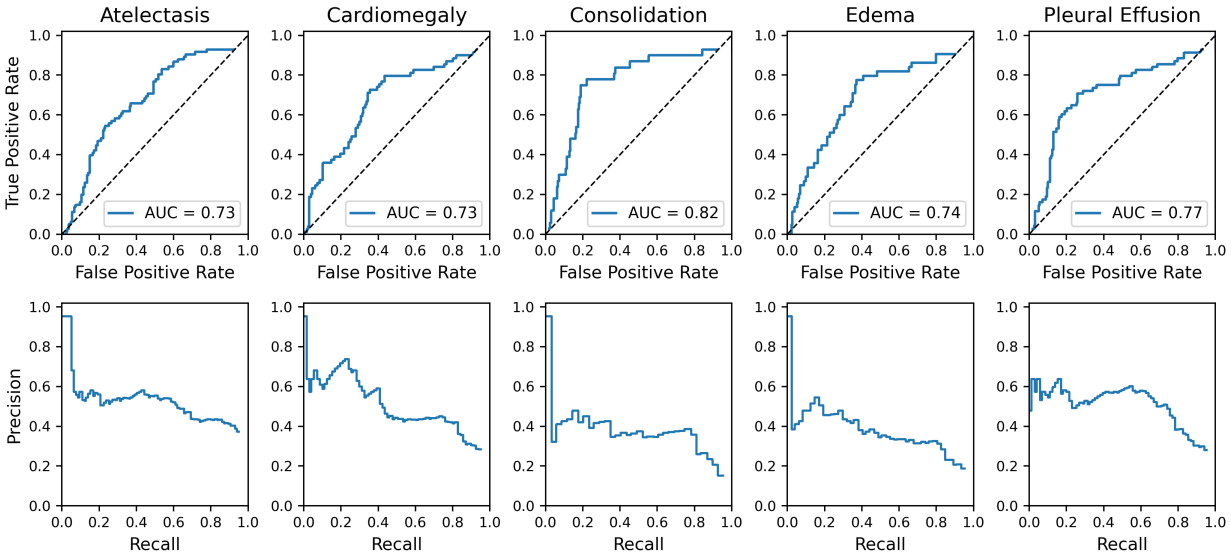
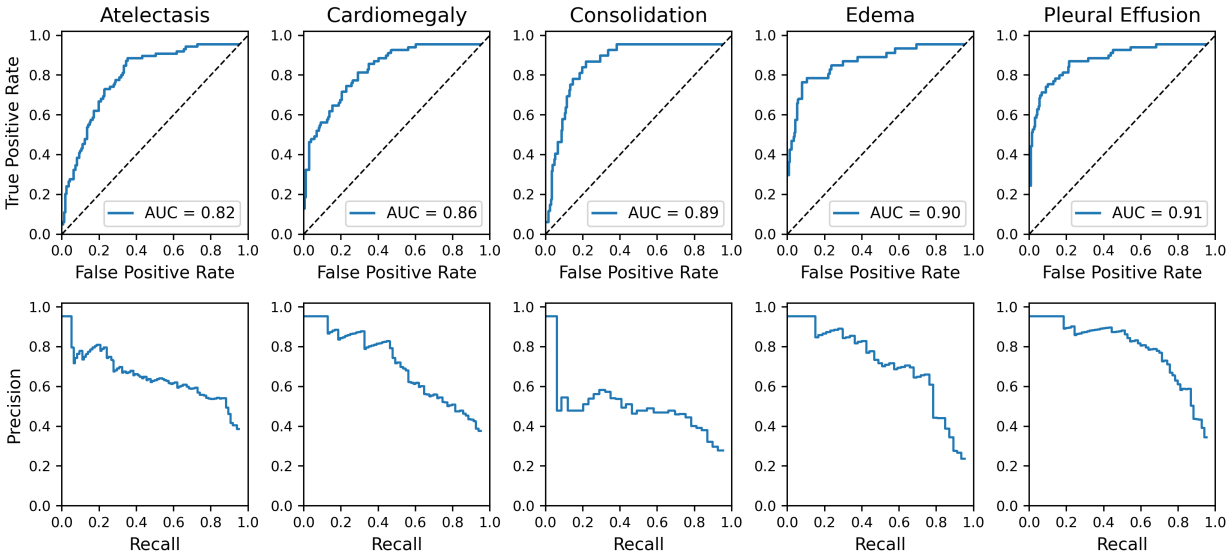


Figure 12

ROC and PR plots for ResNet-152.

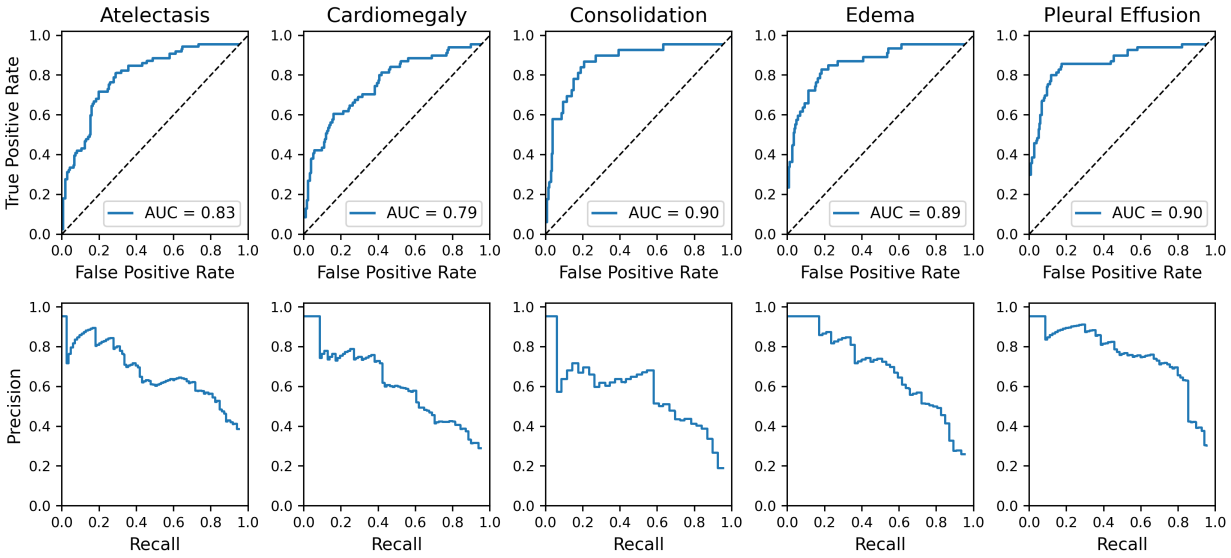


between model size, computational cost, execution time, and energy consumption.

As shown in Table 9, architecture GenENAS-22121211 comprises 1,909,541 parameters and requires a training duration of 43.21 hours. A clarification is warranted regarding the relationship between the number of parameters and the training time reported in Table 9. These two

Figure 13

ROC and PR plots for GenENAS-22121211.



quantities measure different aspects of a model and are not expected to correlate directly. The parameter count reflects the total number of learnable weights and is dominated by the deep, high-channel stages of the network, where the inverted-residual blocks with the largest expansion ratio contribute the most weights. The execution time, in contrast, is governed by the number of floating-point operations (GFLOPs) and by the activation memory traffic incurred during the forward and backward passes, both of which scale with the spatial resolution ($H \times W$) of the feature maps on which each operation is applied. Because convolutional cost grows with the spatial area, placing an expensive `MB6_K5` block (kernel size 5, expansion ratio 6) at an early, high-resolution stage is far more costly in compute and memory than placing the same block in a deeper, low-resolution stage, even though its contribution to the total parameter count is comparatively modest. Please note that the time shown in the table is the individual time required for each neural network architecture to train.

This decoupling explains the apparently counterintuitive cases in Table 9. For instance, architecture `GenENAS-22121211` has 1,909,541 parameters yet requires 43.21 hours of training, whereas `[11212122]` has 2,599,109 parameters—roughly 690,000 more—but completes in only

32.78 hours. The former assigns the two most expensive `MB6_K5` blocks to the first two searchable positions, which operate at the highest spatial resolution (320×320) and therefore incur the greatest computational cost; the latter places inexpensive `MB3_K3` blocks at those early positions and defers the `MB6_K5` blocks to the deeper, low-resolution stages, where their cost is marginal. Conversely, the larger parameter count of `[11212122]` arises precisely because its high-expansion blocks reside in the 256-channel stages, where the weight count is highest. Consequently, an architecture with fewer parameters can require longer training when it concentrates costly operations at high-resolution stages, confirming that parameter count alone is not a reliable proxy for computational cost.

Table 9
Model complexity, memory footprint, training time, and energy consumption.

Arch.	Param.	GFLOPs	Memory [MB]	Time [h]	Cumul. Time [h]	Energy [kJ]	Cumul. Energy [kJ])	Cumul. Emissions [kgCO_2e]
11212122	2599109	3.81	9.91	32.78	32.78	52 513.6	52 513.6	3.78
21212121	2184869	3.76	8.33	38.40	71.18	61 516.8	114 030.4	8.204
22121211	**1909541**	**3.82**	**7.28**	**43.21**	**114.39**	**69222.4**	**183252.8**	**13.184**
21121211	1876325	3.40	7.16	35.50	149.89	56 871.0	240 123.8	17.276
12112122	2586821	3.96	9.87	40.31	190.2	64 576.6	304 700.4	21.922
11212112	2171333	3.47	8.28	32.43	222.63	51 952.9	356 653.3	25.659
22121122	2600357	4.25	9.92	48.04	270.67	76 960.1	433 613.4	31.196
DenseNet-121	6958981	5.84	26.87	17.60		28 195.2		2.028
EfficientNet-B0	4013953	1.57	16.06	11.96		19 159.9		1.378
ResNet-152	58154053	23.20	232.62	33.43		53 554.9		3.853

2.4 Summary of the Serial Case Study

The use of generative algorithms to search for neural network architectures, such as the GenE-NAS method proposed in this chapter, is highly efficient, as it enables the generation of much smaller models in terms of RAM and parameter count, yielding excellent results in medical image classification.

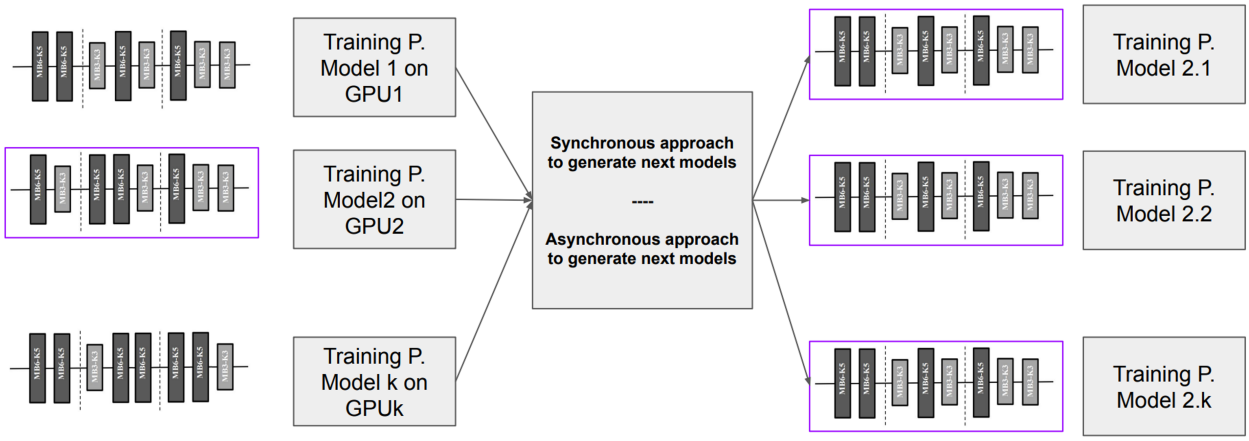
The ResNet-152 architecture performed well in image classification, but it has the disadvantage of having parameter counts and memory usage up to 30 times greater than those of the

GenENAS 22121211 architecture, which has very similar performance. The following section analyzes how this GenENAS strategy can be parallelized with additional computational resources, resulting in GenENAS Parallel, which enables faster development of neural network architectures.

2.5 Generative parallelization of the Neural Architecture Search

Distributing the GenENAS search across multiple GPUs addresses a fundamental bottleneck: evaluating dozens of candidate architectures sequentially on a single device would make the search prohibitively expensive for clinical-scale datasets. In the parallel variant illustrated in Figure 14, the orchestration is implemented with MPI following a master–worker scheme: the Large Language Model acts as a centralised controller hosted on the master process (rank 0) that, at each iteration, proposes as many candidate architectures as there are GPUs available in the HPC node. Each candidate is then dispatched via MPI to a dedicated worker process bound to a single GPU and trained concurrently with the others, so that one full training run is carried out per device in parallel rather than sequentially. Once all concurrent trainings finish, their AUC-ROC scores are gathered back to the controller through a collective MPI operation at an iteration-level synchronisation barrier, and this aggregated feedback is used to propose the next batch of architectures in the following iteration. The wall-clock time of each iteration is therefore bounded by the longest of the parallel trainings rather than by their sum, while the number of architectures explored per iteration scales linearly with the number of available GPUs. This design exploits the natural parallelism of architecture search — each candidate is an independent training job that requires inter-process communication only at iteration boundaries, never during the training itself — making the method directly compatible with standard HPC schedulers and multi-GPU nodes [73].

Figure 14
GenENAS parallel architecture



2.5.1 AI Healthcare Supported by HPC

HPC involves using multiple cores to support parallel processing. This processing can be massive or intensive. HPC supports AI in the design and organization of computer architecture. The well-known expression *Cambrian Explosion of Computing* for AI and Big Data [74] for customizing computer architectures to meet requirements, in this case, AI needs [73]. Then, there are key points to observe:

- Next-Generation HPC Systems: Next-generation HPC systems and HPC architectures are raising the bar for computing and memory performance, I/O, and flexibility. These systems are designed to meet the increasing demands of HPC workloads, including AI and analytics of course. They draw from the cloud and support a diverse range of architectures and compute models, and scaling.

- Unified and Open Programming Models Developers can use unified and open programming models, such as OneAPI[1], CUDA[2] or directives as OpenACC[3] to create a single codebase that

[1] `https://www.oneapi.io/`
[2] `https://developer.nvidia.com/cuda-toolkit`
[3] `https://www.openacc.org/`

can be used across different architectures, including CPUs, GPUs, and FPGAs. This approach enables more productive, high-performance development in the HPC and AI domains.

- Convergence of AI and HPC: The convergence of AI and HPC is an ongoing development. In real-world scenarios, AI architectures can be noisy, incomplete, and heterogeneous, leading to performance that differs from benchmark results. Creating more robust mathematical frameworks for selecting AI architectures and optimizing their performance is a focus for future development.

- Hybrid Workflows: Hybrid workflows that combine traditional compute (i.e., simulation) with data science methods are becoming more efficient. For example, coupling legacy computational fluid dynamics (CFD) codes with AI libraries for forecasting or astrophysics can be challenging, but advancements in HPC architectures, such as GPUs and TPUs, enable more efficient integration of AI and HPC.

- Challenges and Trade-offs: There are challenges and trade-offs when it comes to AI and HPC configurations. AI-heavy workloads often prioritize speed over core count, while HPC workloads prefer greater compute performance with a high core count and more core-to-core bandwidth. Developers face challenges in transitioning software to function on HPC clusters and efficiently programming high-performance parallel computing. It can require a significant time investment, and compatibility across different hardware types and computing models is important.

- Specialized Processing Units Acceleration**: GPUs, TPUs or QPUs play a crucial role in accelerating AI workloads within HPC environments. NVIDIA, for example, offers a complete end-to-end stack and suite of optimized products, infrastructure, and services for high-performance computing and AI workflows. Their full-stack architectural approach ensures optimal performance, fewer servers, and lower energy consumption, resulting in faster insights at lower costs. On the other hand, we can observe the same for other companies,

such as Intel. However, the more impressive option is using QPUs (Quantum Processing Units) to support AI algorithm implementations.

- Energy Sustainability: the convergence of energy sustainability and AI/HPC is an area of growing interest: Green Energy and AI Synergy [75], Energy Efficiency and Sustainability [76], and finally, frugality for AI/HPC [77] implementations are hot topics in the specialized community.

HPC and AI revolutionize the healthcare sector, enabling faster and more accurate diagnoses. By leveraging HPC and AI, healthcare professionals can improve patient care, enhance research capabilities, and optimize healthcare delivery. In computing terms, parallelization plays a crucial role in the field of artificial intelligence (AI), because it is unlikely to implement the different AI algorithms in today's efficient mode without HPC support. Observing AI for healthcare [78] [75], the junction enables faster, more efficient model training, which is essential for developing and deploying AI models in healthcare applications [79] [73]. Some key points about the role of parallelization in AI for healthcare can be:

- Accelerating Model Training: Parallelization allows for the simultaneous execution of multiple computational tasks, significantly speeding up AI model training. This is particularly important in healthcare applications that require real-time responsiveness, such as medical image analysis, to enable quicker diagnosis and treatment planning.

- Handling Large Datasets: Healthcare generates and deals with large amounts of data, including medical records, imaging data, and genomic data. Parallelization enables the efficient processing and analysis of these large datasets, leading to more accurate and meaningful insights.

- Improving Efficiency: By leveraging parallelization techniques, AI systems can process and analyze data faster and lower costs than traditional on-premises infrastructure. This improved efficiency can help healthcare organizations optimize operations, reduce costs, and enhance patient care.

- Real-Time Decision-Making: Parallelization enables AI models to be trained and deployed in real time, enabling healthcare professionals to make faster, more informed decisions. This is particularly valuable in critical situations where timely interventions can significantly impact patient outcomes.

- Advancing Precision Medicine: Parallelization supports the development of precision medicine, which aims to provide tailored healthcare interventions based on individual characteristics and needs. By accelerating model training, parallelization enables the analysis of large-scale datasets and the identification of personalized treatment strategies.

It is important to note that parallelization is just one aspect of AI in healthcare. AI also plays a significant role in medical imaging and diagnostics, virtual patient care, medical research and drug discovery, patient engagement and compliance, rehabilitation, and other administrative applications [73]. Integrating AI into healthcare settings can revolutionize clinical decision-making, hospital operations, medical diagnostics, patient care, and more.

2.6 Experimentation

A simple node of a specific HPC testbed machine was used, and specific materials and methods are proposed, as described in the following subsections.

2.6.1 HPC infrastructure testbed

The node used for the experiments is equipped with an Intel i9 9900K 5.0 Ghz processor, Two (2) NVidia RTX 3090 24 GB Cards - GDDR6X, 384 bits CUDA, 10496 cores. The processor, the Intel Core i9-9900K, is a 9th-generation Intel Core i9 series processor with the key specifications:

- Cache: It has a 16 MB cache.

- Clock speed: It can reach a clock speed of up to 5.00 GHz.

- Cores: It has 8 cores and 16 threads.
- Manufacturing technology: It is manufactured using the 14 nm process.
- Compatibility: It is compatible with the LGA 1151 socket.

It is important to note that cache coherence is a key feature in this configuration, as the node is a multiprocessor system with multiple processors or cores sharing a common memory. In the AI applications of this work, cache coherence plays a crucial function in ensuring the consistency of shared data across different caches, which is essential for accurate and reliable computation [80].

The NVIDIA GeForce RTX 3090 is a high-end GPU based on the Ampere architecture, featuring 10,496 CUDA cores, 24 GB of GDDR6X memory, and a memory bandwidth of 936.2 GB/s.

These are some of the specifications of the NVidia RTX 3090:

- Memory: They have 24 GB of GDDR6X memory.
- Memory bandwidth: They have a 384-bit bus and a memory bandwidth of 936.2 GB/s.
- CUDA cores: They have 10496 CUDA cores.
- Tensor Cores: They have 328 third-generation Tensor Cores.
- RT Cores: They have 82 second-generation RT Cores.

Tensor and RT Cores are specialized hardware components found in NVIDIA GPUs, specifically in their Turing architecture and later generations. These cores serve different purposes and are designed to accelerate specific tasks in AI applications and graphics rendering. Observing the Tensor Cores, they are processing units that are specifically optimized for matrix multiplication and deep learning operations. They can perform mixed-precision matrix operations much faster than traditional GPU cores [73]. Tensor Cores benefit AI applications involving neural networks and deep learning algorithms. They can significantly accelerate tasks such as training and inference in

deep learning models, resulting in faster performance and improved efficiency. Frameworks like TensorFlow[4] utilize Tensor Cores to accelerate AI computations. They are available in limited GPUs, including the GeForce NVidia RTX, Quadro RTX, and Titan families. By leveraging the power of Tensor Cores, these GPUs can deliver enhanced performance in AI, gaming, and content creation applications.

On the other hand, RT Cores are specialized cores designed for real-time ray tracing. Ray tracing is a rendering technique that simulates the behavior of light in a scene, resulting in more realistic and accurate graphics. RT Cores are responsible for accelerating the ray tracing calculations and improving the efficiency of rendering complex lighting effects. They can calculate the paths of light rays and handle tasks such as intersection testing and shadow calculations, which are essential for realistic rendering [73]. Like the Tensor Cores, the RT Cores are a key feature of NVIDIA's Turing architecture and subsequent generations. They enable real-time ray tracing capabilities in GPUs, allowing for more immersive and visually stunning graphics in games and other applications. In this work, we don't use the RT Cores, only the Tensor Cores.

Then, because Tensor Cores are specialized hardware components optimized for matrix multiplication and deep learning operations, while RT Cores are designed for real-time ray tracing, both Tensor Cores and RT Cores contribute to NVIDIA GPUs' enhanced performance and capabilities in AI applications and graphics rendering. However, for this study, the performance guarantee by the Tensor Cores is more relevant.

Figure 15[5] shows a detailed configuration diagram, which includes an i9 9900K processor clocking at 5.0Ghz alongside two high-performance NVidia RTX 3900 graphics cards, boasting 24 GB of GDDR6X memory each.

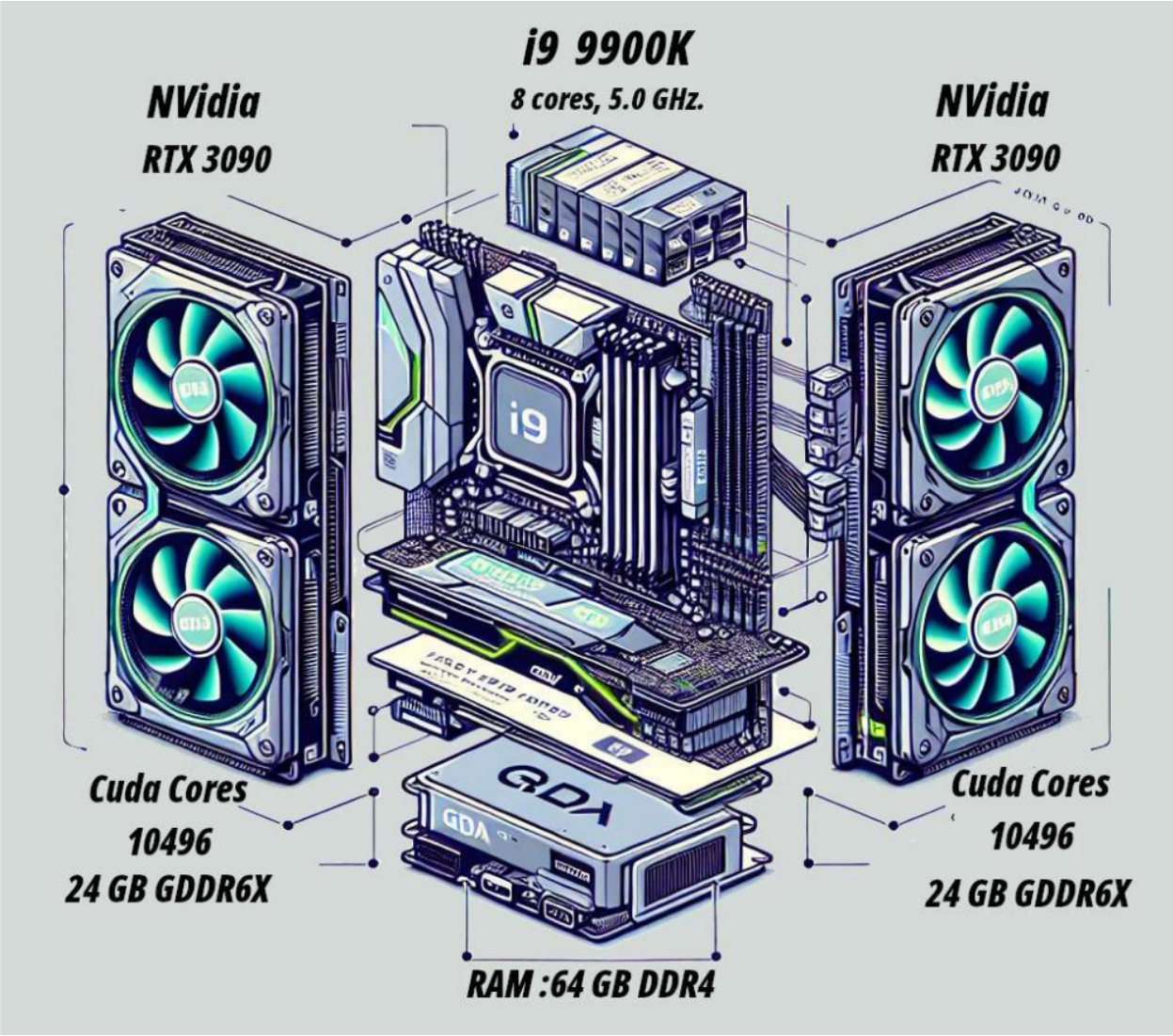
The NVidia RTX 3090 is a high-performance graphics card primarily designed for creation and intensive gaming tasks. They are also used in specialized clusters for AI, visualization, and scientific computing. This node was selected because it could correspond to a professional workstation configuration. Other nodes in a cluster can be used to increase the amount of data or the

[4] `https://www.tensorflow.org`

[5] This Figure 15 was generated by AI.

Figure 15

Computer node testbed architecture: a hybrid computer using Intel CPUs and NVIDIA GPU [73]. Note: Diagram generated using AI and subsequently edited and annotated by the author for technical accuracy.



number of intensive processes. Still, for this work, only one was selected, as we were interested in the parallelization factor across the processing cores for both the CPU and the GPU/TPU. However, several aspects related to internode interconnectivity must be considered.

A hybrid platform based on CPU/GPU/TPUs was used, and it is important to discuss the CPU at this point. The Intel Core i9-9900K processor has a base clock speed of 3.6 GHz and is an octa-core processor with 16 threads, part of Intel's 9th-generation CPUs. The i9-9900K is known for its high performance and is often used in high-performance computing, low-cost nodes, gaming, and high-end desktop systems. The processor features Intel Turbo Boost Technology 2.0, which dynamically increases its clock speed when needed to boost performance for demanding tasks. It also supports Intel Hyper-Threading Technology, which enables each physical core to handle two threads simultaneously, improving multitasking capabilities. In this study, it is important due to the support for intensive calculations.

2.6.2 Results and Discussion

The training dataset used is 178,731 images for training and 44,683 images for testing. Each architecture is trained for 50 epochs using a standard SGD optimizer with a learning rate of 0.0004, decay factor of 0.97, a batch size of 24, and a momentum of 0.9. These models were trained in your serial version on an PC with processor i9 9900K, memory RAM of 64 Gigabytes, and GPU NVidia RTX 3090 24 GB, with 50 epochs used because it was sufficient to obtain enough precision that provides information; additional epochs do not improve the final precision and instead increase the computational cost.

For these tests, both serial and parallel, GenENAS used Llama 2 from 7B, an open-source LLM model developed by Meta AI and released in July 2023. This specific version has 7 billion parameters, making it lightweight, fast, and capable of running on hardware with moderate processing power, which was installed locally on the machine used to run these tests. For execution in serial mode, LLama 2 is asked to recommend a neural network structure based on the macro

structure shown in Table 6, considering the three candidate blocks for each layer. Once a structure is recommended, it is trained and validated to determine the accuracy achievable with the generated model, as shown in Table 10, where the first column, Iteration, lists the completed experimental iterations, while the Architecture column specifies the neural network configuration recommended by the LLM in this case Llama 2 for each respective iteration. Model performance is evaluated using the AUC-ROC metric, where higher values indicate superior classification accuracy and discriminative capacity. Furthermore, the Parameters and Memory [MBytes] columns quantify the structural complexity and the model's footprint, respectively. Finally, the Cumul. Time [Hours], cumul. energy [kJ] and cumul. emissions [KgCO_2e] metrics record the cumulative time, cumulative energy consumption, and cumulative emissions throughout the experimental process.

Once this new model is trained, the Llama 2 is asked again to recommend a new structure that might achieve better accuracy, given the results obtained with the previous architectures. The process is repeated until the accuracies no longer improve and plateau. At this point, the best model is determined as the one with the highest accuracy based on the AUC-ROC. The GenENAS optimization workflow employs a recursive feedback mechanism in which the LLM suggests new topologies based on the historical performance metrics of preceding architectures. To ensure operational efficiency, the algorithm incorporates an early stopping policy that terminates the exploration after two consecutive iterations without significant improvements in the AUC-ROC. Experimental results demonstrate that the most robust configuration, denoted as $[2, 2, 1, 2, 1, 2, 1, 1]$, was identified during the fifth iteration. Nevertheless, the complete validation of the process required a total of seven iterations, accumulating an energy consumption of 456,815.1 kJ, and with cumulative emissions of 32.865 KgCO_2eq, and a temporal investment of 285.15 machine hours, because no AUC-ROC greater than that obtained on the fifth iteration was achieved, as the algorithm was set with a patience of 2 for early stopping. The final model achieved a peak performance with an AUC-ROC of 0.87, as detailed in Table 10.

Following the MPI master–worker scheme described in Section 2.5, the experiments reported here were executed on an HPC node equipped with two NVIDIA RTX 3090 GPUs. Accord-

Table 10
Experiments results serial GenENAS.

Iter	Arch.	AUC-ROC	Param.	Memory [MBytes]	Cumul. Time [Hours]	Cumul. Energy [kJ]	Cumul. Emissions [kgCO_2e]
1	1,0,1,0,0,1,0,0	0.82	540 741	2.06	15.14	24 252.7	1.745
2	1,0,2,0,0,2,0,0	0.82	750 981	2.86	36.60	58 620.4	4.217
3	1,2,1,1,1,2,2,2	0.86	2 635 973	10.06	94.16	150 845.9	10.853
4	2,1,1,1,2,1,2,1	0.86	2 139 365	8.16	140.74	225 471.9	16.221
5	**2,2,1,2,1,2,1,1**	**0.87**	**1909541**	**7.28**	**190.23**	**304750.1**	**21.925**
6	1,1,2,1,2,1,1,2	0.85	2 171 333	8.28	234.29	375 335.8	27.003
7	2,2,1,2,1,1,2,2	0.86	2 600 357	9.92	285.15	456 815.1	32.865

ingly, the LLM (Llama 2) controller proposed two candidate architectures per iteration, which were trained and validated concurrently—one per GPU—before their AUC-ROC scores were returned to guide the next iteration.

The execution of GenENAS Parallel—configured for 10 potential iterations, this parallelized approach yielded superior convergence, achieving an AUC-ROC of 0.87 by the third iteration and fulfilling the termination criteria within only five iterations, because no AUC-ROC greater than that obtained on the third iteration was achieved, as the algorithm was set with a patience of 2 for early stopping. While this strategy successfully identified the optimal architectural configuration $[2, 2, 1, 2, 1, 2, 1, 1]$ with a significant reduction in execution time (totaling 225.52 hours) compared to its serial version which took 285.15 hours to detect the optimal architecture, meaning the parallel version took 59.63 hours less (20.91% less),it nevertheless produced a notable increase in energy expenditure. This is basically reflected in the number of models that each version (serial and parallel) had to train to find the optimal architecture. As can be seen in Tables 10 and 11, the serial version had to train 7 candidate architectures while the parallel version trained 10 candidate architectures. Moreover, each parallel iteration is bounded by the slowest candidate in its batch rather than by the average training time.

This difference in the number of trained models also accounts for the increase in energy expenditure. Since the energy per model is comparable across both versions, the overhead does not stem from parallelization itself but from the interaction between the batch size and the early-

stopping patience: although the optimum was found at the third iteration, the patience of 2 forced the completion of iterations 4 and 5, training four additional non-improving models (two per iteration) instead of the two extra models a serial run incurs for the same patience window. Consequently, the parallel execution consumed 645,435.4 kJ, and with cumulative emissions of 46.435 KgCO_2eq, an overhead of 188,620.3 kJ (41.29%) over the sequential baseline (Table 11). These results show a trade-off between time performance and energy cost: although parallelization accelerates the process of searching for architectures, this benefit is obtained at the expense of higher energy consumption due to the concurrent use of multiple computing resources.

Table 11
Experiments results parallel GenENAS.

Iter	Arch.	AUC-ROC	Param.	Memory [MB]	Cumul. Time [Hours]	Cumul. Energy [kJ]	Cumul. Emissions [kgCO_2e]
1	1,0,1,0,0,1,0,0	0.82	540 741	2.06	19.59	56 071.45	4.034
	1,0,2,0,0,2,0,0	0.82	750 981	2.86			
2	1,2,1,2,2,1,2,1	0.86	2 274 629	8.68	77.33	221 304.2	15.922
	2,1,2,1,1,2,1,2	0.86	2 234 021	8.52			
3	**2,2,1,2,1,2,1,1**	**0.87**	**1909541**	**7.28**	127.09	363 728.7	26.168
	2,1,2,1,1,2,1,1	0.85	1 806 245	6.89			
4	1,2,2,1,1,2,2,1	0.85	2 253 701	9.02	177.95	509 292.9	36.640
	2,1,1,2,2,1,1,2	0.86	2 254 949	9.02			
5	1,2,2,1,1,2,2,2	0.86	2 681 477	10.73	225.52	645 435.4	46.435
	2,1,1,2,2,1,1,1	0.86	1 827 173	7.31			

The classification of chest X-ray images has been approached using various algorithms and machine learning techniques. One common approach is the use of convolutional neural networks (CNNs) . CNNs are deep neural networks well-suited for image classification. They can automatically learn features from the input images and make predictions based on them.

Generative parallelization of Neural Architecture Search (GenENAS Parallel) aims to improve the efficiency of the classification process. NAS involves automatically searching for the optimal neural network architecture for a given task. By parallelizing the search process, we can

explore a larger space of possible architectures and find more efficient models. However, it is necessary to observe computer features to support parallelization in a sustainable energy manner.

Many research studies have focused on developing techniques to improve the efficiency of chest X-ray image classification. These include the generative parallelization of Neural Architecture Search and the augmentation of training data using generative adversarial networks. These techniques aim to enhance the performance of CNN classifiers and enable more accurate and efficient diagnosis of chest X-ray images.

All this demonstrates that generative architecture search is an effective tool for creating compact, competitive models for medical image classification, especially in the study of pulmonary pathologies, where high AUC-ROC scores are achieved, and a strong balance between performance and model size is evident. Evidence shows that well-known architectures like ResNet-152, although highly accurate, have a very large number of parameters, resulting in memory consumption up to 30 times higher than that of the generated architecture while achieving similar performance. This leads to the conclusion that GenENAS not only identifies accurate models but also computationally lightweight and scalable ones, making them competitive for medical diagnostic tasks.

Since GenENAS enables obtaining lighter architectures without significant performance loss, the next step is to analyze how these architectures can be further improved through compression and knowledge transfer techniques. For this purpose, the next chapter introduces the Knowledge Distillation (KD) approach, which is a technique that enables transferring knowledge from a robust and complex model (Teacher) to a more compact and efficient model (Student) without losing its performance and accuracy, continuing the line of Deep Learning model optimization that is worked on throughout this research.

2.7 Summary of the Parallelization Study

GenENAS leverages the generative capabilities of large language models such as Llama 2 to automate the search for optimal neural network architectures based on historical results. This

study examines the combination of parallelization and generative algorithms to optimize neural architecture search for chest X-ray classification, evaluating their impact on the NAS algorithm using the CheXpert dataset [73].

The classification of medical images using deep learning models poses a significant challenge owing to the large volume of data, the high computational cost of these techniques, and the class imbalance inherent in medical datasets caused by the low prevalence of certain pathologies. To minimize computational resource consumption, it is necessary to design more compact and efficient models that can be deployed in clinical settings with limited resources.

The study employs the CheXpert dataset, which comprises 224,316 chest radiographs, to classify five pulmonary pathologies. GenENAS evaluates 6,561 candidate architectures within an eight-layer search space, using AUC-ROC and precision-recall curves as performance metrics. Following training on 187,641 images, the sequential algorithm required 285.15 hours to achieve an AUC-ROC of 0.869.

In the parallel execution across two GPUs, an AUC-ROC of 0.870 was obtained in 225.52 hours, reflecting a reduction in search time but, conversely, a higher energy consumption, since the parallel variant of GenENAS trained ten neural network architectures whereas the sequential version trained only seven.

These results demonstrate that the parallel approach reduced neural architecture search time while maintaining considerable accuracy. Furthermore, the use of GenENAS enables the discovery of more compact architectures without significantly sacrificing performance. This is reflected in critical medical metrics such as AUC-ROC, demonstrating that it is possible to balance computational efficiency and predictive quality, thereby facilitating clinical deployment in diagnostic

support.

3. KD-MLC - Knowledge Distillation for Multi-Label Classification

This chapter develops the second stage of the proposed framework. As established in the previous chapter, the first stage—GenENAS—automatically discovers a compact, high-performing neural architecture for the classification of thoracic pathologies. In this second stage, knowledge is transferred from a high-capacity teacher model to the student model (GenENAS; 1.91 M parameters, 7.28 MB), built upon the architecture selected in the first stage, through the widely known methodology of Knowledge Distillation. Furthermore, the knowledge transfer analysis is carried out using two manually selected compact baselines as a student (one model based on the MobileNetV2 architecture and another on the EfficienNet-B0 architecture), which allows a direct comparison between the student automatically found in the first stage of GenENAS and some already recognized ones, to determine the robustness of the stage. The three models share a single DenseNet-121 teacher and identical data, split, hardware, and optimization settings, so the comparison between the discovered student and the hand-picked students is fair. The chapter therefore answers two coupled questions: does KD produces a compact, accurate, sustainable model? and does the architecture from Stage 1 responded to distillation as well as, or better than, a strong baseline?.

3.1 Knowledge Distillation: Mechanism and Role in the Framework

3.1.1 The distillation mechanism

Knowledge distillation (KD) is a machine learning technique that transfers knowledge from a large, computationally intensive teacher model to a smaller student model without significant loss of

accuracy. Originally introduced by Bucila et al. in 2006 [81] and later formalized by Hinton [5], KD serves as an effective approach for model compression and knowledge transfer, enabling deployment on less powerful hardware.

The growing success of deep neural networks has led to increasingly large models that demand substantial memory and computational resources, limiting their use in environments with restricted hardware capacity. In recent deep learning research, it has become evident that advances in computer vision and natural language processing largely depend on the continuous growth of computational capacity. The current success of deep learning rests on a rather fragile foundation the uncontrolled expansion of computing resources. There is no clear revolution in efficiency, whether through specialized hardware, algorithmic optimization, or the adoption of hybrid paradigms. This situation poses the risk of facing an "AI winter" triggered by reaching the physical and economic limits of computation [82]. Progress in deep learning has been driven primarily by increases in parameters, data, and computational power rather than by substantial algorithmic innovations [82]. Deep learning models are highly dependent on computational capacity, as increasingly larger architectures demand greater processing power. To reduce benchmark error rates by half on datasets such as ImageNet, it is necessary to increase computational power by up to 5,000 times, resulting in exponential growth in both energy and monetary costs, on the order of 10^8 to 10^{16}, along with a significant rise in CO_2 emissions.

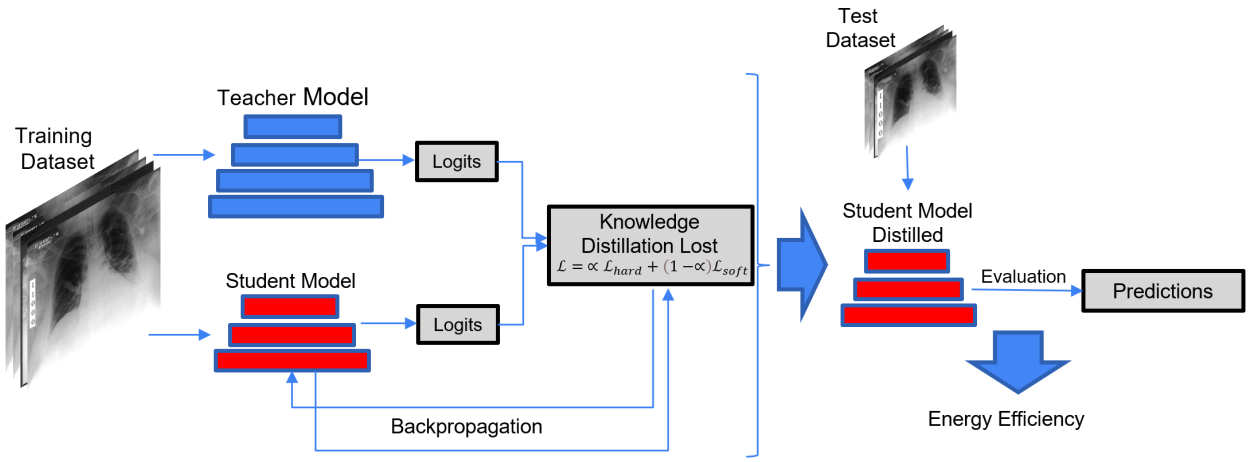
The idea of generating a lighter model stems primarily from the remarkable success of neural networks, which, in most cases, are of massive size. This necessitates systems with substantial memory and computational capacity for their training and deployment. To broaden the use of these models, it is critical to enable their implementation on systems with limited computational resources, such as mobile devices and edge devices. This need is particularly evident in the medical sector, where systems with very limited computational capabilities are deployed in remote areas that require real-time model execution. Consequently, it becomes essential to develop ultra-lightweight and accurate deep learning models, making techniques such as knowledge distillation vital for simplifying models while maintaining accuracy.

The main objective of knowledge distillation is to train a lighter, more compact model that emulates a larger and more complex counterpart, thereby allowing compact models to achieve critical tasks such as vision, natural language processing (NLP), acoustic recognition, medical diagnostics, and applications on edge devices.

Figure 16 shows the knowledge distillation mechanism, which enables the transfer of learned representations and predictive capabilities from a large pretrained teacher model to a smaller computationally efficient student model.

Figure 16

Knowledge distillation process mechanism.



This process is detailed algorithmically in Algorithm 4, and comprises four main stages: The teacher network, typically a high-capacity deep neural architecture with millions of parameters, is first trained on the complete dataset using conventional supervised learning until convergence, ensuring robust generalization and high predictive accuracy. The student network, with a significantly smaller number of parameters, is then trained to emulate the teacher's behavior. During this stage, the student learns from both the ground-truth labels and the teacher's output distributions, referred to as soft targets. The optimization objective combines two complementary components: a hard loss computed from the true labels and a soft loss derived from the teacher's probabilistic outputs. As formalized in Algorithm 4, the combined loss function balances these two terms,

enabling the student to capture the teacher's latent feature relationships and decision boundaries.

The final distilled model is lightweight, memory-efficient, and optimized for deployment on edge or embedded systems with constrained computational resources. Despite its reduced complexity, it maintains predictive performance comparable to that of the teacher model, thereby achieving an effective trade-off among accuracy, efficiency, and sustainability.

The following algorithm 4 formalizes the distillation procedure, based on the general framework proposed in [5], adapted to the multilabel medical image classification setting.

Algorithm 4 Proposed KD-MLC: Knowledge Distillation Transfer for Multi-Label Medical Image Classification

Require:

D: Training dataset (x, y)
M_T: Pre-trained teacher model
M_S: Student model
α: Distillation weighting factor
T: Temperature parameter
w_c: Class weights
$\mathcal{S}$: Uncertainty handling strategy (Zeros or Ones)

for each training epoch **do**
  for each batch (x, y) in D **do**
    Apply uncertainty strategy $\mathcal{S}$ to labels y
    $z_T \leftarrow M_T(x)$ ▷ Teacher logits
    $z_S \leftarrow M_S(x)$ ▷ Student logits
    $p_T \leftarrow \text{Sigmoid}(z_T/T)$
    $p_S \leftarrow \text{Sigmoid}(z_S/T)$ ▷ Temperature-scaled probabilities
    $\mathcal{L}_{hard} \leftarrow \text{Weighted-BCEWithLogitsLoss}(z_S, y, w_c)$
    $\mathcal{L}_{soft} \leftarrow \sum_{c=1}^{C} KL\Big(\text{Bernoulli}(p_T^{(c)}) \parallel \text{Bernoulli}(p_S^{(c)})\Big)$
    ▷ Class-wise Bernoulli KL-Divergence
    $\mathcal{L}_{total} \leftarrow \alpha\, \mathcal{L}_{hard} + (1 - \alpha)\, \mathcal{L}_{soft}$
    Backpropagate $\mathcal{L}_{total}$
    Update parameters of M_S
  end for
end for
return Distilled student model M_S

3.1.2 Integration with Stage 1: the unified GenENAS → KD pipeline

The contribution of this chapter to the framework lies in articulating Stages 1 and 2 into a unified workflow, where GenENAS (Stage 1) generates a compact architecture and Knowledge Distillation (KD) (Stage 2) trains it by transferring knowledge from the teacher model. The resulting model, referred to as Distilled GenENAS, is trained and evaluated under conditions identical to those of the MobileNetV2 and EfficientNet-B0 baseline, including the same teacher model (DenseNet-121), dataset, data split, hardware platform, and optimization settings.

3.2 Case Study: Lung diseases

In the pursuit of developing more compact models that can be easily deployed on devices with limited computational resources, knowledge distillation is applied—a technique recognized as a highly efficient option for compressing deep learning models. Therefore, this work will analyze the impact of this innovative technique on achieving compact deep learning models with accuracy levels close to those of more robust, higher-computational-demand models.

Pulmonary diseases represent one of the leading causes of mortality and morbidity worldwide. Each year, thousands, and even millions, of people are affected by these conditions, with a significant impact on public health. The costs associated with treating these pulmonary pathologies are high due to hospitalizations, prolonged treatments, and work absences, which significantly affect community productivity and the economy. Ongoing research into these diseases has allowed improvements in early diagnosis techniques, thus enabling their timely detection. Early diagnosis significantly increases the likelihood of effective treatments and greatly reduces the risk of death.

Research on these diseases has helped identify risk factors such as smoking, environmental pollution, and infections, among others. This has led to the development of prevention campaigns and public education programs that help reduce the spread of these pathologies, especially given their high transmissibility.

Advancements in technology have driven the development of new tools, such as artificial intelligence, computational models, and advanced diagnostic methods, which have improved the precision and speed of clinical management for these diseases.

Acknowledging the importance of promoting technological innovation and advancing medical research, the scientific community has made available several public datasets that include medical images, such as:

- CheXpert, which contains chest X-rays annotated for 14 pulmonary pathologies and is one of the most widely used datasets for developing and evaluating models [70].
- NIH Chest X-ray, which includes more than 112,000 X-rays labeled for various thoracic diseases, based on radiology reports processed using natural language processing (NLP) techniques [83].
- Lungs Disease, a dataset containing X-ray images for four types of pulmonary diseases: viral pneumonia, bacterial pneumonia, COVID-19, and tuberculosis [84].
- LUNG-PET-CT-DX, which contains CT (Computed Tomography) and combined PET/CT (Positron Emission Tomography combined with CT) images for lung cancer diagnosis and advanced research, with detailed tumor annotations for specialized investigation [85].

These and other datasets have been essential foundations for developing computational models that improve the diagnosis, detection, and monitoring of pulmonary diseases globally. They are available on recognized platforms such as Kaggle, The Cancer Imaging Archive (TCIA), and other open repositories for the scientific community. Their accessibility has driven innovation and scientific progress, accelerating the development of models that greatly assist medical personnel in the automatic detection of diseases, thereby improving diagnostic accuracy, especially in contexts where specialists in this field are in short supply.

3.2.1 Materials and methods

.Study Design This study employs an experimental, quantitative design to evaluate the computational efficiency and diagnostic performance of knowledge distillation for medical image classification.

Using DenseNet-121 as the teacher model, three student architectures were trained under a unified knowledge distillation stage: the lightweight reference architecture *MobileNetV2*, the *GenENAS* architecture identified during stage I of the proposed framework (1.91 million parameters and a memory footprint of 7.28 MB), and *EfficientNet-B0*. The distillation procedure produced a corresponding distilled model for each student architecture, enabling a systematic evaluation of knowledge transfer effectiveness across networks with different architectural characteristics. Performance was assessed using AUC-ROC and F1-score metrics, while computational efficiency was quantified through architectural complexity, training cost, and environmental impact indicators. This experimental design facilitates a comprehensive analysis of the trade-offs between predictive performance, model compactness, computational requirements, and sustainability, thereby providing insights into the practical viability of deploying distilled deep learning models in resource-constrained medical imaging environments.

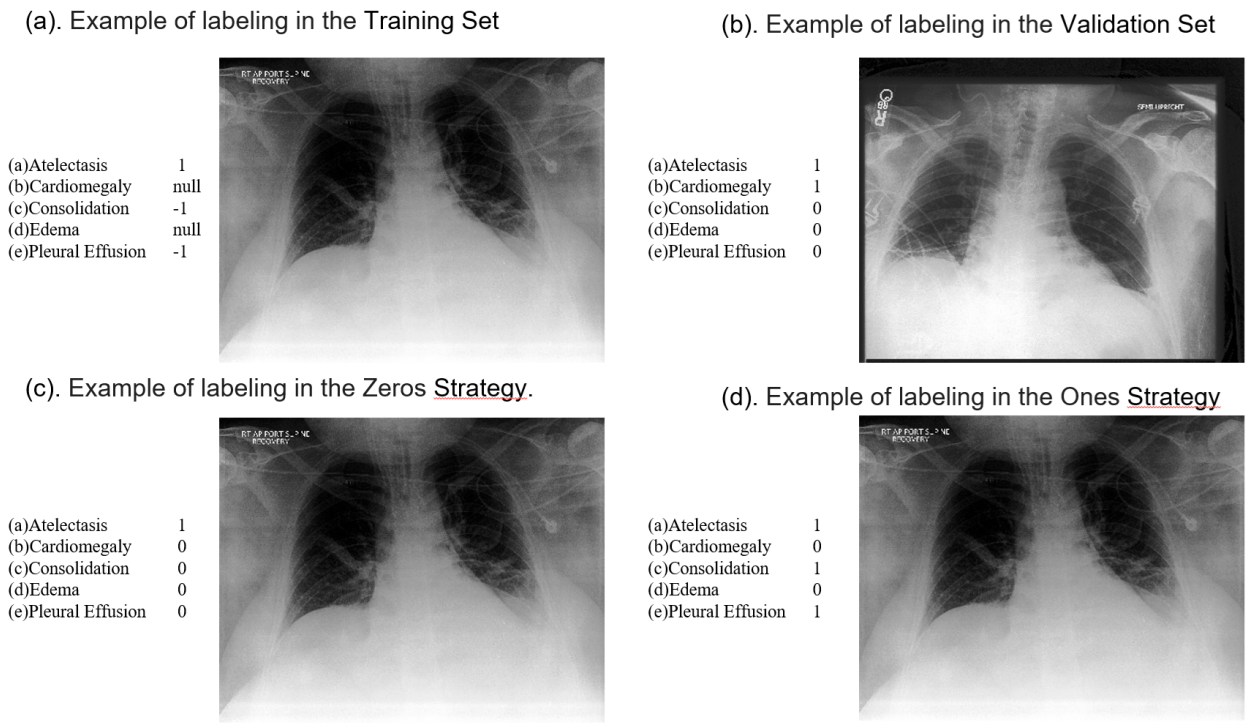
.Data This study uses the CheXpert dataset, previously introduced in Section 2.3.1. In this dataset, a large proportion of blank or uncertain labels can be observed across several pathologies, particularly Atelectasis, Cardiomegaly, Consolidation, Edema, and Pleural Effusion, which are the most frequent and clinically significant thoracic findings.

The five pathologies selected for this study: Atelectasis, Cardiomegaly, Consolidation, Edema, and Pleural Effusion, are the most frequent and clinically significant in the dataset. These cases were chosen as they represent the most frequent and clinically relevant thoracic findings in the CheXpert dataset. The images illustrate the visual variability and diagnostic complexity that the models in this work address. Figure 17 shows an example of an X-ray with its original labeling

and how the label would look for the two labeling strategies, the zeros and the ones.

Figure 17

Representative chest X-rays for the selected pathologies: (a) Labeling training set, (b) Labeling validation set, (c) Labeling Zeros strategy, (d) Labeling Ones strategy.



.Dataset Preparation and Label Mapping Strategy The dataset used in this study was obtained from the Stanford CheXpert repository and consists of two main files: train.csv and valid.csv. The train.csv file contains 223,414 entries. Its first column specifies the relative paths to chest radiographs in JPG format, while the subsequent 14 columns represent radiological findings (pathologies) annotated for each image. Each pathology label can take one of the following values: 1: pathology present, 0: pathology absent, -1: uncertain diagnosis (radiologist not fully confident), null/empty: pathology not mentioned.

Because uncertain labels (1) can significantly affect model learning, this study adopted the label imputation strategies proposed in CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison [70]. Two mapping strategies were defined:

Zero Strategy: Treats uncertain cases (1) as absent (0), reducing false positives and increasing precision but potentially lowering sensitivity.

Ones strategy: Treats uncertain cases (1) as present (1), improving sensitivity and recall but potentially increasing false positives.

Table 12 summaries how the original CheXpert labels are remapped under each uncertainty-handling strategy. The two strategies differ only in how they treat the uncertain value labeled with -1: the Zero strategy treats it as absent (0), prioritising precision by reducing the count of nominally positive cases; the Ones strategy treats it as present (1), prioritising sensitivity by exposing the model to additional borderline positives. Null labels are taken as absent, and the remaining labels (positive and absent) are unaffected.

Table 12
Label mapping strategy

Original Label	**Description**	**Zero Strategy Value**	**Ones strategy Value**
1	*Pathology present*	1	1
0	*Pathology absent*	0	0
-1	*Uncertain / ambiguous case*	0	1
Null / empty	*Not mentioned*	0	0

.Dataset Cohorts Split and label distribution To ensure robust model assessment, approximately 20% of the records from train.csv were reserved as a validation subset, while the remaining 80% were used for training.

The split was performed at the patient level rather than the image level to prevent data leakage. Consequently, all radiographs from a given patient were assigned exclusively to either the training or validation subset, but not both.

This partitioning was carried out using the GroupShuffleSplit function from the scikit-learn library [86]. The method ensures that images from the same patient remain within a single group, enabling an unbiased evaluation of model generalization to unseen subjects. The percentage values are approximate, as the split maintains patient integrity rather than enforcing exact record counts.

Figure 18 shows the subset partition and label distribution per subject for both the original CheXpert annotations (A) and after applying the Zeros strategy (B) and Ones strategy (C). The plots illustrate that the selected split preserved consistent label proportions across the training and validation cohorts, ensuring balanced representation for all five selected pathologies. When applying the Zeros strategy, a value of 0 is assigned to entries labeled -1, i.e., the uncertain cases in which radiologists are not sure about the presence of the pathology; these are treated as negative cases. Conversely, in the Ones strategy, a value of 1 is assigned to entries labeled as -1, meaning that the uncertain cases in which radiologists are unsure about the presence of the pathology are considered positive cases.

Tables 13 and 14 show the number of records and their percentages by pathology for the training and testing datasets under the zero strategy. As observed in the data in this strategy, the pathologies with the highest prevalence of positive cases are pleural effusion (38.61% for training and 38.46% for testing) and edema (23.16% for both training and testing). Conversely, consolidation exhibits the lowest proportion of positive cases, accounting for 6.63% in the training set and 6.56% in the testing set.

Table 13
Pathology case analysis for Training Zeros strategy.

Pathology	Positive	Absent
Pleural Effusion	69001–38.61%	109730–61.39%
Edema	41896–23.16%	136835–76.84%
Consolidation	11852–6.63%	166879–93.37%
Cardiomegaly	21619–12.10%	157112–87.90%
Atelectasis	26762–14.97%	151969–85.03%

By applying the Ones strategy, the following number of records are obtained for each of the pathologies in each of the training and testing datasets shown in tables 15 and 16.

Figure 18

Subset partition and label distribution per subject. (a) Distribution using the original CheXpert annotations. (b) Distribution after applying the Zeros strategy to uncertain labels. (c) Distribution after applying the Ones strategy to uncertain labels.

Table 14
Pathology case analysis for Testing Zeros strategy.

Pathology	Positive	Absent
Pleural Effusion	17186–38.46%	27497–61.54%
Edema	10350–23.16%	34333–76.84%
Consolidation	2931–6.56%	41752–93.44%
Cardiomegaly	5381–12.04%	39302–87.96%
Atelectasis	6614–14.80%	38069–85.20%

In this strategy, the data demonstrates that pleural effusion remains the predominant condition, exhibiting the highest proportion of positive instances with 43.82% in the training set and 43.62% in the testing set. This is followed closely by edema, which accounts for 29.26% and 28.94% of positive cases for training and testing, respectively. Conversely, under this scenario, the lowest prevalence shifts to cardiomegaly, representing 15.74% and 15.57% of positive cases in the training and testing sets, respectively.

Table 15
Pathology case analysis for Training Ones strategy.

Pathology	Positive	Absent
Pleural Effusion	78323–43.82%	100408–56.18%
Edema	52297–29.26%	126434–70.74%
Consolidation	34150–19.11%	144581–80.89%
Cardiomegaly	28130–15.74%	150601–84.26%
Atelectasis	53768–30.08%	124963–69.92%

The validation dataset does not require any strategy since the label values only include present (1) and absent (0). Table 17 shows the number of records for each pathology in this dataset. As the data demonstrates, the pathology with the largest number of positive cases in the validation dataset is now atelectasis, with 80 cases representing 34.19%. It is followed by cardiomegaly with 68 cases (29.06%), and subsequently pleural effusion with 67 positive cases

Table 16
Pathology case analysis for Testing Ones strategy.

Pathology	Positive	Absent
Pleural Effusion	19492–43.62%	25191–56.38%
Edema	12933–28.94%	31750–71.06%
Consolidation	8375–18.74%	36308–81.26%
Cardiomegaly	6957–15.57%	37726–84.43%
Atelectasis	13347–29.87%	31336–70.13%

(28.63%). Next, edema accounts for 45 positive cases (19.23%), and finally, consolidation presents the fewest positive instances with 33 cases (14.10%).

Table 17
Pathology case analysis for Validation

Pathology	Positive	Absent
Pleural Effusion	67–28.63%	167–71.37%
Edema	45–19.23%	189–80.77%
Consolidation	33–14.10%	201–85.90%
Cardiomegaly	68–29.06%	166–70.94%
Atelectasis	80–34.19%	154–65.81%

.Model Architectures and Training Strategies for Knowledge Distillation in Medical Image Classification The term deep learning model architecture refers to the structural design and internal organization of a deep neural network. It specifies how the different layers, whether they're convolutional, recurrent, or fully connected, are arranged and connected. It also defines the number of layers, the activation functions, the regularization mechanisms, and the optimization methods used. This architecture determines the flow of data from the model's input to its output, which in turn defines its ability to learn complex data representations. In deep learning, architecture is crucial for the success of the learning process. It influences the model's ability to capture intricate patterns, its computational efficiency, and its capacity to generalize to new data.

Teacher Model

For the teacher model architecture, the DenseNet-121 was selected. This is a deep convo-

lutional neural network (CNN) architecture from the DenseNet (Densely Connected Convolutional Networks) family, proposed by Gao Huang in 2016 [29]. It was developed as an evolution of ResNet to improve gradient flow and feature reuse. DenseNet is well-known for its efficiency and high accuracy in image classification tasks. Its name comes from its design, where each layer is directly connected to every other layer in a feed-forward manner.

Key Features of DenseNet Architectures DenseNets have two key characteristics that set them apart from other CNN architectures:

- *Dense Blocks*: Each layer in a dense block is connected to all preceding layers, and their feature maps are concatenated and passed as input to the next layer. This dense connectivity improves the flow of information and gradients, making the network easier to train and mitigating the vanishing gradient problem.

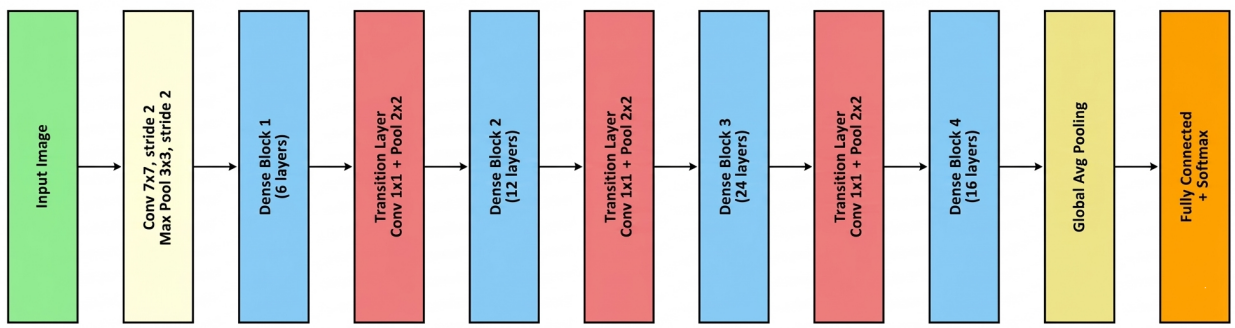
- *Bottleneck Layers*: To manage the rapid growth of feature maps and the resulting computational cost, DenseNets use bottleneck layers. These layers, consisting of a 1x1 convolution, reduce the number of input feature maps before a 3x3 convolution, thereby reducing the number of parameters without sacrificing the network's learning capacity. The architecture of DenseNet-121 can be visualized in Figure 19.

DenseNet-121 is a deep convolutional network characterized by dense connections between layers. Within each dense block, every layer receives the concatenated feature maps from all previous layers as input. The network is composed of:

- *Dense Blocks*: Each block contains multiple layers, with each layer typically including batch normalization, a ReLU activation, and a 3x3 convolution. The concatenation of all previous layers' outputs within a block helps information and gradient flow, facilitating learning and reducing the vanishing gradient problem.

- *Transition Layers*: Located between the dense blocks, these layers use a 1x1 convolution followed by an average pooling operation. Their purpose is to reduce the spatial dimensions and the number of feature channels, thereby maintaining computational efficiency [29].

Figure 19
DenseNet-121 Conceptual Architecture



As its name suggests, DenseNet-121 has 121 layers, including convolutions, dense blocks, and transition layers. The network concludes with a final normalization layer and a fully connected classification layer.

Due to its efficient feature reuse, DenseNet-121 requires fewer parameters than other deep models while achieving excellent results with lower memory consumption. For example, DenseNet-121 has approximately 6.96 million parameters and a disk size of around 27.84 MB, which is significantly more efficient than ResNet-152, which has about 60 million parameters. This makes DenseNet a highly efficient choice for deep learning tasks.

Baseline Student Model

The architecture selected for the Student model was MobileNetV2, which is a lightweight convolutional neural network designed for computational efficiency and suitability for devices with limited resources, while maintaining competitive accuracy in computer vision tasks. MobileNetV2 was developed by Google researchers as an improvement over the original MobileNet. It achieves an excellent balance between size and accuracy, making it ideal for resource-constrained devices.

Its structure is characterized mainly by the use of inverted residual blocks and depthwise separable convolutions. Each block consists of an initial expansion using a 1x1 convolution to increase channel dimensionality, followed by a 3x3 depthwise convolution that processes each channel independently, and a final projection using another 1x1 convolution to reduce dimensionality. In addition, residual connections are incorporated to facilitate the flow of information and

gradients whenever the input and output dimensions match, helping to stabilize and improve the training process [87].

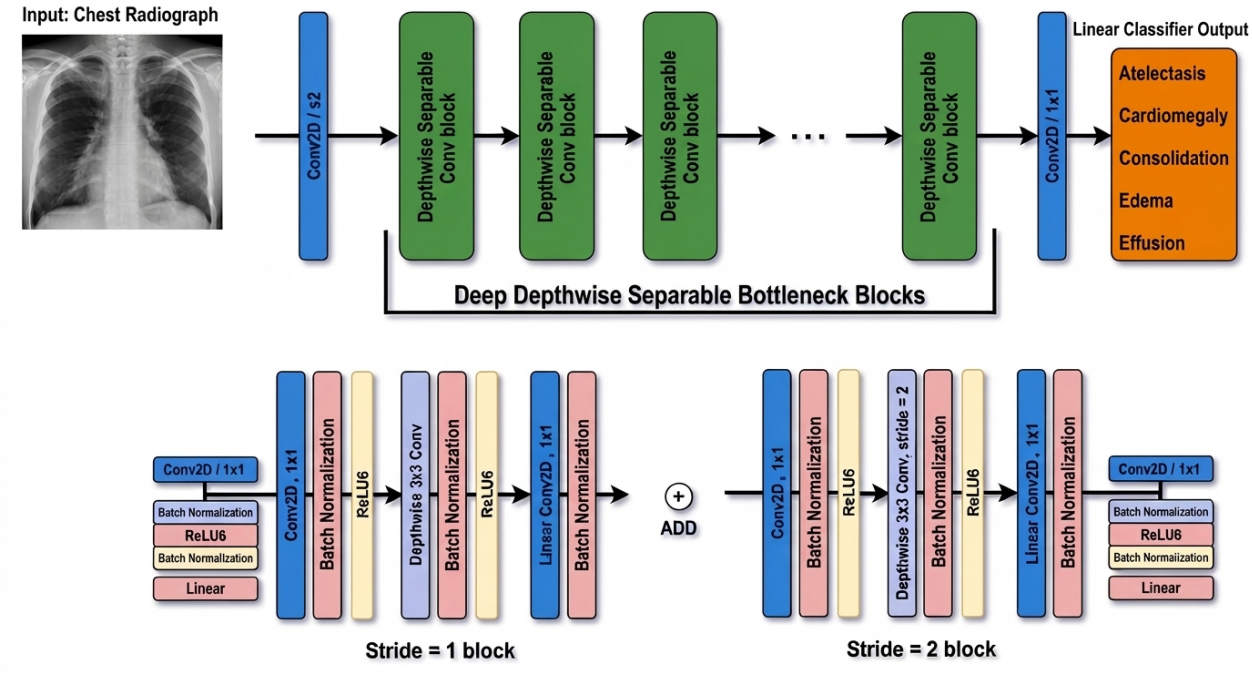
MobileNetV2 contains approximately 2.23 million parameters, making it less than half the size of the DenseNet-121 neural network architecture.

MobileNetV2 replaces the nonlinear activations at the end of compressed blocks with linear activations, thereby preserving important information and preventing the loss of key representations. The network begins with a standard convolution and consists of multiple blocks with varying configurations and depths, ending with a global pooling layer and a fully connected layer for classification. Its modular and parametric design allows the architecture to be adjusted according to precision and resource requirements, resulting in a lightweight yet powerful model for computer vision [88].

The figure 20 shows a general diagram of the MobileNetV2 architecture used as a student model.

Figure 20

MobileNetV2 architecture



.Loss Function The loss function used to train the teacher model and student model is BCEWithLogitsLoss, a commonly used loss function for binary or multilabel classification. It combines the sigmoid function (which converts the model's logits into probabilities) and the binary cross-entropy loss (BCELoss) into a single, numerically stable, and more efficient operation [89].

This loss function is particularly useful because it can be combined with a weight-balancing mechanism. This feature allows different weights to be assigned to each class, which is crucial when dealing with an imbalanced dataset where the number of positive and negative examples differs significantly.

To achieve this, the `pos_weight` argument is added to the BCEWithLogitsLoss function. This argument weights positive examples more heavily in the overall loss, helping the model learn more effectively from the underrepresented class.

For a single example i, the loss function using BCEWithLogitsLoss with a positive class weight p is mathematically expressed as:

$$\mathcal{L}_i = -\,p\,t_i\,\log\big(\sigma(z_i)\big) - \big(1 - t_i\big)\,\log\big(1 - \sigma(z_i)\big) \tag{3.1}$$

where

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.2}$$

is the usual sigmoid function.

Explanation of the Formula

- L_i: This is the calculated loss for the individual example i.
- z_i: This is the logit, or the raw, unnormalized output, from the model for example i.
- $t_i \in \{0, 1\}$: This is the true label for example i. $t_i = 1$ for a positive example, and $t_i = 0$ for a negative example.
- p: This is the weight assigned to the positive class. If $p > 1$, it means you want to give more importance to positive examples.

- $\sigma(z_i)$: This represents the sigmoid function, which converts the logit z_i into a probability—a value between 0 and 1.

The full `BCEWithLogitsLoss` is applied to a batch of data by summing the losses for each example in the batch, and then typically averaging the result. The p in the first half of the equation ($p \cdot t_i$) ensures that only positive examples are affected by the weight, which helps to mitigate the effects of class imbalance in the dataset[89].

When weight balancing is added using the p term (pos_weight), the contribution of the positive samples is scaled to counteract their relative scarcity. This balances the loss function and allows for control over the trade-off between recall and precision in imbalanced problems. For positive classes, the formula is modified to:

$$\ell_i = (1 - t_i) \cdot \log\left(1 + e^{z_i}\right) + t_i \cdot \left(\log\left(1 + e^{-z_i}\right) \cdot p\right) \tag{3.3}$$

The loss function used to train the student model in Knowledge Distillation is based on the classic scheme from Hinton [5]. It combines a hard loss (with true labels) and a distillation loss between "softened" outputs from the teacher and student models, using a temperature T. These two losses are weighted by a parameter α (and the soft term is scaled by T^2 [90].
For each class $c = 1, \ldots, C$ and sample i:

- *Logits* of teacher and student: z_{ic}^t, z_{ic}^s.
- *Labels:* $t_{ic} \in \{0, 1\}$.
- *Temperature:* $T > 0$.

1. *Hard loss (BCE with logits)*

$$\mathcal{L}_{\text{hard}} = \frac{1}{C} \sum_{c=1}^{C} \text{BCEWithLogits}\left(z_{ic}^s,\, t_{ic}\right) \tag{3.4}$$

2. *Softening with temperature-scaled sigmoid and per-class Bernoulli Kullback-Leibler (KL)*

$$q_{ic}^t = \sigma\left(\frac{z_{ic}^t}{T}\right), \qquad q_{ic}^s = \sigma\left(\frac{z_{ic}^s}{T}\right) \tag{3.5}$$

$$\mathrm{KL}_{\mathrm{Bern}}\left(q_{i,c}^t \,\|\, q_{i,c}^s\right) = \mathbb{E}_i\left[q_{ic}^t \log\frac{q_{ic}^t}{q_{ic}^s} + \left(1 - q_{ic}^t\right)\log\frac{1 - q_{ic}^t}{1 - q_{ic}^s}\right]. \tag{3.6}$$

and then

$$\mathcal{L}_{\mathrm{soft}} = T^2 \cdot \frac{1}{C}\sum_{c=1}^{C} \mathrm{KL}_{\mathrm{Bern}}\left(q_{i,c}^t \,\|\, q_{i,c}^s\right). \tag{3.7}$$

(The form of $\mathrm{KL}_{\mathrm{Bern}}$ *can be found, e.g., in standard derivations for the Bernoulli distribution.)*

3. *Combination (Hinton)*

$$\mathcal{L} = \alpha\, \mathcal{L}_{\mathrm{hard}} + (1 - \alpha)\; \mathcal{L}_{\mathrm{soft}}. \tag{3.8}$$

Where α is a balancing factor between the two losses.

.Training Configuration and Weight Balancing In training the different models, the Early Stopping, Optimizer, and Scheduler strategies were applied, which are described below:

Early stopping: this technique was used during model training. Its purpose is to halt training before completing all planned epochs, once it is detected that the model's performance on the validation set ceases to improve or starts to degrade. Early Stopping is a regularization strategy that prevents model overfitting by stopping training at the optimal moment. Additionally, it saves time and computational resources by avoiding unnecessary epochs in which no further improvement is achieved.

The main objective is to ensure that the model does not learn noise or dataset-specific patterns that do not generalize well to new data. During training, the AUC-ROC obtained on the validation set is monitored, which is critical in medical classification models. The patience value was set to 8 for the Teacher model; that is, if the model does not improve its AUC-ROC score for eight consecutive cycles, training is stopped, and the model weights with the best AUC-ROC

are selected. A patience value of 5 was selected for the Student and Distilled Student models (MobileNetV2, GenENAS, EfficientNet-B0) because they have a lighter structure and converge more rapidly; thus, a lower patience helps avoid overfitting, as these models stabilize and reach optimal performance earlier than larger models.

Optimizer: an algorithm that iteratively adjusts the model's parameters to minimize the loss function and improve model performance. The AdamW optimizer was used for the Teacher model and for the student models and distilled student models (MobileNetV2, GenENAS, EfficientNet-B0) training. AdamW is an improved version of Adam that uses weight decay as regularization. AdamW decouples the weight decay term from gradient calculations, applying the penalty directly to the weights, separately from the gradient. This technique is employed in Deep Learning to prevent overfitting during training and to enhance model generalization [91]. Scheduler: the CosineAnnealingLR scheduler was used for the teacher model, this scheduler adjusts the learning rate using a cosine annealing schedule without restarts. It gradually decreases the learning rate from a maximum value to a minimum value over a predefined number of epochs or iterations. The ReduceLROnPlateau scheduler was applied for the student and distilled student models, which automatically adjusts the learning rate by monitoring the AUC-ROC metric [92]. The learning rate is reduced by a factor called the scheduler factor, which was set to 0.5 for training the three models. If no improvement in the AUC-ROC is detected during the scheduler patience, which was set to 2 consecutive epochs, the learning rate is reduced further. This mechanism promotes better convergence and improves training efficiency by adaptively adjusting the learning rate. The Teacher model training begins with a learning rate of 0.0001, and the student and distilled student models start at 0.0003.

Prior to training, all chest X-ray images were processed through a preprocessing pipeline implemented within the dataset loader. Each image was resized to a fixed resolution of 320 × 320 pixels, ensuring consistent input dimensions across all evaluated architectures. To improve model generalization and reduce overfitting, data augmentation techniques were applied to the training set, including random horizontal flipping and random rotations of up to 15 degrees.

Subsequently, images were converted to tensors and normalized using a mean of 0.533 and a standard deviation of 0.034, calculated from the training dataset. Since the original chest radiographs are grayscale images, the single-channel input was replicated across three channels to match the input requirements of convolutional neural network architectures pretrained on ImageNet. For validation and testing, only resizing, tensor conversion, normalization, and channel expansion were applied, ensuring a consistent evaluation protocol without introducing artificial variations.

For training, the configuration used in terms of hyperparameters differs in each model, whether it is the teacher, the student, and the distilled student, for which the following values were established:

- **General hyperparameters:**
  - Batch size: 32
  - Number of epochs: 50
- **Teacher (DenseNet-121:**
  - Learning Rate: 0.0001
  - Optimizer: AdamW
  - Weight Decay: 0.001
  - Scheduler: CosineAnnealingLR
  - Scheduler Factor: 0.5
  - Scheduler patience: 2
  - Patience: 8
- **Student (MobileNetV2, GenENAS, EfficientNet-B0):**
  - Learning Rate: 0.0003
  - Optimizer: AdamW
  - Weight Decay: 0.0001
  - Scheduler: ReduceLROnPlateau

- Scheduler Factor: 0.5
- Scheduler patience: 2
- Patience: 5

- **Distilled Student (MobileNetV2, GenENAS, EfficientNet-B0):**
  - Learning Rate: 0.0003
  - Optimizer: AdamW
  - Weight Decay: 0.0001
  - Scheduler: ReduceLROnPlateau
  - Scheduler Factor: 0.5
  - Scheduler patience: 2
  - Patience: 5
  - Alpha: 0.5
  - Temperature: 6.0

In the training of deep neural networks, the loss function is typically computed by averaging the differences between the model's predictions and the true labels. However, when the training dataset is imbalanced, the model tends to favor the majority classes, leading to poor performance on the minority ones [93]. To mitigate this issue, a balancing strategy was used to calculate the error, adjusting the contribution of each class by assigning weights proportional to the class imbalance.

Each class is assigned a weight w_i, calculated as follows:

$$w_i = \frac{n_i}{p_i}$$

Where:

- n_i: number of *negative* samples in class
- p_i: number of *positive* samples in class

This implies that the fewer samples the pathology has, the greater its weight and, therefore, its influence on the error calculation. The weight values shown in Table 18 for the zeros strategy and Table 19 for the Ones strategy were applied for each of the pathologies analyzed.

The contrast between Tables 18 and 19 reflects directly the prevalence shifts of Tables 13 and 15. Under the Zeros strategy, Consolidation is the rarest positive class (6.63% prevalence in training) and consequently receives the largest weight (14.019), nearly twice the weight of the next most imbalanced class, Cardiomegaly (7.300, with 12.10% prevalence). Under the Ones strategy this ordering changes: because the Ones strategy reassigns uncertain (-1) labels as positives and Consolidation has a large fraction of uncertain annotations (Table 4), its prevalence almost triples (6.63% $\rightarrow$ 19.11%), which causes its weight to drop from 14.019 to 4.227 — a 70% reduction. Cardiomegaly, whose prevalence rises only marginally (12.10% $\rightarrow$ 15.74%), becomes the rarest of the five classes under Ones and consequently takes over as the most weighted pathology (5.393). Pleural Effusion, the most prevalent pathology under both strategies, remains the only class with a weight close to unity (1.580 in Zeros, 1.274 in Ones).

Table 18
Weights for each pathology in Zeros strategy.

Pathology	Weight
Atelectasis	5.682 210
Cardiomegaly	7.300 089
Consolidation	14.019 138
Edema	3.257 244
Pleural Effusion	1.580 300

Table 19
Weights for each pathology in Ones strategy.

Pathology	Weight
Atelectasis	2.319 379
Cardiomegaly	5.393 239
Consolidation	4.227 341
Edema	2.411 762
Pleural Effusion	1.274 000

.Testbed Infrastructure

- **Processor:** Intel Core i9-14900K
  - Generation: Raptor Lake Refresh (14th generation)
  - Cores and Threads: 24 cores (8 Performance cores + 16 efficient cores) and 32 threads.
  - Frequency: Up to 6.0 GHz turbo boost, making it one of the fastest CPUs in single-thread performance currently.
  - Advantages: Exceptional single-core performance—ideal for simulations, compiles, and software that does not scale well with parallelism. Excellent multitasking and mixed workload handling due to the hybrid architecture with Efficient cores. Optimized for deep-learning workflows such as data preprocessing, virtualization, model compiling, and coordinating GPU tasks.
- **RAM:** 128 GB
  - Capacity: well above common standards even in workstations.
  - Applications: training deep-learning models with large datasets requiring in-memory dataloaders; virtualization; 8K video editing; scientific simulations; CAD/CAE.
  - Advantage: prevents bottlenecks due to disk swapping—critical in HPC/AI pipelines.
- **GPU:** NVIDIA RTX 4090 (24 GB GDDR6X)
  - Architecture: Ada Lovelace.
  - CUDA Cores: 16,384.
  - Tensor Cores (4^{th} gen): 512.
  - RT Cores (3^{rd} gen): 128.
  - Memory: 24 GB VRAM.
  - FP16/FP32 performance: ~82 TFLOPS (FP32); up to 330+ effective TFLOPS (FP16 with Tensor Cores).
  - Key advantage: enables training of fairly large vision/NLP models on a single GPU; suitable for PyTorch, TensorFlow, JAX, and for rendering in Blender or Unreal Engine.

3.2.2 Experiments and results

The analysis evaluates the effectiveness of KD for efficient and sustainable multi-label medical image classification on the CheXpert dataset across five target pathologies. Experiments are conducted under both the *Zeros* and *Ones* uncertainty-handling strategies and include the teacher model, the two undistilled student models, and the two distilled student models. The evaluation reports AUC–ROC (mean and standard deviation over multiple runs), F1-score, model footprint, and computational cost in terms of execution time, energy consumption, and CO_2 emissions.

.Model Complexity and Parameter Comparison Table 20 highlights substantial differences in the architectural complexity of the evaluated models. As the teacher network, DenseNet-121 exhibits the largest computational footprint, comprising 6,958,981 parameters, a memory requirement of 26.87 MB, and 5.84 GFLOPs. These characteristics make it considerably more resource-intensive than the student architectures considered in this study. At the opposite end of the spectrum, GenENAS emerges as the most compact architecture, with only 1,909,541 parameters, a memory footprint of 7.28 MB, and 3.82 GFLOPs, followed closely by MobileNetV2 with 2,230,277 parameters and 8.65 MB of memory usage. EfficientNet-B0 occupies an intermediate position among the student models but remains substantially more demanding than both GenENAS and MobileNetV2, containing 4,013,953 parameters and requiring 16.06 MB of memory, approximately twice the memory footprint of the latter architectures.

From the perspective of the static footprint (Table 20), *GenENAS* stands out as the architecture with the lowest structural complexity among all evaluated models. With only 1,909,541 parameters, it contains approximately 27.4% of the parameters of DenseNet-121 and 85.6% of those of MobileNetV2, making it the most compact architecture in the study. Its memory footprint is 7.28 MB, the lowest among all evaluated models, while its computational complexity reaches 3.82 GFLOPs. Although this value remains below that of the teacher model, it is substantially higher than the 1.22 GFLOPs required by MobileNetV2. Consequently, GenENAS requires more

than three times the number of floating-point operations performed by MobileNetV2 to process the same input. This observation suggests that, despite its reduced parameter count, the architecture exhibits a higher computational density per parameter, indicating that parameter efficiency does not necessarily translate into proportional reductions in computational workload.

It is important to note that the knowledge distillation process does not alter the structural configuration of the student network. Consequently, the number of parameters, memory consumption, and computational complexity measured in GFLOPs remain unchanged between each baseline student model and its distilled counterpart. Therefore, the benefits of knowledge distillation stem from improvements in predictive performance and knowledge transfer rather than from reductions in architectural complexity or computational requirements. This finding is particularly significant because it highlights that parameter count alone does not provide a complete characterization of a model's computational efficiency. Despite being the most compact architecture in terms of learnable parameters, *GenENAS* does not achieve the lowest computational complexity when measured in GFLOPs. This discrepancy suggests that the architecture incorporates connections or computational blocks with a relatively high operational cost, leading to increased processing requirements per parameter. Consequently, the results emphasize the importance of jointly considering parameter count and computational complexity when assessing the suitability of lightweight neural networks for resource-constrained deployment scenarios.

In Deep Learning, there is a well-known inverse relationship between model compactness and raw learning capacity; however, advanced techniques such as knowledge distillation help mitigate this limitation. This process transfers structured information from the teacher to the student model, allowing the latter to retain high predictive performance despite its reduced size and complexity.

Consequently, the distilled student models emerges as a compact, efficient, and energy-aware alternative that effectively balances parameter and memory reduction with strong generalization. These findings support the concept of Green AI, promoting the development of lightweight, interpretable, and environmentally sustainable deep learning models.

Table 20
Static Model Footprint.

Model	Parameters	Memory FootPrint [MB]	GFlops
DenseNet-121 (Teacher)	6 958 981	26.87	5.84
MobileNetV2 (Student)	2 230 277	8.65	1.22
Distilled MobileNetV2	2 230 277	8.65	1.22
GenENAS (Student)	1 909 541	7.28	3.82
Distilled GenENAS	1 909 541	7.28	3.82
EfficientNet-B0 (Student)	4 013 953	16.06	1.57
Distilled EfficientNet-B0	4 013 953	16.06	1.57

.Energy consumption and Computational Sustainability Analysis Table 21 provides a comparative summary of the computational and environmental costs associated with training the different models involved in the knowledge distillation pipelines, evaluated under two weight initialization strategies: *Zeros* and *Ones*. The reported metrics include total training time (in seconds), number of training epochs, energy consumption (in kilojoules), and carbon dioxide equivalent emissions ($kgCO_2eq$). Together, these indicators provide a comprehensive assessment of the computational requirements and environmental impact of each training configuration, enabling a detailed comparison of the trade-offs between predictive performance, computational efficiency, and sustainability.

Under the *Zeros* strategy, the teacher model completed training in 4,114.22 seconds over 9 epochs, consuming 1,412.38 kJ of energy and generating 0.1018 $kgCO_2eq$ emissions. When the *Ones* strategy was employed, the training process became considerably more demanding, requiring 6,776.31 seconds and 15 epochs, with energy consumption and carbon emissions increasing to 2,324.08 kJ and 0.1675 $kgCO_2eq$, respectively. This represents an increase of approximately 64.7% in training time and 64.6% in energy consumption. The observed behavior suggests that treating uncertain labels as positive examples introduces additional optimization complexity, resulting in slower convergence and a higher computational and environmental cost for training the teacher model.

Regarding the evaluated knowledge distillation pipelines—*MobileNetV2*, *GenENAS*, and *EfficientNet-B0*—the results demonstrate notable differences in computational efficiency and envi-

ronmental impact. The *MobileNetV2*-based pipeline consistently achieved the lowest training cost across both uncertainty-handling strategies. Under the *Zeros* strategy, the cumulative training time amounted to 4,125.43 seconds, requiring 1,338.20 kJ of energy and generating 0.0964 $kgCO_2eq$ emissions. Comparable results were obtained under the *Ones* strategy, with a total training time of 4,122.91 seconds, energy consumption of 1,334.35 kJ, and emissions of 0.0961 $kgCO_2eq$. The near-identical values observed across both configurations indicate that the computational behavior of *MobileNetV2* is largely insensitive to the treatment of uncertain labels, suggesting a high degree of optimization stability and predictable resource requirements.

This characteristic makes *MobileNetV2* particularly attractive for resource-constrained environments where computational efficiency and environmental sustainability are important considerations.

One of the most noteworthy findings obtained by analyzing *GenENAS* in conjunction with Table 21 is the apparent paradox between its architectural compactness and its high training cost. Despite being the model with the smallest number of parameters and the lowest memory footprint, the *GenENAS*-based knowledge distillation pipeline consistently exhibits the highest computational cost in terms of training time and energy consumption among the three evaluated pipelines.

Under the *Zeros* strategy, the cumulative training time of the *GenENAS* pipeline reaches 5,762.34 seconds, compared to 4,125.43 seconds for *MobileNetV2* and 4,886.31 seconds for *EfficientNet-B0*, representing increases of 39.7% and 17.9%, respectively. In terms of energy consumption, the accumulated cost of 2,106.23 kJ exceeds that of *MobileNetV2* (1,338.20 kJ) by 57.4% and that of *EfficientNet-B0* (1,646.71 kJ) by 27.9%. Carbon dioxide equivalent emissions follow the same pattern, with *GenENAS* generating 0.1519 $kgCO_2eq$, compared with 0.0964 $kgCO_2eq$ for *MobileNetV2* and 0.1187 $kgCO_2eq$ for *EfficientNet-B0*.

Under the *Ones* strategy, this trend remains consistent. The *GenENAS* pipeline accumulates 5,667.53 seconds of training time, consumes 2,061.37 kJ of energy, and produces 0.1486 $kgCO_2eq$ emissions, remaining the most computationally expensive configuration among the evaluated student architectures despite a slight reduction relative to the *Zeros* strategy. It is also noteworthy

Table 21
Summary of training cost under the Zeros and Ones strategies. The teacher (DenseNet-121) is trained once and shared by all distillation pipelines.

Strategy	Model	Total Time [seg]	Epochs	Energy [kJ]	Emissions [$KgCO_2eq$]
Zeros	DenseNet-121 (Teacher, shared)	4114.22	9	1412.38	0.1018
	MobileNetV2 (Student)	1440.04	6	406.74	0.0293
	Distilled MobileNetV2	2685.39	6	931.46	0.0671
	Subtotal (MobileNet pipeline)	**4125.43**	**–**	**1338.20**	**0.0964**
	GenENAS (Student)	2571.96	8	888.67	0.0641
	Distilled GenENAS	3190.38	6	1217.56	0.0878
	Subtotal (GenENAS pipeline)	**5762.34**	**–**	**2106.23**	**0.1519**
	EfficientNet-B0 (Student)	1807.22	6	553.99	0.0399
	Distilled EfficientNet-B0	3079.09	6	1092.72	0.0788
	Subtotal (EfficientNet-B0 pipeline)	**4886.31**	**–**	**1646.71**	**0.1187**
Ones	DenseNet-121 (Teacher, shared)	6776.31	15	2324.08	0.1675
	MobileNetV2 (Student)	1471.58	6	412.60	0.0297
	Distilled MobileNetV2	2651.33	6	921.75	0.0664
	Subtotal (MobileNet pipeline)	**4122.91**	**–**	**1334.35**	**0.0961**
	GenENAS (Student)	1933.57	6	667.12	0.0481
	Distilled GenENAS	3733.96	7	1394.25	0.1005
	Subtotal (GenENAS pipeline)	**5667.53**	–	**2061.37**	**0.1486**
	EfficientNet-B0 (Student)	1821.34	6	543.66	0.0392
	Distilled EfficientNet-B0	3594.59	7	1252.63	0.0903
	Subtotal (EfficientNet-B0 pipeline)	**5415.93**	**–**	**1796.29**	**0.1295**

that the distilled *GenENAS* model trained under the *Ones* strategy required one additional training epoch (seven epochs compared with six under the *Zeros* strategy). This behavior suggests that the optimization dynamics induced by the *Ones* initialization strategy differ for this particular architecture, potentially affecting convergence patterns and resource utilization during the knowledge distillation process.

.Evaluation Metrics and Diagnostic Performance Table 22 presents the evaluation metrics in terms of AUC-ROC, standard deviation, and F1-score for each model under the two uncertainty-label initialization strategies. Overall, the teacher model, DenseNet-121, achieved competitive performance under both conditions, obtaining AUC-ROC values of 0.884 and 0.891 for the *Zeros* and *Ones* strategies, respectively, with corresponding F1-scores of 0.673 and 0.664. Nevertheless, the performance gap between the teacher and the distilled models remains remarkably small, supporting the central premise of knowledge distillation: predictive capability can be effectively transferred from a large teacher network to substantially lighter student architectures.

Under the *Zeros* strategy, Distilled MobileNetV2 achieved an AUC-ROC of 0.880 and an F1-score of 0.630, yielding performance levels very close to those of the teacher despite containing less than one-third of its parameters. Distilled GenENAS attained an AUC-ROC of 0.877 and an F1-score of 0.653, surpassing the non-distilled MobileNetV2 student model in terms of F1-score (0.629) in the Zeros strategy.

Under the *Ones* strategy, most distilled models exhibited improved predictive performance. Distilled MobileNetV2 achieved its best overall results, reaching an AUC-ROC of 0.884 and an F1-score of 0.670, effectively matching the teacher model in terms of AUC-ROC while surpassing its own performance under the *Zeros* configuration. Distilled GenENAS, by contrast, was the exception to this trend: its AUC-ROC decreased from 0.877 to 0.867 and its F1-score declined from 0.653 to 0.641, indicating that for this particular architecture, treating uncertain labels as positive examples did not facilitate more effective knowledge transfer. Distilled EfficientNet-B0, in line with the majority, improved its AUC-ROC from 0.871 to 0.879, while achieving an F1-score

of 0.644.

Overall, the results suggest that the *Ones* strategy favors more effective knowledge transfer for most of the distilled models, particularly in terms of AUC-ROC performance. The behavior of Distilled GenENAS, however, shows that this effect is architecture-dependent: for this model both metrics declined under the *Ones* strategy, emphasizing that ranking-based and threshold-dependent measures such as the F1-score must be evaluated jointly when assessing distilled models.

Table 22
Evaluation Metrics

Strategy	Model	AUC-ROC	SD	F1-Score
Zeros	DenseNet-121	0.884	$1.3\text{x}10^{-3}$	0.673
	MobileNetV2 (Student)	0.877	$7\text{x}10^{-4}$	0.629
	Distilled MobileNetV2	0.880	$1.1\text{x}10^{-3}$	0.630
	GenENAS (Student)	0.846	$1.1\text{x}10^{-3}$	0.576
	Distilled GenENAS	0.877	$1.2\text{x}10^{-3}$	0.653
	EfficientNet-B0 (Student)	0.851	$8\text{x}10^{-4}$	0.603
	Distilled EfficientNet-B0	0.871	$7\text{x}10^{-4}$	0.639
Ones	DenseNet-121	0.891	$1.4\text{x}10^{-3}$	0.664
	MobileNetV2	0.876	$6\text{x}10^{-4}$	0.651
	Distilled MobileNetV2	0.884	$1\text{x}10^{-3}$	0.670
	GenENAS (Student)	0.854	$1.1\text{x}10^{-3}$	0.629
	Distilled GenENAS	0.867	$1.2\text{x}10^{-3}$	0.641
	EfficientNet-B0 (Student)	0.859	$1\text{x}10^{-3}$	0.614
	Distilled EfficientNet-B0	0.879	$9\text{x}10^{-4}$	0.644

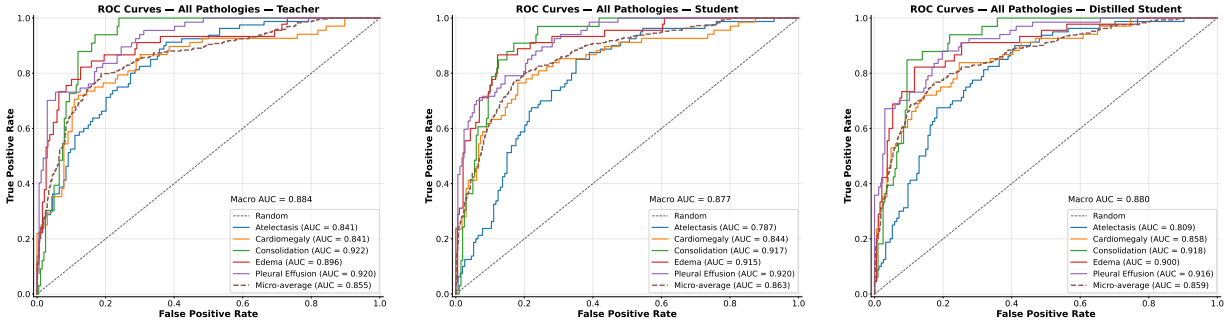
Figure 21 shows the ROC curves per pathology obtained under the Zeros and Ones strategies for the teacher (DenseNet-121), the student (MobileNetV2), and the distilled student. The three families of curves are closely grouped, with the non-distilled student lying between the teacher and the distilled student: in terms of the macro AUC reported in the figure, the distilled student reaches 0.880 under Zeros and 0.884 under Ones, approaching the teacher (0.884 and 0.891, respectively). Across all models, the per-pathology ordering is stable: Pleural Effusion and Edema yield the highest curves, while Cardiomegaly remains the most difficult target in the Ones strategy for the student models, which differs from what the Zeros strategy shows, as the pathology with the lowest

AUC value is atelectasis for both student models (distilled and non-distilled).

Figure 21

ROC Curves for each pathology obtained with the Teacher (DenseNet-121, Student (MobileNetV2) and Distilled Student Models (Distilled MobileNetV2). (a) ROC Curves using the Zeros strategy. (b) ROC Curves using the Ones strategy.

(a) DenseNet-121 / MobileNetV2 - ROC Curves Zeros strategy



(b) DenseNet-121 / MobileNetV2 - ROC Curves Ones strategy

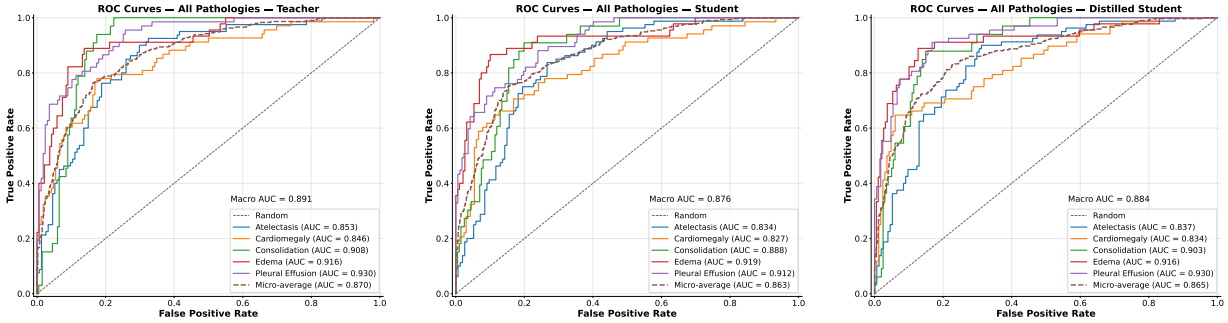
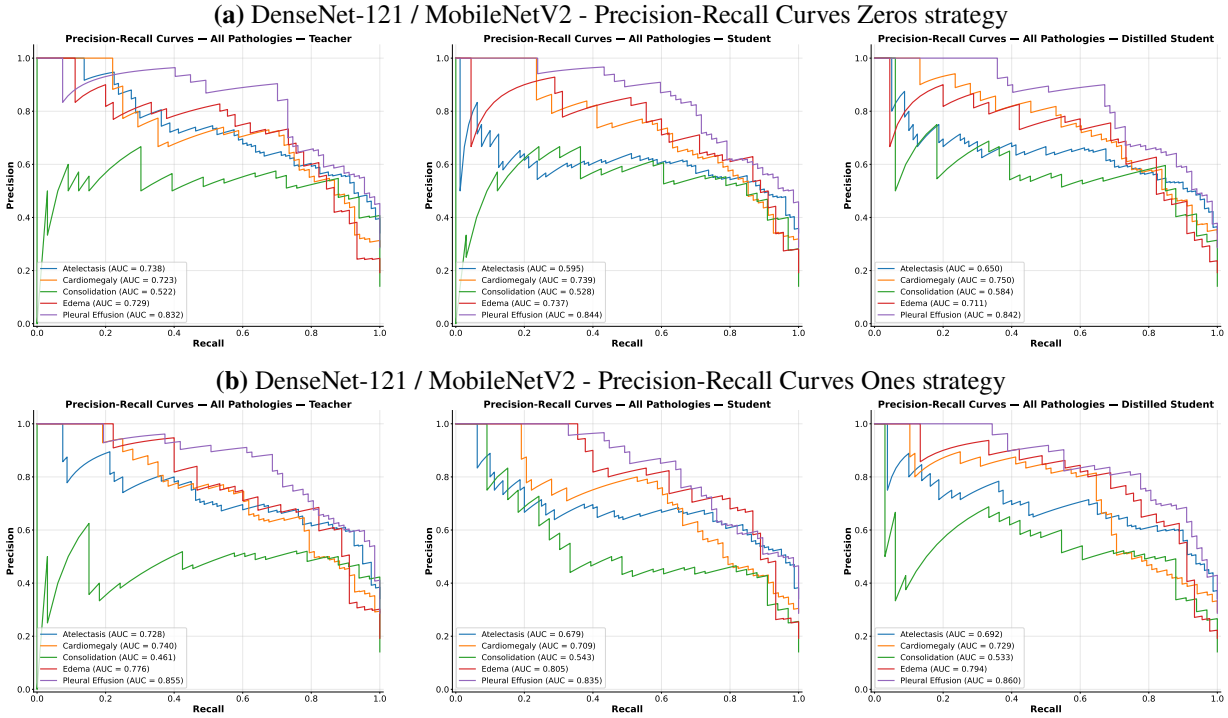


Figure 22 presents the corresponding precision–recall curves, which are more sensitive to class imbalance than the ROC analysis. The precision–recall trade-off confirms the same hierarchy among models and shows that the Ones strategy shifts the curves of the distilled student upward for the minority pathologies, which explains its higher F1-score in Table 22 (0.670 versus 0.630 under Zeros). The precision-recall curve shows the trade-off between precision and recall for different threshold values. Precision is the proportion of true positives among all positive predictions made by the model, and recall is the proportion of true positives among all actual positive instances in the dataset. This curve is used to compare classifier performance, especially when class imbalance is severe. The X-axis corresponds to recall, and the Y-axis to precision. The area under the curve summarizes the model's performance; the higher it is, the better the balance between precision and recall.

Figure 22

Precision-Recall Curves for each pathology obtained with the Teacher (DenseNet-121, Student (MobileNetV2) and Distilled Student Models (Distilled MobileNetV2). (a) Precision-Recall Curves using the Zeros strategy. (b) Precision-Recall Curves using the Ones strategy.



The effect of distillation on the architecture discovered in Stage 1 is illustrated in Figures 23 and 24, which compare the ROC and precision–recall curves of the GenENAS-22121211 student before and after distillation, under both labeling strategies. The improvement is clearly visible in the ROC analysis (Figure 23): the curves of the distilled model dominate those of the non-distilled student over the entire operating range, raising the macro AUC from 0.846 to 0.877 under the Zeros strategy and from 0.854 to 0.867 under Ones. The precision–recall curves (Figure 24) show, however, that these ranking gains are unevenly distributed across pathologies, which is consistent with the decrease of the F1-score from 0.653 in zeros to 0.641 under the Ones strategy reported in Table 22: for the minority classes, the precision of the distilled model degrades at high recall levels. This visual evidence reinforces the conclusion that ranking-based and threshold-dependent metrics must be examined jointly, and confirms at the curve level that the generatively discovered architecture is an effective receptacle for the teacher's knowledge.

Figure 23

ROC Curves for each pathology obtained with Student (GenENAS-22121211) and Distilled Student Models (Distilled GenENAS-22121211). (a) GenENAS - ROC Curves using the Zeros strategy. (b) GenENAS - ROC Curves using the Ones strategy.

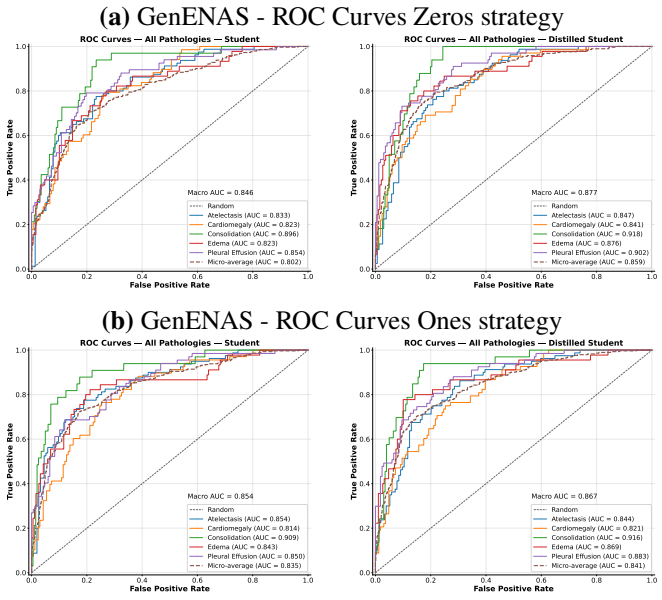
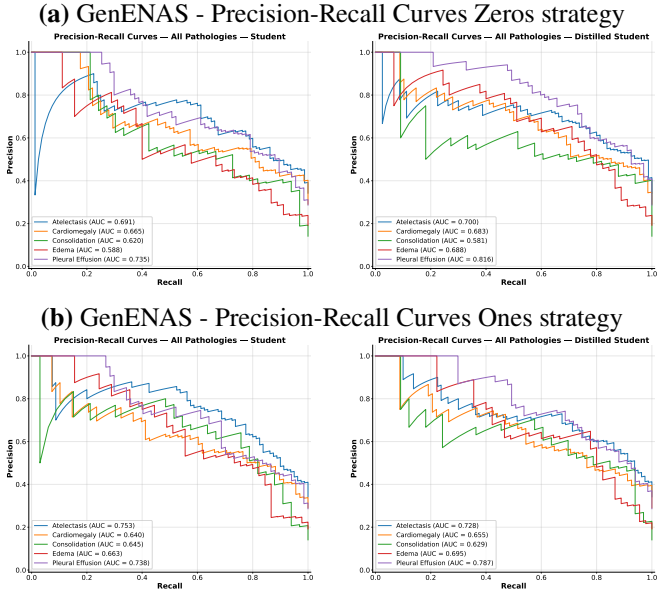


Figure 24

Precision-Recall Curves for each pathology obtained with the Student (GenENAS-22121211) and Distilled Student Models (Distilled GenENAS-22121211). (a) GenENAS - Precision-Recall Curves using the Zeros strategy. (b) GenENAS - Precision-Recall Curves using the Ones strategy.



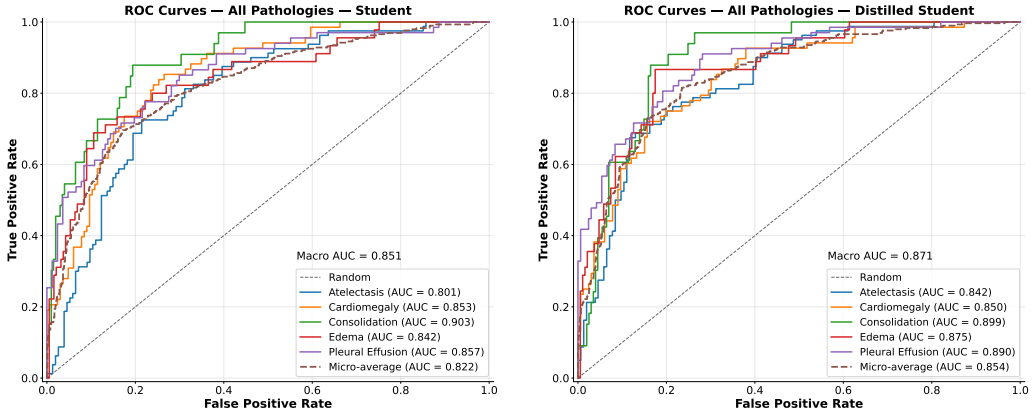
An analogous behavior is observed for the third student architecture in Figures 25 and 26,

which report the ROC and precision–recall curves of EfficientNet-B0 and its distilled version. Distillation lifts the ROC curves of the student toward the teacher's operating region under both strategies, increasing the macro AUC from 0.851 to 0.871 under Zeros and from 0.859 to 0.879 under Ones (Figure 25), while the precision–recall analysis (Figure 26) shows a more uniform improvement across pathologies than in the GenENAS case, in line with the simultaneous gains in AUC-ROC and F1-score reported in Table 22. Taken together, Figures 21 through 26 provide curve-level evidence that knowledge distillation systematically narrows the gap between the compact students and the teacher across the full range of decision thresholds, and not merely at a single operating point.

Figure 25

ROC Curves for each pathology obtained with Student (EfficientNet-B0) and Distilled Student Models (Distilled EfficientNet-B0). (a) EfficientNet-B0 - ROC Curves using the Zeros strategy. (b) EfficientNet-B0 - ROC Curves using the Ones strategy.

(a) EfficientNet-B0 - ROC Curves Zeros strategy



(b) EfficientNet-B0 - ROC Curves Ones strategy

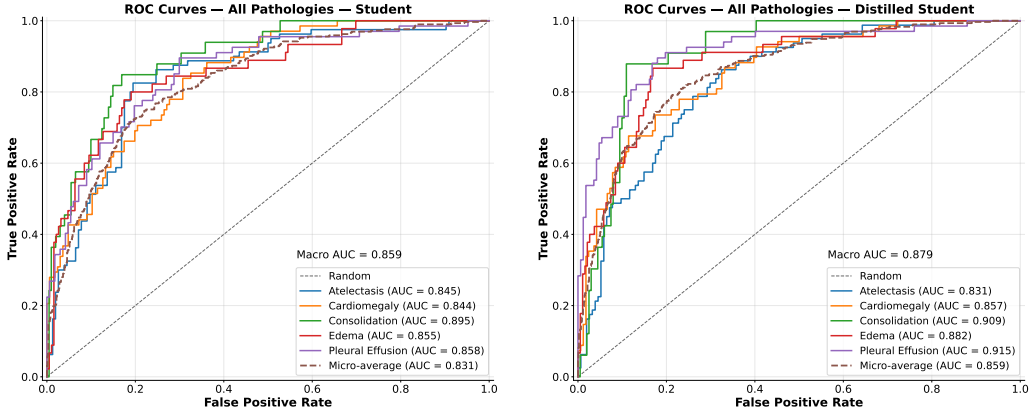
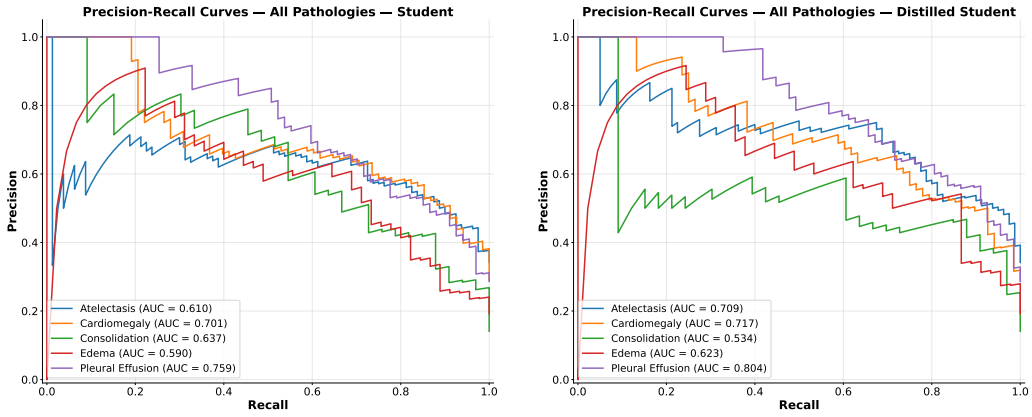


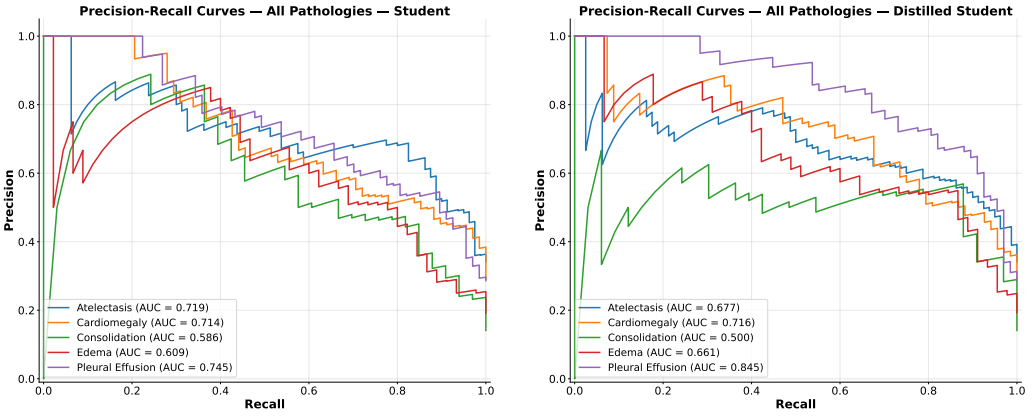
Figure 26

Precision-Recall Curves for each pathology obtained with the Student (EfficientNet-B0) and Distilled Student Models (Distilled EfficientNet-B0). (a) EfficientNet-B0 - Precision-Recall Curves using the Zeros strategy. (b) EfficientNet-B0 - Precision-Recall Curves using the Ones strategy.

(a) Precision-Recall Curves Zeros strategy



(b) Precision-Recall Curves Ones strategy



The experimental evidence presented in this section demonstrates the effectiveness of Knowledge Distillation as a technique for transferring the predictive capability of a robust, high-capacity model to substantially lighter architectures without a meaningful sacrifice in performance. Under the Ones strategy, the distilled MobileNetV2 recovered nearly all of the teacher's diagnostic ability (AUC-ROC of 0.884 versus 0.891, with a higher F1-score of 0.670 versus 0.664) using one third of its parameters, while the distilled versions of GenENAS and EfficientNet-B0 improved upon their non-distilled counterparts by 0.013 and 0.020 points of AUC-ROC, respectively, as shown in Table 22 and in the curve-level analysis of Figures 21–26. The standard deviations across

the five independent runs remained in the order of 10^{-3}–10^{-4} for all configurations, confirming that the distilled models deliver stable and reproducible predictions. These results establish that high diagnostic accuracy, low resource consumption, reduced inference time, and fully local deployment can coexist in a single compact model—precisely the combination required by sensitive medical applications in resource-constrained environments.

3.3 Summary

This chapter developed and validated the second stage of the framework proposed in this thesis: Knowledge Distillation for Multi-Label Classification (KD-MLC). Within the end-to-end pipeline, this stage receives as input the compact architecture discovered generatively in Stage 1 (GenENAS-22121211) together with two reference student architectures (MobileNetV2 and EfficientNet-B0), and endows them with the predictive capability of a high-capacity teacher, DenseNet-121, before they are handed over to Stage 3 for energy certification. To operate in the multi-label medical domain, Hinton's original formulation was adapted by replacing the Softmax output with independent Sigmoid functions per pathology, reformulating the hard loss as a class-weighted binary cross-entropy that compensates the severe class imbalance of CheXpert, redefining the soft loss as a per-class Bernoulli KL divergence with temperature scaling, and evaluating two strategies for the dataset's uncertain labels: Zeros (uncertainty as absence) and Ones (uncertainty as presence).

The experimental results (Table 22) confirm that distillation transfers near-teacher performance to substantially lighter models. Under the Ones strategy, the distilled MobileNetV2 achieved an AUC-ROC of 0.884 and an F1-score of 0.670, effectively matching the teacher (AUC-ROC of 0.891, F1 of 0.664) while reducing the model size by 67.9% (from 26.87 to 8.65 MB) and the parameter count from 6.96 M to 2.23 M. The Ones strategy outperformed Zeros in AUC-ROC for most of the distilled models, indicating that treating uncertain labels as positive examples generally facilitates a more effective knowledge transfer. The exception was Distilled GenENAS, whose AUC-ROC and F1-score both declined under the Ones strategy (from 0.877 to 0.867 and from 0.653

to 0.641, respectively), showing that this effect is architecture-dependent and that gains in ranking metrics must always be contrasted with threshold-dependent metrics before clinical deployment. Most importantly for the coherence of the framework, distillation produced consistent AUC-ROC gains for the compact students: under the Ones strategy the GenENAS architecture improved from 0.854 to 0.867 and EfficientNet-B0 from 0.859 to 0.879. This result validates the interface between Stages 1 and 2, demonstrating that generatively discovered architectures are effective receptacles for distilled knowledge and that automatic architecture search and knowledge distillation can operate jointly within a unified pipeline rather than as isolated techniques.

At the close of this stage, the framework has produced a family of compact models whose diagnostic capability is no longer the limiting factor for deployment. The question that remains open—and that ultimately determines their viability in real clinical settings—is their operational cost: the energy consumed, the CO_2 emissions produced, and the computational demand incurred each time a prediction is made. The next chapter addresses precisely this question, developing the third stage of the framework: a systematic evaluation of energy consumption, emissions, and energy efficiency of the teacher, the students, and the distilled students under realistic inference workloads, which completes the pipeline by certifying that the models obtained here are not only accurate and compact, but also sustainable.

4. Energy Efficiency Analysis of the Proposed Framework

This chapter develops the third and final stage of the framework proposed in this thesis. While Stage 1 (Chapter 2) generatively discovers a compact architecture through the GenENAS LLM-guided search process, and Stage 2 (Chapter 3) transfers the predictive capability of a high-capacity teacher model to that architecture through Knowledge Distillation, this stage subjects the resulting models to an energy and environmental assessment. Its purpose is to determine whether the

computational savings achieved throughout the pipeline translate into measurable reductions in energy consumption and CO_2 emissions during inference—the phase of the model lifecycle that, due to its continuous use in clinical deployment, accumulates the highest energy burden—without compromising diagnostic performance.

To this end, the energy consumption, carbon emissions, and computational requirements during inference (throughput, latency, and peak memory usage) of the teacher model, the baseline student models, and their distilled counterparts—including the distilled GenENAS architecture produced end-to-end by the proposed framework—are evaluated under identical hardware and data conditions across workloads of increasing size. In this way, the pipeline is completed with a quantitative assessment of its energy efficiency within the Green AI paradigm.

This concern is not incidental: the rapid growth of deep learning models has led to major advances in computer vision and natural language processing. All this progress has resulted in a significant increase in energy consumption for model training and inference, raising both economic and environmental challenges. Artificial intelligence entails a high energy demand, pushing it to concerning levels within the Green AI paradigm, thereby encouraging the development of techniques to reduce the computational and carbon footprint generated by these processes [94].

Green artificial intelligence promotes the development and use of more efficient and sustainable technologies, demonstrating that innovation can be achieved without compromising the planet [95]. This has been achieved thanks to improvements in hardware, model optimization, and the use of clean energy, which is why many organizations are reducing their environmental footprint while driving innovation.

By applying techniques such as KD, the aim is not only to improve model performance while reducing computational cost, but also to optimize energy efficiency by substantially reducing the resources required during inference, both in terms of energy and execution time [96].

High energy consumption is mainly due to the complexity and scale of modern neural networks. It is estimated that training just one of the large commercial LLMs, such as ChatGPT, Gemini, Llama or GROK, requires consuming tens of gigawatt-hours (GWh) of electricity [94].

4.1 Energy consumption by phases of the model lifecycle

Energy consumption in the AI model lifecycle is distributed across several main phases, including data collection and preparation, model development and design, model training, deployment or inference for real-time prediction, and, finally, ongoing maintenance of the model in production. Of these phases, training and inference consume the most energy, with the latter accounting for the greatest cumulative energy load due to its continuous use. Together, these phases reflect the necessary balance between functional capability and energy sustainability in AI model lifecycle management, highlighting the urgent need to optimize algorithms and use renewable energy sources to mitigate environmental impact [96].

Training: During this phase, the Deep Learning model requires significant computational power, as operations such as matrix multiplication, activation of deep layers, and recurrent layers are performed here. The hardware used for training is typically GPUs, TPUs, and high-performance servers, and the duration of training for a DL model can range from hours to days or even weeks. Energy consumption in this phase increases not only with the number of operations, but also with hyperparameters, dataset size, model architecture, number of epochs, and the application of strategies such as early stopping and weight decay, among others [97].

Inference: once the model has been trained, it is deployed for use in predictions. It is important to note that each prediction requires less time and computational resources than those needed during training. However, when a model is in production and receives thousands or even millions of queries per day, its cumulative energy consumption can eventually exceed that during training [98].

4.2 Measuring energy consumption and emissions

For energy consumption calculations, the energy usage of each hardware component, such as the CPU, GPU, and memory, is measured, as these are the components that consume the most energy

during training and inference.

$$E_{Total} = E_{CPU} + E_{GPU} + E_{RAM} \tag{4.1}$$

Sustainable computing applied to AI defines emissions as the amount of greenhouse gases released into the atmosphere as a consequence of the energy consumption of a computational process, which are expressed in kilograms of carbon dioxide equivalent $KgCO_{2e}$.

When computational resources are used to create, train, and run AI models, electricity is consumed. This electricity is generated from various energy sources, such as coal, natural gas, oil, hydroelectric, solar, wind, and nuclear power, many of which emit greenhouse gases (GHGs) into the atmosphere.

The CO_2 emissions from data centers are primarily produced by the electrical consumption required to power servers and maintain the supporting infrastructure (cooling, networking, etc.).

The calculation of carbon dioxide equivalent (CO_{2e}) is based on the following general formula:

$$CO_{2e} = E_{Total} * FE * PUE \tag{4.2}$$

Where:

- E_{Total} : Energy consumed (in kilowatt-hours, kWh)
- FE : Emission factor of the electricity source ($kgCO_{2e}/kWh$)
- PUE : Power Usage Effectiveness

Total Energy is the amount of computational resources consumed during the execution of a process. The emission factor depends on each country, taking into account the energy mix used by each country to produce energy, whether it is generated from renewable resources or fossil fuels. This factor is higher when fossil fuels predominate and lower when more environmentally friendly energy sources predominate. Carbon emissions were estimated using CodeCarbon v3.2.8.

The framework automatically detected the location of execution in Colombia and applied a carbon intensity of 0.2595 $kgCO_2e/kWh$, based on the emission database integrated into CodeCarbon. This value was used consistently across all experiments to ensure reproducibility and comparability of the reported results. Power Usage Effectiveness (PUE) is the global standard for measuring the energy efficiency of a data center. It is defined as the ratio between the total energy consumed by the facility and the energy effectively used by the IT equipment (servers, storage, networks). The average PUE is around 1.58, although high-efficiency facilities can achieve values from 1.10 to 1.20 [99].

4.3 Energy Efficiency

Refers to the relationship between a model's performance and the amount of energy required to achieve it. In other words, an energy-efficient model is one that achieves high predictive performance while consuming the least possible energy [52].

$$\textit{Energy Efficiency}\,(\text{EE}) = \frac{\textit{Performance metrics}\,(\text{AUC-ROC})}{\textit{Energy Consumption}\,\left(E_{\text{Total}}\right)}. \tag{4.3}$$

Where:

- Performance metrics corresponds to the model's performance (accuracy, AUC-ROC, F1, throughput, etc.).
- Energy consumed includes CPU, GPU, RAM, and other components during training or inference.

$$\text{EE} = \frac{\text{AUC-ROC}}{E_{\text{TOTAL}}} \tag{4.4}$$

The current research focuses on analyzing the energy efficiency achieved during the inference of Deep Learning models, i.e., reducing energy consumption during the execution of already

trained models for prediction or classification without affecting their performance. This is very important to develop sustainable solutions that can be used on edge devices, mobile phones, and embedded systems with limited resources.

Reducing energy consumption during inference helps decrease environmental impact and prolong the lifetime of battery-powered devices, such as autonomous systems, IoT devices, and health devices. With this, the goal is to achieve a balance among performance, accuracy, and energy consumption through the development of specialized algorithms and hardware, promoting greener, more sustainable AI.

4.4 Case Study: CheXpert Dataset

Inference was performed on datasets of 1,000, 5,000, 10,000, 20,000, 30,000, and 40,000 images from the training set for DenseNet-121, MobileNetV2, GenENAS, EfficientNet-B0, and their respective distilled models, with DenseNet-121 serving as the teacher model. Using the Ones strategy, which achieved the best AUC-ROC metrics.

The same computational resource used in the previous chapter on Knowledge Transfer by Distillation was employed to run these energy consumption and energy efficiency tests.

.Testbed Infrastructure

- **Processor:** Intel Core i9-14900K
- **RAM:** 128 GB
- **GPU:** NVIDIA RTX 4090 (24 GB GDDR6X)

Table 23 summarizes the predictive performance (AUC-ROC and its standard deviation across five independent runs), the energy consumed, and the CO_2-equivalent emissions for the Teacher (DenseNet-121), the three student architectures (MobileNetV2, GenENAS, and EfficientNet-B0), and their distilled counterparts, evaluated over inference workloads ranging from

1,000 to 40,000 chest X-ray images. The results reveal a clear inverse relationship between model compactness and energy consumption. DenseNet-121 consistently achieved the highest AUC-ROC (0.807–0.811), but at the highest energy cost, reaching 8,909.72 J and 6.42×10^{-4} $kgCO_2e$ for 40,000 images. In contrast, the distilled MobileNetV2 maintained an AUC-ROC of 0.778–0.782 while consuming 4,904.74 J at the largest workload, a reduction of approximately 45% in both energy and emissions with respect to the teacher, at the cost of roughly 0.03 points of AUC-ROC. The benefit of knowledge distillation is most evident in the weaker students: Distilled GenENAS improved its AUC-ROC by 0.026–0.033 and Distilled EfficientNet-B0 by 0.034–0.038 with respect to their non-distilled versions, with only a marginal energy overhead (below 4%). Furthermore, the standard deviations remain in the order of 10^{-3}–10^{-4} and decrease as the sample size grows, indicating that the measurements are stable and reproducible; at 40,000 images, the distilled MobileNetV2 (SD = 8.19×10^{-4}) exhibits greater run-to-run stability than the teacher (SD = 1.11×10^{-3}).

Table 23
Performance and energy metrics by sample size. Energy is reported in joules; emissions in $kgCO_2eq$.

Samples [images]	Model	AUC-ROC	SD	Energy [J]	Emissions [$kgCO_2e$]
1000	DenseNet-121 (Teacher)	0.807	1.05×10^{-2}	312.17	2.25×10^{-5}
	MobileNetV2 (Student)	0.775	4.82×10^{-3}	213.63	1.54×10^{-5}
	Distilled MobileNetV2	0.782	1.24×10^{-2}	199.08	1.44×10^{-5}
	GenENAS (Student)	0.726	3.43×10^{-3}	252.45	1.82×10^{-5}
	Distilled GenENAS	0.759	9.43×10^{-3}	261.28	1.88×10^{-5}
	EfficientNet-B0 (Student)	0.738	3.87×10^{-3}	198.87	1.43×10^{-5}
	Distilled EfficientNet-B0	0.776	1.20×10^{-2}	207.93	1.50×10^{-5}

Continued on the next page

Table 23 – continued

Samples [images]	Model	AUC-ROC	SD	Energy [J]	Emissions [kgCO_2e]
	DenseNet-121 (Teacher)	0.808	4.04×10^{-3}	1163.21	8.38×10^{-5}
	MobileNetV2 (Student)	0.778	1.81×10^{-3}	685.33	4.94×10^{-5}
	Distilled MobileNetV2	0.780	2.21×10^{-3}	689.44	4.97×10^{-5}
5000	GenENAS (Student)	0.730	2.30×10^{-3}	944.83	6.82×10^{-5}
	Distilled GenENAS	0.757	2.45×10^{-3}	978.62	7.05×10^{-5}
	EfficientNet-B0 (Student)	0.741	1.45×10^{-3}	651.68	4.70×10^{-5}
	Distilled EfficientNet-B0	0.776	2.75×10^{-3}	649.48	4.68×10^{-5}
	DenseNet-121 (Teacher)	0.809	1.75×10^{-3}	2301.69	1.66×10^{-4}
	MobileNetV2 (Student)	0.778	3.13×10^{-3}	1284.04	9.25×10^{-5}
	Distilled MobileNetV2	0.779	2.13×10^{-3}	1266.59	9.13×10^{-5}
10000	GenENAS (Student)	0.732	2.02×10^{-3}	1775.64	1.28×10^{-4}
	Distilled GenENAS	0.756	2.54×10^{-3}	1907.64	1.38×10^{-4}
	EfficientNet-B0 (Student)	0.742	2.55×10^{-3}	1198.82	8.64×10^{-5}
	Distilled EfficientNet-B0	0.776	2.22×10^{-3}	1222.35	8.81×10^{-5}
	DenseNet-121 (Teacher)	0.810	1.75×10^{-3}	4514.34	3.25×10^{-4}
	MobileNetV2 (Student)	0.778	1.30×10^{-3}	2451.24	1.77×10^{-4}
	Distilled MobileNetV2	0.779	2.05×10^{-3}	2443.34	1.76×10^{-4}
20000	GenENAS (Student)	0.731	1.82×10^{-3}	3655.70	2.64×10^{-4}
	Distilled GenENAS	0.756	2.12×10^{-3}	3724.85	2.69×10^{-4}
	EfficientNet-B0 (Student)	0.741	1.27×10^{-3}	2294.09	1.65×10^{-4}
	Distilled EfficientNet-B0	0.775	1.95×10^{-3}	2336.87	1.68×10^{-4}

Continued on the next page

Table 23 – continued

Samples [images]	Model	AUC-ROC	SD	Energy [J]	Emissions [$kgCO_2e$]
	DenseNet-121 (Teacher)	0.810	1.14×10^{-3}	6829.77	4.92×10^{-4}
	MobileNetV2 (Student)	0.776	7.43×10^{-4}	3575.24	2.58×10^{-4}
	Distilled MobileNetV2	0.778	9.45×10^{-4}	3670.15	2.65×10^{-4}
30000	GenENAS (Student)	0.730	7.40×10^{-4}	5246.71	3.78×10^{-4}
	Distilled GenENAS	0.756	6.81×10^{-4}	5554.38	4.00×10^{-4}
	EfficientNet-B0 (Student)	0.739	1.04×10^{-3}	3398.38	2.45×10^{-4}
	Distilled EfficientNet-B0	0.775	6.99×10^{-4}	3469.91	2.50×10^{-4}
	DenseNet-121 (Teacher)	0.811	1.11×10^{-3}	8909.72	6.42×10^{-4}
	MobileNetV2 (Student)	0.777	7.76×10^{-4}	4795.53	3.46×10^{-4}
	Distilled MobileNetV2	0.778	8.19×10^{-4}	4904.74	3.54×10^{-4}
40000	GenENAS (Student)	0.730	1.39×10^{-3}	7102.44	5.12×10^{-4}
	Distilled GenENAS	0.756	1.10×10^{-3}	7379.58	5.32×10^{-4}
	EfficientNet-B0 (Student)	0.740	1.40×10^{-3}	4537.06	3.27×10^{-4}
	Distilled EfficientNet-B0	0.774	8.84×10^{-4}	4633.79	3.34×10^{-4}

Table 24 reports the computational demand of inference in terms of throughput (images per second), latency per image, and peak GPU memory for each model and workload size. Two patterns stand out. First, throughput increases with the size of the inference batch for all models—for instance, from 1,172.58 img/s (1,000 images) to 1,766.78 img/s (40,000 images) in the case of the teacher—reflecting the amortization of initialization and data-loading overheads over larger workloads. Second, the lightweight models consistently outperform the teacher in serving capacity: at 40,000 images, MobileNetV2 reaches 1,920.06 img/s with a latency of 0.526 ms, compared to 1,766.78 img/s and 0.566 ms for DenseNet-121, while its distilled version retains essentially the same profile (1,872.37 img/s, 0.534 ms). Peak memory, however, does not follow the parameter count: the GenENAS models exhibit the largest peak GPU memory (903–941 MB), exceeding the teacher (532–569 MB) despite having fewer parameters, which is attributable to the wider

intermediate activation maps of their inverted-residual blocks. This observation indicates that parameter count alone is an insufficient proxy for deployment cost, and that activation memory must be considered when targeting devices with strict memory budgets.

Table 24
Inference computational demand by sample size. Throughput in images per second; latency in milliseconds per inference; peak memory in MB.

Samples [images]	Model	Throughput [img/s]	Latency [ms]	Peak GPU memory [MB]
1000	DenseNet-121 (Teacher)	1172.58	0.853	532.27
	MobileNetV2 (Student)	1330.09	0.752	571.56
	Distilled MobileNetV2	1340.87	0.746	610.23
	GenENAS (Student)	1263.73	0.791	903.08
	Distilled GenENAS	1199.90	0.833	903.08
	EfficientNet-B0 (Student)	1296.53	0.771	623.22
	Distilled EfficientNet-B0	1250.95	0.799	623.36
5000	DenseNet-121 (Teacher)	1644.44	0.608	540.65
	MobileNetV2 (Student)	1743.70	0.573	580.93
	Distilled MobileNetV2	1722.25	0.581	619.61
	GenENAS (Student)	1663.73	0.601	912.46
	Distilled GenENAS	1635.78	0.611	912.46
	EfficientNet-B0 (Student)	1701.26	0.588	632.59
	Distilled EfficientNet-B0	1661.62	0.602	632.74

Continued on next page

Table 24 – continued

Samples [images]	Model	Throughput [img/s]	Latency [ms]	Peak memory [MB]
	DenseNet-121 (Teacher)	1693.57	0.590	541.15
	MobileNetV2 (Student)	1813.08	0.551	580.93
	Distilled MobileNetV2	1823.60	0.548	619.61
10000	GenENAS (Student)	1758.04	0.569	912.46
	Distilled GenENAS	1688.89	0.592	912.46
	EfficientNet-B0 (Student)	1769.99	0.565	632.59
	Distilled EfficientNet-B0	1758.23	0.569	632.74
	DenseNet-121 (Teacher)	1735.42	0.576	550.53
	MobileNetV2 (Student)	1861.57	0.537	590.36
	Distilled MobileNetV2	1871.10	0.534	629.03
20000	GenENAS (Student)	1704.81	0.587	921.83
	Distilled GenENAS	1729.52	0.578	921.83
	EfficientNet-B0 (Student)	1812.01	0.552	641.97
	Distilled EfficientNet-B0	1803.58	0.554	642.11
	DenseNet-121 (Teacher)	1721.42	0.581	569.27
	MobileNetV2 (Student)	1920.06	0.521	609.06
	Distilled MobileNetV2	1862.82	0.537	647.73
30000	GenENAS (Student)	1786.22	0.560	940.58
	Distilled GenENAS	1748.06	0.572	940.58
	EfficientNet-B0 (Student)	1827.99	0.547	660.72
	Distilled EfficientNet-B0	1801.88	0.555	660.86

Continued on next page

Table 24 – continued

Samples [images]	Model	Throughput [img/s]	Latency [ms]	Peak memory [MB]
	DenseNet-121 (Teacher)	1766.78	0.566	550.52
	MobileNetV2 (Student)	1900.74	0.526	590.36
	Distilled MobileNetV2	1872.37	0.534	629.03
40000	GenENAS (Student)	1772.86	0.564	921.83
	Distilled GenENAS	1750.95	0.571	921.83
	EfficientNet-B0 (Student)	1827.12	0.547	642.11
	Distilled EfficientNet-B0	1808.83	0.553	642.11

Figure 27 depicts the total energy consumed during inference as a function of the workload size (1,000 to 40,000 images) for each of the seven evaluated models. Energy grows almost perfectly linearly with the number of processed images for all models, which indicates a stable per-image energy cost and allows reliable extrapolation to larger-scale deployments. Three well-separated bands can be distinguished. The teacher, DenseNet-121, defines the upper bound throughout the entire range, with an effective cost of approximately 0.22 J per image. The GenENAS pair occupies an intermediate band ($\approx$0.18 J per image), despite having the smallest parameter count, confirming that architectural topology—rather than model size alone—drives inference cost. Finally, the MobileNetV2 and EfficientNet-B0 families form the lower band, with per-image costs between 0.11 and 0.12 J, roughly half of the teacher's. The distilled variants overlap almost exactly with their non-distilled counterparts, which demonstrates that knowledge distillation improves predictive quality (Table 23) without altering the energy profile of the student architecture.

Figure 28 shows the CO_2-equivalent emissions produced during inference for the same workload range. The curves mirror the behavior observed in Figure 27: emissions scale linearly with the number of inferences and preserve the same three-band structure, with DenseNet-121 as the most emitting model (6.42×10^{-4} kgCO_2e at 40,000 images), the GenENAS pair in an intermediate

Figure 27

Energy consumption in Inference

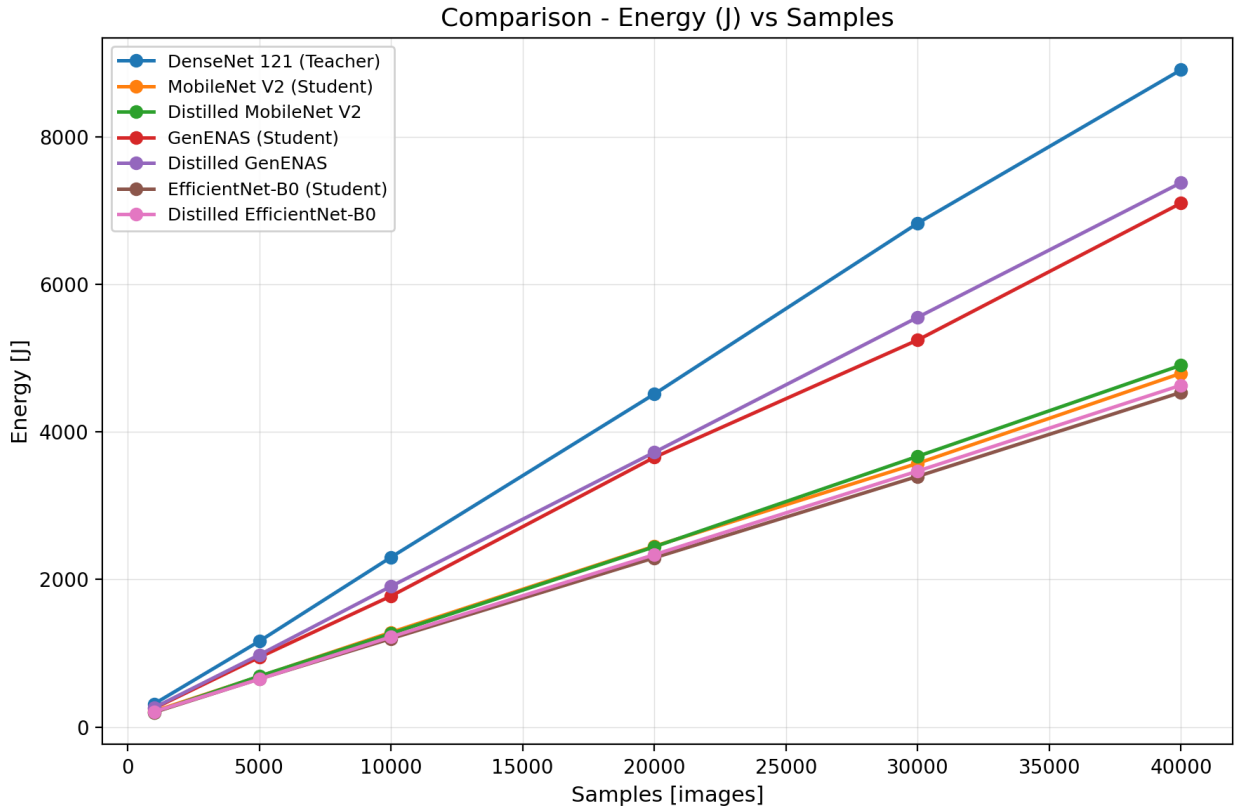
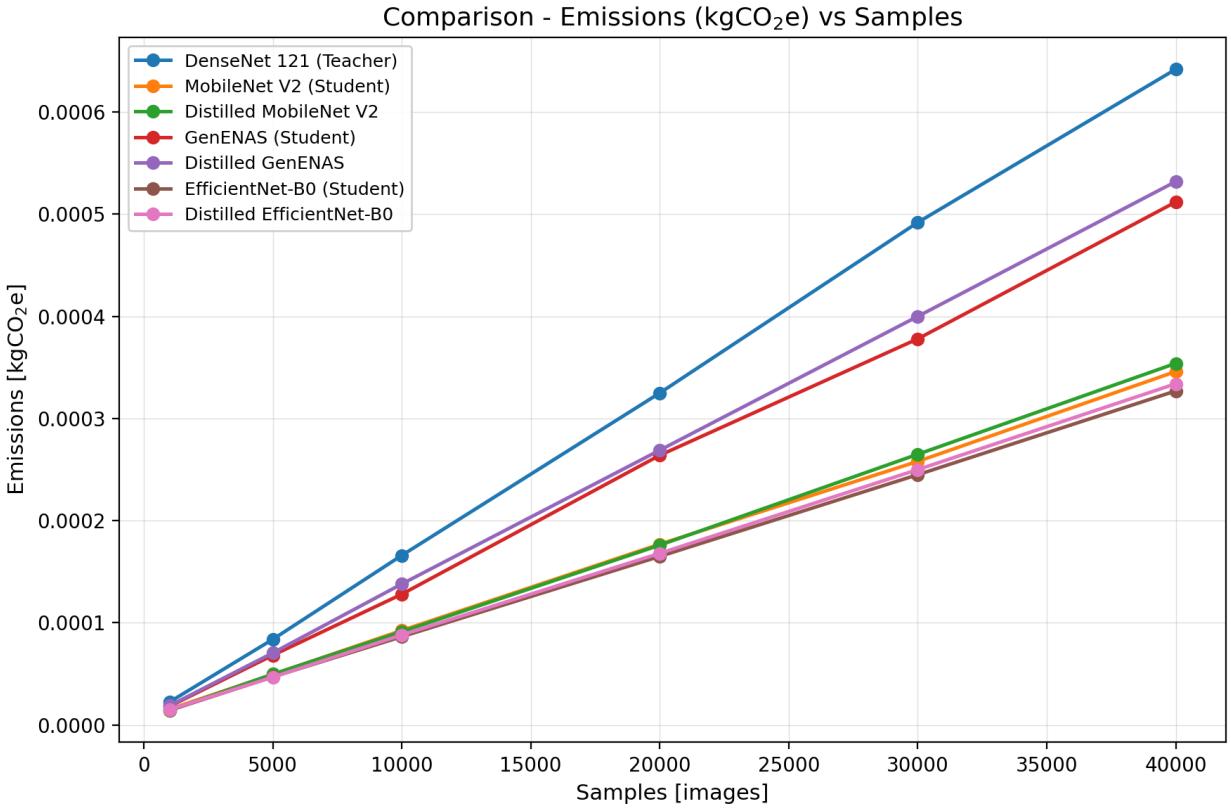


Figure 28
Emissions produced in Inference



position (5.12–5.32 $\times$ 10^{-4} $kgCO_2e$), and the MobileNetV2 and EfficientNet-B0 families at the bottom (3.27–3.54 $\times$ 10^{-4} $kgCO_2e$). In practical terms, replacing the teacher with a distilled compact model reduces the carbon footprint of each diagnosis by approximately 45%, a saving that becomes substantial when projected onto clinical scenarios involving thousands of daily inferences, and that aligns the proposed framework with the sustainability goals promoted by the Green AI paradigm.

Table 25 presents the energy efficiency (EE), computed according to Equation 4.4 as the ratio between the AUC-ROC and the total energy consumed during the inference of 40,000 images from the training dataset. The teacher model is, by a wide margin, the least efficient configuration (9.10$\times$ 10^{-5} AUC-ROC/J): its moderate advantage in predictive performance does not compensate for its energy consumption, which nearly doubles that of the compact models. All lightweight alternatives achieve efficiencies between 1.02×10^{-4} and 1.67×10^{-4}, with Distilled EfficientNet-B0 emerging

as the most energy-efficient model overall (1.67×10^{-4}, an 83% improvement over the teacher), since distillation raised its AUC-ROC from 0.740 to 0.774 with a negligible energy overhead. For MobileNetV2 and GenENAS, the distilled versions attain efficiency values essentially on par with their non-distilled counterparts (1.59×10^{-4} vs. 1.62×10^{-4}, and 1.02×10^{-4} vs. 1.03×10^{-4}, respectively), as their accuracy gains are offset by a slight increase in energy consumption. Overall, these results confirm that the combination of compact architectures and knowledge distillation delivers the best balance between diagnostic performance and energy cost, achieving up to 1.8 times the efficiency of the teacher model.

Table 25
Energy Efficiency.

Model	Energy Efficiency
DenseNet-121 (Teacher)	9.10×10^{-5}
MobileNetV2 (Student)	1.62×10^{-4}
Distilled MobileNetV2	1.59×10^{-4}
GenENAS (Student)	1.03×10^{-4}
Distilled GenENAS	1.02×10^{-4}
EfficientNet-B0 (Student)	1.63×10^{-4}
Distilled EfficientNet-B0	1.67×10^{-4}

4.5 Cumulative Energy, Time and Carbon Footprint Analysis

This section integrates the computational costs associated with the three stages of the proposed framework. While Chapters 2, 3, and 4 independently evaluated the architecture search process, knowledge distillation, and energy-efficiency analysis, respectively, the present section provides a cumulative assessment of the execution time, energy consumption, and carbon emissions required to obtain and analyze the final distilled model.

As shown in Table 26, the cumulative analysis reveals that GenENAS stage dominates the total framework development cost, accounting for over 98% of total energy consumption and carbon emissions. This result is expected, as neural architecture search is a distinct optimization process

Table 26
Total energy consumption and carbon emissions across the proposed framework.

Stage	Component	Time [hours]	Energy [kJ]	Emissions [kgCO_2eq]
1	Architecture Search – GenENAS (Table 10)	285.15	456,815.1	32.865
2	KD-MLC (Zeros and Ones) (Table 21)	4.32	5,579.98	0.402
3	Inference – Distilled GenENAS Model (40,000 Images) (Table 23)	–	7.38	5.32×10^{-4}
Total Development Cost		**289.47**	**462,402.46**	**33.268**

that evaluates multiple candidate architectures.

Although the GenENAS stage incurs a substantial computational investment, it is executed only once. The resulting architecture can subsequently be trained, distilled, and deployed multiple times without repeating the search process.

KD-MLC stage introduces only a marginal computational overhead compared to the GenENAS stage while providing significant gains in predictive performance and model robustness.

Once discovered and distilled, the final distilled GenENAS model exhibits an extremely small operational footprint, requiring only 7.38 kJ to process 40,000 chest X-ray images. The proposed framework follows a Green AI paradigm in which a one-time computational investment is transformed into a highly efficient and sustainable model suitable for long-term deployment.

The final distilled GenENAS model requires approximately 0.185 J per chest X-ray image during inference, demonstrating its suitability for deployment in resource-constrained environments.

The cumulative results demonstrate that the proposed framework successfully transforms a computationally intensive architecture discovery process into a compact and highly energy-efficient diagnostic model. While the architecture search stage accounts for the majority of the development cost, the resulting distilled GenENAS architecture achieves a negligible operational footprint during inference, validating the effectiveness of combining generative neural architecture

search, knowledge distillation, and energy-aware evaluation within a unified framework.

4.6 Summary

This chapter developed and validated the third and final stage of the framework proposed in this thesis, closing the pipeline that begins with the automatic discovery of compact architectures through GenENAS (Stage 1, Chapter 2) and continues with the transfer of predictive capability from a high-capacity teacher to lightweight students via Knowledge Distillation (Stage 2, Chapter 3). Whereas the first two stages answer the question of *how to obtain* compact and accurate models, this stage answers the question that ultimately determines their viability in real clinical settings: *at what energy and environmental cost do they operate once deployed*. To this end, the models produced by the previous stage under the Ones labeling strategy—the teacher DenseNet-121, the students MobileNetV2, GenENAS, and EfficientNet-B0, and their distilled counterparts—were benchmarked during inference over workloads ranging from 1,000 to 40,000 chest X-ray images, measuring the AUC-ROC and its standard deviation across five independent runs, the energy consumed, the associated CO_2-equivalent emissions, and the computational demand in terms of throughput, latency, and peak memory.

The results quantify a clear trade-off between model capacity and sustainability. The teacher achieved the highest predictive performance (AUC-ROC up to 0.811) but also the highest operational cost, with an energy consumption and a carbon footprint that nearly double those of the compact models (8,909.72 J and 6.42×10^{-4} $kgCO_2e$ for 40,000 inferences). The distilled students preserved most of the diagnostic capability (AUC-ROC of 0.774–0.782 for Distilled MobileNetV2 and Distilled EfficientNet-B0) while reducing energy and emissions by approximately 45%, improving throughput and latency, and exhibiting greater run-to-run stability than the teacher. Energy consumption and emissions scaled linearly with the workload for all models, so these per-image savings extrapolate directly to large-scale clinical deployments. In terms of the integrated energy-efficiency metric (Equation 4.4), all compact models nearly doubled the efficiency of the teacher,

with Distilled EfficientNet-B0 standing out as the most efficient configuration (1.67×10^{-4} AUC-ROC/J, an 83% improvement over the teacher), which confirms that knowledge distillation adds predictive value without altering the energy profile of the student architecture. The analysis also revealed that parameter count alone does not determine operational cost: the GenENAS models, despite being the smallest in parameters, exhibited intermediate energy consumption and the largest memory footprint, evidencing the importance of evaluating deployment metrics empirically rather than inferring them from model size.

Taken together, these findings complete the validation of the proposed framework as an end-to-end methodology: generative architecture search identifies compact candidates, knowledge distillation endows them with near-teacher predictive capability, and the energy analysis of this chapter certifies that the resulting models are sustainable, stable, and deployable on resource-constrained infrastructure. This positions the framework as a concrete instrument of the Green AI paradigm, enabling high-quality, low-footprint diagnostic support in hospitals and regions where high-performance computing resources, stable connectivity, or cloud access cannot be taken for granted.

5. Conclusions and Future Work

5.1 Conclusions

This thesis set out to demonstrate that computationally efficient deep learning models for medical imaging can be obtained not through isolated optimization tricks, but through a unified, end-to-end framework in which a single model is progressively discovered, strengthened, and certified for sustainable deployment. The framework chains three stages under one computational-efficiency objective: Stage 1 (GenENAS) automatically discovers a compact architecture; Stage 2 (KD-

MLC) transfers the predictive capability of a high-capacity teacher into compact students through multi-label knowledge distillation; and Stage 3 (Energy Efficiency Analysis) quantifies the energy, emissions, and computational demand of the resulting models under realistic inference workloads. The conclusions of this research are therefore presented not as independent findings, but as the validation of each stage, of the interfaces between them, and of the framework as a whole.

On the framework as a whole

1. **The unified framework is the central contribution, and it works end-to-end.** The experimental evidence confirms that the three stages are complementary and mutually reinforcing: GenENAS reduces parameters and memory footprint at design time, knowledge distillation maximizes the diagnostic performance of the compact models without increasing their size, and the energy analysis certifies that the resulting models operate at roughly half the energy and carbon cost of the reference teacher. No single stage, in isolation, achieves this combination: architecture search alone yields compact but weaker models (AUC-ROC of 0.854 for the GenENAS student under the Ones strategy), distillation alone presupposes a suitable student architecture, and energy analysis alone optimizes nothing. Chained together, they deliver models that are simultaneously compact, accurate, stable, and sustainable.

2. **The framework refutes the computational brute-force paradigm in the medical domain.** The dominant strategy of scaling parameters and data shows diminishing returns for chest X-ray classification: standard architectures such as ResNet-152 (58.15 M parameters, 232.62 MB) offer only a marginal advantage (AUC-ROC of 0.875 versus 0.869) over an automatically discovered architecture approximately 30 times smaller (1.91 M parameters, 7.28 MB). Incorporating hardware, memory, and energy constraints from the design stage—rather than as an afterthought—is therefore not a limitation but a viable and necessary design principle, particularly for institutions that cannot rely on high-performance computing infrastructure.

On Stage 1: Generative architecture discovery (GenENAS)

3. **Large language models are effective black-box optimizers for neural architecture search.** GPT-4 and Llama 2 demonstrated the latent capability to reason over a combinatorial search space of 3^8 = 6,561 architectures, guiding the search toward high-performing configurations in very few iterations. The optimal architecture, GenENAS-22121211, was identified at the fifth iteration of the serial execution (285.15 GPU-hours and 456,815.1 kJ in total), reaching an AUC-ROC of 0.869 across the five target CheXpert pathologies. The fact that an open-source 7B-parameter model (Llama 2) suffices for this task increases the accessibility and reproducibility of the methodology in environments where proprietary APIs are restricted or economically unfeasible.

4. **Generative parallelization accelerates the search but at a quantified energy cost.** Distributed over two NVIDIA RTX 3090 GPUs, GenENAS Parallel located the best architecture at its third iteration (127.09 accumulated hours, versus 190.23 hours for the serial run) and completed the search in 225.52 hours in total, 20.9% faster than the 285.15 hours of the serial version. This acceleration, however, required training ten candidate architectures instead of seven, increasing energy consumption by 41.29% (645,435.4 kJ versus 456,815.1 kJ). Making this time–energy trade-off explicit is itself a contribution of the framework: parallel NAS should be chosen when search time is the binding constraint, and serial NAS when the energy budget is.

On Stage 2: Knowledge transfer via multi-label distillation (KD-MLC)

5. **Distillation transfers near-teacher performance to models one third the size.** Under the Ones strategy, the distilled MobileNetV2 reached an AUC-ROC of 0.884 and an F1-score of 0.670, effectively matching the teacher DenseNet-121 (AUC-ROC of 0.891, F1 of 0.664) while reducing the model size by 67.9% (from 26.87 to 8.65 MB) and the parameter count

from 6.96 M to 2.23 M. The benefit was largest precisely where the framework needs it most—on the compact students: distillation raised the AUC-ROC of the GenENAS architecture from 0.854 to 0.867 and that of EfficientNet-B0 from 0.859 to 0.879. This validates the interface between Stages 1 and 2: the architectures discovered generatively are effective receptacles for distilled knowledge.

6. **Handling clinical uncertainty is a first-class design decision.** Treating CheXpert's uncertain labels as positive (Ones strategy) consistently produced better AUC-ROC results than treating them as negative (Zeros strategy) across the distilled models. At the same time, the case of Distilled GenENAS—whose AUC-ROC improved while its F1-score decreased from 0.653 to 0.641—shows that gains in ranking metrics do not automatically translate into gains in threshold-dependent metrics. In clinical applications, where error costs are asymmetric, both families of metrics must be evaluated jointly before deployment.

On Stage 3: Energy efficiency and sustainable deployment

7. **Higher energy consumption does not imply better performance.** For 40,000 inference samples, the teacher consumed 8,909.72 J to reach an AUC-ROC of 0.811, whereas the distilled MobileNetV2 required only 4,904.74 J—a 45% reduction in energy and emissions (3.54×10^{-4} versus 6.42×10^{-4} $kgCO_2e$)—for an AUC-ROC of 0.778. Sacrificing approximately 0.03 points of AUC-ROC halves the operational footprint, a trade decisively favorable for screening scenarios with massive inference volumes. Because energy and emissions scale linearly with the workload, these per-image savings extrapolate directly to large-scale clinical deployments, where the inference phase dominates the cumulative environmental impact of a model in production.

8. **Energy efficiency must be measured, not inferred from model size.** Under the integrated metric EE = AUC-ROC$/E_{\text{Total}}$, the compact models nearly doubled the efficiency of the

teacher (9.10×10^{-5} AUC-ROC/J), with Distilled EfficientNet-B0 as the most efficient configuration (1.67×10^{-4}, an 83% improvement). Yet the GenENAS-derived models, despite having the fewest parameters, exhibited intermediate energy consumption and the largest peak memory (903–941 MB), exceeding even the teacher. Parameter count is therefore an insufficient proxy for operational cost: deployment metrics—energy, latency, throughput, and activation memory—must be benchmarked empirically, which is exactly the role Stage 3 plays within the framework.

9. **The resulting models are deployable, stable, and clinically plausible for resource-constrained settings.** With 8.65 MB of disk size, throughput above 1,870 images per second, latency of 0.534 ms per image, and greater run-to-run stability than the teacher (SD of 8.19×10^{-4} versus 1.11×10^{-3} at 40,000 images), the distilled student can run entirely on smartphones, medical tablets, or mid-range workstations, without cloud connectivity and without transmitting patient data. This directly addresses the motivation of the thesis: democratizing advanced medical AI in hospitals and regions where high-performance infrastructure, stable connectivity, or cloud access cannot be assumed.

Closing remarks

Although the framework was instantiated and validated on chest X-ray classification with CheXpert, its three stages constitute transferable methodologies: by adjusting the optimization metrics, the teacher and student architectures, the uncertainty-handling strategies, and the loss-balancing mechanisms, the same pipeline can be adapted to other application domains where accuracy, computational cost, and sustainability must be balanced. The framework thus offers a concrete, reproducible instrument of the Green AI paradigm, and a methodological alternative to computational brute force for bringing high-quality diagnostic support to the environments that need it most.

5.2 Domain-Specific Medical Considerations and Framework Generalization

The framework proposed in this research comprising Generative Enhanced Neural Architecture Search (GenENAS), Knowledge Distillation Transfer Multi-Label Classification (KDT-MLC), and comprehensive energy efficiency analysis has been designed, implemented, and validated within the context of medical image classification of chest X-rays using the CheXpert dataset. However, it is pertinent to discuss both the particularities that make the medical domain an especially challenging scenario for the deployment of deep learning models, as well as the generalization capability of the framework toward other application domains.

.Particularities of Deep Learning in Medical Applications First, the medical domain is characterized by an asymmetric tolerance for diagnostic errors. In pathology classification, a false negative (the failure to detect a present disease) can have severe consequences for the patient, including delayed treatment and a worsened prognosis. Conversely, while a false positive generates an undesirable alarm, it can typically be resolved through complementary studies without direct harm to the patient's health. This asymmetry in error costs justifies the adoption of metrics such as AUC-ROC and F1-Score, which evaluate model performance across multiple decision thresholds, rather than relying on accuracy, which is inadequate in the presence of class imbalance.

Second, the management of uncertain and noisy labels is an inherent complexity of the radiological domain. In the CheXpert dataset, four possible states are incorporated for each label: positive, negative, uncertain, and not mentioned. This reflects the clinical reality where even experienced radiologists may disagree on the diagnosis of certain pathologies. Consequently, the *Zeros* and *Ones* strategies evaluated in this thesis represent a specific contribution to handling such diagnostic ambiguity.

Third, the regulatory framework imposes requirements that have no equivalent in most deep learning application domains. A diagnostic model deployed in a real clinical setting must comply with approval regulations from governmental bodies, which demand exhaustive documentation

regarding model validation, reproducibility, behavior under out-of-distribution data, and, in some cases, the explainability of its predictions.

Fourth, computational infrastructure constraints in hospital settings constitute the central motivation of this research. Hospitals—particularly those in regions lacking high-end GPU resources or stable cloud connectivity—frequently require model deployment on edge devices such as tablets, mid-range workstations, or embedded systems integrated with image acquisition equipment. This restriction justifies both the search for compact architectures via GenENAS and model compression through knowledge distillation. Furthermore, medical data privacy requirements often limit or prohibit the transmission of patient images to external servers, necessitating models capable of performing local inference without cloud dependency.

Finally, the class distribution and pathology prevalence—which led to the incorporation of weights inversely proportional to the frequency of each pathology in the loss function—constitute a direct adaptation to this medical domain reality.

.Generalization of the Framework to Other Domains The three components of the framework proposed in this thesis constitute transferable methodologies adaptable to various deployment scenarios. However, the specific instantiation presented in this research is tailored to the constraints and requirements of the medical imaging domain. The variation of the framework's components depends on the target application domain. Among the elements that may vary are the optimization metrics, the teacher and student architectures utilized, the necessity of uncertainty management strategies, constraints regarding size and parameter count, modifications to loss functions, and the adjustment or removal of class-balancing mechanisms, among others.

5.3 Hypothesis Validation

The research hypothesis is supported by the experimental evidence obtained throughout this thesis. It was hypothesized that population-based training approaches are constrained in distributed com-

puting environments because they rely on transferring and synchronizing complete weight vectors among population members, leading to memory centralization and communication overhead that limit computational efficiency.

The proposed framework addresses this limitation through a fundamentally different search paradigm. Rather than exchanging model weights between nodes, the architecture search process is guided by a Large Language Model (LLM) that operates on compact textual encodings of neural architectures. Consequently, the search process eliminates the need for weight transfer during architecture generation and significantly reduces the memory and communication requirements typically associated with population-based optimization methods.

The experimental results provide evidence supporting this hypothesis. The automatically discovered architecture, containing only 1.91 million parameters, achieved an AUC–ROC of 0.869, closely matching the performance of ResNet-152 (AUC–ROC of 0.875) despite being approximately thirty times smaller. Furthermore, the distilled models derived from the proposed framework achieved substantial reductions in computational complexity, memory footprint, energy consumption, and carbon emissions when compared with high-capacity teacher models, while maintaining competitive diagnostic performance.

These findings demonstrate that the computational limitations traditionally associated with centralized population-based training can be effectively mitigated by replacing weight-centric information exchange with compact architectural representations. As a result, the proposed framework achieves a better balance among diagnostic accuracy, computational efficiency, and energy sustainability, thereby validating the central hypothesis of this research.

5.4 Further Work

In this thesis, a unified three-stage framework—generative architecture search, multi-label knowledge distillation, and energy efficiency analysis—has proven highly effective for obtaining compact models without sacrificing diagnostic performance. Precisely because the framework is modular,

the most natural directions for future research are extensions of its individual stages and of the pipeline as a whole, which are organized below according to the stage they strengthen.

Extensions to Stage 1: Generative Enhanced Neural architecture search - GenENAS

1. **Hybrid CNN–Transformer search spaces:** The current search space of GenENAS is restricted to convolutional operations. It is suggested to extend it to incorporate Vision Transformers (ViTs), which better capture global relationships within an image—an essential capability for detecting subtle patterns in the lungs. The generative engine described in Chapter 2 could thus be used to discover hybrid architectures that combine the efficiency of convolutions (for edges and textures) with the global modeling capabilities of Transformers. Since Stage 3 revealed that parameter count alone does not predict operational cost, the LLM-guided search could additionally receive energy and peak-memory measurements as feedback signals, turning GenENAS into an explicitly multi-objective, energy-aware search.

Extensions to Stage 2: Knowledge distillation multilabel classification - KD-MLC

2. **New distillation schemes, including multi-teacher and cross-architecture distillation:** In multi-teacher distillation, several teacher models transfer knowledge to a single student, allowing the strengths of each teacher to be combined. Recent studies have shown that a student model can aggregate knowledge from multiple teachers and achieve superior performance compared to a model trained from scratch, while maintaining constant computational complexity [100]; the student matches and sometimes exceeds the performance of the individual teachers by condensing their collective knowledge into a single model. Complementarily, and in synergy with the hybrid search proposed for Stage 1, distillation can be applied across architectural families—using a Transformer as the teacher and a CNN as the student—enabling the transfer of global attention patterns into the CNN's efficient structure. In both cases,

the KD-MLC adaptations developed in this thesis (per-class Bernoulli KL divergence, class weighting, and uncertainty-handling strategies) remain directly applicable.

Extensions to Stage 3: Compression and deployment efficiency

3. **Aggressive quantization and structured pruning:** Post-training quantization and quantization-aware training allow converting floating-point models into 8-bit or even 4-bit integer formats. Since the deployment analysis of Chapter 4 showed that activation memory does not shrink proportionally with parameter count—the GenENAS models exhibited the largest peak memory (903–941 MB) despite having the fewest parameters—quantization becomes particularly attractive: reducing precision from FP32 to INT8 decreases both weight and activation memory by a factor of four, while accelerating inference on portable ARM/DSP hardware optimized for integer operations. Structured pruning would additionally remove entire channels from convolutional layers that contribute minimally to predictions, based on the importance of the weights learned during the distillation process. Crucially, the framework itself provides the instrument to validate these techniques: the energy, emissions, and efficiency benchmark of Stage 3 should be re-applied to the quantized and pruned models to verify that the theoretical compression gains translate into measured energy savings without an unacceptable loss of AUC-ROC.

Framework-level validation

4. **Domain generalization of the complete pipeline:** The present study primarily uses the CheXpert dataset, which may introduce domain bias, as chest X-ray images can vary depending on the machine, technician, and patient demographics. It is therefore recommended to evaluate the distilled models produced by the framework (e.g., the distilled MobileNetV2) on completely different external datasets without retraining. Doing so would demonstrate

the robustness of the framework's end products when exposed to data from rural hospitals or from developing countries, acquired with equipment different from that used to create CheXpert—precisely the deployment scenarios that motivate this thesis.

5. **Clinical robustness and explainability of the compressed models:** Aggressive compression can sometimes harm model calibration. Future work should therefore include generating heatmaps for both the teacher and the distilled student, enabling a quantitative comparison of their activation patterns across the stages of the framework. This would verify whether the compact model continues to focus on the same lung regions as the teacher after architecture search, distillation, and any subsequent compression, ensuring that the efficiency gains certified by Stage 3 are not obtained at the expense of clinical reliability.

6. **Prospective clinical validation in a hospital environment:** The validation carried out in this thesis is retrospective and conducted on a curated research dataset (CheXpert). A natural and necessary continuation is the prospective evaluation of the framework's end products within a real hospital setting, which entails three complementary efforts. First, integration with the hospital information system and the PACS/RIS infrastructure, so that the distilled model operates on radiographs acquired under routine clinical conditions rather than on a pre-assembled benchmark. Second, evaluation under the actual clinical workflow, measuring not only diagnostic performance but also operational factors such as inference latency on the institution's hardware, behavior under heterogeneous acquisition protocols, and robustness to image-quality variability. Third, blinded assessment by radiologists and clinical specialists, comparing the model's predictions and activation heatmaps against expert reads to establish diagnostic concordance and clinical trust. Such a study would also require the corresponding ethical and regulatory approvals (institutional review board, data-protection compliance, and, where applicable, medical-device certification). This prospective validation lies beyond the computational-efficiency scope of the present work—whose objective concerns the implementation of deep learning algorithms on large-scale architectures—but

constitutes the decisive step toward the eventual deployment of the resulting models in the resource-constrained clinical environments that motivate this thesis.

Bibliographic references

[1] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th. Hoboken, NJ: Pearson, 2021, ISBN: 978-0-13-461099-3.

[2] P. Jackson, *Introduction to Expert Systems*, 3rd. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1986.

[3] G. Litjens et al., "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, pp. 60–88, Dec. 2017, ISSN: 13618415. DOI: 10.1016/j.media.2017.07.005 [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1361841517301135

[4] M. Zheng et al., *Can GPT-4 perform neural architecture search?* Aug. 2, 2023. DOI: 10.48550/arXiv.2304.10970 arXiv: 2304.10970[cs.LG]. [Online]. Available: http://arxiv.org/abs/2304.10970

[5] G. Hinton, O. Vinyals, and J. Dean, *Distilling the knowledge in a neural network*, Version Number: 1, 2015. DOI: 10.48550/ARXIV.1503.02531 [Online]. Available: https://arxiv.org/abs/1503.02531

[6] J. H. Gennari et al., "The evolution of protégé: An environment for knowledge-based systems development," *International Journal of Human-Computer Studies*, vol. 58, no. 1, pp. 89–123, Jan. 2003, ISSN: 10715819. DOI: 10.1016/S1071-5819(02)00127-1 [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1071581902001271

[7] E. H. Shortliffe, *Computer-Based Medical Consultations: MYCIN* (Artificial Intelligence Series). New York: American Elsevier, 1976, ISBN: 978-0-444-00179-5.

[8] R. A. Miller, H. E. Pople, and J. D. Myers, "*Internist-I*, an experimental computer-based diagnostic consultant for general internal medicine," *New England Journal of Medicine*, vol. 307, no. 8, pp. 468–476, Aug. 19, 1982, ISSN: 0028-4793, 1533-4406. DOI: 10.1056/NEJM198208193070803 [Online]. Available: http://www.nejm.org/doi/abs/10.1056/NEJM198208193070803

[9] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997, ISBN: 978-0070428072.

[10] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995, ISSN: 0885-6125, 1573-0565. DOI: 10.1007/BF00994018 [Online]. Available: http://link.springer.com/10.1007/BF00994018

[11] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001, ISSN: 0885-6125, 1573-0565. DOI: 10.1023/A:1010933404324 [Online]. Available: https://link.springer.com/10.1023/A:1010933404324

[12] L. Melkumova and S. Shatskikh, "Comparing ridge and LASSO estimators for data analysis," *Procedia Engineering*, vol. 201, pp. 746–755, 2017, ISSN: 18777058. DOI: 10.1016/j.proeng.2017.09.615 [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1877705817341474

[13] A. Rajkomar, J. Dean, and I. Kohane, "Machine learning in medicine," *New England Journal of Medicine*, vol. 380, no. 14, pp. 1347–1358, Apr. 4, 2019, ISSN: 0028-4793, 1533-4406. DOI: 10.1056/NEJMra1814259 [Online]. Available: http://www.nejm.org/doi/10.1056/NEJMra1814259

[14] G. Y. Vybhavi, G. Sriramya, V. Y. Bharadwaj, and G. Ramesh, "Clustering algorithms in data science: Evaluating the time and space complexities of k-means, DBSCAN, and hierarchical methods," presented at the 15TH INTERNATIONAL CONFERENCE ON MATERIALS PROCESSING AND CHARACTERIZATION 2023, Newcastle, England,

2024, p. 050 004. DOI: 10.1063/5.0215042 [Online]. Available: https://pubs.aip.org/aip/acp/article-lookup/doi/10.1063/5.0215042

[15] D. A. Aliyu, E. A. Patah Akhir, N. A. Osman, J. A. Salisu, Y. Saidu, and J. S. Yalli, "Optimization techniques in reinforcement learning for healthcare: A review," in *2024 8th International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, Pune, India: IEEE, Aug. 23, 2024, pp. 1–6, ISBN: 979-8-3503-9177-0. DOI: 10.1109/ICCUBEA61740.2024.10774698 [Online]. Available: https://ieeexplore.ieee.org/document/10774698/

[16] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 28, 2015, ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature14539 [Online]. Available: https://www.nature.com/articles/nature14539

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 24, 2017, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3065386 [Online]. Available: https://dl.acm.org/doi/10.1145/3065386

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90

[19] A. Vaswani et al., "Attention is all you need," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2017, pp. 5998–6008. [Online]. Available: https://papers.nips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[20] F. Shamshad et al., "Transformers in medical imaging: A survey," *Medical Image Analysis*, vol. 88, p. 102 802, Aug. 2023, ISSN: 13618415. DOI: 10.1016/j.media.2023.102802 [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1361841523000634

[21] M. Jaderberg et al., *Population based training of neural networks*, Nov. 28, 2017. DOI: 10.48550/arXiv.1711.09846 arXiv: 1711.09846[cs.LG]. [Online]. Available: http://arxiv.org/abs/1711.09846

[22] J. Liang, S. Gonzalez, H. Shahrzad, and R. Miikkulainen, *Regularized evolutionary population-based training*, Jul. 21, 2021. DOI: 10.48550/arXiv.2002.04225 arXiv: 2002.04225[cs.NE]. [Online]. Available: http://arxiv.org/abs/2002.04225

[23] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, no. 2, Jun. 1994, ISSN: 0960-3174, 1573-1375. DOI: 10.1007/BF00175354 [Online]. Available: http://link.springer.com/10.1007/BF00175354

[24] S. Gonzalez and R. Miikkulainen, "Improved training speed, accuracy, and data utilization through loss function optimization," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, Glasgow, United Kingdom: IEEE, Jul. 2020, pp. 1–8, ISBN: 978-1-7281-6929-3. DOI: 10.1109/CEC48606.2020.9185777 [Online]. Available: https://ieeexplore.ieee.org/document/9185777/

[25] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, 2009. [Online]. Available: https://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf

[26] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. [Online]. Available: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/37648.pdf

[27] S. Zagoruyko and N. Komodakis, *Wide residual networks*, Jun. 14, 2017. DOI: 10.48550/arXiv.1605.07146 arXiv: 1605.07146[cs.CV]. [Online]. Available: http://arxiv.org/abs/1605.07146

[28] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, PMLR, 2019, pp. 6105–6114. [Online]. Available: https://proceedings.mlr.press/v97/tan19a.html

[29] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI: IEEE, Jul. 2017, pp. 2261–2269, ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.243 [Online]. Available: https://ieeexplore.ieee.org/document/8099726/

[30] C. Szegedy et al., "Going deeper with convolutions," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html

[31] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[32] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, Jul. 2015, pp. 448–456. [Online]. Available: https://proceedings.mlr.press/v37/ioffe15.html

[33] A. G. Howard et al., *MobileNets: Efficient convolutional neural networks for mobile vision applications*, Apr. 17, 2017. DOI: 10.48550/arXiv.1704.04861 arXiv: 1704.04861[cs.CV]. [Online]. Available: http://arxiv.org/abs/1704.04861

[34] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, *Learning transferable architectures for scalable image recognition*, Apr. 11, 2018. DOI: 10.48550/arXiv.1707.07012 arXiv: 1707.07012[cs.CV]. [Online]. Available: http://arxiv.org/abs/1707.07012

[35] Y. Huang et al., "GPipe: Efficient training of giant neural networks using pipeline parallelism," *arXiv preprint arXiv:1811.06965*, vol. 32, 2019. [Online]. Available: https://arxiv.org/abs/1811.06965

[36] K. He, G. Gkioxari, P. Dollár, and R. Girshick, *Mask r-CNN*, Jan. 24, 2018. DOI: 10.48550/arXiv.1703.06870 arXiv: 1703.06870[cs.CV]. [Online]. Available: http://arxiv.org/abs/1703.06870

[37] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, *Focal loss for dense object detection*, Feb. 7, 2018. DOI: 10.48550/arXiv.1708.02002 arXiv: 1708.02002[cs.CV]. [Online]. Available: http://arxiv.org/abs/1708.02002

[38] M. Tan, B. Chen, R. Ning, T. Pang, and Q. V. Le, "MnasNet: Platform-aware neural architecture search for mobile," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2820–2828, 2019. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2019/html/Tan_MnasNet_Platform-Aware_Neural_Architecture_Search_for_Mobile_CVPR_2019_paper

[39] F. Chollet, *Xception: Deep learning with depthwise separable convolutions*, Apr. 4, 2017. DOI: 10.48550/arXiv.1610.02357 arXiv: 1610.02357[cs.CV]. [Online]. Available: http://arxiv.org/abs/1610.02357

[40] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, Feb. 12, 2017, ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v31i1.11231 [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11231

[41] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5987–5995. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/html/Xie_Aggregated_Residual_Transformations_CVPR_2017_paper.html

[42] X. Zhang, Z. Li, C. C. Loy, and D. Lin, *PolyNet: A pursuit of structural diversity in very deep networks*, Jul. 17, 2017. DOI: 10.48550/arXiv.1611.05725 arXiv: 1611.05725[cs.CV]. [Online]. Available: http://arxiv.org/abs/1611.05725

[43] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, *Squeeze-and-excitation networks*, May 16, 2019. DOI: 10.48550/arXiv.1709.01507 arXiv: 1709.01507[cs.CV]. [Online]. Available: http://arxiv.org/abs/1709.01507

[44] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 1, pp. 4780–4789, Jul. 17, 2019, ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v33i01.33014780 [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4405

[45] C. Liu et al., *Progressive neural architecture search*, Jul. 26, 2018. DOI: 10.48550/arXiv.1712.00559 arXiv: 1712.00559[cs.CV]. [Online]. Available: http://arxiv.org/abs/1712.00559

[46] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, *AutoAugment: Learning augmentation policies from data*, Apr. 11, 2019. DOI: 10.48550/arXiv.1805.09501 arXiv: 1805.09501[cs.CV]. [Online]. Available: http://arxiv.org/abs/1805.09501

[47] J. A. García Henao, F. Precioso, P. Staccini, and M. Riveill, "Diagnosenet: Automatic framework to scale neural networks on heterogeneous systems applied to medical diagnosis," in *IT Convergence and Security*, H. Kim and K. J. Kim, Eds., vol. 712, Series Title: Lecture

Notes in Electrical Engineering, Singapore: Springer Singapore, 2021, pp. 3–17, ISBN: 978-981-15-9353-6 978-981-15-9354-3. DOI: 10.1007/978-981-15-9354-3_1 Accessed: Jun. 16, 2026. [Online]. Available: http://link.springer.com/10.1007/978-981-15-9354-3_1

[48] J. A. García H., E. Hernández B., C. E. Montenegro, P. O. Navaux, and C. J. Barrios H., "enerGyPU and enerGyPhi monitor for power consumption and performance evaluation on nvidia tesla GPU and intel xeon phi," in *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, Cartagena: IEEE, May 2016, pp. 718–725, ISBN: 978-1-5090-2453-7. DOI: 10.1109/CCGrid.2016.100 [Online]. Available: https://ieeexplore.ieee.org/document/7515761/

[49] V. Bolón-Canedo, L. Morán-Fernández, B. Cancela, and A. Alonso-Betanzos, "A review of green artificial intelligence: Towards a more sustainable future," *Neurocomputing*, vol. 599, p. 128 096, Sep. 2024, ISSN: 09252312. DOI: 10.1016/j.neucom.2024.128096 Accessed: Jun. 2, 2026. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0925231224008671

[50] N. Alizadeh and F. Castor, "Green AI: A preliminary empirical study on energy consumption in DL models across different runtime infrastructures," in *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI*, Apr. 14, 2024, pp. 134–139. DOI: 10.1145/3644815.3644967 arXiv: 2402.13640[cs.SE]. [Online]. Available: http://arxiv.org/abs/2402.13640

[51] A. Berthelot, E. Caron, M. Jay, and L. Lefèvre, "Understanding the environmental impact of generative AI services," *Communications of the ACM*, vol. 68, no. 7, pp. 46–53, Jul. 2025, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3725984 [Online]. Available: https://dl.acm.org/doi/10.1145/3725984

[52] C. E. Tripp et al., *Measuring the energy consumption and efficiency of deep neural networks: An empirical analysis and design recommendations*, Version Number: 1, 2024. DOI: 10.

48550/ARXIV.2403.08151 [Online]. Available: https://arxiv.org/abs/2403.08151

[53] K. Kirkpatrick, "The carbon footprint of artificial intelligence," *Communications of the ACM*, vol. 66, no. 8, pp. 17–19, Aug. 2023, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3603746 [Online]. Available: https://dl.acm.org/doi/10.1145/3603746

[54] S. Li, C. Yan, and Y. Liu, "Energy efficiency and coding of neural network," *Frontiers in Neuroscience*, vol. 16, p. 1 089 373, Jan. 11, 2023, ISSN: 1662-453X. DOI: 10.3389/fnins.2022.1089373 [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2022.1089373/full

[55] OECD, "Measuring the environmental impacts of artificial intelligence compute and applications," 2022. [Online]. Available: https://www.oecd.org/content/dam/oecd/en/publications/reports/2022/11/measuring-the-environmental-impacts-of-artificial-intelligence-compute-and-applications_3dddded5/7babf571-en.pdf

[56] S. Hanifi, A. Cammarono, and H. Zare-Behtash, "Advanced hyperparameter optimization of deep learning models for wind power prediction," *Renewable Energy*, vol. 221, p. 119 700, Feb. 2024, ISSN: 09601481. DOI: 10.1016/j.renene.2023.119700 [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0960148123016154

[57] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019. [Online]. Available: http://jmlr.org/papers/v20/18-598.html

[58] C. Yu et al., "GPT-NAS: Neural architecture search meets generative pre-trained transformer model," *Big Data Mining and Analytics*, vol. 8, no. 1, pp. 45–64, Feb. 2025, ISSN: 2096-0654, 2097-406X. DOI: 10.26599/BDMA.2024.9020036 [Online]. Available: https://ieeexplore.ieee.org/document/10681256/

[59] B. Zoph and Q. V. Le, *Neural architecture search with reinforcement learning*, Feb. 15, 2017. DOI: 10.48550/arXiv.1611.01578 arXiv: 1611.01578[cs.LG]. [Online]. Available: http://arxiv.org/abs/1611.01578

[60] E. Real et al., "Large-scale evolution of image classifiers," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, PMLR, 2017, pp. 2902–2911.

[61] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing, "Neural architecture search with bayesian optimisation and optimal transport," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/f33ba15effa5c10e873bf3842afb46a6-Paper.pdf

[62] H. Liu, K. Simonyan, and Y. Yang, *DARTS: Differentiable architecture search*, Version Number: 2, 2018. DOI: 10.48550/ARXIV.1806.09055 [Online]. Available: https://arxiv.org/abs/1806.09055

[63] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *International Conference on Machine Learning (ICML)*, vol. 80, PMLR, 2018, pp. 4095–4104. [Online]. Available: https://proceedings.mlr.press/v80/pham18a.html

[64] H. Benmeziane, I. Hamzaoui, Z. Cherif, and K. El Maghraoui, "Medical neural architecture search: Survey and taxonomy," in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, 2024, pp. 7932–7940. DOI: 10.24963/ijcai.2024/878 [Online]. Available: https://www.ijcai.org/proceedings/2024/878

[65] C. Yang et al., *Large language models as optimizers*, Apr. 15, 2024. DOI: 10.48550/arXiv.2309.03409 arXiv: 2309.03409[cs.LG]. Accessed: May 31, 2026. [Online]. Available: http://arxiv.org/abs/2309.03409

[66] J. A. Baktash and M. Dawodi, *Gpt-4: A review on advancements and opportunities in natural language processing*, Version Number: 1, 2023. DOI: 10.48550/ARXIV.2305.03195 Accessed: May 25, 2026. [Online]. Available: https://arxiv.org/abs/2305.03195

[67] A. Koubaa, *GPT-4 vs. GPT-3.5: A concise showdown*, Mar. 24, 2023. DOI: 10.20944/preprints202303.0422.v1 Accessed: May 25, 2026. [Online]. Available: https://www.preprints.org/manuscript/202303.0422/v1

[68] H. Touvron et al., *Llama 2: Open foundation and fine-tuned chat models*, Jul. 19, 2023. DOI: 10.48550/arXiv.2307.09288 arXiv: 2307.09288[cs.CL]. Accessed: May 31, 2026. [Online]. Available: http://arxiv.org/abs/2307.09288

[69] C. Garbin, P. Rajpurkar, J. Irvin, M. P. Lungren, and O. Marques, *Structured dataset documentation: A datasheet for CheXpert*, Version Number: 1, 2021. DOI: 10.48550/ARXIV.2105.03020 [Online]. Available: https://arxiv.org/abs/2105.03020

[70] J. Irvin et al., *CheXpert: A large chest radiograph dataset with uncertainty labels and expert comparison*, Jan. 21, 2019. DOI: 10.48550/arXiv.1901.07031 arXiv: 1901.07031[cs.CV]. [Online]. Available: http://arxiv.org/abs/1901.07031

[71] H. H. Pham, T. T. Le, D. Q. Tran, D. T. Ngo, and H. Q. Nguyen, *Interpreting chest x-rays via CNNs that exploit hierarchical disease dependencies and uncertainty labels*, Version Number: 3, 2019. DOI: 10.48550/ARXIV.1911.06475 [Online]. Available: https://arxiv.org/abs/1911.06475

[72] Z. Yuan, Y. Yan, M. Sonka, and T. Yang, "Large-scale robust deep AUC maximization: A new surrogate loss and empirical studies on medical image classification," 2020, Version Number: 2. DOI: 10.48550/ARXIV.2012.03173 [Online]. Available: https://arxiv.org/abs/2012.03173

[73] F. Mejía Cajicá, J. A. García Henao, C. J. Barrios Hernández, and M. Riveill, "Efficiency in the classification of chest x-ray images through generative parallelization of the neural architecture search," *ACI Avances en Ciencias e Ingenierías*, vol. 17, no. 1, May 9, 2025,

ISSN: 2528-7788, 1390-5384. DOI: 10.18272/aci.v17i1.3707 [Online]. Available: https://revistas.usfq.edu.ec/index.php/avances/article/view/3707

[74] S. Matsuoka, "Cambrian explosion of computing and big data in the post-moore era," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, Tempe Arizona: ACM, Jun. 11, 2018, pp. 105–105, ISBN: 978-1-4503-5785-2. DOI: 10.1145/3208040.3225055 [Online]. Available: https://dl.acm.org/doi/10.1145/3208040.3225055

[75] Shedrack Onwusinkwue et al., "Artificial intelligence (AI) in renewable energy: A review of predictive maintenance and energy optimization," *World Journal of Advanced Research and Reviews*, vol. 21, no. 1, pp. 2487–2799, Jan. 30, 2024, ISSN: 25819615. DOI: 10.30574/wjarr.2024.21.1.0347 Accessed: May 25, 2026. [Online]. Available: https://wjarr.com/node/4358

[76] I. Hidalgo et al., *Sustainable artificial intelligence systems: An energy efficiency approach*, Dec. 2, 2023. DOI: 10.36227/techrxiv.24610899.v1 [Online]. Available: https://www.techrxiv.org/doi/full/10.36227/techrxiv.24610899.v1

[77] B. C. Rao, "Frugal computing for artificial intelligence and other applications," in *Frugal Engineering*. Singapore: Springer Nature Singapore, 2024, pp. 209–213, Series Title: Design Science and Innovation, ISBN: 978-981-99-9699-5 978-981-99-9700-8. DOI: 10.1007/978-981-99-9700-8_11 [Online]. Available: https://link.springer.com/10.1007/978-981-99-9700-8_11

[78] A. Al Kuwaiti et al., "A review of the role of artificial intelligence in healthcare," *Journal of Personalized Medicine*, vol. 13, no. 6, p. 951, Jun. 5, 2023, ISSN: 2075-4426. DOI: 10.3390/jpm13060951 Accessed: May 25, 2026. [Online]. Available: https://www.mdpi.com/2075-4426/13/6/951

[79] T. Davenport and R. Kalakota, "The potential for artificial intelligence in healthcare," *Future Healthcare Journal*, vol. 6, no. 2, pp. 94–98, Jun. 2019, ISSN: 25146645. DOI:

10.7861/futurehosp.6-2-94 Accessed: May 25, 2026. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2514664524010592

[80] I. Calciu et al., "Using local cache coherence for disaggregated memory systems," *ACM SIGOPS Operating Systems Review*, vol. 57, no. 1, pp. 21–28, Jun. 26, 2023, ISSN: 0163-5980. DOI: 10.1145/3606557.3606561 Accessed: May 25, 2026. [Online]. Available: https://dl.acm.org/doi/10.1145/3606557.3606561

[81] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, Philadelphia PA USA: ACM, Aug. 20, 2006, pp. 535–541, ISBN: 978-1-59593-339-3. DOI: 10.1145/1150402.1150464 [Online]. Available: https://dl.acm.org/doi/10.1145/1150402.1150464

[82] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, *The computational limits of deep learning*, Version Number: 2, 2020. DOI: 10.48550/ARXIV.2007.05558 [Online]. Available: https://arxiv.org/abs/2007.05558

[83] Kaggle. "Nih chest x-rays," Accessed: Sep. 1, 2025. [Online]. Available: https://www.kaggle.com/datasets/nih-chest-xrays/data

[84] Omkarmanohardalvi, *Lungs disease dataset (4 types)*, 2022. [Online]. Available: https://www.kaggle.com/datasets/omkarmanohardalvi/lungs-disease-dataset-4-types

[85] P. Li, S. Wang, T. Li, J. Lu, Y. HuangFu, and D. Wang, *A large-scale CT and PET/CT dataset for lung cancer diagnosis*, version 5, 2020. DOI: 10.7937/TCIA.2020.NNC2-0461 [Online]. Available: https://www.cancerimagingarchive.net/collection/lung-pet-ct-dx/

[86] Scikit-learn Developers, *Sklearn.modelselection.groupshuffflesplit*, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GroupShuffleSplit.html

[87] S. Patel. "Mobilenet v2 architecture in computer vision." Last updated: 23 Jul, 2025, GeeksforGeeks, Accessed: Sep. 1, 2025. [Online]. Available: https://www.geeksforgeeks.org/computer-vision/mobilenet-v2-architecture-in-computer-vision/

[88] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, *MobileNetV2: Inverted residuals and linear bottlenecks*, Mar. 21, 2019. DOI: 10.48550/arXiv.1801.04381 arXiv: 1801.04381[cs.CV]. [Online]. Available: http://arxiv.org/abs/1801.04381

[89] Pytorch, *Bcewithlogitsloss*, *PyTorch Documentation* (stable), version 2.x. [Online]. Available: https://docs.pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html

[90] P. Yang et al., "Multi-label knowledge distillation," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, Paris, France: IEEE, Oct. 1, 2023, pp. 17 225–17 234, ISBN: 979-8-3503-0718-4. DOI: 10.1109/ICCV51070.2023.01584 [Online]. Available: https://ieeexplore.ieee.org/document/10378178/

[91] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, Version Number: 3, 2017. DOI: 10.48550/ARXIV.1711.05101 [Online]. Available: https://arxiv.org/abs/1711.05101

[92] scikit-learn developers, *Scikit-learn: Machine learning in python*, 2024. [Online]. Available: https://scikit-learn.org

[93] E. Ben-Baruch et al., *Asymmetric loss for multi-label classification*, Version Number: 4, 2020. DOI: 10.48550/ARXIV.2009.14119 [Online]. Available: https://arxiv.org/abs/2009.14119

[94] A. De Vries, "The growing energy footprint of artificial intelligence," *Joule*, vol. 7, no. 10, pp. 2191–2194, Oct. 2023, ISSN: 25424351. DOI: 10.1016/j.joule.2023.09.004 [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2542435123003653

[95] D. Mhlanga, *Energy efficiency in AI and how power consumption impedes innovation*, 2025. DOI: 10.2139/ssrn.5222616 [Online]. Available: https://www.ssrn.com/abstract=5222616

[96] V. Mehlin, S. Schacht, and C. Lanquillon, *Towards energy-efficient deep learning: An overview of energy-efficient approaches along the deep learning lifecycle*, Version Number: 1, 2023. DOI: 10.48550/ARXIV.2303.01980 [Online]. Available: https://arxiv.org/abs/2303.01980

[97] T. Yarally, L. Cruz, D. Feitosa, J. Sallou, and A. Van Deursen, "Uncovering energy-efficient practices in deep learning training: Preliminary steps towards green AI," in *2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, Melbourne, Australia: IEEE, May 2023, pp. 25–36, ISBN: 979-8-3503-0113-7. DOI: 10.1109/CAIN58948.2023.00012 [Online]. Available: https://ieeexplore.ieee.org/document/10164743/

[98] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, "Compute and energy consumption trends in deep learning inference," 2021, Version Number: 2. DOI: 10.48550/ARXIV.2109.05472 [Online]. Available: https://arxiv.org/abs/2109.05472

[99] T. Aslan, P. Holzapfel, L. Stobbe, A. Grimm, N. F. Nissen, and M. Finkbeiner, "Toward climate neutral data centers: Greenhouse gas inventory, scenarios, and strategies," *iScience*, vol. 28, no. 1, p. 111 637, Jan. 2025, ISSN: 25890042. DOI: 10.1016/j.isci.2024.111637 [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2589004224028645

[100] K. Zuchniak, *Multi-teacher knowledge distillation as an effective method for compressing ensembles of neural networks*, Feb. 14, 2023. DOI: 10.48550/arXiv.2302.07215 arXiv: 2302.07215[cs.LG]. [Online]. Available: http://arxiv.org/abs/2302.07215