An Exploratory Study on Quantifying the Operational Carbon Footprint of Vessel Segmentation

Noémie Debroux[1][0000−0002−5329−5485], Mathéo Galuba[1], and Antoine Vacavant[1][0000−0001−9616−3282]

Université Clermont Auvergne, Clermont Auvergne INP, CNRS, Institut Pascal, Clermont–Ferrand, F-63000, France
noemie.debroux@uca.fr

Abstract. With the increasing number of deep learning-based solutions for a wide range of problems, the environmental impact of these models has become crucial. With the digital field representing an increasing rate of carbon emissions, it becomes urgent to question the ecological footprint of software developments. That's the question we address in this work with an exploratory study relying on the CodeCarbon tool to quantify carbon emissions. An extensive comparative study of recent deep learning models for retinal vessel segmentation is conducted. We look at several quality metrics against carbon emissions and show that some architectures are energy-intensive with a very low improvement in the quality.

Keywords: Sustainable software · Energy accounting · CodeCarbon · Carbon emissions · Vessel segmentation

1 Introduction

The rise in computational demand with recent digital innovations and digital transition comes with serious preoccupations about its environmental impact. Additionally, the energy required to power data centers, communication networks, and users' terminals follows an exponential growth nourished by the widespread adoption of resource-intensive technologies such as generative Artificial Intelligence (AI). This trajectory requires urgent decarbonization efforts across all sectors. Within this context, improving software energy efficiency has emerged as an essential aspect of software development, shifting the focus from performance to a balance against sustainability [1, 4, 8].

The first step towards developing more sustainable software is establishing robust and standardized carbon accounting methodologies [10]. However, quantifying carbon emissions is a complex, multi-variable problem. Those variables are highly dynamic and often require reliance on estimations rather than precise measurements. Indeed, expanding the scope of measurement while aiming for comprehension can introduce greater uncertainty and inaccuracy, making the pursuit of a precise environmental impact assessment a serious challenge [13].

This field currently lacks the mature tools and clear benchmarks needed to turn this conceptual awareness into actionable practice. This paper addresses this gap through a series of empirical studies that provide a multifaceted analysis of software development's carbon emissions, using the CodeCarbon tool [3]. We first review current carbon accounting methodologies and tools and justify our choice. We then compare multiple solutions through an applied case study in medical image analysis, including a non-AI algorithm and state-of-the-art deep learning models, demonstrating how performance can be balanced against environmental cost. This comparative study provides actionable insights for developing more sustainable software along with a toolkit and methodologies applicable across multiple paradigms [7]. To the best of our knowledge, this is the first time that such a paper provides an extensive evaluation of recent deep learning models by combining performance metrics with carbon footprint, with a focus on the retinal vessel segmentation task, which is crucial for health care applications.

Our analysis is confined to quantifying operational carbon emissions originating directly from electricity consumption during computation. This scope necessarily excludes the embodied carbon attributable to the manufacturing, transportation, and end-of-life processing of the hardware itself, which constitutes a non-trivial portion of the technology sector's total footprint. While we acknowledge this as a significant limitation for a complete life-cycle assessment, the focus on operational emissions is justified as it represents the primary factor directly influenced by the design and runtime decisions of software developers.

2 Review of Carbon Emission Assessment Software

2.1 From energy consumption to carbon emissions

Converting Energy Use to Carbon Emissions The conversion of a software's energy consumption into carbon emissions hinges on two variables: the total energy consumed (E) during the software runtime, measured in kilowatt-hours (kWh), and the carbon intensity (CI) of the electricity grid the machine is attached to, expressed in grams of CO_2-equivalent per kWh (gCO_2eq/kWh). These are related through: Emissions $= E \times CI$. This multiplicative relationship underscores a critical insight: identical energy usage can yield vastly different carbon footprints depending on the power grid. Therefore, cloud and delocalised computing should be addressed carefully.

Carbon Intensity Significance and Variability The carbon intensity is calculated using a weighted average of emissions from various energy sources, represented by electricity from source i (E_i) and their corresponding lifecycle emission factors (EF_i). For example, coal power exhibits an emission factor of $EF_{coal} = 888$ tons CO_2eq/GWh reflecting its carbon-intensive combustion process. Whereas wind energy has a low emission factor of $EF_{wind} = 26$ tons CO_2eq/GWh, primarily from turbine manufacturing and maintenance [14].

Carbon intensity is a critical determinant in software emission accounting due to its multiplicative relationship with energy consumption and inherent spatiotemporal variability. As demonstrated by [4], computational workloads can achieve carbon reductions through strategic geographical shifting (e.g., routing to low-carbon grids) or temporal shifting (e.g., pausing/resuming operations during high-renewable availability periods), without altering accuracy.

Existing Methodologies and Tools Carbon emission accounting for software systems employs methodologies ranging from predictive estimation to direct measurement, each offering distinct advantages and limitations. These approaches are categorized by their data collection mechanisms and implementation requirements.

Online calculators such as *Green Algorithms* (https://www.green-algorithms.org/) and *ML CO2 Impact*(https://mlco2.github.io/impact/) provide accessible emission estimates based on user-input parameters, including hardware configuration, runtime duration, and geographical region. These tools apply standardized hardware power models multiplied by regionally averaged carbon intensity values to compute emissions. These calculators serve best for preliminary assessments rather than precise accounting [1].

Runtime monitoring tools like *CodeCarbon* [3], *Carbontracker* (https://carbontracker.org/), and *Eco2AI* (https://github.com/sb-ai-lab/Eco2AI) represent an intermediate approach that directly profiles energy consumption during execution. By accessing hardware performance counters (e.g., *Intel RAPL*, *NVIDIA SMI*) and operating system telemetry, these tools measure actual power usage at process or system levels while frequently integrating time-resolved carbon intensity data through APIs. Software-based approaches offer the best balance between accuracy and practicality for ongoing operational assessment.

External power measurement devices such as *Yokogawa WT310E* (https://cdn.tmi.yokogawa.com/IMWT310E-01EN.pdf) provide the highest accuracy through physical measurement of current and voltage between power sources and target systems. These instruments calculate full-system consumption, including peripheral devices. Practical implementation, however, requires physical access to equipment, device-specific calibration, and faces challenges in isolating individual process consumption within shared environments. These constraints limit their deployment primarily to controlled settings where maximum precision is required, such as datacenters.

2.2 In-Depth Analysis of CodeCarbon

To quantify the carbon emissions of our experiments, we adopt CodeCarbon [3] as a measurement tool. This choice is driven by its trade-off between measurement accuracy and practical ease of use. Furthermore, CodeCarbon benefits from active maintenance, growing popularity, and empirical validation with multiple studies [1, 8] demonstrating strong agreement with hardware-based energy monitoring with an error inferior to 15%.

CodeCarbon dynamically adapts its measurement strategy based on system capabilities and execution environment. Its accuracy depends on factors such as operating system, hardware support, and privilege levels—with optimal precision achieved when running on GNU/Linux systems with root access. The tool dynamically resolves geographic context via geolocation APIs or user-defined coordinates, fetching real-time regional carbon intensity (gCO_2eq/kWh) from sources like *ElectricityMap.org*. This approach ensures contextually appropriate accuracy across heterogeneous experimental environments.

While CodeCarbon provides state-of-the-art emissions estimates, several inherent limitations should be noted. The tool does not account for energy consumption from peripheral components, including storage, networking, or power supply inefficiencies. Additionally, the monitoring process itself introduces computational overhead, though empirical studies consistently measure this at below 1% [1, 8] of total runtime. Although this marginal impact does not substantially affect our comparative analyses, it is important to acknowledge these limitations. For absolute energy accounting, supplementary hardware-based measurements may be necessary.

3 Evaluation of Deep Retinal Vessel Segmentation Models

The case study in medical image analysis is useful, since clinical deployment can involve large-scale inference where per-image emissions matter.

3.1 Clinical Dataset

Fundus photography is a common task in ophthalmology as it is a non-invasive and low-cost imaging technique. The aspect and vascular structure of the retina enable clinicians to detect at an early stage cardiovascular and ocular pathologies such as diabetic retinopathy, hypertension, and glaucoma. Automating this process addresses the limitations of manual analysis, which is time-intensive and prone to observer variability, thereby enhancing the scalability of screening programs, particularly in resource-limited settings.

We use the CHASE_DB1 dataset, a publicly available subset of the Child Heart and Health Study in England (CHASE). It contains 28 999×960 retinal fundus images from both eyes of 14 children [5]. Two experts manually annotated each image, and the first annotation served as the ground truth. CHASE_DB1 is widely used and compatible with all model codebases we selected. Additionally, it presents relevant segmentation challenges such as low vessel contrast, uneven illumination, and reflection. Different factors, including eye pathologies, generate such artefacts.

3.2 Classical Approach

Theoretical Foundation The Frangi filter is a classic image processing technique designed to detect tubular structures like blood vessels by analyzing local patterns. This algorithm leverages the Hessian matrix, a mathematical tool that

measures how intensity values change in different directions around each pixel. When applied to retinal images, it identifies vessels by detecting their characteristic "ridge-like" structure. The model relies on the following formula:

$$F = \begin{cases} 1 - exp(-\frac{R_a^2}{2\alpha^2})exp(-\frac{R_b^2}{2\beta^2})(1 - exp(-\frac{S^2}{2\gamma^2})) & \text{if } \lambda_2, \lambda_3 < 0 \\ 0 & \text{else} \end{cases},$$

$$R_a = \frac{|\lambda_2|}{|\lambda_3|}, R_b = \frac{|\lambda_1|}{\sqrt{|\lambda_2\lambda_3|}}, S = \|H\| = \sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2},$$

λ_1, λ_2, and λ_3 being the eigen values of the Hessian matrix H of the original image, with $\lambda_1 \leq \lambda_2 \leq \lambda_3$.

It involves a multiscale approach, which applies the same analysis across scales $\sigma \in [\sigma_{\min}, \sigma_{\max}]$ with step size σ_{step}. The final response combines these scales by preserving the strongest vessel probability across all resolutions: $V = \max_\sigma (V_\sigma)$. Three correction constants refine the detection: α penalizes plate-like deviations; β suppresses blob-like structures; γ normalizes responses in high-variance regions. This algorithm outputs each pixel's likelihood of belonging to a tubular structure.

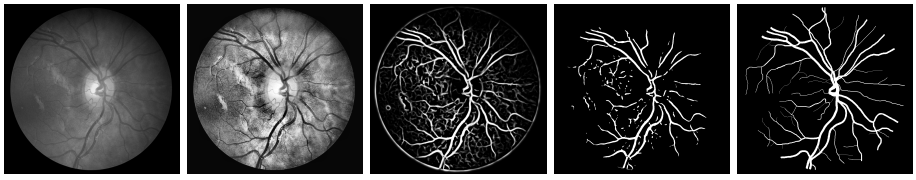
Implementation for Fundus Imaging As illustrated in Figure 1, we also implement a few pre- and post-processing steps. First, fundus images undergo local contrast transformation (CLAHE Figure 1b). Post-processing (Figure 1d) then addresses two characteristic artefacts: the acquisition-induced circular boundary is eliminated through a predefined mask, while isolated pixel noise is suppressed via morphological area opening that removes disconnected components smaller than 10 pixels. These adaptations maintain the method's computational frugality, adding negligible overhead (0.1203 ± 0.0157 sec/image) to the core Frangi filter operation (2.4906 ± 0.0031 sec/image) on the CHASE_DB1 dataset.

Optimization For a fair comparison with learning-based approaches, we propose to take into account the tedious phase of parameter optimization. Given the seven parameters governing Frangi filter performance, we employ a 3-stage grid search strategy to efficiently navigate the 7D optimization space. We denote $\mathcal{U}^n_{[a,b]}$: Uniform grid of n values in $[a, b]$

1. Scale: $\begin{matrix} \sigma_{\min} \in \mathcal{U}^{20}_{[0.1,\ 10]} \\ \sigma_{\max} \in \mathcal{U}^{20}_{[0.1,\ 10]} \\ \sigma_{\text{step}} \in \mathcal{U}^{10}_{[0.1,\ 4.0]} \end{matrix}$ s.t. $\begin{matrix} \sigma_{\max} - \sigma_{\min} \geq 0.1 \\ \left\lfloor \frac{\sigma_{\max} - \sigma_{\min}}{\sigma_{\text{step}}} \right\rfloor < 5 \end{matrix}$

   During this stage, α, β, and γ are fixed to default suboptimal values. The feasible space represents 1526 combinations.
2. Correction constants: $\alpha \in \mathcal{U}^{20}_{[0.001,\ 1.0]}$, $\beta \in \mathcal{U}^{20}_{[1.0,\ 20.0]}$, $\gamma \in \mathcal{U}^{5}_{[0.001,\ 0.01]}$. The search space is $20 \times 20 \times 5 = 2000$
3. Threshold: $\tau \in \mathcal{U}^{50}_{[0.01,\ 0.99]}$.

This staged approach achieves a search space size significantly smaller than the 7D unconstrained search space by eliminating lots of unnecessary tests.



(a) Input (b) CLAHE (c) Frangi (d) Post-proc. (e) GT

Fig. 1: Stages of the segmentation pipeline with the Frangi filter algorithm

3.3 Modern AI approaches

In Table 1, we summarise the main technical implementation details of the studied models. Hereafter, we briefly describe each deep model.

Table 1: Main features of deep models implemented in our study. For all models, the optimizer is Adam.

Model	**# epochs**	**Batch size**	**Learning rate**	**# params**	**Loss**	**# train images**
U-Net	40 ep	4	0.0001	31 million	CE	28
W-Net	40 cycles, 50 ep	4	10^{-8} to 0.01	68,000	CE	6
Chivet *et al.*	150 ep	2	0.001	62 million	BCE+BZ	4
FR-UNet	40 ep	512	0.0001	7 million	BCE	20
FSG-Net	100 cycles, 20 ep warmup	4	0.001	18 million	Dice+BCE	14

U-Net The U-Net architecture [11] features a symmetric encoder-decoder structure with skip connections that preserve spatial details during segmentation. This design combines high-resolution features from the contracting path with upsampled features from the expanding path. For our implementation, we employed a standard 4-stage U-Net.

W-Net W-Net [6] is a lightweight model composed of two cascaded little 3-stage U-Nets. The cascade architecture allows progressive refinement across stages. Despite its minimal size and training on only 6 images, this configuration achieves strong segmentation accuracy by progressively refining features across cascaded stages.

Chivet et al. Approach This model from [2] adopts a cascaded architecture of two standard U-Nets incorporating second-order features derived from the Hessian matrix, which effectively highlight thin tubular structures like blood vessels.

FR U-Net The Full Resolution U-Net [9] preserves high-resolution representations throughout its architecture to maintain pixel-level accuracy, incorporating Multiresolution Convolution Interactive (MRCI) mechanisms and Feature Aggregation Modules (FAMs) for multi-scale segmentation. The segmentation is refined with Dual-Threshold Iteration post-processing to enhance topological coherence by recovering weak vessel pixels.

FSG-Net The Full-Scale Guided Network [12] extends the U-Net architecture with Guided Residual Modules (GRMs) that refine features using attention maps from full-scale inputs. Training employed a WarmupCosine scheduler (100 cycles, 20-epoch warmup) with early stopping and data augmentation.

3.4 Measuring Setup

Performance Metrics Given the inherent class imbalance in vascular segmentation tasks, where background pixels dominate the image, the commonly reported accuracy is not particularly informative, as models biased toward negative predictions may yield misleadingly high scores. To ensure a more rigorous evaluation, we employ other appropriate metrics: the Dice coefficient (F1-score) and mean Intersection over Union (mIoU) quantify pixel-wise overlap between predictions and ground truth, emphasizing spatial coherence. The area under the ROC curve (AUC-ROC) evaluates threshold-agnostic performance by analyzing sensitivity-specificity balance. Sensitivity and specificity delineate error sources. Dice and mIoU are prioritized as they are discriminative in this context.

Carbon Emissions Quantification Environmental impact was quantified using CodeCarbon in `machine` mode. To avoid diurnal fluctuations, emissions are normalized using France's 2024 mean carbon intensity. This time-agnostic approach enables consistent comparisons while reflecting real-world operational emissions since the physical machines are located in France. Hyperparameters followed original publications without emission-oriented tuning.

Visualization Strategy Results were visualized through per-metric scatter plots, with emissions on the x-axis and performance metrics on the y-axis. This dual-axis approach positions optimal models in the top-left quadrant (high performance, low emissions), enabling immediate visual identification of efficient solutions across sustainability and performance. A composite metric combining performance and emissions was deliberately avoided due to inherent limitations: such an aggregation would impose restrictions, making the existing trade-offs between the two dimensions blurry. Our visualization preserves the independence of every variable, which makes the model selection easier. This approach also reveals non-linear relationships that a single metric would fail to capture.

3.5 Training Emissions

Experimental results are presented with per-metric scatter plots in Figure 2 and numerical values in Table 2 (columns 2-8).

Firstly, the dual-axis plots reveal no clear correlation between carbon emissions and model performance, underscoring the importance of a per-model evaluation. Instead, we observe significant variability on the emissions axis. Most strikingly, FSG-Net's consumption is an outlier, residing an order of magnitude higher than its nearest competitor, an immense cost that is not proportionally justified by its performance gains.

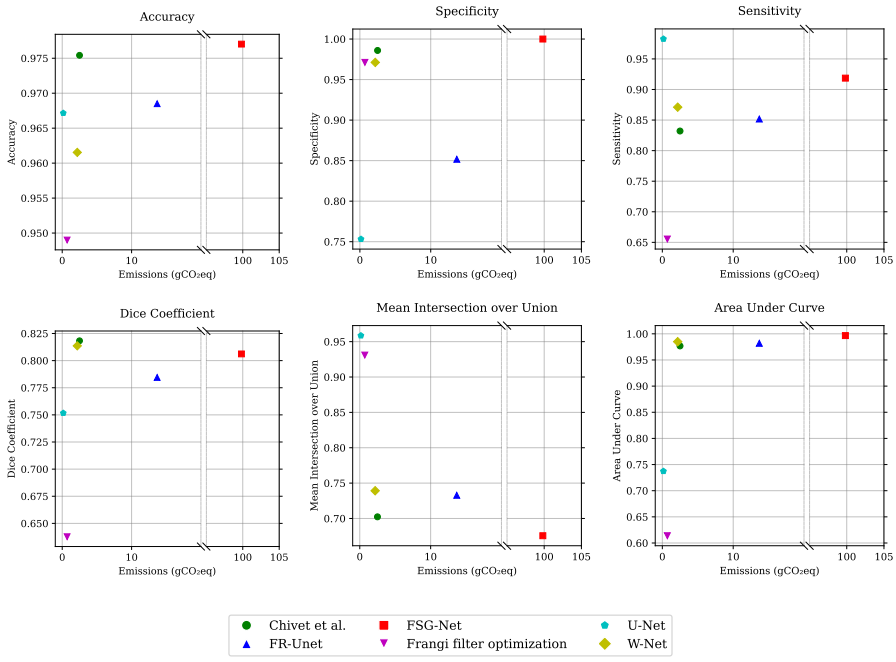


Fig. 2: Comparison of performance assessed with eight metrics versus carbon emissions for training multiple models on the task of retinal vessel segmentation.

Surprisingly, the Frangi filter optimization, the baseline non-AI model, does not emit the least carbon; slightly above the U-Net, its emissions remain negligible compared to the deep learning cohort. Ultimately, the filter's limitations, such as its dependency on input image quality and inability to distinguish vessels from certain artefacts, are easily recognizable on the performance axis compared to other solutions.

Focusing on other U-Net-based models, the analysis reveals a complex landscape where architectural efficiency does not always dictate environmental cost. This is exemplified by the significant disparity in emissions between models. For instance, while FSG-Net delivers strong results, simpler and more efficient architectures like W-Net or the one provided by Chivet et al. achieve comparable—and in some metrics, even higher—performance with a drastically lower environmental footprint. This disproportionate increase in emissions for marginal performance gains challenges the necessity of such resource-intensive approaches. The emissions difference is therefore largely unwanted, representing an inefficiency rather than a justified trade-off for superior results. It underscores that the pursuit of state-of-the-art performance, without consideration for architectural efficiency, can lead to models whose environmental cost is unsustainable,

Table 2: Quantitative results, obtained for training phase (carbon emissions and segmentation metrics), and carbon footprint for inference phase (for 1 image). The best result for each metric is in bold, and the second best is underlined.

Model	**Em. train (gCO2e)**	**ACC**	**SEN**	**SPE**	**DICE**	**MIOU**	**AUC**	**Em. inf. (10^{-3} gCO2e)**
Frangi	<u>0.71</u>	0.949	0.665	0.971	0.638	<u>0.931</u>	0.614	5.371
U-Net	**0.15**	0.967	**0.982**	0.753	0.752	**0.956**	0.737	**0.353**
Chivet et al.	2.49	<u>0.975</u>	0.832	<u>0.985</u>	**0.818**	0.702	0.977	1.401
FR-Unet	13.68	0.968	0.851	0.851	0.784	0.733	0.982	15.824
FSG-Net	99.82	**0.977**	<u>0.918</u>	**1.000**	0.806	0.676	**0.996**	5.707
W-Net	2.17	0.961	0.871	0.971	<u>0.814</u>	0.739	<u>0.985</u>	<u>0.762</u>

especially for widespread clinical deployment, where the practical difference in output may be minimal.

3.6 Inference Emissions

To evaluate the long-term operational impact of model deployment, we extended our emissions analysis beyond the training phase to quantify the carbon cost per inference presented in Table 2 (last column). Most importantly, the hierarchy seen during training is distinct from the emissions pattern during inference. The classical Frangi filter exhibited a surprisingly high inference cost, a direct consequence of our CPU-based Python implementation compared to the other GPU-accelerated deep learning models. Conversely, the standard U-Net proved to emit the least emissions during inference, benefiting from highly optimized deep learning frameworks. At the far end of the spectrum, the FR U-Net, which implements a custom post-processing algorithm (dual threshold), rendered it the most computationally expensive model per image. The conclusion is that inference costs are predominantly dictated by implementation choices and architectural complexity rather than raw performance. Since this per-image cost is multiplied by the scale of clinical deployment, selecting a model that is both accurate and efficient at inference becomes critical for sustainability.

Finally, we acknowledge a limitation of this analysis: the absence of real-world deployment data. While our per-image metric provides a valuable standardized unit for comparison, the actual operational carbon footprint of any model is a function of scale. A more informative metric for assessing the true annual environmental impact of a deployed screening system would be tons of CO_2 equivalent per year (tCO_2e q/year), for example, which incorporates the volume of images processed. This calculation remains impossible without data on clinical adoption rates and usage patterns.

4 Conclusion

In this article, we have employed the CodeCarbon framework to evaluate the carbon footprint of traditional and modern models dedicated to retinal vessel

segmentation. Thanks to this benchmark, we have enumerated various insightful observations, and we can address corresponding perspectives.

Our original and extensive evaluation of various metrics compared to carbon footprint upon the study of retinal vessel segmentation, makes this size increase of deep models questionable concerning the target application. Research and development communities should dedicate an important effort to more sustainable approaches in proposing new AI models. Their increasing consumption in both training and inference should impose another viewpoint than a race to the performance, and should include systematic environmental metrics as we proposed in this article.

Moreover, to better understand the causes of our observations, we plan to conduct further Pareto-optimality analysis, to measure marginal performance gains per unit of carbon emitted, and to consider multi-objective optimizations.

References

1. Bouza, L., et al.: How to estimate carbon footprint when training deep learning models? A guide and review. Environmental Research Communications (2023)
2. Chivet, B., et al.: Hessian-based deep retinal vessel segmentation with extremely few annotations. In: Medical Image Understanding and Analysis (2025)
3. Courty, B., et al.: mlco2/codecarbon: v2.4.1 (2024)
4. Dodge, J., et al.: Measuring the carbon intensity of ai in cloud instances. In: Proceedings of the 2022 ACM Conference FAccT. New York, NY, USA (2022)
5. Fraz, M.M., et al.: An ensemble classification-based approach applied to retinal blood vessel segmentation. IEEE Transactions on Biomedical Engineering **59**(9), 2538–2548 (2012)
6. Galdran, A., et al.: The little w-net that could: State-of-the-art retinal vessel segmentation with minimalistic models (2020)
7. Gowda, S.N., et al.: Watt for what: Rethinking deep learning's energy-performance relationship. In: Del Bue, A., et al. (eds.) Computer Vision – ECCV 2024 Workshops. Springer Nature Switzerland, Cham (2025)
8. Jay, M., et al.: An experimental comparison of software-based power meters: focus on CPU and GPU. In: 23rd IEEE/ACM international symposium CCGrid. IEEE, Bangalore, India (2023)
9. Liu, W., et al.: Full-resolution network and dual-threshold iteration for retinal vessel and coronary angiograph segmentation (2022)
10. Lottick, K., et al.: Energy usage reports: Environmental awareness as part of algorithmic accountability. NeurIPS Workshop on Tackling Climate Change with Machine Learning (2019)
11. Ronneberger, O., et al.: U-Net: Convolutional networks for biomedical image segmentation. In: MICCAI 2015. pp. 234–241. Springer (2015)
12. Seo, S., et al.: Full-scale representation guided network for retinal vessel segmentation (2025)
13. Shehabi, A., et al.: 2024 United States data center energy usage report (2024)
14. World Nuclear Association: Comparison of lifecycle greenhouse gas emissions of various electricity generation sources (2022)