An empirical study using lightweight convolutions to achieve Green AI

Lívia de Azevedo[1], **Luan Teylo**[2], **João Santos**[1], **Mariza Ferro**[1]

[1]Instituto de Computação – Universidade Federal Fluminense – Niterói – RJ – Brazil

[2]Centre Inria de l'université de Bordeaux – Talence – France

{liviaazevedosilva, j_p_santos}@id.uff.br, luan.teylo@inria.fr, mariza@ic.uff.br

Abstract. *Convolutional Neural Networks (CNNs) are widely used in computer vision but larger and deeper CNNs often require substantial computational resources and energy, increasing their environmental impact. Thus, Green AI highlights the need to balance predictive performance and sustainability. In this context, lightweight convolutions have been proposed to reduce computational complexity and model parameters. Therefore, this work empirically evaluates the environmental and predictive impacts of six lightweight convolutions integrated into the STMixGAN across two datasets. Results show energy savings of 0.8%–8.9% across training and inference, while predictive performance degradation differs between datasets, ranging from 3.5%–21.6% and 1.6%–5.2%.*

1. Introduction

Convolutional Neural Networks (CNNs) have revolutionized fields such as object detection, image classification and semantic segmentation [Mei et al. 2025, Zhao et al. 2024]. However, the pursuit of ever-better performance, even if only by a few tenths of accuracy, has led to increasingly complex architectures [Liu et al. 2024], such as the VGG-19, which contains 19 layers and over 144 million parameters [Simonyan and Zisserman 2014]. This trend poses significant challenges: first, these models are often incompatible with resource-constrained edge devices due to limited memory and power [Shuvo et al. 2023]; second, the massive computational demand for training and deploying such models carries a substantial energy consumption and consequently, environmental overload.

To address these issues, the "Green AI" approach [Schwartz et al. 2020] advocates for efficiency (minimization of energy consumption and hardware requirements) as a primary evaluation metric alongside accuracy. While lightweight architectures, such as the MobileNet [Howard et al. 2017, Sandler et al. 2018] and SqueezeNet [Iandola et al. 2016, Gholami et al. 2018] families, have been developed to optimize the accuracy-efficiency trade-off for running on resource-constrained devices [Chen et al. 2024, Mei et al. 2025], research often focuses strictly on the number of parameters, computational complexity, and memory consumption [Chen et al. 2024] rather than broader environmental impacts.

A promising strategy within Green AI is the use of lightweight convolution modules [Zhou et al. 2023, Tabbakh et al. 2024], such as Depthwise Separable and Fire Convolutions, which can be integrated into existing architectures to reduce operations. Despite their potential, there is a lack of empirical studies that systematically quantify the

environmental gains of these modules, especially in computationally expensive frameworks like Generative Adversarial Network (GAN).

This work fills this gap by conducting a comprehensive empirical study to evaluate the environmental, performance, and accuracy impacts of six lightweight convolution modules. Specifically, we address the following research question: *What are the environmental and accuracy impacts of integrating lightweight convolutions into Deep Learning (DL) models, and to what extent can they enhance model sustainability*? To answer this, we evaluate these modules within a GAN architecture — STMixGAN [He et al. 2025] — covering both training and inference phases using the Moving MNIST [Srivastava et al. 2015] and TAASRAD19 [Franch et al. 2020] (radar reflectivity) datasets.

The main contribution of this work is a novel comparative sustainability evaluation of different lightweight convolutions within the same experimental context. The results provide practitioners with evidence of the possible trade-offs between predictive performance and environmental impact, supporting the adoption of lightweight convolutions when appropriate. In addition, our analysis contributes to the Green AI field by measuring not only accuracy metrics (SSIM, RMSE, and MAE) and hardware performance (peak memory usage and average memory utilization), but also critical sustainability indicators: Energy Consumption, Carbon Equivalent Emissions (CO_2), and Water Usage.

This work is organized as follows. In Section 2, related work is presented; Section 3 shows the methodology, data preprocessing, model configuration, and the evaluation used. In Section 4 the experiments and results are presented. Finally, Section 5 presents the final considerations and future work.

2. Related Work

Lightweight CNN architectures have been widely explored as a means to design computationally efficient DL models. SqueezeNet [Iandola et al. 2016] and SqueezeNext [Gholami et al. 2018] introduce the Fire module and the SqueezeNext block as core components of their architectures. These works typically evaluate efficiency gains by comparing full models against state-of-the-art architectures such as AlexNet, MobileNet, and VGG. Notably, SqueezeNext reports energy consumption during inference. MobileNet [Howard et al. 2017] employs Depthwise Separable Convolution, while MobileNetV2 [Sandler et al. 2018] introduces inverted residuals and linear bottlenecks. However, these studies primarily evaluate complete architectures, rather than isolating and analyzing the impact of the proposed convolutional modules within different network backbones. In contrast, GhostNet [Han et al. 2020] partially addresses this limitation by integrating its Ghost module into architectures such as ResNet and VGG, enabling a more direct assessment of its effectiveness. Similarly, the Global Convolutional Network [Peng et al. 2017] introduces the Global Convolution module and evaluates it within a ResNet-based architecture by comparing baseline accuracy with that of the modified model. Although originally proposed for semantic segmentation, this module can be incorporated into other architectures to reduce computational requirements. Despite these advances, none of these works systematically evaluate environmental sustainability metrics, such as energy consumption or carbon emissions, leaving an important gap in the context of Green AI.

In parallel, several lightweight convolutional modules have been proposed

independently of specific architectures. Examples include Tied Convolution [Wang and Yu 2021], the LS^2CM module [Meng et al. 2022], Partial Convolution [Chen et al. 2023], and Cheap Convolution [Zou et al. 2025]. These methods address distinct challenges, such as reducing channel redundancy (Tied Convolution), improving hyperspectral image classification (LS^2CM), increasing FLOPs throughput (Partial Convolution), and enabling efficient parcel segmentation (Cheap Convolution). Although they often achieve reductions in computational cost compared to standard convolutions, their evaluations are typically limited to the respective proposed convolutional modules. Consequently, they lack comprehensive comparative analyses across different lightweight convolutional modules and do not report environmental sustainability metrics.

Additionally, some studies have compared lightweight CNN architectures. For instance, the MESNET framework [Rao et al. 2025] combines models from the MobileNet, ShuffleNet, and EfficientNet families within an ensemble framework and reports both predictive performance and environmental metrics for each model. The work in [Abu Talib et al. 2025] evaluates lightweight CNNs for deployment on edge devices, defining lightweight models as those that can run on NVIDIA Jetson Nano hardware. Similarly, [Attallah and Pacal 2026] presents a large-scale comparison of lightweight CNNs and Vision Transformers in a resource-constrained medical imaging task. However, these studies are limited to comparisons at the architecture level and do not investigate the impact of individual convolutional modules.

Therefore, there is a clear gap in the literature regarding module-level evaluations of lightweight convolutional modules, particularly when jointly considering predictive performance and environmental sustainability metrics. To address this limitation, this work presents a comparative study of lightweight convolutional modules under a common experimental setting, enabling a direct and fair comparison among them.

3. Methodology

This section describes the systematic approach used to evaluate the impact of lightweight convolutions. We focus on a Spatio-temporal Prediction task using a GAN-based framework, measuring the trade-off between predictive quality and environmental sustainability. The proposed methodology consists of four steps, detailed in next sections: (i) baseline architecture selection, (ii) integration of lightweight modules, (iii) experimental setup definition, and (iv) Green AI metrics measurement protocol. The source code of this work is publicly available in a repository [1].

3.1. Base Architecture: STMixGAN

To evaluate the impact of lightweight convolutions in a high-demand scenario, we selected the STMixGAN [He et al. 2025] as our baseline. The model is composed of an encoder–decoder generator (STMixNet) that combines a Swin Transformer [Liu et al. 2021] branch for global context with a CNN branch for local features, and a fully convolutional discriminator (DC-Net). To integrate global and local representations, the architecture utilizes an MLP-Mixer mechanism [Tolstikhin et al. 2021] for information fusion.

This sophisticated combination of attention mechanisms, multi-layer mixers and convolutional layers results in high computational requirements. Our intervention tar-

[1] https://github.com/RioNowcast-Green/empirical-study-lightweight-convolutions-green-ai

gets only the CNN components, replacing all standard 2D convolutions with kernel sizes greater than 1×1 with the six lightweight convolutional modules evaluated in this study. By keeping the macro-architecture intact during this replacement, we ensure that any observed changes in environmental metrics or predictive accuracy can be directly attributed to the efficiency of the convolutional design.

3.2. Lightweight Convolutions

The following modules were selected to replace the standard convolutional layers with a $k \times k$ filter size, each employing a unique strategy to minimize computational cost (Figure 1). **Depthwise Separable Convolution (DWS)** [Howard et al. 2017] factorises a $k \times k$ convolution into a depthwise step (one $k \times k$ filter per input channel) followed by a 1×1 pointwise step that produces the requested number of output channels. **Depthwise Separable Convolution Bottleneck (DWSB)** [Sandler et al. 2018] prepends a 1×1 pointwise layer that, in our adaptation, reduces rather than expands the channel dimension before the depthwise step, so that the bottleneck contributes to parameter savings. **SqueezeNext Block (SqNxt)** [Gholami et al. 2018] retains a single 1×1 squeeze followed by the asymmetric $1 \times k$ and $k \times 1$ pair, dropping the second pointwise reduction and the final pointwise expansion of the original block to maximize the parameter and computation reduction. **Fire Convolution** [Iandola et al. 2016] uses a single 1×1 squeeze channel and a single 1×1 plus 3×3 expand channel pair (the three squeeze/expand hyperparameters are all set to one), the most aggressive setting permitted by the module. **Global Convolution** [Peng et al. 2017] uses two parallel branches composed of $1 \times k$ followed by $k \times 1$ convolutions, whose outputs are summed. When replacing a standard convolution whose input and output differ only in the channel dimension, the Global Convolution uses a kernel size of $k = 3$ with padding equal to $k/2$. **Tied Block Convolution** [Wang and Yu 2021] uses $B = 60$ shared $k \times k$ filter groups, with any remaining input channels appended to the output to remove the divisibility constraint of the original formulation. The number of input and output channels must be divisible by B; otherwise, a standard convolution is used instead of the Tied convolution.

In the case of Fire and Global Convolution, the intermediate output height, width, and number of channels may differ from the expected output dimensions. The expected spatial dimensions are computed based on the kernel size, padding, stride, and input dimensions. To adjust the spatial dimensions, if the output height and width are smaller than expected, nearest-neighbour interpolation is applied. Otherwise, the output is sliced to match the desired size. To adjust the number of channels, if the number of output channels is lower than the desired value, the channels are replicated until the target number is reached. Otherwise, only the required number of channels is retained, and the remaining ones are discarded. The Tied Block Convolution uses the slicing strategy to adjust spatial dimensions. The architectural designs of these modules are detailed in Figure 1.

These six modules were chosen to cover a broad spectrum of efficiency techniques in lightweight convolutions, enabling a comprehensive assessment of the impacts of different lightweight convolution designs during both training and inference. In addition, we focus on lightweight convolutions rather than other Green AI approaches, such as pruning, quantization, or knowledge distillation, because they can be integrated into existing architectures as plug-and-play alternatives, requiring no or some adaptations to the lightweight convolution modules to match the output dimensions of standard convolutions.

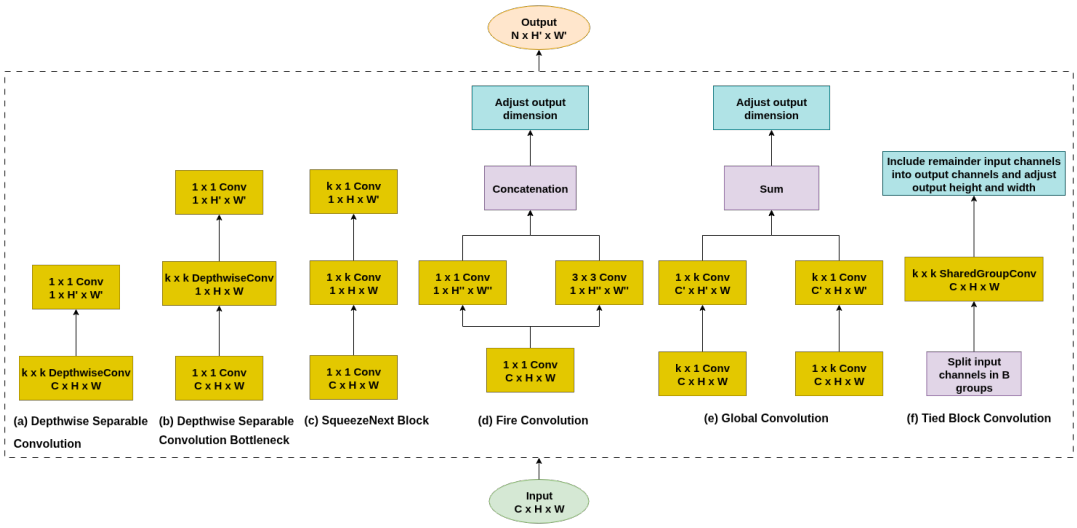


Figure 1. Schematic illustration of the six lightweight convolutional modules examined in this study. These alternative designs decrease the number of parameters and computational cost relative to conventional convolution. Source: The authors.

3.3. Experimental Setup and Datasets

Datasets: To mitigate experimental bias and evaluate the generalization of lightweight convolutions across different domains, we selected two distinct spatio-temporal datasets: Moving MNIST [Srivastava et al. 2015] and TAASRAD19 [Franch et al. 2020]. Moving MNIST consists of sequences of two handwritten digits moving independently within a 64×64 grid. We used 10,000 sequences, each with 20 frames, following the standard split of 9,000 sequences for training and 1,000 for testing. TAASRAD19 is a high-resolution radar reflectivity dataset collected in the Italian Alps, comprising nine years of data with a 5-minute temporal resolution. For this study, we focused on the years 2016–2018 for training (8,800 sequences) and 2019 for testing (3,454 sequences), representing a challenging real-world meteorological scenario. We selected the last four years of data because this period contains the highest number of storm events, which represents a challenging predictive meteorological task.

Data Preprocessing: To ensure an analysis level related to domain across datasets, the TAASRAD19 frames were bilinearly resized from their original 480×480 resolution to 64×64, aligning the input dimensions with the Moving MNIST baseline. For both datasets, sequences were truncated to 10 frames: the first 5 frames are used as input to predict the subsequent 5 frames. Before training and inference, the input data were normalized to the [0-1] range, as in the original STMixGAN.

Execution Environment and Training Setup: All experiments were executed on PlaFRIM, the experimental platform of the Inria Bordeaux research center. We used a node equipped with two NVIDIA A100 GPUs of 40 GB; only one A100 was used per run so that runtime and energy readings are not contaminated by inter-GPU synchronization. Single-GPU execution is enforced in the training script, which pins `CUDA_VISIBLE_DEVICES=0`. All models were trained with the same hyper-parameters and loss functions as the original STMixGAN. We used the Adam optimizer [Goodfellow et al. 2020] with a learning rate of 10^{-4} and $(\beta_1, \beta_2) = (0.5, 0.9)$

for both the generator and the critic. The training was conducted with a batch size of 10 for 100 epochs using the PyTorch 2.10 framework. To control for stochastic variation and ensure the statistical significance of the results, each configuration was trained four times using independent random seeds. The reported metrics represent the mean value across these four runs.

3.4. Evaluation Metrics and Sustainability Protocol

Predictive and Hardware Performance Evaluation Metrics: To assess the predictive quality of the models, we employed three standard metrics for spatio-temporal forecasting, following the protocol established in [He et al. 2025]: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Structural Similarity Index Measure (SSIM). The image sequence SSIM is computed as the mean SSIM across all images in the sequence. Wall-clock execution time, peak GPU memory, and average GPU memory are recorded during both training and inference. These metrics expose runtime efficiency that is not visible in parameter or Float Point Operations (FLOPs) counts. The IOPS framework was used to orchestrate the experiments and collect system-level GPU metrics through its resource tracing layer, as detailed in the paper's repository.

Sustainable Evaluation: This work assesses energy consumption, water usage, and carbon emissions for each execution. GPU energy consumption is obtained from system-level GPU metrics collected by the IOPS framework. Water usage and carbon emissions are estimated using the wAIter [Breder and Ferro 2025] methodology. The carbon [Strubell et al. 2019] is calculated as $\mathrm{CO_2e} = c_i \cdot e$, where c_i denotes the country's carbon intensity and e is the energy consumed. The water usage [Li et al. 2025] is calculated as $w = e \cdot [\rho_{s1} + \theta \cdot \rho_{s2}]$, where ρ_{s1} refers to water usage within the data center, ρ_{s2} denotes the water intensity of the national power generation mix, and θ represents the server Power Usage Effectiveness (PUE). Since our experiments were conducted in France, we adopt $c_i = 56.039$ and $\rho_{s2} = 3.67145$, following the wAIter definitions. Additionally, $\rho_{s1} = 0$, as PlaFRIM does not use water for cooling, and default $\theta = 1$ is assumed, since the exact PUE value of the PlaFRIM facility is not available.

4. Experiments and Results

This section reports the experimental results and analysis. All tables present performance metrics as "mean $\pm$ standard deviation" over four runs. The best and second-best results are shown in bold and underlined, respectively, while colors indicate improvements (green) or degradations (red) relative to the standard convolution.

Environmental and Accuracy Impacts: Tables 1 and 2 show the environmental metrics for training and inference, while Table 3 presents the inference performance of the lightweight convolutions. During training, most lightweight convolutions reduce energy consumption compared to the standard convolution on both datasets, except for Tied, which slightly increases consumption. Fire and SqNxt achieve the best reductions on Moving MNIST (8.5% and 8.1%), while Fire and DWSB perform best on TAASRAD19 (8.2% and 8.1%). Overall, energy savings are consistent across datasets, although the most energy-efficient lightweight convolutions differ between them. During inference, most lightweight convolutions also reduce energy consumption, with some exceptions (Global on Moving MNIST; Fire and Global on TAASRAD19). The best results are obtained by SqNxt and DWSB on Moving MNIST (8.9% and 7.3%) and by DWS and Tied

on TAASRAD19 (5.8% and 4.6%). Global performs worst in both datasets (increases of 10.1% on Moving MNIST and 7.4% on TAASRAD19). Notably, some methods exhibit phase-dependent behavior: Tied saves energy only during inference, Global only during training, and Fire on TAASRAD19 exclusively during training. These indicate that energy efficiency is not consistent across training and inference.

When analyzing the inference performance of the models that achieved the best environmental results, a clear trade-off between sustainability and accuracy emerges. On Moving MNIST, most lightweight convolutions lead to notable degradations, around 29% in MAE, 11–15% in RMSE, and 8–10% in SSIM for Fire and SqNxt, while on TAASRAD19 these impacts are considerably smaller, typically below 9% in MAE, 3% in RMSE, and 2% in SSIM. DWSB follows a similar pattern, with modest degradations on TAASRAD19 (approximately 8% in MAE, 3% in RMSE, and 2% in SSIM) but larger reductions on Moving MNIST (23%, 12%, and 3%, respectively). An exception is DWS, which improves performance on Moving MNIST (reducing MAE by 7.4%, RMSE by 1.9%, and increasing SSIM by 3.9%), but still shows slight degradations on TAASRAD19 (around 1–2% across metrics). Tied degraded performance by 2.4% in MAE, 1.4% in RMSE, and 0.9% in SSIM on TAASRAD19. Overall, the accuracy impact of lightweight convolutions is more pronounced on Moving MNIST and relatively mild on TAASRAD19. In some cases, they can even outperform the standard convolution, but this behavior is not consistent across datasets, reinforcing the hypothesis that the environmental–accuracy trade-off is probably problem-dependent.

Table 1. Environmental impact during training across datasets.

	Moving MNIST			TAASRAD19		
	Carbon (gCO_2) (↓)	**Water (L) (↓)**	**Energy (kWh) (↓)**	**Carbon (gCO_2) (↓)**	**Water (L) (↓)**	**Energy (kWh) (↓)**
Standard	158.98665 ± 0.72676	10.41617 ± 0.04761	2.83707 ± 0.01297	154.8759 ± 1.36746	10.14685 ± 0.08959	2.76372 ± 0.0244
DWS	157.73226 ± 1.21428	10.33398 ± 0.07955	2.81469 ± 0.02167	152.42705 ± 2.70961	9.98641 ± 0.17752	2.72002 ± 0.04835
DWSB	147.01185 ± 1.30509	9.63163 ± 0.0855	2.62338 ± 0.02329	142.31209 ± 1.96228	9.32372 ± 0.12856	2.53952 ± 0.03502
Fire	**145.51676 ± 2.182**	**9.53367 ± 0.14296**	**2.59671 ± 0.03894**	**142.2066 ± 1.61326**	**9.3168 ± 0.10569**	**2.53764 ± 0.02879**
Global	155.36838 ± 3.31743	10.17911 ± 0.21734	2.7725 ± 0.0592	151.04348 ± 2.46076	9.89576 ± 0.16122	2.69533 ± 0.04391
SqNxt	146.11328 ± 2.63238	9.57275 ± 0.17246	2.60735 ± 0.04697	144.86734 ± 2.06848	9.49113 ± 0.13552	2.58512 ± 0.03691
Tied	160.47803 ± 1.90756	10.51388 ± 0.12498	2.86368 ± 0.03404	156.08583 ± 2.30095	10.22612 ± 0.15075	2.78531 ± 0.04106

Table 2. Environmental impact during inference across datasets.

	Moving MNIST			TAASRAD19		
	Carbon (gCO_2) (↓)	**Water (L) (↓)**	**Energy (kWh) (↓)**	**Carbon (gCO_2) (↓)**	**Water (L) (↓)**	**Energy (kWh) (↓)**
Standard	0.13821 ± 0.00592	0.00905 ± 0.00039	0.00247 ± 0.00011	0.42565 ± 0.02095	0.02789 ± 0.00137	0.0076 ± 0.00037
DWS	0.12868 ± 0.01226	0.00843 ± 0.0008	0.0023 ± 0.00022	**0.40098 ± 0.00777**	**0.02627 ± 0.00051**	**0.00716 ± 0.00014**
DWSB	0.12832 ± 0.00943	0.00841 ± 0.00062	0.00229 ± 0.00017	0.40759 ± 0.00805	0.0267 ± 0.00053	0.00727 ± 0.00014
Fire	0.1373 ± 0.00728	0.009 ± 0.00048	0.00245 ± 0.00013	0.42727 ± 0.01706	0.02799 ± 0.00112	0.00762 ± 0.0003
Global	0.15241 ± 0.00646	0.00999 ± 0.00042	0.00272 ± 0.00012	0.45746 ± 0.01606	0.02997 ± 0.00105	0.00816 ± 0.00029
SqNxt	**0.12614 ± 0.00807**	**0.00826 ± 0.00053**	**0.00225 ± 0.00014**	0.41225 ± 0.01016	0.02701 ± 0.00067	0.00736 ± 0.00018
Tied	0.13121 ± 0.00665	0.0086 ± 0.00044	0.00234 ± 0.00012	0.40642 ± 0.01791	0.02663 ± 0.00117	0.00725 ± 0.00032

Training GPU metrics: Table 4 presents the GPU metrics during training. All lightweight convolutions used less memory than the standard convolution, except for the peak memory usage of DWS. The slower variants (DWS, Global, and Tied) decompose standard convolution into operations with lower arithmetic intensity, which can increase memory access costs and introduce additional GPU kernel launch and synchronization overhead [Ma et al. 2018]. A key indicator is the average power draw (Figure 2), which is consistently lower than that of the standard convolution and the faster variants for all

Table 3. Predictive performance comparison across datasets.

	Moving MNIST			TAASRAD19		
	SSIM (↑)	RMSE (↓)	MAE (↓)	SSIM (↑)	RMSE (↓)	MAE (↓)
Standard	0.85719 ± 0.01058	28.13193 ± 0.23217	7.10815 ± 0.11498	**0.83945 ± 0.00179**	**1.80964 ± 0.00902**	**0.78031 ± 0.00411**
DWS	**0.89036 ± 0.0016**	**27.59137 ± 0.21815**	**6.58184 ± 0.07294**	0.83141 ± 0.00094	1.85247 ± 0.00808	0.7924 ± 0.00737
DWSB	0.83083 ± 0.01878	31.46575 ± 0.51739	8.74304 ± 0.29835	0.8234 ± 0.00418	1.87047 ± 0.02526	0.84148 ± 0.01793
Fire	0.77231 ± 0.01502	31.36238 ± 0.26598	9.19899 ± 0.25459	0.82291 ± 0.00477	1.85953 ± 0.00825	0.84364 ± 0.02995
Global	0.71368 ± 0.04163	31.35979 ± 0.49583	9.71483 ± 0.38076	0.82022 ± 0.00166	1.85737 ± 0.01979	0.86287 ± 0.00677
SqNxt	0.783 ± 0.06714	32.25786 ± 1.86331	9.2139 ± 1.36874	0.82896 ± 0.00429	1.85781 ± 0.00417	0.81481 ± 0.01919
Tied	0.85127 ± 0.02565	29.41523 ± 0.05239	7.48869 ± 0.1385	0.83215 ± 0.00393	1.83527 ± 0.01007	0.79915 ± 0.01237

slower convolutions, except for Tied. This reduced power consumption suggests that the streaming multiprocessors (SMs), which rely on high arithmetic intensity to sustain throughput, are likely stalling on global memory accesses. Tied may further suffer from kernel launch overhead due to its use of B=60 groups, in addition to higher memory access costs and overhead from output dimension adjustments. However, a deeper investigation using low-level profiling tools is required to confirm these hypotheses.

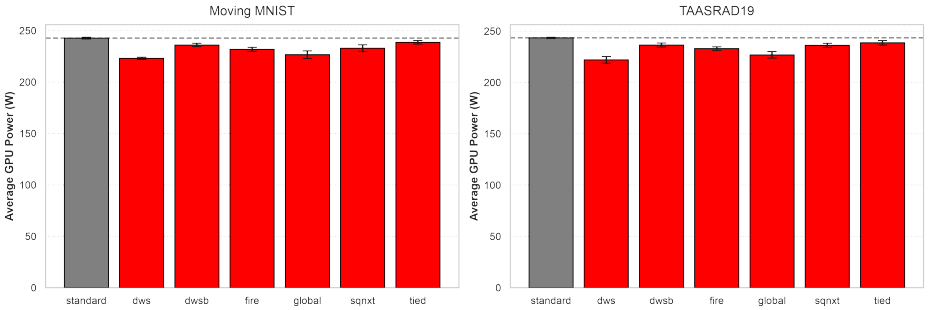


Figure 2. Average GPU power (W) during training, mean of four runs.

The best training times were achieved by the DWSB and SqNxt convolutions on Moving MNIST (4.9% and 4.2% reductions) and by the DWSB and Fire convolutions on TAASRAD19 (5.4% and 4.04%). Our results challenge the common assumption that energy consumption is strictly proportional to execution time. While certain convolution modules demonstrate faster training, they do not necessarily yield the lowest environmental footprint. Fire achieves the best environmental performance across both datasets, but it is not the fastest to train. Furthermore, slower variants such as DWS and Global still achieve lower environmental impact than the standard convolution.

Table 4. GPU performance during training across datasets.

	Moving MNIST			TAASRAD19		
	Mem Peak (MiB) (↓)	Avg Mem Util. (MiB) (↓)	Time (min) (↓)	Mem Peak (MiB) (↓)	Avg Mem Util. (MiB) (↓)	Time (min) (↓)
Standard	12259	8805.376 ± 139.11836	698.8312 ± 3.89254	12259	8912.896 ± 43.98832	680.3718 ± 4.23421
DWS	12987	**7450.624 ± 58.5981**	755.9265 ± 3.22618	12987	**7501.824 ± 69.06752**	734.01814 ± 2.08278
DWSB	12097	8077.312 ± 30.28456	**664.47573 ± 4.4636**	12097	8131.584 ± 111.03988	**643.81639 ± 3.8486**
Fire	12095	7993.344 ± 44.68208	671.17863 ± 4.60171	12095	8034.304 ± 32.33852	652.82723 ± 3.14
Global	**12025**	8649.728 ± 109.77356	733.09031 ± 3.7055	**12025**	8711.168 ± 114.70628	711.43199 ± 1.38527
SqNxt	12147	8163.328 ± 27.98101	669.58586 ± 3.32927	12147	8120.32 ± 101.5775	654.48596 ± 3.18859
Tied	12083	8375.296 ± 30.90151	719.51016 ± 3.05962	12083	8378.368 ± 83.44186	699.61929 ± 4.46577

Inference GPU metrics: Table 5 presents the GPU metrics during inference. Across all lightweight convolutions, less memory was used than with the standard convolution, except for Global in terms of memory utilization. Regarding execution time, the

standard convolution outperformed all lightweight variants. Similar to the training phase, one possible explanation for this behavior is the increased memory access cost, additional GPU kernel launches, and synchronization overhead compared to standard convolution, as well as the extra computations introduced in this work for convolutions that require output size adjustments. Once again, the most environmentally efficient convolutions were not the fastest ones. DWS was the second fastest in execution time; however, it was the third most energy-efficient method on Moving MNIST.

Table 5. GPU performance during inference across datasets.

	Moving MNIST			TAASRAD19		
	Mem Peak (MiB) (↓)	**Avg Mem Util. (MiB) (↓)**	**Time (min) (↓)**	**Mem Peak (MiB) (↓)**	**Avg Mem Util. (MiB) (↓)**	**Time (min) (↓)**
Standard	1223	851.968 ± 90.04992	**1.14262 ± 0.00913**	2193	1057.792 ± 65.73835	**3.89302 ± 0.04349**
DWS	1131	547.84 ± 70.34117	1.14352 ± 0.00886	2101	**675.84 ± 38.56916**	3.91096 ± 0.02274
DWSB	**1129**	535.552 ± 63.22319	1.196 ± 0.01642	**2099**	704.512 ± 35.86437	4.05055 ± 0.03813
Fire	**1129**	**518.144 ± 59.68553**	1.26836 ± 0.01324	**2099**	716.8 ± 25.46996	4.33549 ± 0.07056
Global	**1129**	1325.056 ± 209.24374	1.3718 ± 0.02646	**2099**	1826.816 ± 12.0583	4.58885 ± 0.03512
SqNxt	**1129**	574.464 ± 16.5115	1.21202 ± 0.01564	**2099**	713.728 ± 24.2034	4.12102 ± 0.03683
Tied	**1129**	545.792 ± 48.62293	1.20172 ± 0.02101	**2099**	712.704 ± 47.05946	4.03311 ± 0.02606

What are the environmental and accuracy impacts of integrating lightweight convolutions into DL models, and to what extent can they enhance model sustainability? Our findings suggest that lightweight convolutions can improve model sustainability during training, inference, or both, depending on the convolution. However, the gains remain relatively limited, with energy savings ranging from 0.8% to 8.5% during training and from 0.8% to 8.9% during inference. These improvements come with varying impacts on predictive performance. On the Moving MNIST dataset, predictive performance deteriorates by between 3.5% and 21.6% on average across the evaluated metrics, whereas on TAASRAD19 the impact is less pronounced, with mean reductions ranging from 1.6% to 5.2%. In contrast, when employing DWS on Moving MNIST, an improvement in performance is observed without any increase in energy consumption during either training or inference. Therefore, we suggest that the viability of lightweight convolutions as a Green AI strategy depends on both the problem and the convolution. While some methods achieve a favorable trade-off (e.g., DWS on TAASRAD19, reducing training and inference energy consumption by 1.6% and 5.8%, respectively, with only a 1.6% average reduction in predictive metrics), others do not (e.g., SqNxt on Moving MNIST, yielding energy savings of 8.1% and 8.8% but a 17.6% average reduction in predictive metrics).

5. Final Considerations and Future Work

In this work, experiments were conducted using a GAN-based model to compare six lightweight convolutional modules and evaluate their effectiveness as a Green AI strategy, considering both environmental impact and predictive performance. The experimental results show that some lightweight convolutions can achieve modest reductions in carbon footprint, water usage, and energy consumption during both training and inference or only in one of these phases. Moreover, depending on the convolution and the dataset, the impact on predictive performance may vary from negligible to significant. In some cases, improvements in training time and reductions in memory requirements are also observed. This work contributes a comparative Green AI evaluation of different lightweight convolutions under a common experimental setting, enabling a fair assessment of the trade-offs between predictive performance and environmental sustainability.

Despite the findings, some limitations of this work should be addressed. First, although we evaluated six diverse lightweight modules, we focused on a single architectural configuration for each without performing an extensive hyperparameter search, which is available for some modules. Second, the experiments were limited to a single model architecture and two datasets focused on the same frame prediction task. Furthermore, only six lightweight convolutions were selected, while other relevant alternatives exist that could be explored. Finally, the higher execution time observed during inference, and for some models during training, compared to the standard convolution requires further investigation to verify if these findings align with observations reported in the literature.

For future work, additional experiments can be conducted using a broader set of lightweight convolutions and different CNN-based architectures, such as ResNet and MobileNet, across more diverse datasets. To further extend this study, these future investigations could explore different hyperparameter configurations for the evaluated modules, such as varying the parameter B in the Tied Convolution and the squeeze and expand ratios in the Fire Convolution. Lastly, a deeper investigation into the high execution times observed during training and inference would be valuable, supported by computational profiling tools such as PyTorch Profiler and NVIDIA Nsight Compute to measure low-level performance metrics like memory bandwidth utilization.

Ethical Declaration: During this work, ChatGPT and Writefull were used to improve English writing. The authors reviewed all suggestions and take full responsibility for the final content.

Acknowledgments: This work was supported by the Serrapilheira Institute (Grant No. Serra—2211-41897) and SERRAPILHEIRA/FAPERJ— 2023 (Grant No. E-26/210.242/2024).

References

Abu Talib, M. A., Setumin, S., Che Ani, A. I., and Abu Bakar, S. J. (2025). An analysis of lightweight convolutional neural network models for image classification task on edge device. In *2025 IEEE 15th International Conference on Control System, Computing and Engineering (ICCSCE)*, pages 175–180.

Attallah, O. and Pacal, I. (2026). Comparative evaluation of lightweight convolutional neural network and vision transformer models for multi-class brain tumor classification using merged large mri datasets. *Chemometrics and Intelligent Laboratory Systems*, 269:105609.

Breder, G. and Ferro, M. (2025). waiter: Serving awareness - environmental footprint estimator for artificial intelligence models. DOI: 10.5281/zenodo.17247278.

Chen, F., Li, S., Han, J., Ren, F., and Yang, Z. (2024). Review of lightweight deep convolutional neural networks. *Archives of Computational Methods in Engineering*, 31(4):1915–1937.

Chen, J., Kao, S.-h., He, H., Zhuo, W., Wen, S., Lee, C.-H., and Chan, S.-H. G. (2023). Run, don't walk: Chasing higher flops for faster neural networks. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12021–12031.

Franch, G., Maggio, V., Coviello, L., Pendesini, M., Jurman, G., and Furlanello, C. (2020). Taasrad19, a high-resolution weather radar reflectivity dataset for precipitation nowcasting. *Scientific Data*, 7(1):234.

Gholami, A., Kwon, K., Wu, B., Tai, Z., Yue, X., Jin, P., Zhao, S., and Keutzer, K. (2018). Squeezenext: Hardware-aware neural network design. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1719–171909.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Commun. ACM*, 63(11):139–144.

Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., and Xu, C. (2020). Ghostnet: More features from cheap operations. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1577–1586.

He, L., Zheng, K., Ruan, H., Yang, S., Zhang, J., Luo, C., Tang, S., Yi, Y., Tian, Y., and Cheng, J. (2025). A spatiotemporal mixed-enhanced generative adversarial network for radar-based precipitation nowcasting. *Computers & Geosciences*, 200:105919.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.

Li, P., Yang, J., Islam, M. A., and Ren, S. (2025). Making ai less 'thirsty'. *Commun. ACM*, 68(7):54–61.

Liu, Y., Xue, J., Li, D., Zhang, W., Chiew, T. K., and Xu, Z. (2024). Image recognition based on lightweight convolutional neural network: Recent advances. *Image and Vision Computing*, 146:105037.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9992–10002.

Ma, N., Zhang, X., Zheng, H.-T., and Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision – ECCV 2018*, pages 122–138, Cham. Springer International Publishing.

Mei, D., Gong, J., Qu, M., and Bian, S. (2025). The evolution of convolutional neural network architectures: A review. *IEEE Access*, 13:200446–200496.

Meng, Z., Jiao, L., Liang, M., and Zhao, F. (2022). A lightweight spectral-spatial convolution module for hyperspectral image classification. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5.

Peng, C., Zhang, X., Yu, G., Luo, G., and Sun, J. (2017). Large kernel matters — improve semantic segmentation by global convolutional network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1743–1751.

Rao, R. S., Kashyap, A., Yadav, M., and Mishra, A. (2025). Mesnet: integrating lightweight cnns and real-time carbon tracking for sustainable image classification. *Discover Sustainability*, 6(1):1386.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520.

Schwartz, R., Dodge, J., Smith, N. A., and Etzioni, O. (2020). Green ai. *Communications of the ACM*, 63(12):54–63.

Shuvo, M. M. H., Islam, S. K., Cheng, J., and Morshed, B. I. (2023). Efficient acceleration of deep learning inference on resource-constrained edge devices: A review. *Proceedings of the IEEE*, 111(1):42–91.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Srivastava, N., Mansimov, E., and Salakhutdinov, R. (2015). Unsupervised learning of video representations using lstms. In *Proceedings of the 32nd International Conference on Machine Learning - Volume 37*, ICML'15, page 843–852. JMLR.org.

Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. In Korhonen, A., Traum, D., and Màrquez, L., editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Tabbakh, A., Amin, L. A., Islam, M., Mahmud, G. M. I., Chowdhury, I. K., and Mukta, M. S. H. (2024). Towards sustainable ai: a comprehensive framework for green ai. *Discover Sustainability*, 5(1):408.

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., Lucic, M., and Dosovitskiy, A. (2021). Mlp-mixer: an all-mlp architecture for vision. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA. Curran Associates Inc.

Wang, X. and Yu, S. X. (2021). Tied block convolution: Leaner and better cnns with shared thinner filters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11):10227–10235.

Zhao, X., Wang, L., Zhang, Y., Han, X., Deveci, M., and Parmar, M. (2024). A review of convolutional neural networks in computer vision. *Artificial Intelligence Review*, 57(4):99.

Zhou, Y., Lin, X., Zhang, X., Wang, M., Jiang, G., Lu, H., Wu, Y., Zhang, K., Yang, Z., Wang, K., Sui, Y., Jia, F., Tang, Z., Zhao, Y., Zhang, H., Yang, T., Chen, W., Mao, Y., Li, Y., Bao, D., Li, Y., Liao, H., Liu, T., Liu, J., Guo, J., Zhao, X., WEI, Y., Qian, H., Liu, Q., Wang, X., Kin, W., Chan, Li, C., Li, Y., Yang, S., Yan, J., Mou, C., Han, S., Jin, W., Zhang, G., and Zeng, X. (2023). On the opportunities of green computing: A survey.

Zou, B., Li, W., and Chen, Z. (2025). Cheapconv: efficient parcel segmentation and detection for real-time logistics automation. *Journal of Real-Time Image Processing*, 22(6):209.