Custom CNN Model vs Pre-trained Model: Designing a Lightweight CNN for Green Computing Using the MNIST and CIFAR-10 Dataset

Term Project – CSE 407

Authors

Abrar Hossain Zahin
2022-2-60-040

Rafid Rahman
2022-2-60-116

Ahmed Mustafa Amlan
2021-3-60-253

Most Rabeya Begum
2022-1-60-367

Abstract & Motivation

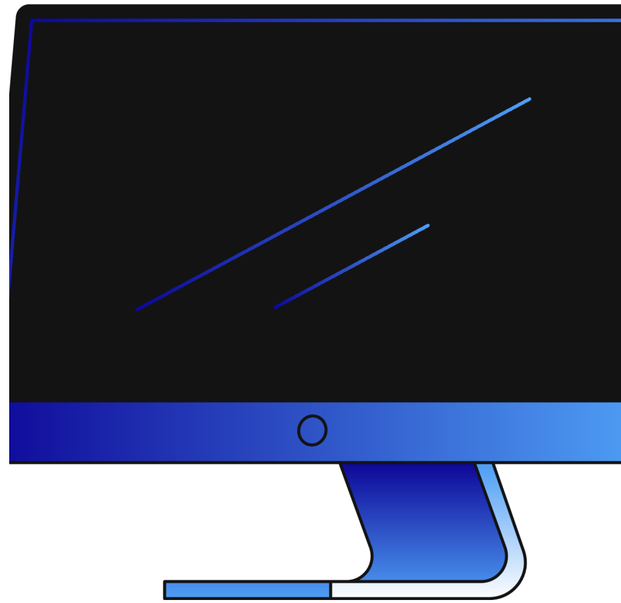
- Deep learning growth → huge energy & CO_2 cost
- Goal: build a tiny custom CNN that matches pre-trained accuracy on MNIST and CIFAR-10 but with far less compute & emissions
- Compare against Pruned MobileNetV2 and ResNet18
- Metrics: Accuracy | Parameters | GFLOPs| Model size | Inference time | CO_2 emission
- Techniques: Depth-wise separable conv, pruning.

Introduction and Problem Statement

- AI models are getting bigger → electricity & carbon footprint exploding
- Training one large model ≈ lifetime emissions of 5 cars (Strubell 2019)
- Problem: modern SOTA models too heavy for mobile/edge/IoT
- Custom lightweight models often lose accuracy

Objectives

- Design ultra-light custom CNN (low params & GFLOPs)
- Benchmark one suitable pre-trained model on MNIST and CIFAR-10
- Measure: accuracy, params, GFLOPs, size, inference time, CO_2 emission
- Apply pruning (L1, L2, Hybrid)
- Identify the greenest viable option



Research Gap

01. Limited Focus on Ground-Up Efficiency

02. Incomplete Green Metrics Evaluation

03. Edge Device Deployment Gap

Research Questions

RQ1. How custom lightweight CNN compares to pruned pretrained models on accuracy,GFLOPS,size, latency, CO_2

RQ2. To what extent do unstructured pruning and weight masking reduce the resource footprint of the custom CNN

RQ3. Is the custom CNN environmentally efficient, and which optimization achieves the greatest carbon reduction with the least accuracy loss on MNIST and CIFAR-10?

Methodology

Dataset Collection

Data Preparation

Model Design

Optimizations

Training

Evaluation

MNIST and CIFAR-10 Dataset

Load and Preprocess MNIST and CIFAR-10

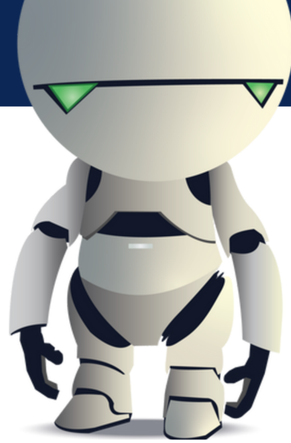
Custom CNN and Pre-trained

Pruning (L1, L2, Hybrid, 30%)

Adam Optimizer with Code Carbon Tracking

Compute Metrics (Accuracy, GFLOPs, Inf Time, CO_2)

Custom Model & Pre-trained selection



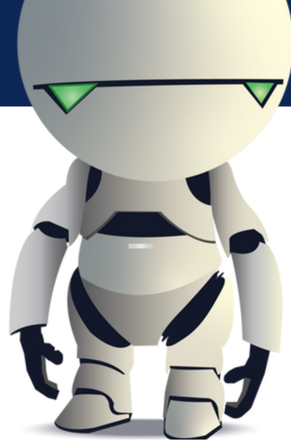
- **UltraLightCNN (custom design)**

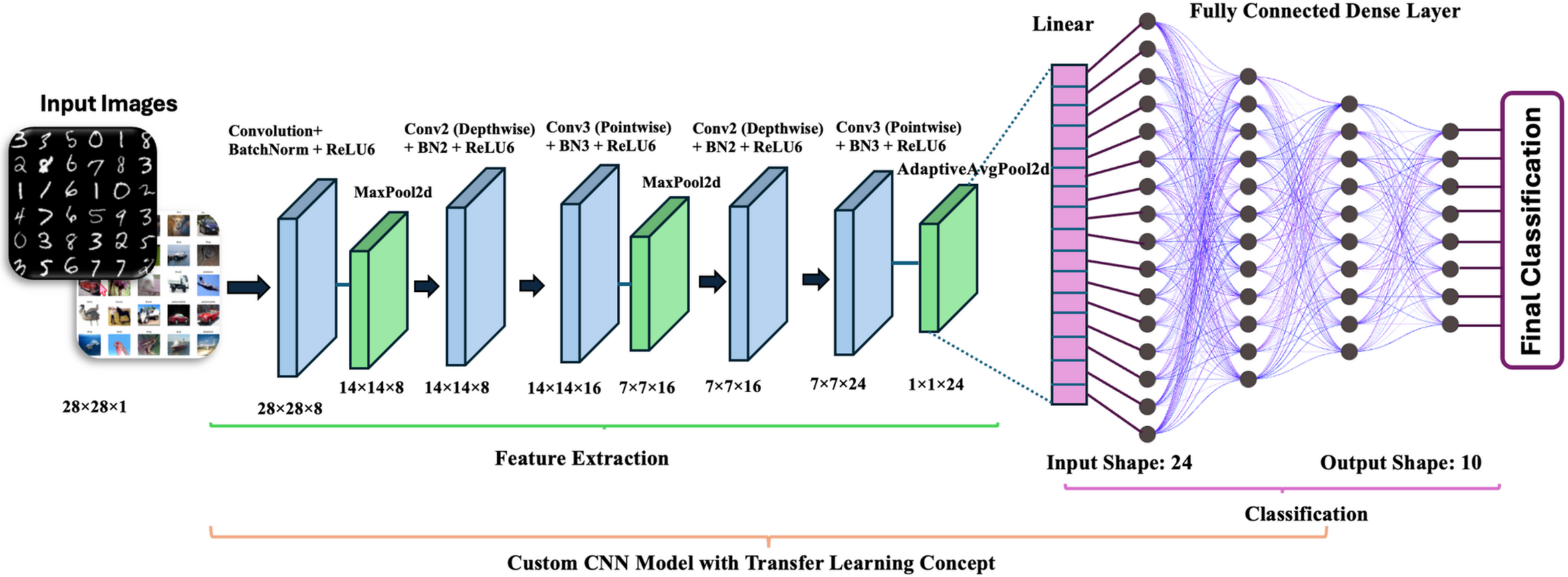
  Depthwise + pointwise separable convolutions, small channel widths, ReLU6, BatchNorm, max-pooling and global adaptive average pooling.
  Extremely small parameter footprint (target: <1 MB) and designed for quantization and edge deployment.
  Input adapted per dataset: 1-channel for MNIST, 3-channel for CIFAR-10.

- **Pre-trained baselines selected**
  MobileNetV2 (mobile-friendly, separable convs) and **ResNet18** (lightweight residual backbone).
  Both adapted for dataset channels and **structured-pruned (≈30%)** to reduce parameters/GFLOPs for fair green comparison.

- **Green optimizations used**
  Structured pruning (30% channels), and fine-tuning after pruning.

Custom CNN Architecture





Result and Analysis

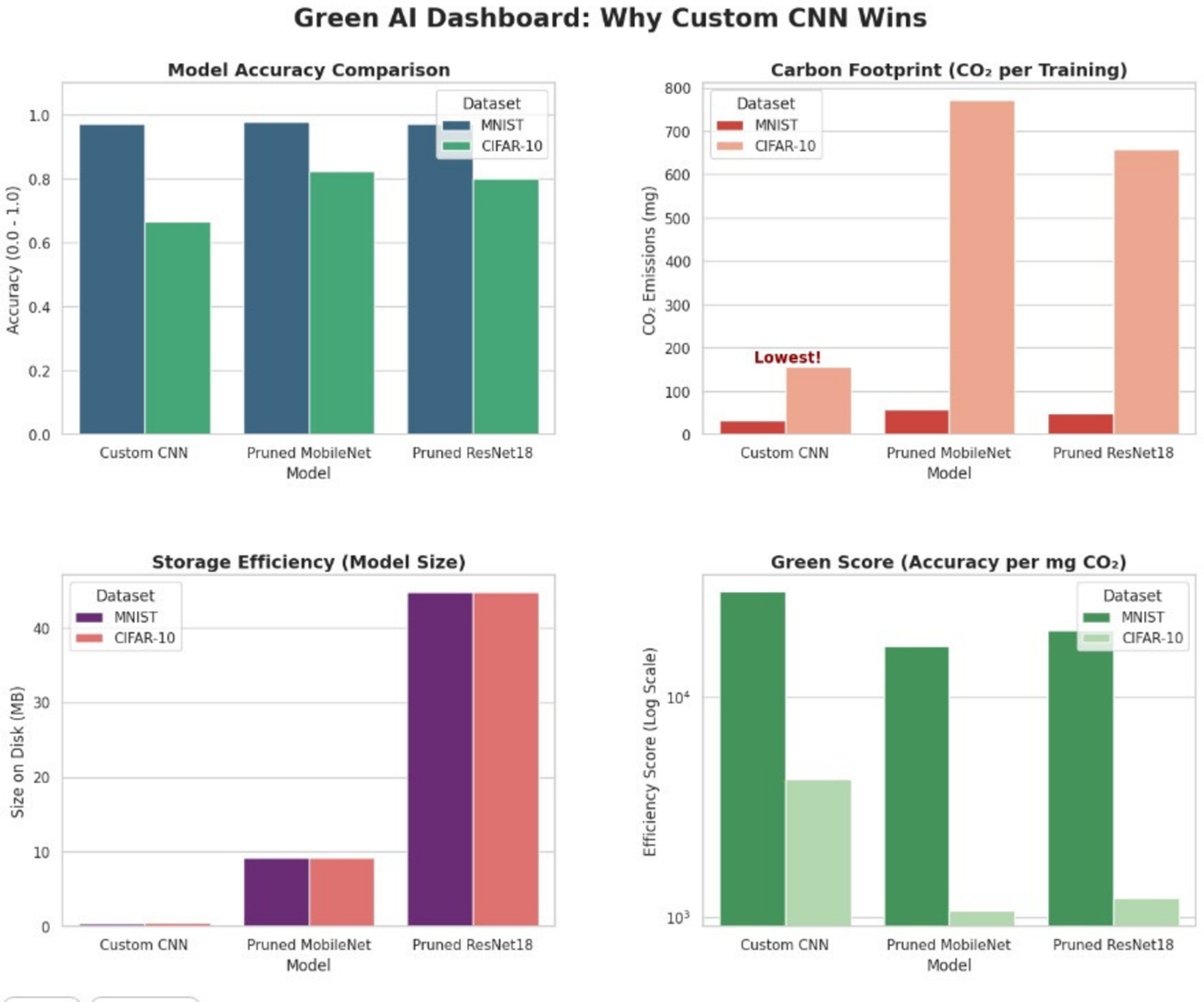
MNIST Comparison

	Model	Accuracy	Precision	Loss	Params	GFLOPs	Size (MB)	Inf Time (ms/img)	CO2 (kg)
0	Custom CNN	0.9759	0.9759	0.0832	101,930	421.000	0.41	0.07	0.000061
1	Pruned MobileNet	0.9879	0.9879	0.0423	2,236,106	5.940	9.17	0.07	0.000108
2	ResNet18 Pruned	0.9754	0.9754	0.0780	11,175,370	33.200	44.77	0.07	0.000093

CIFAR-10 Comparison

	Model	Accuracy	Precision	Loss	Params	GFLOPs	Size (MB)	Inf Time (ms/img)	CO2 (kg)
0	Custom CNN	0.6593	0.6620	0.9660	132,938	844.550	0.54	0.09	0.000295
1	Pruned MobileNet	0.7913	0.7982	0.6131	2,236,682	6.520	9.17	0.10	0.001489
2	Pruned ResNet18	0.8051	0.8100	0.8618	11,181,642	37.250	44.80	0.12	0.001263

Result and Analysis



Discussion

RQ1: Benchmarking and Comparison

What we did: We measured accuracy alongside five ”Green” metrics: CO2 emissions, GFLOPs, inference time, parameter count and storage size.

Results: The Custom CNN achieved nearly identical accuracy to the pruned pretrained models on MNIST (97%) while using significantly fewer resources.

Significance: This proves that for specific tasks the massive pretrained backbones are not efficient. Our Custom CNN achieved the smallest size and best green score (accuracy per mg CO2) confirming that custom architectures can be more eco-friendly than reduced versions of large models.

Discussion

RQ2: Compression and Optimization Impact

What we did: We applied unstructured Pruning to remove entire channels and evaluated the model's footprint. We specifically analyzed the transition from masking weights (logical pruning) to physical parameter reduction to see real world speedups.

Results: Standard unstructured pruning in PyTorch often acts as a weight mask but smaller compressed file sizes reduce the energy required to transmit the model over a network to an edge device. By using it we got rid of hardware limitations.

Significance: The Custom CNN's small size (< 1 MB) allows it to run entirely within the SRAM of an edge device, preventing the energy-heavy data movement required by the 44 MB ResNet18.

Discussion

RQ3: Green Effectiveness and Deployment Viability

What we did: We calculated a ”Green Score” (Accuracy per mg of CO2) to determine the viability of deployment.

Results: Our Custom CNN attained the highest Green Score across both datasets. For CIFAR-10, even though accuracy was slightly lower than MobileNet, the carbon cost was roughly 5x lower, making the Custom CNN the clear winner for sustainable AI.

Significance: The Custom CNN is ”Green” because it is Edge-Ready.1 Its storage efficiency (small MB size) means it can be deployed with minimal network energy, extending the battery life of IoT sensors and reducing the e-waste associated with needing high-end hardware for AI inference

Limitations

- Benchmarks limited to MNIST and CIFAR-10 (simple/low-resolution datasets).
- Training was short (few epochs), possibly underestimating peak CIFAR-10 performance.
- CO_2 estimates from CodeCarbon on Kaggle hardware — location/hardware variance affects absolute numbers.
- Limited compression methods evaluated (basic pruning + 8-bit quantization); no hardware-specific deployment measurements.

Future Work

- Evaluate on larger / real-world datasets (e.g., ImageNet or custom domain data) and extend training/augmentation.
- Deploy and measure on edge hardware (TinyML, MCUs, mobile devices) to capture real energy/latency gains.
- Explore NAS, knowledge distillation, mixed-precision & advanced quantization (4-bit) to close accuracy gaps.
- Run large-scale, hardware-diverse benchmarks (MLPerf/TinyML) and open-source the code for reproducibility.

References

[1] Strubell, E., Ganesh, A., & McCallum, A. (2019)
"Energy and Policy Considerations for Deep Learning in NLP"
Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 3645–3650.

[2] Han, S., Pool, J., Tran, J., & Dally, W. (2015)
"Learning both Weights and Connections for Efficient Neural Network"
Advances in Neural Information Processing Systems, vol. 28, pp. 1135–1143.

[3] Howard, A. G. et al. (2017)
"MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications"
arXiv preprint arXiv:1704.04861.

THANK YOU.