HIGHLIGHTING THE IMPORTANCE OF REDUCING RESEARCH BIAS AND CARBON EMISSIONS IN CNNS

Ahmed Badar, Arnav Varma, Adrian Staniec, Mahmoud Gamal, Omar Magdy, Haris Iqbal, Elahe Arani*and Bahram Zonooz*

Advanced Research Lab, Navinfo Europe, Eindhoven, The Netherlands

ABSTRACT

Convolutional neural networks (CNNs) have become commonplace in addressing major challenges in computer vision. Researchers are not only coming up with new CNN architectures but are also researching different techniques to improve the performance of existing architectures. However, there is a tendency to over-emphasize performance improvement while neglecting certain important variables such as simplicity, versatility, the fairness of comparisons, and energy efficiency. Overlooking these variables in architectural design and evaluation has led to research bias and a significantly negative environmental impact. Furthermore, this can undermine the positive impact of research in using deep learning models to tackle climate change. Here, we perform an extensive and fair empirical study of a number of proposed techniques to gauge the utility of each technique for segmentation and classification. Our findings restate the importance of favoring simplicity over complexity in model design (Occam's Razor). Furthermore, our results indicate that simple standardized practices can lead to a significant reduction in environmental impact with little drop in performance. We highlight that there is a need to rethink the design and evaluation of CNNs to alleviate the issue of research bias and carbon emissions.

Keywords Green AI · Carbon emmissions · Image Segmentation · Classification

1 Introduction

Deep neural networks (DNNs) have achieved remarkable results in recent years [1], in applications such as image classification [2, 3], speech recognition [4] and automation [5]. A number of techniques have been proposed to further improve the performance of the networks which target different aspects of the learning and inference process. This has led to development of techniques including different learning rate schedulers [6, 7, 8, 9], loss functions [10, 11], optimizers [12, 13] and other network customizations [14, 15, 16, 17, 18]. On the other hand, larger networks are being designed with increased depth and width to improve accuracy.

To further optimize these networks, there is an overemphasis on adding more complexity to either the network architecture or to the training procedure which can come at the cost of simplicity, explainability, inference time and energy efficiency. Moreover due to the energy inefficiency, the overall carbon footprint of the model increases significantly which has an adverse environmental impact [19], for instance, training a common deep learning model can have a larger carbon footprint than half the life cycle of a car [20]. In an effort to create a low carbon society to tackle climate change which is one of the major challenges faced by humanity today, researchers recently proposed using machine learning to decarbonize major pollution contributors such as the transport and the industrial sector [21]. Paradoxically, this approach itself is becoming a major contributor to CO_2 emissions. Thus, addressing the energy efficiency and decarbonization of deep learning models during design and evaluation is vital.

In conjunction with the aforementioned issues, the lack of a standardized methodology for experimentation has led to a non-rigorous evaluation of the utility of the different techniques. The importance of this issue is also underscored by *Bouthillier et al.*, [22]

*Equal Advising

"Reproducibility is not only about code sharing, but most importantly about experiment design".

Furthermore, the lack of a standardized pipeline resulting in unfair comparisons and the neglect of important variables such as energy consumption, design simplicity and evaluation of neural networks can play important roles in reinforcing "research bias", i.e. the process where scientists overlook or influence certain observations to report results in order to portray a certain outcome.

In this paper, we conduct an exhaustive and fair empirical study to evaluate architectural design and a wide spectrum of techniques that have been proposed to improve neural networks performance. Our findings echo the importance of the Occam's Razor principle and further demonstrate that following simple standardized practices can curtail the environmental impact with little to no drop in performance. We conclude that it is crucial to rethink the design and evaluation of neural networks to ameliorate the problem of research bias. Our main contributions are as follows:

- We assess the performance of a diverse set of techniques on different datasets and tasks under common settings.
- We evaluate the effect of each technique on energy consumption and carbon emissions to estimate its environmental impact.
- We address the relationship between computational cost, energy efficiency, and performance gain.
- We highlight the necessity of a standard pipeline for neural network design and evaluation, and to curtail their carbon footprint.
- We show that following simple standardized practices can help in fair comparison of various techniques, thereby mitigating research bias.

2 Related Work

As a consequence of the increasing need for computational resources for deep learning methods, some researchers are now interested in studying and documenting the effect of the carbon footprint of these techniques. Strubell et al. [20] study the energy consumption of different DL models during training. They conclude that the energy required for training models has increased significantly in recent years to a point where the large carbon footprint is becoming a major concern, i.e. energy consumption of the average deep learning model training can produce more than half the carbon emissions of the entire life cycle of an automobile. Schwartz et al. [19] argue that energy efficiency of DL models is as important as the accuracy and put forward multiple methods to estimate the carbon emissions of AI models. The authors contend that researchers should report energy consumption and floating point operations per model as an evaluation metric. Following this recommendation, we report these metrics in our study.

Inference, in general, contributes more towards the carbon footprint of a model during its life cycle compared to the training. To this end, a number of studies have proposed to optimize the inference process. Wang et al. [23] provide a general framework to optimize inference on mainstream integrated GPUs. Moons et al. [24] suggest the use of precision scaling for CNNs to reduce the overall energy consumption. By representing weights and values that are least affected by quantization with a lower precision, they reduce the energy consumption. Dami et al. [25] introduce the idea of "Data Echoing" whereby they optimize the pre-optimizer steps by reusing the idle upstream times of the GPUs to speed up training. However, the focus of all the aforementioned methods is on hardware optimization and they do not investigate how the deep learning architectures themselves affect environment and performance related metrics. After completion of this work, we became aware of the paper by Musgrave et al. [26] where they too emphasize fair comparisons and highlight flaws in experimental setups. However, the scope of their study is metric learning. In this study, we focus on weighing the utility of different recent methods for classification and segmentation tasks on both the training and inference efficiency in terms of overall performance and the energy expended.

3 Methodology

Our methodology is designed to ensure that the DL algorithms are evaluated with minimal effect of the hardware and software used for training, testing and inference. First, we use the same computing environment for all our experiments (Intel 8700 and NVIDIA RTX 2080 Ti). Second, we check all the techniques for each task (segmentation and classification) on the same network that performs competitively on well-known public datasets. We compare each technique with one another alongside the baseline network to determine the overall value these different techniques provide. Third, we provide a comprehensive analysis by not only reporting performance, but also energy consumption and inference time. For segmentation training, we report the mean intersection over union (mIoU) on training and validation sets and the overall energy consumed. For segmentation inference, the total number of parameters, floating point operations (FLOPs), inference time and energy per image are provided. For classification tasks, we report the

same metrics as above with the exception of mIoUs, where classification accuracy is reported instead. In case of multiple possible hyperparameter settings, we report the best results for each technique.

3.1 Base Network Selection

Segmentation. BiSeNet [27] (with a ResNet-18 backbone pretrained on ImageNet [2]) is selected for the segmentation task. We use two datasets for segmentation training and inference, Cityscapes [28] and COCO-Stuff [29]. For both datasets, we use a poly learning rate decay with a decay rate of 0.9, stochastic gradient descent (SGD) as optimizer and cross-entropy (CE) as the loss function. For Cityscapes, we train for 1000epochs with an initial lr of 0.025, batch size 16, and random rescaling between 0.5 to 2 with a base size of 1024, followed by randomly cropped windows of size 512×512. For COCO-Stuff on the other hand, we train for 60 epochs with an initial lr of 0.01, batch size 12, and random rescaling between 0.5 to 2 with a base size of 640, followed by randomly cropped windows of size 640×640.

Classification. ResNet-50 with no pre-training is chosen for the classification task. We train and test the network on CIFAR-100 [30] and Tiny-ImageNet [31] datasets. We use an initial lr of 0.1 with a step-wise lr decay, batch size 128, SGD optimizer and CE as the loss function with maximum number of epoch set to 200.

These networks are selected as they are well-studied and known to perform well in their respective domains. Moreover, we are interested in real-time applications and want to reduce the time spent on multiple training runs. The hyperparameter values for baseline experiments are the same as in the original publications. Unless otherwise stated, we follow the standard training scheme given above.

3.2 Categories for Experimentation

We divide the experiments into several subcategories. Grouping similar categories allows for a concrete comparison of different techniques based on the multiple reported metrics. A summary of our grouping is provided below:

- Architecture Modification
- Learning Rate Scheduler
- Data Augmentation
- Optimizer
- Loss Function
- Custom Nodes and Layer

3.3 How do we measure energy consumption?

We spawn a new thread running parallel to training or inference code. In this thread, we poll GPU power usage (using NVIDIA Management Library [32]) and integrate it over time, similar to the method described in [33]. However, we only sample the power every 10ms as it was the shortest time giving consistent results. Since we use the same data pipeline and dataloader, the CPU power remains same throughout the experiments, making it essentially an offset that we exclude from our reports. To measure training energy, we run all experiments on the same machine for 10 epochs and scale it up to 1000 epochs. The only exception is the progressive resizing technique, where we run the measurement for all epochs as the number of computations varies during the run. To measure inference energy, we run 10000 forward passes with mini-batch size 1 of randomly generated "images". We repeat this procedure three times and report the average energy per image. Lacoste et al. [34] is used to calculate the CO_2 emissions in kg per week (kg/wk) with an RTX 2080 Ti. The CO_2 emissions are calculated for the models on 168 hours of video (1 week) recorded at 2048×1024 and 30 FPS.

4 Experimental Evaluation

We perform systematic experimentation to test each technique with all the major metrics in perspective. Each section is organized as follows: we first describe the relevant techniques, we then report a summary of the results. For techniques that do not modify the network architecture we do not report the inference metrics as they are same as the baseline.

4.1 Architecture Modification

Spatial Branch Ablation (One-Branch). Many recent CNNs use a multi-branch architecture for learning the spatial details and global context separately and then fusing these features to segment the images [27, 35, 36]. In such networks, the context branch is deep (including many layers), and usually performs calculations on an image that is scaled down by a certain factor either via strided convolutions or bilinear resizing, while the spatial branch consists of fewer layers to preserve the spatial detail. Yu et al. [27] show that adding the spatial branch has a very small yet positive effect on

Table 1: Effect of different techniques on mIoU calculated for both training and validation sets, and total GPU energy used for training per run for segmentation on Cityscapes (with image size of 512×512) and COCO-Stuff (with image size of 640x640). All values that are same or better than the baseline are in bold and the best results are highlighted.

Method	Cityscapes			COCO-Stuff		
	mIoU train.	mIoU valid.	Energy train.	mIoU train.	mIoU valid.	Energy train.
Baseline	84.4%	69.3%	19.6 MJ	35.4%	27.1%	65.8 MJ
One-Branch	83.5%	68.6%	**17.1 MJ**	34.48%	26.6%	**64.1 MJ**
Random Grad.	**85.1%**	**69.7%**	19.7 MJ	**38.13%**	22.6%	68.9 MJ
Cyclic LR	80.5%	67.6%	**19.5 MJ**	35.13%	**27.2%**	**65.3 MJ**
Poly LR 1/2 Epochs	82.3%	**69.3%**	**9.8 MJ**	31.51%	26.2%	**35.6 MJ**
Prog. Resize	80.3%	64.5%	**8.0 MJ**	29.08%	24.3%	**30.3 MJ**
Mixup	41.8%	68.1%	20.3 MJ	16.19%	25.5%	84.5 MJ
RAdam	82.5%	67.8%	20.1 MJ	26.79%	21.9%	73.1 MJ
LookAhead	**84.9%**	68.8%	**19.5 MJ**	**39.82%**	**27.4%**	71.6 MJ
Label Relaxation	81.6%	67.4%	33.7 MJ	32.22%	24.9%	95.2 MJ
Dice Loss	**85.6%**	68.5%	25.7 MJ	**36.67%**	26.7%	77.5 MJ
Focal Loss	83.5%	68.5%	20.8 MJ	**36.01%**	**27.6%**	82.7 MJ
BlurPool	**84.6%**	67.3%	26.2 MJ	**36.96%**	**27.4%**	91.1 MJ
SwitchNorm	84.3%	68.7%	19.7 MJ	**36.23%**	**27.3%**	76.8 MJ
Sp. Bottleneck	79.8%	66.0%	20.8 MJ	31.38%	24.9%	71.5 MJ
GE-θ	84.3%	68.7%	**19.6 MJ**	32.15%	24.9%	72.2 MJ
GE-θ^-	84.4%	68.3%	**18.8 MJ**	32.39%	25.2%	71.0 MJ
CoordConv	83.1%	69.0%	24.3 MJ	32.29%	25.3%	83.1 MJ

Table 2: Effect of different techniques on accuracy calculated for both training and validation/test sets, and total GPU energy used for training per run for classification on CIFAR-100 and Tiny-ImageNet. All values that are same or better than the baseline are in bold and the best results are highlighted.

Method	CIFAR-100			Tiny-ImageNet		
	Accuracy train.	Accuracy test	Energy train.	Accuracy train.	Accuracy valid.	Energy train.
Baseline	99.96%	78.65%	2.40 MJ	99.95%	87.15%	6.89 MJ
Random Grad.	99.93%	78.07%	2.43 MJ	98.76%	86.36%	8.16 MJ
Cyclic LR	99.92%	74.08%	**2.35 MJ**	82.46%	74.65%	8.86 MJ
Poly LR 1/2 Epochs	99.89%	77.26%	**1.36 MJ**	99.95%	86.49%	**3.47 MJ**
Mixup	99.35%	**80.65%**	2.62 MJ	96.6%	**87.27%**	7.81 MJ
RAdam	99.50%	73.65%	4.31 MJ	**99.99%**	83.48%	8.91 MJ
LookAhead	**99.97%**	**79.10%**	2.45 MJ	**99.99%**	**88.03%**	9.51 MJ
Focal Loss	99.60%	78.16%	2.41 MJ	98.60%	86.55%	7.68 MJ
BlurPool	99.93%	**79.44%**	3.18 MJ	**99.97%**	86.73%	12.11 MJ
SwitchNorm	99.50%	77.12%	4.63 MJ	99.92%	86.36%	17.14 MJ
Sp. Bottleneck	99.93%	72.76%	4.25 MJ	97.02%	85.22%	11.67 MJ
GE-θ	99.94%	**78.98%**	3.37 MJ	**99.97%**	**87.43%**	22.24 MJ
GE-θ^-	99.93%	78.57%	2.70 MJ	99.95%	87.08%	7.01 MJ
CoordConv	99.95%	**78.82%**	2.79 MJ	99.91%	86.61%	9.12 MJ

mIoU and negligibly effects the inference time. However, no tests are conducted on how this branch can affect energy consumption. Here, we consider the effect of the spatial branch on the energy consumption.

Table 1 shows that there is a negligible difference between the full model (baseline) and the One-Branch in terms of mIoU. The validation mIoU is 0.7 percentage points (pp) and 0.5 pp below the baseline for Cityscapes (CS) and COCO-Stuff respectively, while both the required training energy and inference time (Table 3) are significantly reduced. This highlights the importance of the trade-off between energy consumption and the performance, questioning the utility of such architectural complexity at the cost of energy and computational efficiency.

Table 3: Per image inference metrics for segmentation (Cityscapes 512×512) and (COCO-Stuff 640×640). All values that are same or better than the baseline are in bold and the best results are highlighted. The CO_2 emissions are calculated for full resolution 2048×1024 images.

Method	Cityscapes					COCO-Stuff				
	Params (M)	FLOPs (G)	Time (ms)	Energy (J)	CO_2 (kg/wk)	Params (M)	FLOPs (G)	Time (ms)	Energy (J)	CO_2 (kg/wk)
Baseline	14.01	25.93	6.51	1.57	28.6	14.01	36.3	7.81	4.41	34.3
One-Branch	**13.88**	**22.85**	**5.07**	**1.26**	**22.2**	**13.88**	**32.0 G**	**6.52**	**3.8**	**28.6**
BlurPool	**14.01**	34.16	11.7	2.892	51.4	14.01	47.8	13.9	6.6	61.1
SwitchNorm	**14.01**	25.93	7.25	1.589	31.9	14.01	36.3	8.68	4.8	38.2
Sp. Bottleneck	**14.01**	**15.36**	**6.51**	1.608	**28.6**	14.01	**21.5**	**7.76**	**4.35**	**34.1**
GE-θ	21.88	**25.93**	6.73	1.635	29.5	25.05	36.3	8.06	**4.40**	35.4
GE-θ^-	**14.01**	**25.93**	6.71	1.711	29.5	14.01	36.3	8.04	**4.39**	35.3
CoordConv	14.02	27.16	8.10	2.009	35.6	14.02	38.0	9.71	4.79	42.7

Table 4: Classification CIFAR-100 and Tiny-ImageNet. All values that are same or better than the baseline are in bold and the best results are highlighted. The CO_2 emissions are calculated for resolution 64×64 images.

Method	CIFAR-100					Tiny-ImageNet				
	Params (M)	FLOPs (G)	Time (ms)	Energy (J)	CO_2 (kg/wk)	Params (M)	FLOPs (G)	Time (ms)	Energy (J)	CO_2 (kg/wk)
Baseline	2.37	1.3	**6.60**	0.98	**29.0**	2.39	1.3	**6.60**	**0.92**	**29.0**
BlurPool	**2.37**	1.93	6.70	1.0	29.4	**2.39**	1.93	6.70	0.94	**29.4**
SwitchNorm	**2.37**	**1.3**	23.1	2.4	101.5	**2.39**	**1.3**	22.9	2.2	100.7
Sp. Bottleneck	**2.37**	**0.86**	7.32	1.1	32.1	**2.39**	**0.86**	7.21	1.0	31.7
GE-θ	2.42	**1.3**	6.97	1.0	30.6	2.59	**1.3**	6.90	1.0	30.3
GE-θ^-	**2.37**	**1.3**	6.96	1.0	30.6	**2.39**	**1.3**	6.90	0.98	30.3
CoordConv	2.42	**1.3**	6.66	**0.97**	**29.2**	2.43	**1.3**	6.70	0.94	**29.4**

4.2 Learning Rate Scheduler

A number of learning rate manipulation techniques have been suggested over the years to improve training and inference for CNNs. Below we introduce and analyze the effect of a number of learning rate optimization techniques.

Random Gradient. Random gradient [9] multiplies the learning rate for each mini-batch by a random number sampled from $U([0, 1])$. The authors claim this minimizes fluctuations in the optimization process. This technique has been shown to perform well in many fields.

Cyclic Learning Rate. Cyclic learning rate (CLR) [8], developed to remove the need for a learning rate hyperparameter, is based on cyclically changing the learning rate throughout the training. The variant giving the best result is named *triangular2*, which varies the learning rate linearly and halves the upper limit every cycle. The authors show that it can improve the training and overall accuracy of different types of neural networks. For segmentation, we tried multiple hyperparameter settings and report the best results achieved with `base_lr`=0.0001 and `base_lr`=0.00004, `max_lr`=0.01 and `max_lr`=0.004, and `stepsize`=50 and `stepsize`=3 for Cityscapes and COCO-Stuff respectively. For classification, we set a `base_lr`=0.0001, `max_lr`=0.01 and a `stepsize`=20 epochs for both datasets.

Poly LR with ½ Maximum Epochs. In our base experiment, learning rate follows a polynomial curve. This means that the rate at which the learning rate decreases with each epoch is inversely proportional to maximum number of epochs. We test how having a smaller number of maximum epochs affects training and validation. To ensure that this result is not due to the variability in experiments we run three experiments and report the average mIoU.

Results. In Table 1, we show that for segmentation compared to the baseline, random gradient experiments yield an improvement in mIoU for CS, but a decrease in COCO-Stuff mIoU at the cost of a small increase in energy. On the other hand, for cyclic learning rate, a significant drop in mIoU is observed for CS, whereas a minor increase in mIoU is observed for COCO-Stuff. Running the network for half the number of epochs with a poly lr reduces the energy

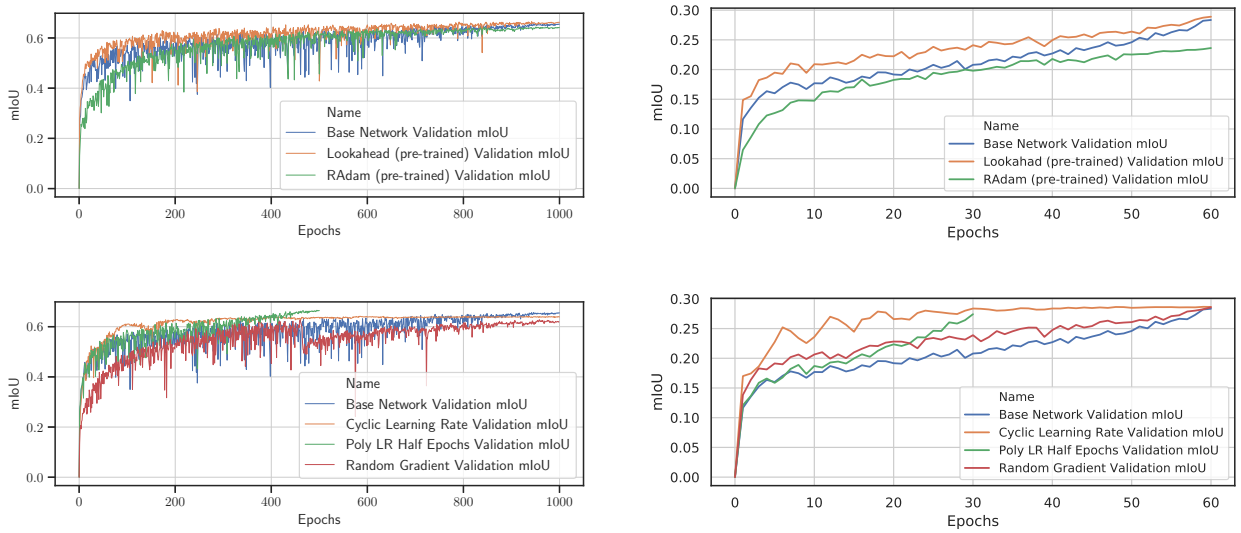


Figure 1: **(Top)** The validation mIoU (calculated on square center crops) profiles for different optimizers(Cityscapes on **Left** and COCO-Stuff on **Right**) indicate that LookAhead optimizer has a better convergence than the baseline while the convergence for RAdam is slower than the baseline. **(Bottom)** The validation mIoU for different learning rate modifications(Cityscapes on **Left** and COCO-Stuff on **Right**). Poly LR with half epochs still gives competent results, even outperforming the remaining methods on Cityscapes. Cycles are visible for CLR. Values don't exactly match the Table as mIoU was calculated only on the square center crops.

consumption (19.6 MJ to 9.8 MJ for CS and 65.8 MJ to 35.6 MJ for COCO) with little to no drop in validation mIoU. This emphasizes that proper experimental design can play an important role in environmental impact.

For classification, we observe that neither CLR nor random gradient affect the energy consumption significantly for CIFAR-100 while for the larger training run with Tiny-ImageNet the training energies go up from the baseline. The overall test/validation accuracy with random gradient is slightly lower, while CLR reports considerably lower accuracy than the baseline. For half the number of maximum epochs with poly lr, we observe that the energy consumption is almost halved while the test/validation accuracy is 1.39 pp and 0.66 pp lower than the baseline for CIFAR-100 and Tiny-ImageNet respectively. However, we noted that our evaluation matches that of the original CLR publication, the only difference being that they use a constant lr for the base experiment.

4.3 Data Augmentation

Different types of data augmentation techniques have been applied to neural networks to deal with the scarcity of training data. These techniques reuse data after applying image processing techniques, such as random crops, random flipping, resizing, brightness and contrast modification. They have, in several cases, shown to improve the robustness of CNNs [37]. We run our baseline experiment with the aforementioned techniques.

Progressive Resizing. Progressive resizing is a technique where the training is sped up by training on downsized images for a given number of epochs, followed by training progressively on relatively larger images until the point where final epochs are done on images of original size [38, 39]. This has been shown to reduce training times significantly. We run our experiment with image size $\frac{1}{8}$ of the original image, and upscale the image by a factor of 2 after every 250 epochs for Cityscapes, and 15 epochs for COCO-Stuff. This is done until we get the original image size, which is kept constant for the rest of the training.

Mixup. Mixup [17] is a simple data augmentation scheme, which trains the network on convex combinations of pairs of data points and the corresponding one-hot representations of their labels, to improve generalization of deep neural networks.

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1-\lambda) x_j \\ \tilde{y} &= \lambda y_i + (1-\lambda) y_j \end{aligned} \tag{1}$$

Where $0 \leq \lambda \leq 1$, x_i, x_j are raw input vectors, and y_i, y_j are one-hot label encodings. Here, we evaluate how the mixup affects the overall training process.

Results. For segmentation, progressive resizing results in an inferior mIoU but considerably better training energy (see Table 1), while for mixup the negative effect on mIoU is less pronounced with a slight increase in energy consumption for both datasets. For classification, we observe that mixup leads to a significant improvement in test/validation accuracy with almost identical energy consumption for CIFAR-100 and increased consumption for Tiny-ImageNet. We do not employ progressive resizing for classification as the CIFAR-100 and Tiny-ImageNet images are too small (32×32) to retain enough information after resizing.

4.4 Optimizer

DNNs are trained using variants of stochastic gradient descent. Here, we evaluate recently proposed optimizers which claim to improve the performance of SGD.

RAdam Optimizer. The RAdam optimizer [12] employs an adaptive learning rate which is rectified to ensure a consistent variance. It effectively provides an automated warm-up custom tailored to the current dataset. Warm-up learning rate strategy sets up a small learning rate in the starting phase of training to cater for the undesirably large variance. After a specific number of epochs, the learning rate is stepped up to a larger value.

$$lr = \begin{cases} x_e \cdot lr_0 & e < n \\ x_n \cdot lr_0 & e \geq n \end{cases} \tag{2}$$

where n is the number of threshold epochs for the warm-up, x_i is the learning rate modifier for ith epoch, e is the current epoch, and lr_0 denotes the base learning rate. In case of the spiking learning rate warm-up we have,

$$lr = \begin{cases} x \cdot lr_0 & e \; mod \; n = 0 \\ lr_0 & else \end{cases} \tag{3}$$

RAdam was run with default hyperparameters used by Liu et al. [12] while keeping all other settings of the optimizer same as the baseline.

LookAhead Optimizer. LookAhead optimizer uses "fast weights" multiple times using another standard optimizer before updating the "slow weights" in the direction of the final fast weights [13] and has been shown to reduce the variance, resulting in better convergence and versatility with little hyperparameter tuning. We run LookAhead with outer optimizer looking ahead every 5 iterations and interpolation coefficient $\alpha = 0.5$. Inner optimizer has the same optimization routine and learning rate schedule as the baseline. Same hyperparameters for LookAhead were chosen as those by Zhang et al. [13] for ImageNet data.

Results. For segmentation, Table 1 shows that RAdam produces an mIoU that is 1.5 pp and 5.1 pp lower than the baseline on Cityscapes and COCO-Stuff respectively. On the other hand, it consumes around 0.5 MJ more energy on Cityscapes, and 7.3 MJ more energy on COCO-Stuff. LookAhead optimizer performs relatively better with the reported mIoU only 0.5 pp below the Cityscapes baseline with similar energy consumption, and 0.3 pp above the COCO-Stuff baseline with 5.8 MJ more energy consumption. Figure 1 shows that lookahead optimizer improves the convergence over the baseline (SGD). For classification, in Table 2 we observe a test accuracy drop of 5.0 pp for RAdam and an increase of 0.45 pp for lookahead on CIFAR-100, and a validation accuracy drop of 3.67 pp for RAdam and an increase of 0.88 pp for lookahead on Tiny-ImageNet, indicating that the former reduces accuracy while the latter improves it. Lookahead has almost the same energy as the CIFAR-100 baseline but consumes more energy on Tiny-ImageNet, while the RAdam optimizer consumes significantly more energy on both datasets.

4.5 Loss Function

There is no loss function that suits all deep learning algorithms. The choice of loss function can have a significant effect on the overall performance of a deep learning model as well as the speed of convergence.

Boundary Label Relaxation. The boundary label relaxation [11] leverages the idea that segmentation boundaries close to one another can easily be classified as one or the other without any marked impact on the quality of segmentation. Based on this, boundary label relaxation calculates a soft cross-entropy loss value, where the summation of the probabilities of all the neighbouring classes at a given pixel is used as the final class probability for that pixel.

$$P(C) = P(A \cup B) = P(A) + P(B) \tag{4}$$

$$L_{Relax} = -log \sum_{C \in N} P(C) \tag{5}$$

where A and B are the mutually exclusive classes and $P(.)$ represents the softmax class probability. We perform the experiment with label relaxation where a 3×3 kernel is used to check for neighbouring pixels.

Soft Dice Loss. The dice coefficient gives a measure of overlap between two sample sets. Dice loss applies the dice coefficient to measure the overlap between the softmax output and the ground-truth [40, 41].

$$L_{Dice}(p, \hat{p}) = 1 - \frac{2 \sum_{pixels} p\hat{p}}{\sum_{pixels} p + \sum_{pixels} \hat{p}} \tag{6}$$

We use,

$$L_{SoftDice} = L_{CE} + \gamma L_{Dice} \tag{7}$$

where γ is a scaling factor for the dice loss. When the network is trained with dice loss alone, significantly lower performance is observed. Therefore, we tested with multiple values of γ and got the best performance with $\gamma = 1$.

Focal Loss. Lin *et al.* [42] introduced focal loss after observing that two stage detectors are more accurate because of the inherent extreme foreground-background class imbalance. They introduce focal loss to address this class imbalance by focusing more on hard negatives and down-weighing the easier samples. In this study, we adapt the focal loss for segmentation by treating each pixel as a separate sample.

$$L_{Focal}(p, \hat{p}) = - \sum_{pixels} [((1 - \hat{p})^{\gamma} \log(\hat{p})) \cdot p], \tag{8}$$

where $\hat{p}$ is softmax output vector, $\cdot$ is a dot product, and p is a one-hot label vector. We train with $\gamma = 2$ as according to author it yields the best results.

Results. For segmentation, the boundary label relaxation lowers the mIoU by 1.9 pp and 2.2 pp for CS and COCO-Stuff respectively while consuming significantly more energy. For dice loss, we get a lower mIoU (-0.8 pp for CS and -0.4 pp for COCO-Stuff) relative to the baseline and 6.1 MJ and 11.7 MJ increase in energy consumption. Additionally, an improvement (1.2 pp for both datasets) in training mIoU is observed. For the focal loss, we observe a drop of 0.8 pp and an increase of 0.5 pp in validation mIoU with more energy consumption for CS and COCO-Stuff respectively. For classification, we test only focal loss and observe that it marginally reduces the test/validation accuracy (0.49 pp and 0.60 pp for CIFAR-100 and Tiny-ImageNet respectively).

4.6 Custom Nodes and Layers

A number of studies have been carried out to improve the neural networks performance by introducing new custom nodes and layers.

BlurPool (Shift-Invariant Pooling). Zhang [18] introduced a new way to perform the pooling operation for resizing features, called BlurPool. This challenges the common assumption that CNNs are inherently shift-invariant. This is evident from the fact that even small translational perturbations (e.g. due to max pooling, resizing etc.) of an image can produce a drastically erroneous result. To counter the effect of this uncertainty, anti-aliasing is used in conventional image processing. The study uses the same idea for neural networks by adding anti-aliasing filters prior to downsampling. We run the BlurPool experiment by modifying all downsampling operations with BlurPool layers of kernel size 5×5 while keeping the strides same as the original downsampling operation.

Switchable Normalization. A number of normalization techniques have been put forward in recent years, such as batch normalization (BN) [43], instance normalization (IN) [44] and layer normalization (LN) [45]. The relative performance of each technique depends on the task at hand. Switchable Normalization (SN) [46] learns to combine different normalization techniques. This experiment is carried out with all the batch normalization layers replaced by switchable normalization layers. For segmentation, we use switchable normalization everywhere except the backbone. We report these results in Table 1 and 3.

Spatial Bottleneck. Spatial bottlenecks [47] decomposes convolutions into two stages for computational efficiency. The first stage reduces the spatial computation using stride k for the convolution layer. The second stage uses transposed convolution layer with the same stride k to recover the original size. According to authors, this reduces computations by a factor of $2/k^2$, and can hence be used to reduce computation with a tolerable reduction in accuracy. The spatial bottleneck experiments are run with the ResNet18 backbone modified to include the ResBlocks with spatial bottleneck modules instead of the standard convolution and deconvolution.

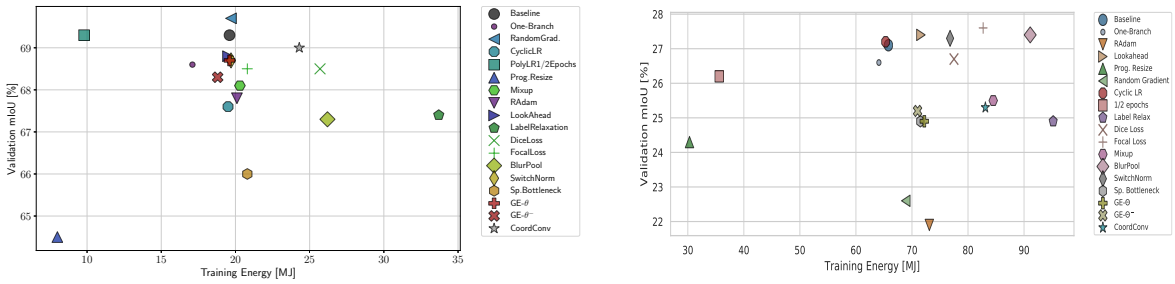
Gather-Excite Operators. Hu et al. [15] introduced gather-excite (GE) operators to better capture long range dependencies and interactions across an image. The gather operator aggregates the features with a large receptive field and then the excite operator redistributes the condensed information to the local features without adding much to the computational overhead. We refer to the GE modules with trainable parameters as GE-θ and the non-trainable modules as GE-θ^-. GE-θ and GE-θ^- experiments are also carried out by attaching these modules to the outputs of each of the ResBlocks in the ResNet18 backbone.

Coordinate Convolution. The Coordinate Convolution (CoordConv), introduced by Liu et al. [16], concatenates pixel coordinates to the input tensor before a convolution, to propagate spatial information to deeper layers. The coordinates are added as additional channels having x, y and radial coordinates correspondingly. The authors claim that the technique improves localization accuracy for GANs and classification tasks. For the CoordConv experiment, we concatenate the Cartesian and radial features just to the first layer of the backbone and the first layer of the spatial path. We use the same range of co-ordinates for training and testing, despite cropping half the pixels in training as this yields the higher mIoU.

Results. In segmentation training for both Cityscapes and COCO-Stuff, we observe similar trends with a few exceptions and notable results. For Cityscapes, we observe that Random Gradient yields the best training and validation mIoU, that is 0.7 pp and 0.4 pp higher, respectively, than the baseline. This improvement could be attributed to the lower lr for training as Random Gradient multiplies the lr with $0 < \alpha < 1$. Another result of note is that for $1/2$ the number of epochs, we get the same validation mIoU, indicating that training for more epochs (in the baseline) leads to overfitting. For COCO-Stuff on the other hand, we observe that focal loss yields the best result as the data balancing from focal loss plays a greater role for the larger and more imbalanced dataset. The most notable results for training energy we observe are on progressive resizing and $1/2$ epoch run for both the datasets. Even though progressive resizing runs for twice the number of epochs, it consumes less energy than the $1/2$ epoch run, indicating that image size has a greater impact on training energy than the number of epochs.

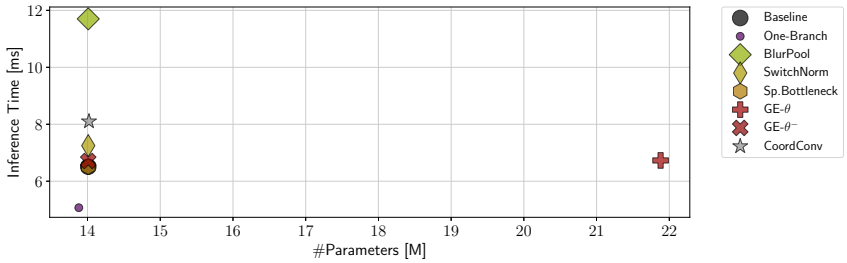
For both the classification datasets, most of the train accuracy values are over 99% with much lower test accuracy, indicating that all the methods overfit on the training set. For test accuracy, we observe that methods such as Mixup and Lookahead yield the best results. Lookahead optimizer performs well as it reduces the variance of the gradients of randomly picked batch sizes, while Mixup improves performance because of the strong regularization it provides. In terms of training energy, as expected, running half the number of epochs leads to lowest consumption.

For segmentation inference, an interesting observation we make is that the inference speed is not directly proportional to FLOPs as variables such as data parallelism and memory transfer can play a role in overall inference speed. Additionally, note that the difference in inference times observed for COCO-Stuff and Cityscapes is because we report the results for them at different resolutions (640×640 and 512×512 respectively). In case of classification inference, we observe that all the methods are slower than the baseline and thus possess a larger carbon footprint than the baseline.

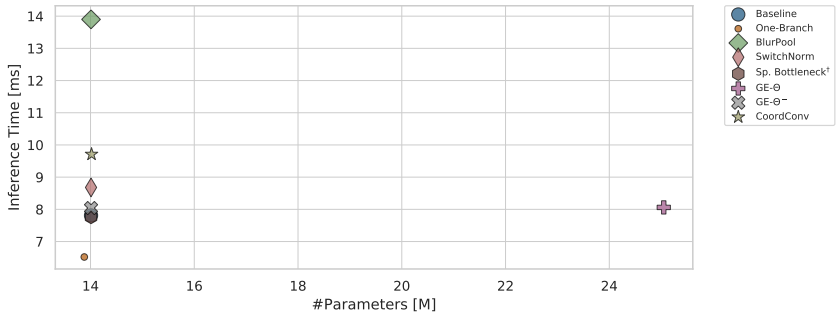


(a) (b)

Figure 2: Validation mIoU vs. training energy for each technique on **(a)** Cityscapes and **(b)** COCO-Stuff: no correlation between the training energy and validation mIoU is observed.



(a)



(b)

Figure 3: Inference Time vs. Number of parameters for each technique on **(a)** Cityscapes and **(b)** COCO-Stuff: The number of parameters is not the only determinant for inference time. This highlights the need to make effective use of parameters in NN design, for instance, ensuring minimal memory overhead, and high parallelization.

5 Discussion

This study empirically evaluated architectural design and a broad spectrum of methods proposed to improve CNNs performance to highlight the issue of research bias in deep learning. Training an average deep learning model can have an immense impact on the environment. For instance, a model can have as much as more than half the carbon footprint of a car's entire life-cycle [20]. On the other hand, researchers have recently proposed the use of machine learning models to tackle climate change [21]. Paradoxically, these models themselves can have a considerable environmental impact. We therefore perform an extensive evaluation in order to assess the suitability of different techniques for distinct tasks. We used BiSeNet(ResNet-18 backbone) as our baseline for segmentation task, and observed that removing the spatial branch had a modest impact on the accuracy but considerably improved the carbon footprint and inference time. This again emphasizes the importance of model simplicity in addition to the overall performance during the design process. On the other hand, for training, we also showed that by simply reducing the maximum number of epochs, we can get the same accuracy at half the carbon cost. We evaluated a wide variety of other techniques, such as CoordConv, anti-aliasing for shift-invariance, optimizers such as LookAhead, RAdam, and loss functions such as Focal Loss, Dice Loss, and observe that a majority of these techniques have a negligible effect on the overall network performance. Moreover, some of these techniques considerably increased the complexity, carbon footprint and inference times. From the results in Figure 2, we could not observe a clear trade-off between accuracy and energy savings. In addition, Figure 3 illustrated that the number of parameters is not the only determinant of the inference time. These results highlighted the need to design networks with a broader spectrum of variables under consideration. More extensive demonstrations of the lack of correlation between these variables can be found in Appendix A. While machine learning competitions such as Kaggle have been a great boon towards advancing research, they have exacerbated the issue of research bias by causing researchers to overlook several important variables such as environmental impact, and versatility. This appears to be a good example of Goodhart's Law which states that "When a measure becomes a target it ceases to be a good one" [48]. While we report our results on vision based applications, the same conclusions may extend to other DL intensive applications such as NLP [49, 50]. In this study, we call the community's attention to these important variables, and ask them to rethink the experimental design for deep learning; first by taking these important variables into account, and second by following a standard pipeline. Simple standardized practices can lead to significant reduction in environmental impact, as evidenced by Section 4.2. We also suggest the research community to

take a step back and focus on the bigger picture when designing the networks, instead of targeting minute improvements with narrow applicability at the cost of simplicity, versatility, and energy.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

[4] Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8599–8603. IEEE, 2013.

[5] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[6] Elad Hazan, Adam R. Klivans, and Yang Yuan. Hyperparameter optimization: A spectral approach. *CoRR*, abs/1706.00764, 2017. URL `http://arxiv.org/abs/1706.00764`.

[7] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016. URL `http://arxiv.org/abs/1608.03983`.

[8] Leslie N. Smith. No more pesky learning rate guessing games. *CoRR*, abs/1506.01186, 2015. URL `http://arxiv.org/abs/1506.01186`.

[9] Jiakai Wei. Fast, better training trick - random gradient. *CoRR*, abs/1808.04293, 2018. URL `http://arxiv.org/abs/1808.04293`.

[10] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017. URL `http://arxiv.org/abs/1708.02002`.

[11] Yi Zhu, Karan Sapra, Fitsum A. Reda, Kevin J. Shih, Shawn D. Newsam, Andrew Tao, and Bryan Catanzaro. Improving semantic segmentation via video propagation and label relaxation. *CoRR*, abs/1812.01593, 2018. URL `http://arxiv.org/abs/1812.01593`.

[12] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[13] Michael R. Zhang, James Lucas, Geoffrey E. Hinton, and Jimmy Ba. Lookahead optimizer: k steps forward, 1 step back. *CoRR*, abs/1907.08610, 2019. URL `http://arxiv.org/abs/1907.08610`.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, September 2015. ISSN 0162-8828. doi:10.1109/TPAMI.2015.2389824.

[15] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Andrea Vedaldi. Gather-excite: Exploiting feature context in convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 9401–9411, 2018.

[16] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. *CoRR*, abs/1807.03247, 2018. URL `http://arxiv.org/abs/1807.03247`.

[17] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[18] Richard Zhang. Making convolutional networks shift-invariant again. *arXiv preprint arXiv:1904.11486*, 2019.

[19] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *arXiv preprint arXiv:1907.10597*, 2019.

[20] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

[21] David Rolnick, Priya L Donti, Lynn H Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, et al. Tackling climate change with machine learning. *arXiv preprint arXiv:1906.05433*, 2019.

[22] Xavier Bouthillier, César Laurent, and Pascal Vincent. Unreproducible research is reproducible. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 725–734, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/bouthillier19a.html.

[23] Leyuan Wang, Zhi Chen, Yizhi Liu, Yao Wang, Lianmin Zheng, Mu Li, and Yida Wang. A unified optimization approach for CNN model inference on integrated gpus. *CoRR*, abs/1907.02154, 2019. URL http://arxiv.org/abs/1907.02154.

[24] Bert Moons, Bert De Brabandere, Luc Van Gool, and Marian Verhelst. Energy-efficient convnets through approximate computing. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8. IEEE, 2016.

[25] Dami Choi, Alexandre Passos, Christopher J. Shallue, and George E. Dahl. Faster neural network training with data echoing. *CoRR*, abs/1907.05550, 2019. URL http://arxiv.org/abs/1907.05550.

[26] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. *arXiv preprint arXiv:2003.08505*, 2020.

[27] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation. *CoRR*, abs/1808.00897, 2018. URL http://arxiv.org/abs/1808.00897.

[28] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[29] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1209–1218, 2018.

[30] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[31] H. Pouransari and Saman Ghili. Tiny imagenet visual recognition challenge. 2014.

[32] NVDIA. pynvml: Python bindings to the nvidia management library (nvml). https://pypi.org/project/pynvml/, 2019.

[33] Martín Pi Puig, Laura Cristina De Giusti, Marcelo Naiouf, and Armando Eduardo De Giusti. Gpu performance and power consumption analysis: A dct based denoising application. In *XXIII Congreso Argentino de Ciencias de la Computación (La Plata, 2017).*, 2017.

[34] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

[35] Rudra P. K. Poudel, Ujwal Bonde, Stephan Liwicki, and Christopher Zach. ContextNet: Exploring Context and Detail for Semantic Segmentation in Real-time. *CoRR*, abs/1805.04554, 2018. URL http://arxiv.org/abs/1805.04554.

[36] Rudra P. K. Poudel, Stephan Liwicki, and Roberto Cipolla. Fast-SCNN: Fast Semantic Segmentation Network. *CoRR*, abs/1902.04502, 2019. URL http://arxiv.org/abs/1902.04502.

[37] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[38] Elahe Arani, Shabbir Marzban, Andrei Pata, and Bahram Zonooz. RGPNet: A real-time general purpose semantic segmentation. *arXiv preprint arXiv:1912.01394*, 2019.

[39] Jeremy Howard et al. fast.ai. https://github.com/fastai/fastai, 2018.

[40] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571. IEEE, 2016.

[41] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. *Lecture Notes in Computer Science*, page 240–248, 2017. ISSN 1611-3349. doi:10.1007/978-3-319-67558-9_28. URL http://dx.doi.org/10.1007/978-3-319-67558-9_28.

[42] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[43] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[44] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization, 2016.

[45] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[46] Ping Luo, Jiamin Ren, Zhanglin Peng, Ruimao Zhang, and Jingyu Li. Differentiable learning-to-normalize via switchable normalization, 2018.

[47] Junran Peng, Lingxi Xie, Zhaoxiang Zhang, Tieniu Tan, and Jingdong Wang. Accelerating deep neural networks with spatial bottleneck modules, 2018.

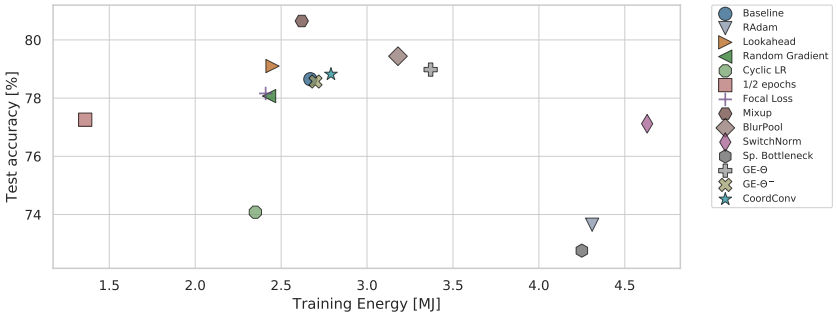
[48] Marilyn Strathern. 'improving ratings': audit in the british university system. *European Review*, 5(3):305–321, 1997. doi:10.1002/(SICI)1234-981X(199707)5:3<305::AID-EURO184>3.0.CO;2-4.

[49] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency; Association for Computing Machinery: New York, NY, USA*, 2021.

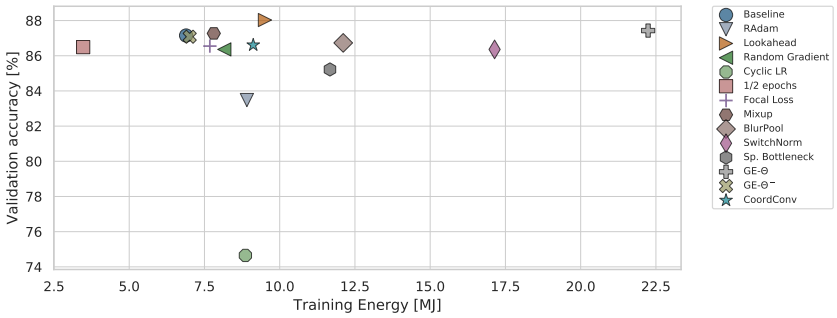
[50] Kawin Ethayarajh and Dan Jurafsky. Utility is in the eye of the user: A critique of nlp leaderboards, 2020.

A Appendix

Figures provided below further demonstrate the lack of correlation between accuracy/mIoU, and training energy and CO_2 emissions.

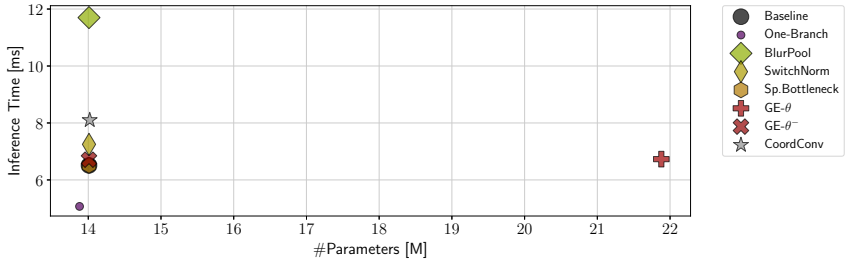


(a)

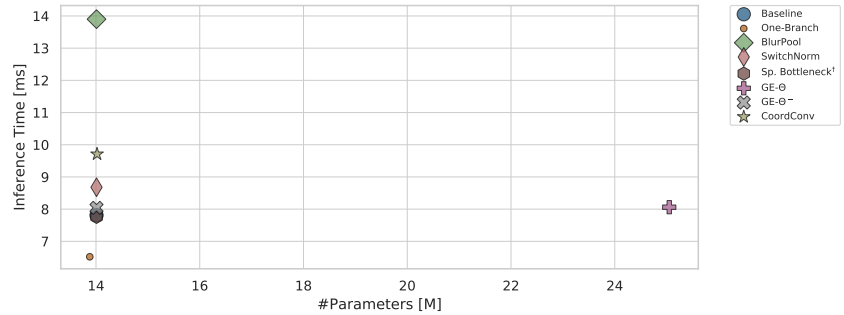


(b)

Figure 4: Accuracy vs. training energy for each technique on **(a)** CIFAR-100 test set and **(b)** Tiny-ImageNet validation set : no correlation between the training energy and validation/test accuracy is observed.

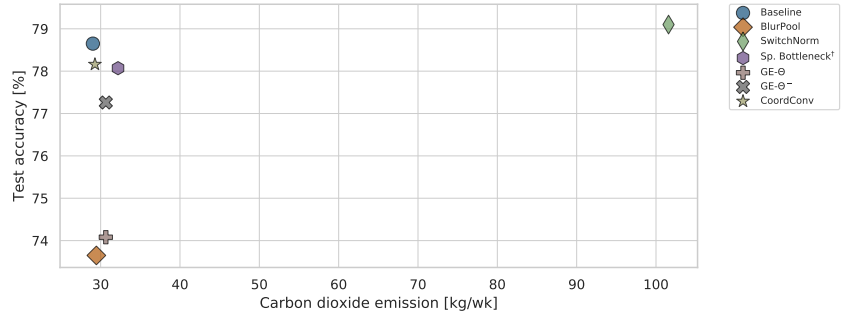


(a)

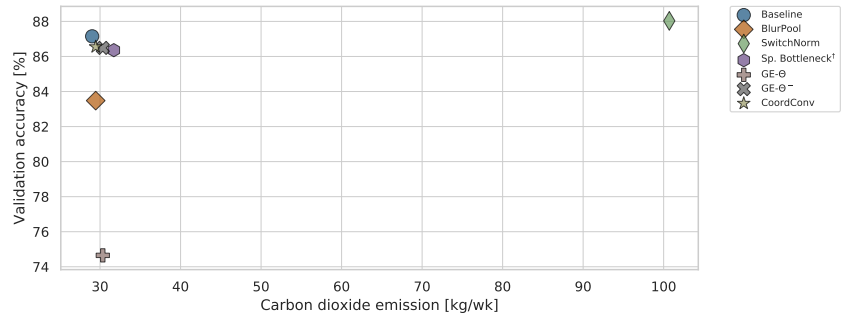


(b)

Figure 5: Inference Time vs. Number of parameters for each technique on **(a)** CIFAR-100 test set and **(b)** Tiny-ImageNet validation set : The number of parameters is not the only determinant for inference time. This highlights the need to make effective use of parameters in NN design, for instance, ensuring minimal memory overhead, and high parallelization.

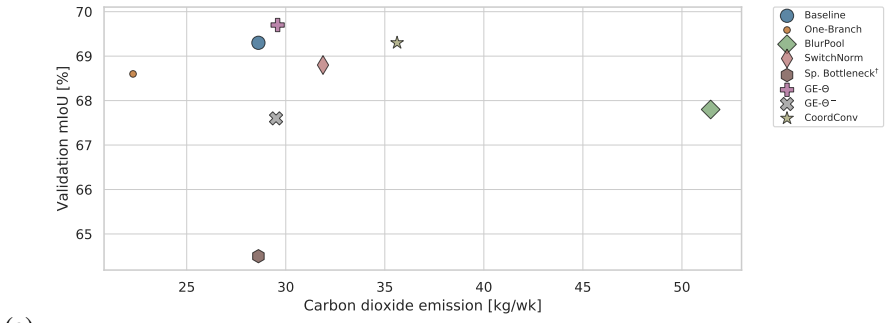


(a)

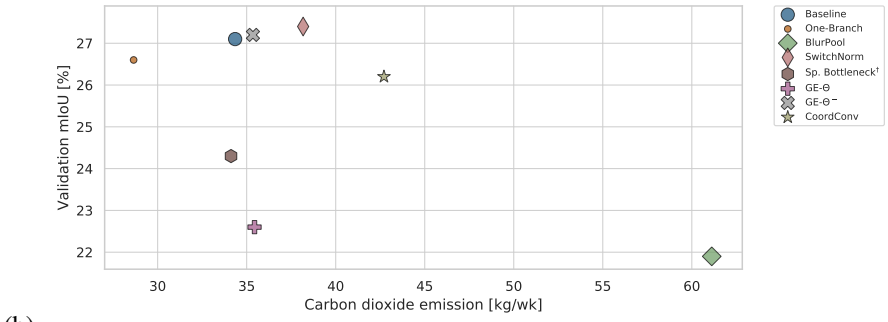


(b)

Figure 6: Accuracy vs. CO_2 emissions for each technique on **(a)** CIFAR-100 test set and **(b)** Tiny-ImageNet validation set : No correlation between test/val accuracy and CO_2 emission.



(a)



(b)

Figure 7: Validation mIoU vs. CO_2 emissions for each technique on **(a)** Cityscapes and **(b)** COCO-Stuff : No correlation between validation mIoU and CO_2 emission.