Sustainable Computing: Informatics and Systems 38 (2023) 100857

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

Trends in AI inference energy consumption: Beyond the performance-vs-parameter laws of deep learning

Radosvet Desislavov[1], Fernando Martínez-Plumed[*,1], José Hernández-Orallo[1]

VRAIN, Universitat Politecnica de Valencia, Spain

ARTICLE INFO

Keywords:
Artificial Intelligence
Deep learning
Inference
Energy consumption
Performance analysis
Performance evaluation
AI progress

ABSTRACT

The progress of some AI paradigms such as deep learning is said to be linked to an exponential growth in the number of parameters. There are many studies corroborating these trends, but does this translate into an exponential increase in energy consumption? In order to answer this question we focus on inference costs rather than training costs, as the former account for most of the computing effort, solely because of the multiplicative factors. Also, apart from algorithmic innovations, we account for more specific and powerful hardware (leading to higher FLOPS) that is usually accompanied with important energy efficiency optimisations. We also move the focus from the first implementation of a breakthrough paper towards the consolidated version of the techniques one or two year later. Under this distinctive and comprehensive perspective, we analyse relevant models in the areas of computer vision and natural language processing: for a sustained increase in performance we see a much softer growth in energy consumption than previously anticipated. The only caveat is, yet again, the multiplicative factor, as future AI increases penetration and becomes more pervasive.

1. Introduction

As Deep Neural Networks (DNNs) become more widespread in all kinds of devices and situations, what is the associated energy cost? In this work we explore the evolution of different metrics of deep learning models, paying particular attention to *inference*, i.e., deployment of a trained model, and its associated computational cost and energy consumption. The full impact, and its final carbon footprint, not only depends on the internalities (hardware and software directly involved in their operation) but also on the externalities (all social and economic activities around it). From the AI research community, we have more to say and do about the former. Accordingly, more effort is needed, within AI, to better account for the internalities, as we do in this paper.

In our study, we differentiate between training and inference. At first look it seems that training cost is higher. However, for deployed systems, inference costs exceed training costs, because of the multiplicative factor of using the system many times. Training, even if it involves repetitions, is done once but inference is done repeatedly. Several sources, including companies in the technology sector such as Amazon or NVIDIA, estimate that inference can exceed the cost of training in pervasive systems, and that inference accounts for up to 90% of the machine learning costs for deployed AI systems [1–5]. There are several studies about training computation and its environmental impact [6–11] but there are very few focused on inference costs and their associated energy consumption (and many of them focused on post-processing techniques to reduce the energy consumption [12,13]).

DNNs are deployed almost everywhere [14], from smartphones to automobiles, all having their own compute, temperature and battery limitations. Precisely because of this, there has been a pressure to build DNNs that are less resource demanding, even if larger DNNs usually outperform smaller ones. Alternatively to this in-device use, many larger DNNs are run on data centres, with people accessing them repeated in a transparent way, e.g., when using social networks [15]. Millions of requests imply millions of inferences over the same DNN.

Many studies report that the size of neural networks is growing exponentially [16,17]. However, this does not necessarily imply that the cost is also growing exponentially, as more weights could be implemented with the same amount of energy, mostly due to hardware specialisation but especially as the energy consumption per unit of compute is decreasing. Also, there is the question of whether the changing costs of energy and their carbon footprint [18] should be added to the equation. Finally, many studies focus on the state-of-the-art (SOTA) or the cutting-edge methods according to a given metric of performance, but many algorithmic improvements usually come in the months or few years after a new technique is introduced, in the form of *general use* implementations having similar results with much lower compute requirements. All these elements have been studied separately, but a

* Corresponding author.
E-mail address: fmartinez@dsic.upv.es (F. Martínez-Plumed).

[1] All authors have contributed equally.

https://doi.org/10.1016/j.suscom.2023.100857
Received 16 January 2022; Received in revised form 21 October 2022; Accepted 20 February 2023
Available online 26 February 2023

more comprehensive and integrated analysis is necessary to properly evaluate whether the impact of AI on energy consumption and its carbon footprint is alarming or simply worrying, in order to calibrate the measures to be taken in the following years and estimate the effect in the future.

For conducting our analysis we chose two representative domains: Computer Vision (CV) and Natural Language Processing (NLP). For CV we analysed image classification models, and ImageNet [19] more specifically, because there is a great quantity of historical data in this area and many advances in this domain are normally brought to other computer vision tasks, such as object detection, semantic segmentation, action recognition, or video classification, among others. For NLP we analysed results for the General Language Understanding Evaluation (GLUE) benchmark [20], since language understanding is a core task in NLP.

We focus our analysis on inference FLOPs (Floating Point Operations) required to process one input item (image or text fragment). We collect inference FLOPs for many different DNNs architectures following a comprehensive literature review. Since hardware manufacturers have been working on specific chips for DNN, adapting the hardware to a specific case of use leads to performance and efficiency improvements. We collect hardware data over the recent years, and estimate how many FLOPs can be obtained with one Joule with each chip. Having all this data we finally estimate how much energy is needed to perform one inference step with a given DNN. Our main objective is to study the evolution of the required energy for one prediction over the years.

The main findings and contributions of this paper are to (1) showcase that better results for DNN models are in part attributable to algorithmic improvements and not only to more computing power; (2) determine how much hardware improvements and specialisation is decreasing DNNs energy consumption; (3) report that, while energy consumption is still increasing exponentially for new cutting-edge models, DNN inference energy consumption could be maintained low for increasing performance if the efficient models that come relatively soon after the breakthrough are selected.

We provide all collected data and performed estimates as a data set, publicly available in the appendixes and a repository in: https://bit.ly/3DTHvFC. The rest of the paper covers the background, introduces the methodology and presents the analysis of hardware and energy consumption of DNN models and expounds on some forecasts. Discussion and future work close the paper.

2. Background

In line with other areas of computer science, there is some previous work that analyses compute and its cost for AI, and DNNs more specifically [6–11]. Many of these studies focus on training costs. For instance, the ImageNet dataset has usually been used to determine trends in compute requirements versus performance, usually referred to as ‘AI efficiency’ [21]. This study shows that 44 times less compute was required in 2020 to train a network with the same performance AlexNet achieved seven years before.

Despite this increase in efficiency, the demand for better task performance, linked with more complex DNNs and larger volumes of data, has motivated a growth in AI compute that may not be compensated by this increased efficiency. Thompson et al. [11] report the computational demands of several Deep Learning applications, showing that progress in them is still strongly reliant on increases in computing power. This is expressed in the form of ‘scaling laws’. As a result, it has been estimated that AI models have doubled the computational power they use every 3.4 months since 2012 [6]. Gholami et al. [7] report similar scaling rates for AI training compute but they forecast that DNNs memory requirements will soon become a problem. The focus is being put on whether this exponential trend may have a limit on how far we can improve performance in the future without a paradigm change.

The studies on compute, performance and resources are more important for another reason. They must help society and AI researchers realise the issues about efficiency and energy consumption. For instance, Schwartz et al. [22] analyses training costs and propose that researchers should put more attention on efficiency and they should report always the number of FLOPs. Strubell et al. [23] estimate the energy consumption, cost and CO_2 emissions of training some popular NLP models. Henderson et al. [24] perform a systematic reporting of the energy and carbon footprints of reinforcement learning algorithms. Section 5.3 in Bommasani et al. [25] seeks to identify the assumptions that shape the environmental impact for foundation models. All these studies contribute to a better assessment of the problem and create more incentives for their solution. For instance, new algorithms and architectures such as EfficientNet [26] and EfficientNetV2 [27] have aimed at this reduction in compute.

Compared to training costs, there are fewer studies on inference costs, despite using a higher share of compute and energy, because of multiplicative factors (number of inferences) [28]. Canziani et al. [8] is one of the first studies focusing on inference costs. They analysed accuracy, memory footprint, parameters, operations count, inference time and power consumption of 14 ImageNet models. To measure the power consumption they ran DNNs on a NVIDIA Jetson TX1 board. A similar study [9] measures energy efficiency, Joules per image, for a single forward and backward propagation iteration (a training step). This study benchmarks 4 Convolutional Neural Networks (CNNs) on CPUs and GPUs on different frameworks. Their work shows that GPUs are more efficient than CPUs (for the analysed CNNs). Both publications study model efficiency, but they do this for very concrete cases. In this paper we will analyse a greater number of DNNs and hardware components in a longer time frame.

3. Methodology

When dealing about computing effort and computing speed (hardware performance), terminology is usually confusing. For instance, the term ‘compute’ is used ambiguously, sometimes applied to the number of operations or the number of operations per second. However, it is important to clarify what kind of operations and the acronyms for them. In this regard, we will use the acronym FLOPS to measure hardware performance, by referring to the number of floating point operations *per second*, as standardised in the industry, while FLOPs will be applied to the amount of computation for a given task (e.g., a prediction or inference pass), by referring to the number of operations, counting a multiply-add operation pair as two operations. An extended discussion about this can be found in the Appendix.

We collect most of our information directly from research papers that report results, compute and other data for one or more newly introduced techniques for the benchmarks and metrics we cover in this work. We manually read and inspected the original paper and frequently explored the official GitHub repository, if exists. However, often there is missing information in these sources, so we need to get the data from other sources, namely:

- *Related papers*: usually the authors of another paper that introduces a new model compare it with previously existing models, providing further information.
- *Model implementations*: `PyTorch` [29] contains many (pre-trained) models, and their performance is reported. Other projects do the same (see, e.g., [30,31]).
- *Existing data compilations*: there are some projects and public databases collecting information about deep learning architectures and their benchmarks, e.g., [7,32–35].
- *Measuring tools*: when no other source was available or reliable, we used the `ptflops` library [36] or similar tools to calculate the model’s FLOPs and parameters (when the implementation is available).

Given this general methodology, we now discuss in more detail how we made the selection of CV and NLP models, and the information about hardware.

3.1. CV models data compilation

There is a huge number of models for image classification, so we selected models based on two criteria: popularity and accuracy. For popularity we looked at the times the paper presenting the model has been cited (from sources such as in Scopus[2] or Google Scholar[3]) and whether the model appears mentioned in other papers (e.g., for comparative analyses). We focused on model's accuracy as well because having the best models per year in terms of accuracy is necessary for analysing progress. To achieve this we used existing compilations [35] and filtered by year and accuracy. For our selection, accuracy was more important than popularity for recent models, as they are less cited than the older ones because they have been published for a shorter time. Once we selected the sources for image classification models, we collected the following information: Top-1 accuracy on ImageNet, number of parameters, FLOPs per forward pass, release date and training dataset. Further details about model selection, FLOPs estimation, image cropping [37] and resolution [38,39] can be found in the Appendix (and Table B.2).

3.2. NLP models data compilation

For NLP models we noted that there is much less information about inference (e.g., FLOPs) and the number of models for which we can get the required information is smaller than for CV. We chose GLUE for being sufficiently representative and its value determined for a good number of architectures. To keep the numbers high we just included all the models since 2017 for which we found inference compute estimates [40]. Further details about FLOPs estimation and counting can be found in the Appendix (selected models in Table I.7).

3.3. Hardware data compilation

Regarding hardware evolution, we collected data for Nvidia GPUs.[4] We chose Nvidia GPUs because they represent one of the most efficient hardware platforms for DNN[5] and they have been used for Deep Learning in the last 10 years, so we have a good temporal window for exploration. Note that for this analysis we only focus on GPUs, as this is the current trend in the area of AI compared to FPGAs [28,41,42]. As of today, in AI applications where speed and reaction times are critical, GPUs deliver benefits in learning and reaction time, and they have the capability to process large amounts of data needed for AI and neural networks, compared to FPGAs. Only in those cases where the algorithm we want to implement is "simple", can FPGAs be faster and more energy efficient than a GPU [42,43]. This is because FPGAs turn out to be more limited in terms of their usage and therefore require that the algorithm can be easily implemented by combinational logic, and that the programmable processor is not needed (i.e., instruction fetching, decoding, control, etc. is just unnecessary overhead).

In particular, we collected GPU data for Nvidia GPUs from 2010 to 2021. The collected data is: FLOPS, memory size, power consumption (reported as Thermal Design Power, TDP) and launch date. As explained before, FLOPS is a measure of computer performance. From the FLOPS and power consumption we calculate the efficiency, dividing

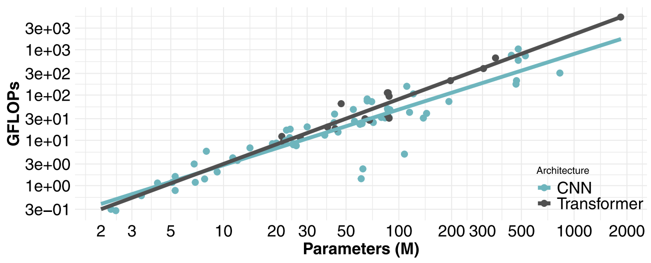

Fig. 1. Relation between the number of parameters and FLOPs (both axes are logarithmic).

FLOPS by Watts. We use TDP and the reported peak FLOPS to calculate efficiency. This means we are considering the efficiency (GLOPS/Watt) when the GPU is at full utilisation. In practice the efficiency may vary depending on the workload, but we consider this estimate ("peak FLOPS"/TDP) accurate enough for analysing the trends and for giving an approximation of energy consumption. In our compilation there are desktop GPUs and server GPUs. We pay special attention to server GPUs released in the last years, because they are more common for AI, and DNNs in particular. We do not include in our analysis those low-power embedded systems designed for standalone machines and other embedded applications (e.g., NVIDIA Jetson family[6]). If this were the case, the results on energy consumption would be optimistic, taking into account that many implementations do not use these systems. Also, to make any claim about compensating energy consumption more credible, we have preferred to be conservative in this respect. A discussion about discrepancies between theoretical and real FLOPS as well as issues regarding Floating Point (FP) precision operations can be found in the Appendix.

4. Computer vision analysis

In this section, we analyse the evolution of ImageNet [44] (one pass inference) according to performance and compute. Further details in the Appendix.

4.1. Number of parameters and FLOPs

The number of parameters is usually reported, but it is not directly proportional to compute. For instance, in CNNs, convolution operations dominate the computation: if d, w and r represent the network's depth, width and input resolution, the FLOPs grow following the relation [26]:

$$\text{FLOPs} \propto d + w^2 + r^2$$

This means that FLOPs do not directly depend on the number of parameters. The number of parameters affect network depth (d) or width (w). However, distributing the same number of parameters in different ways (e.g., using different layer shapes, filters, etc.) will result in different numbers of FLOPs. Moreover, the resolution (r) does not depend on the number of parameters directly, because the input resolution can be increased without increasing network size.

Despite this, Fig. 1 shows a linear relation between FLOPs and parameters. We attribute this to the balanced scaling of w, d and r. These dimensions are usually scaled together with bigger CNNs using higher resolution. Note that recent transformer models [45] do not follow the growth relation presented above. However, the correlation between the number of parameters and FLOPs for CNNs is 0.772 and the correlation for transformers is 0.994 (Fig. 1). This suggests that usually in both architectures parameters and FLOPs scale in tandem. We will use FLOPs, as they allow us to estimate the needed energy relating hardware FLOPS with required FLOPs for a model [40,46].

[2] https://www.scopus.com
[3] https://scholar.google.com/
[4] https://developer.nvidia.com/deep-learning
[5] We considered Google's TPUs (https://cloud.google.com/tpu?hl=en) for the analysis but there is not enough public information about them, as they are not sold but only available as a service.

[6] https://developer.nvidia.com/embedded/jetson-modules

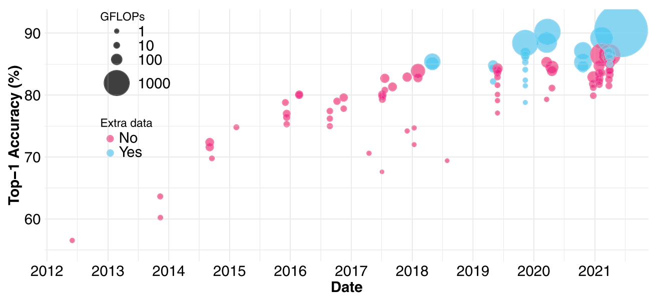

Fig. 2. Accuracy evolution over the years. The size of the balls represent the GFLOPs of one forward pass.

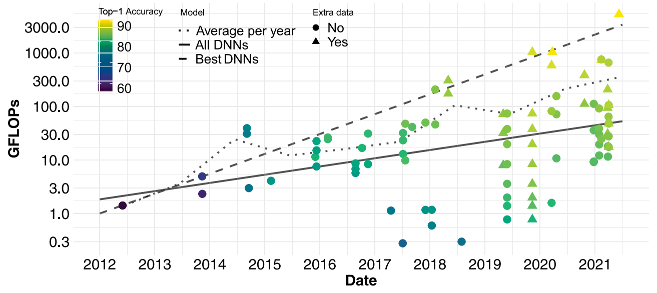

Fig. 3. GFLOPs over the years. The dashed line is a linear fit (note the logarithmic y-axis) for the models with highest accuracy per year. The solid line includes all points.

4.2. Performance and compute

There has been very significant progress for ImageNet. In 2012, AlexNet achieved 56% Top-1 accuracy (single model, one crop). In 2021, Meta Pseudo Labels (EfficientNet-L2) achieved 90.2% Top-1 accuracy (single model, one crop). However, this increase in accuracy comes with an increase in the required FLOPs for a forward pass. A forward pass for AlexNet is 1.42 GFLOPs while for EfficientNet-L2 is 1040 GFLOPs (details in the Appendix).

Fig. 2 shows the evolution from 2012 to 2021 in ImageNet accuracy (with the size of the bubbles representing the FLOPs of one forward pass). In recent papers some researchers began using more data than those available in ImageNet1k for training the models. However, using extra data only affects training FLOPs, but does not affect the computational cost for *inferring* each classification (forward pass).

If we only look at models with the best accuracy for each year we can see an exponential growth in compute (measured in FLOPs). This can be observed clearly in Fig. 3: the dashed line represents an exponential growth (shown as a linear fit since the y-axis is logarithmic). The line is fitted with the models with highest accuracy for each year. However not all models released in the latest years need so much compute. This is reflected by the solid line, which includes all points. We also see that for the same number of FLOPs we have models with increasing accuracy as time goes by.

In Table 1 there is a list of models having similar number of FLOPs as AlexNet. In 2019 we have a model (EfficientNet-B1) with the same number of operations as AlexNet achieving a Top-1 accuracy of 79.1% without using extra data, and a model (NoisyStudent-B1) achieving Top-1 accuracy of 81.5% using extra data. In a period of 7 years, we have models with similar computation with much higher accuracy. We observe that when a SOTA model is released it usually has a huge number of FLOPs, and therefore consumes a large amount of energy, but in a couple of years there is a model with similar accuracy but with much lower number of FLOPs. These models are usually those that become popular in many industry applications. This observation confirms that better results for DNN models of *general use* are in part attributable to algorithmic improvements and not only to the use of more computing power.

Table 1
Results for several DNNs with a similar number of FLOPs as AlexNet.

Model	Top-1 Accuracy	GFLOPs	Year
AlexNet [37]	56.52	1.42	2012
ZFNet [47]	60.21	2.34	2013
GoogleLeNet [48]	69.77	3.00	2014
MobileNet [49]	70.6	1.14	2017
MobileNetV2 1.4 [50]	74.7	1.18	2018
EfficientNet-B1 [26]	79.1	1.40	2019
NoisyStudent-B1 [51]	81.5	1.40	2019

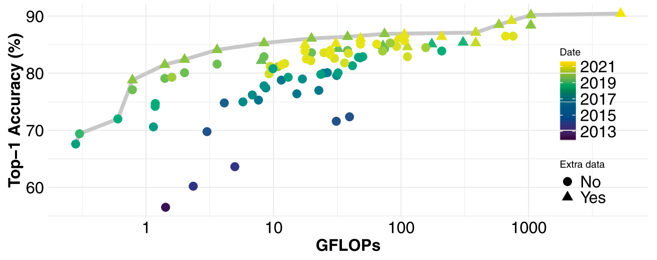

Fig. 4. Relation between accuracy and GFLOPs.

Finally, Fig. 4 shows that the Pareto frontier (in grey) is composed of new models (in yellow and green), whereas old models (in purple and dark blue) are relegated below the Pareto. As expected, the models which use extra data are normally those forming the Pareto frontier. Let us note again that extra training data does not affect inference GFLOPs.

5. Natural language analysis

In this section, we analyse the trends in performance and inference compute for NLP models. To analyse performance we use GLUE, which is a popular benchmark for natural language understanding, one key task in NLP. The GLUE benchmark[7] is composed of nine sentence understanding tasks, which cover a broad range of domains. The description of each task can be found in [20].

5.1. Performance and compute

We represent the improvement on the GLUE score in relation to GFLOPs over the years in Fig. 5 (and in Fig. G.15 in the Appendix). GFLOPs are for single input of length 128, which is a reasonable sequence length for many use cases, being able to fit text messages or short emails. We can observe a very similar evolution to the evolution observed in ImageNet: SOTA models require a large number of FLOPs, but in a short period of time other models appear, which require much fewer FLOPs to reach the same score. There are many models that focus on being efficient instead of reaching high score, and this is reflected in their names too (e.g., MobileBERT [52] and SqueezeBERT [53]). We note that the old models become inefficient (they have lower score with higher number of GLOPs) compared to the new ones, as it happens in CV models.

5.2. Compute trend

In Fig. 6 we include all models (regardless of having performance results) for which we found inference FLOPs estimates. The dashed line adjusts to the models with higher GFLOPs (models that, when released, become the most demanding model) and the solid line to all

[7] Many recent models are evaluated on SUPERGLUE, but we choose GLUE to have a temporal window for our analysis.

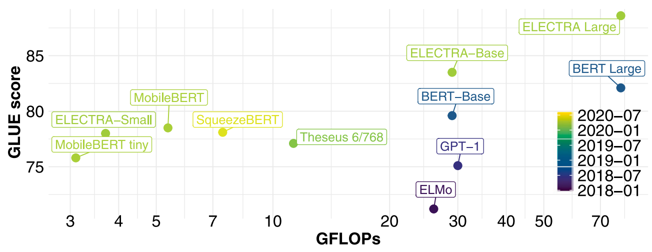

Fig. 5. GFLOPs per token analysis for NLP models.

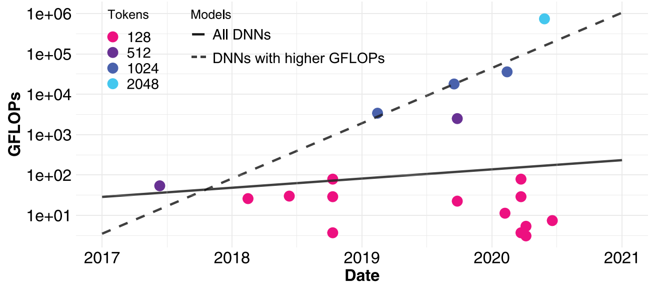

Fig. 6. GFLOPs per token analysis for NLP models.

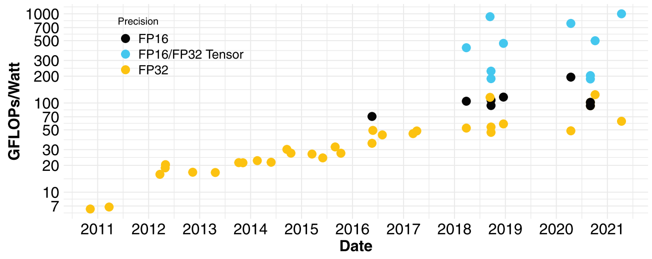

Fig. 7. Theoretical Nvidia GPUs GFLOPS per Watt. Data in Table J.8 in the appendix.

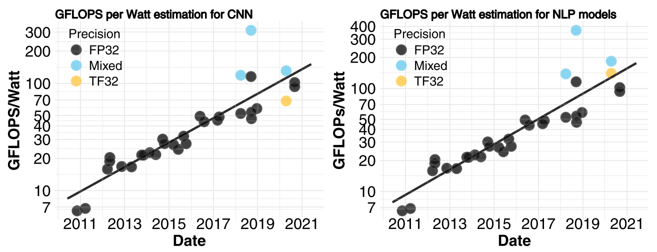

Fig. 8. Nvidia GPU GFLOPS per Watt adapted for CV (CNNs) and NLP models. Data in Table J.9 in the appendix.

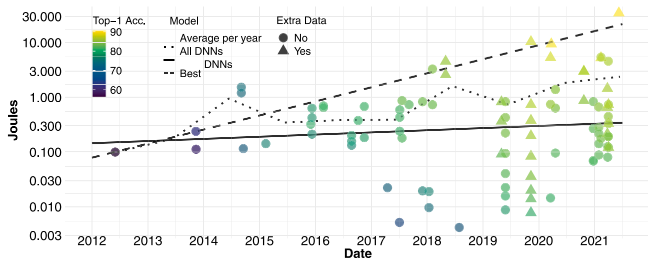

Fig. 9. Estimated Joules of a forward pass (CV). The dashed line is a linear fit (logarithmic y-axis) for the models with highest accuracy per year. The solid line fits all models.

NLP models. In this plot we indicate the input sequence length, because in this plot we represent models with different input sequence lengths. We observe a similar trend as in CV: the GFLOPS of the most cutting-edge models have a clear exponential growth, while the general trend, i.e., considering all models, does not scale so aggressively. Actually, there is a good pocket of low-compute models in the last year.

6. Hardware progress

We use FLOPS as a measure of hardware performance and FLOPS/Watt as a measure of hardware efficiency. We collected performance for different precision formats and tensor cores for a wide range of GPUs. The results are shown in Fig. 7. Note that the y-axis is in logarithmic scale. Theoretical FLOPS for tensor cores are very high in the plot. However, if we follow a more realistic estimation for the Nvidia GPUs (V100, A100 and T4),[8] the actual performance for inference using tensor cores is not that high. The details of this estimation are shown in Table E.3 in the appendix. Note that we have not included ‘edge’ AI low-power accelerators in the analysis since their performance and efficiency cannot be reliably estimated with the used methodology. In the Appendix L, we have included an extended discussion and insights regarding the performance of these accelerators.

With these estimates we have been able to obtain good linear fits (with the y-axis in logarithmic scale) for each data set, one for CV and another for NLP, as shown by the solid lines in Fig. 8. Note that there are some particular points in Fig. 8 that stand out among the others by a large margin. This corresponds to GPUs using mixed precision, i.e., GPUs optimised for DNNs. Particularly, the highest point corresponds to the GPU T4, a GPU that is specifically designed for inference, and this is why it is so efficient for this task.

7. Energy consumption analysis

Once we have estimated the inference FLOPs for a range of models and the GFLOPS per Watt for different GPUs, we can estimate the energy (in Joules) consumed in one inference. We do this by dividing the FLOPs for the model by FLOPS per Watt for the GPU. But how can we choose the FLOPS per Watt that correspond to the model? We use the models presented in Fig. 8 to obtain an estimate of GLOPS per Watt *for the model’s release date*. In this regard, the FLOPs for DNNs can be misleading sometimes (as reported in [24]), usually due to underlying optimisations of the firmware, frameworks, memory and hardware that can influence energy efficiency. They show that energy and FLOPs are highly correlated when analysing the same architecture, but the correlation decreases when different architectures are mixed. We consider that this low correlation does not significantly affect our estimates, as we analyse trends over years and adjust for exponential scaling, where the dispersion is reduced.

To perform a more precise analysis it would be necessary to measure power consumption for each network with the original hardware and software, as unfortunately the required energy per (one) inference is rarely reported. Still, our estimates are comparable to real data extracted from other experiments in the literature (see Appendix K).

We can express the efficiency metric FLOPS per Watt as FLOPs per Joule, as shown in Eq. (1). Having this equivalence we can use it to divide the FLOPs needed for a forward pass and obtain the required Joules, see Eq. (2). Doing this operation we obtain the consumed energy in Joules.

$$\text{Efficiency} = \frac{\text{HW Perf.}}{\text{Power}} \quad \text{in units:} \quad \frac{FLOPS}{\text{Watt}} = \frac{FLOPs/\text{s}}{\text{Joules/s}} = \frac{FLOPs}{\text{Joule}} \tag{1}$$

$$\text{Energy} = \frac{\text{Fwd. Pass}}{\text{Efficiency}} \quad \text{in units:} \quad \frac{FLOPs}{FLOPs/\text{Joule}} = \text{Joule} \tag{2}$$

[8] Specifications in: https://www.nvidia.com/en-us/data-center/

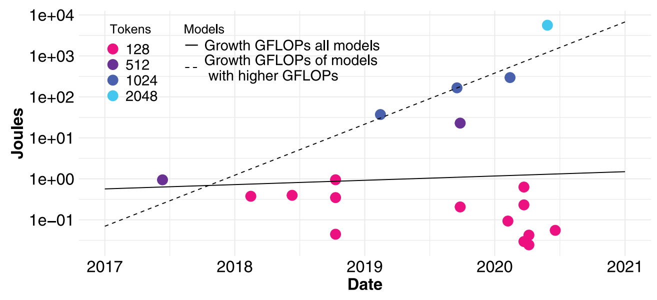

Fig. 10. Estimated Joules of a forward pass (NLP). Same interpretation as in Fig. 9.

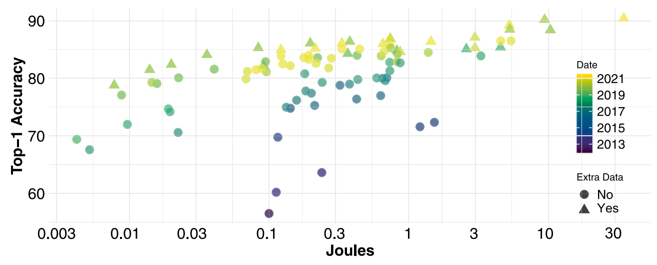

Fig. 11. Relation between Joules and Top-1 Accuracy over the years (CV, ImageNet).

Applying this calculation to all collected models we obtain Fig. 9 for CV. The dashed line represents an exponential trend (a linear fit as the y-axis is logarithmic), adjusted to the models with highest accuracy for each year, like in Fig. 2, and the dotted line represent the average Joules for each year. By comparing both plots we can see that hardware progress softens the growth observed for FLOPs, but the growth is still clearly exponential for the models with high accuracy. The solid line is almost horizontal, but in a logarithmic scale this may be interpreted as having an exponential growth with a small base or a linear fit on the semi log plot that is affected by the extreme points. In Fig. 10 we do the same for NLP models and we see a similar picture.

Fig. 11 shows the relation between Top-1 Accuracy and Joules. Joules are calculated in the same way as in Fig. 9. The relation is similar as the observed in Fig. 4, but in Fig. 11 the older models are not only positioned further down in the y-axis (performance) but they tend to cluster on the bottom right part of the plot (high Joules), so their position on the y-axis is worse than for Fig. 4 due to the evolution in hardware. This is even more clear for NLP, as seen in Fig. 12.

8. Forecasting and multiplicative effect

In our analysis we see that DNNs as well as hardware are improving their efficiency and do not show symptoms of standstill. This is consistent with most studies in the literature: performance will continue growing as compute grows, but at the same time efficiency is increasing. However, this is the first work that analyses whether these two things cancel, especially when we analyse inference and not training. Our conclusion is that they not cancel out for the cutting-edge models of each moment but this is less clear for the regular models in *general use* by industries and individuals.

However, since we are focusing on inference costs, we need to consider the multiplicative factor. How many inferences are performed *per capita*? This has definitely increased very significantly with the use of smart devices, Internet of things and many other devices around us, which are incorporating DNN-based services. However, how many inference passes per capita do we have at this moment, and how is this growing? This is very difficult to estimate, and we leave it for future work. However, it is interesting to analyse possible hypotheses: assume there is one inference pass of a neural network application per second per capita. What would this imply in terms of energy consumption?

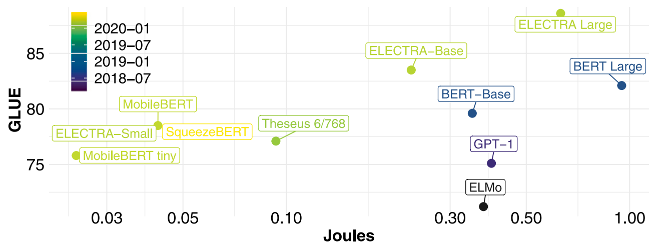

Fig. 12. Relation between Joules and GLUE score over the years (NLP, GLUE).

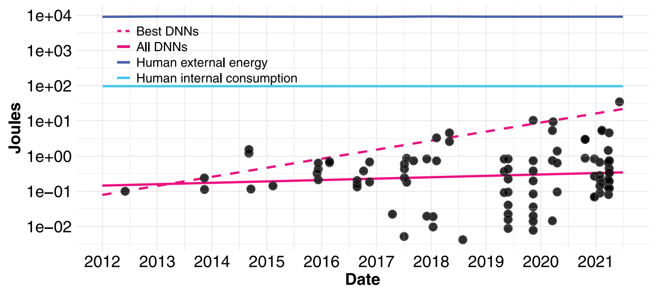

Fig. 13. Estimated Joules per forward pass (e.g., one prediction) compared to human energy consumption in 1s (CV).

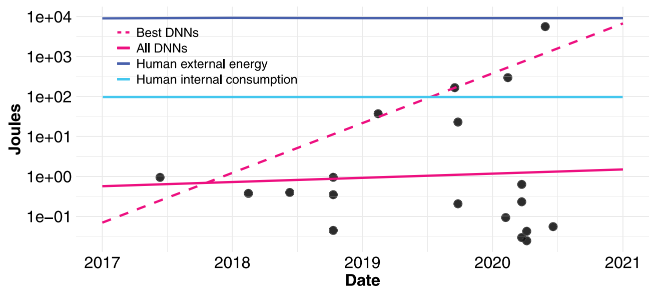

Fig. 14. Estimated Joules per forward pass (e.g., one prediction) compared to human consumption in 1s (NLP).

In order to put this inference energy consumption in context we calculate the value of average human body energy consumption (we will refer to it as somatic or internal consumption) in one second and the average energy that a human being consumes in one second with all their commodities (we will refer to it as external consumption). The internal consumption is calculated assuming 2000 KCal per person day, and converting this to Joules/s, giving approximately 100 Joules/s. The external consumption is the sum of total energy consumption, including electricity, transport and heating, using the USA as a Ref. [54]. This suggests 79,897 Kwh/year in 2019, which is approximately 10,000 Joules every second. The comparison of these two references with the trends can be seen in Fig. 13 (CV). As we see, the energy consumed for one inference of the best models approaches the energy consumed by the human body in one second but stills far from the external energy consumed in one second. If each human did an AI-based decision implying a forward pass every second during the whole day (and night), this would be still well below their internal consumption. However, AI-based decisions are becoming more ubiquitous. For instance, a self-driving car or a surveillance camera may be making many forward passes per second. For NLP, the trends are similar but the best models are growing much faster, as we see in Fig. 14, while the regular models may even decrease. Here, the interpretation in terms of how many decisions are made in a second is also hard to determine. For instance, a language model interfaced by a human does not require more than the basic 128-token windows per second. However, many applications of language models can process data without interacting with humans at a much higher speed.

9. Discussion and future work

In this work we have combined the analysis of several elements about AI, compute and energy consumption that allow us to have a different and more comprehensive perspective about the energy impact of AI. The most distinctive element of our analysis is that we focus on inference cost, which is usually lower than the training cost when both are reported in research papers, but because of multiplicative factors, it is much higher overall. Many DNN models are trained once and applied millions of times (forward passes).

Our findings are very different from the unbridled exponential growth that is usually reported when just looking at the number of parameters of new deep learning models [55–57]. When we focus on inference costs of these networks, the energy that is associated is not growing so fast, because of several factors that partially compensate the growth, such as algorithmic improvements, hardware specialisation and hardware consumption efficiency. The gap gets closer when we analyse those models that settle, i.e., those models whose implementation become very popular one or two years after the breakthrough algorithm was introduced. These *general-use* models can achieve systematic growth in performance at an almost constant energy consumption. The main conclusion is that even if the *energy* used by AI were kept constant, the improvement in performance could be sustained with algorithmic improvements and fast increase in the number of parameters.

This conclusion has an important limitation. It assumes a constant multiplicative factor. As more and more devices use AI (locally or remotely) the energy consumption can escalate just by means of increased penetration, in the same way that cars have become more efficient in the past two decades but there are many more cars in the world today.

We hope this paper contributes to the increasing debate about AI and energy consumption by analysing the inference costs. As these are dominated by multiplicative factors, this should encourage not only AI researchers but economists and social scientists to participate in this analysis. Future studies would be enriched by socio-economic indicators about the use of AI (the degree of penetration), the cost of energy and devices as well as the carbon footprint per Joule [18]. Similarly, comparing energy consumption by AI and trends in human salaries could help determine where automation becomes cost effective in economic terms.

Finally, this paper has many limitations that originate from the limited information reported in scientific papers. Many papers include the number of hyperparameters, but it is less common to have complete information about FLOPs and energy consumption. It is even rarer when looking for inference costs. This information is not only necessary for the transparency of the field but it is of utmost relevance for producing studies such as the one we have presented here, with a larger number of benchmarks and models. Also, it is important that new techniques are reported with new but also old benchmarks, so that we can have larger temporal windows where we can analyse the evolution of the field. We hope that future studies can build on this one and better publishing practices.

CRediT authorship contribution statement

Radosvet Desislavov: Conceptualization, Methodology, Data curation, Visualisation, Investigation, Writing – original draft, Reviewing and editing. **Fernando Martínez-Plumed:** Conceptualization, Methodology, Data curation, Visualisation, Investigation, Writing – original draft, Reviewing and editing. **José Hernández-Orallo:** Conceptualization, Methodology, Data curation, Visualisation, Investigation, Writing – original draft, Reviewing and editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

We thank the reviewers for their insightful remarks, which have helped improve the paper significantly. This work has been partially supported by the MIT-Spain - INDITEX Sustainability Seed Fund under project COST-OMIZE, the grant PID2021-122830OB-C42 funded by MCIN/AEI/10.13039/501100011033 and “ERDF A way of making Europe”, Generalitat Valenciana under INNEST/2021/317 and PROMETEO/2019/098, EU’s Horizon 2020 research and innovation programme under grant agreement No. 952215 (TAILOR), the Future of Life Institute, FLI, under grant RFP2-152, US DARPA HR00112120007 (RECoG-AI).

Appendix A. Flops vs flops

The terminology regarding computing effort and speed is usually confusing. The term ‘compute’ is often ambiguous, and is sometimes applied to a *number of operations* or to the *number of operations per second*. In this sense, we use the acronym FLOPS to measure hardware performance, referring to the number of floating point operations *per second*, as standardised in the industry. For its part, FLOPs is applied to the amount of computation for a given task (e.g., a prediction or inference pass), referring to the number of operations, counting a pair of multiply-add operations as two operations.

For instance, we found out that the acronym FLOP may be misleading. By FLOP, we mean one floating point operation, a measure of the amount of compute (computing effort) and by FLOPS, we mean floating point operations *per second*, i.e., FLOPS = FLOP/s. However, many papers, especially CV papers, use the terms FLOPs and FLOPS to refer to the number of operations, but we will be just use FLOPs as the plural of FLOP, never as FLOPS. Then there is the question of what a FLOP is. When dealing with DNN, this is usually associated with the number of multiply-add operations, even there are other type of operations involved when executing a DNN. This is done this way because it is usually a good estimate [40,46]. More specifically, we will count one fused multiply-add operation as 2 FLOPs (note the lowercase ‘s’). Hardware manufacturers count them in this manner [58], because in fact there are two mathematical operations. However, CV research papers count a multiply-add operation as only one operation. In this case, we will multiply the number of operations reported by 2. In sum, the acronym FLOPS will be applied to measure hardware performance, by referring to the number of floating point operations *per second*, as standardised in the industry, while FLOPs will be applied to the amount of computation for a given task (e.g., a prediction or inference pass), by referring to the number of operations, counting a multiply-add operation pair as two operations.

Appendix B. Methodology details for CV models

Accuracy and FLOPs metrics were collected carefully, taking into account that there are different sampling techniques to reach a given accuracy. For instance, in the AlexNet paper [37], to classify a single image they make 10 predictions, they take 10 different crops[9] from the original image and average the 10 predictions to get the final prediction. While this is a useful trick, it is not fair to compare the accuracy of a model achieved with 10 crops with another achieved

[9] Cropping is a common image manipulation process: while cropping the middle square (down-sampling) from input images is a good practice for data preparation, random cropping is also a good practice for train-data augmentation

Table B.2
CV models data set. A citation next to a given value means that this value is extracted from that source, otherwise the values are from the paper (cited in model column). The symbol † means that this value was obtained or checked from a model implementation using model analysis tools, and the symbol ∗ means that we estimated the value.

Model	Top-1 Acc.	Params (M)	GFLOPs	Extra data	Date	Architecture
AlexNet [37]	56.52 [29]	61.00 †	1.42 †	No	01/06/2012	CNN
ZFNet-b [47]	63.63 [31]	107.63 [31]	4.96 [31]	No	11/11/2013	CNN
ZFNet [47]	60.21 [31]	62.36 [31]	2.34 [31]	No	12/11/2013	CNN
VGG-19 [38]	72.37 [29]	144.00	39.34 †	No	04/09/2014	CNN
VGG-16 [38]	71.59 [29]	138.00	31.00 †	No	04/09/2014	CNN
Inception V1/GoogleLeNet [48]	69.77 [29]	6.80	3.00	No	17/09/2014	CNN
Inception V2/Incepton BN [59]	74.80	11.29 [31]	4.10 [31]	No	11/02/2015	CNN
Inception V3 [60]	78.80	23.83	11.48	No	02/12/2015	CNN
ResNet-50 [61]	75.30 [62]	26.00 [63]	7.60	No	10/12/2015	CNN
ResNet-101 [61]	76.40 [62]	45.00 [63]	15.20	No	10/12/2015	CNN
ResNet-152 [61]	77.00 [62]	60.00 [63]	22.60	No	10/12/2015	CNN
Inception V4 [64]	80.00	42.68 [31]	24.60 [31]	No	23/02/2016	CNN
Inception ResNet V2 [64]	80.10	55.84 [31]	26.38 [31]	No	23/02/2016	CNN
Densenet-121 [65]	74.98	7.98 [31]	5.74 [31]	No	25/08/2016	CNN
Densenet-169 [65]	76.20	14.15 [31]	6.80 [31]	No	25/08/2016	CNN
Densenet-201 [65]	77.42	20.01 [31]	8.68 [31]	No	25/08/2016	CNN
Xception [66]	79.00	22.86	16.80 [31]	No	07/10/2016	CNN
ResNeXt-50 (32 × 4d) [67]	77.80	25.00	8.40	No	16/11/2016	CNN
ResNeXt-101 (64 × 4d) [67]	79.60	83.46	31.20 †	No	16/11/2016	CNN
MobileNet [49]	70.60	4.20	1.14	No	17/04/2017	CNN
ShuffleNet x1.0 (g=8) [68]	67.60	2.43 [31]	0.28	No	04/07/2017	CNN
DPN-131 (40 ×4d) [69]	80.07	79.50	32.00	No	06/07/2017	CNN
DPN-98 (40 ×4d) [69]	79.80	61.70	23.40	No	06/07/2017	CNN
DPN-92 (32 ×3d) [69]	79.30	37.80	13.00	No	06/07/2017	CNN
NASNet-A (6 @ 4032) [70]	82.70	88.90	47.60	No	21/07/2017	CNN
NASNet-A (7 @ 1920) [70]	80.80	22.60	9.86	No	21/07/2017	CNN
SENet-154 [71]	81.32	115.09 [31]	41.50 [31]	No	05/09/2017	CNN
PNASNet-5 (N = 4, F = 216) [72]	82.90	86.10	50.00	No	02/12/2017	CNN
PNASNet-5 (N = 3, F = 54) [71]	74.20	5.10	1.18	No	02/12/2017	CNN
MobileNetV2 [50]	72.00	3.40	0.60	No	13/01/2018	CNN
MobileNetV2 1.4 [50]	74.70	6.90	1.18	No	13/01/2018	CNN
AmoebaNet-A (N=6, F=190) [73]	82.80	86.70	46.20	No	05/02/2018	CNN
AmoebaNet-A (N=6, F=448) [73]	83.90	469.00	208.00	No	05/02/2018	CNN
ResNeXt-101 32×32d [74]	85.10	466.00	174.00	Instagram 940M	02/05/2018	CNN
ResNeXt-101 32×48d [74]	85.40	829.00	306.00	Instagram 940M	02/05/2018	CNN
ShuffleNetV2 x1.0 [75]	69.40	2.28 [31]	0.30	No	30/07/2018	CNN
ResNeXt-101 32 × 16d [76,77]	84.80	193.00	72.00	Custom 940M	02/05/2019	CNN
ResNeXt-101 32 × 8d [76,77]	84.30	88.00	32.00	Custom 940M	02/05/2019	CNN
ResNeXt-50 32 × 4d [76,77]	82.20	25.00	8.00	Custom 940M	02/05/2019	CNN
EfficientNet-B0 [26]	77.10	5.30	0.78	No	28/05/2019	CNN
EfficientNet-B1 [26]	79.10	7.80	1.40	No	28/05/2019	CNN
EfficientNet-B2 [26]	80.10	9.20	2.00	No	28/05/2019	CNN
EfficientNet-B3 [26]	81.60	12.00	3.60	No	28/05/2019	CNN
EfficientNet-B4 [26]	82.90	19.00	8.40	No	28/05/2019	CNN
EfficientNet-B5 [26]	83.60	30.00	19.80	No	28/05/2019	CNN
EfficientNet-B6 [26]	84.00	43.00	38.00	No	28/05/2019	CNN
EfficientNet-B7 [26]	84.30	66.00	74.00	No	28/05/2019	CNN
NoisyStudent-B0 [51]	78.80	5.30	0.78	JFT 300M	11/11/2019	CNN
NoisyStudent-B1 [51]	81.50	7.80	1.40	JFT 300M	11/11/2019	CNN
NoisyStudent-B2 [51]	82.40	9.20	2.00	JFT 300M	11/11/2019	CNN
NoisyStudent-B3 [51]	84.10	12.00	3.60	JFT 300M	11/11/2019	CNN
NoisyStudent-B4 [51]	85.30	19.00	8.40	JFT 300M	11/11/2019	CNN
NoisyStudent-B5 [51]	86.10	30.00	19.80	JFT 300M	11/11/2019	CNN
NoisyStudent-B6 [51]	86.40	43.00	38.00	JFT 300M	11/11/2019	CNN
NoisyStudent-B7 [51]	86.90	66.00	74.00	JFT 300M	11/11/2019	CNN
NoisyStudent-L2 [51]	88.40	480.00	1040.00 ∗	JFT 300M	11/11/2019	CNN
FixEfficientNet-L2 [78]	88.50	480.00	585.00 ∗	JFT 300M	18/03/2020	CNN
FixEfficientNet-B7 [78]	85.30	66.00	82.00 ∗	No	18/03/2020	CNN
FixEfficientNet-B0 [78]	79.30	5.30	1.60 ∗	No	18/03/2020	CNN
Meta Pseudo Labels L2 [79]	90.20	480.00	1040.00 ∗	JFT 300M	23/03/2020	CNN
ResNeSt-269 [80]	84.50	111.00	155.8 †	No	19/04/2020	CNN
ResNeSt-200 [80]	83.90	70.00	71.56 †	No	19/04/2020	CNN
ResNeSt-50 [80]	81.13	27.50	10.78	No	19/04/2020	CNN
ViT-L/16 [81]	85.30	304.00 [27]	384.00 [27]	ImageNet 21k	22/10/2020	Transformer
ViT-L/16 [81]	87.12	304.00 [27]	384.00 [27]	JFT 300M	22/10/2020	Transformer
ViT-B/16 [81]	84.60 [27]	87.00 [27]	112.00 [27]	ImageNet 21k	22/10/2020	Transformer
DeiT-small [82,83]	79.90	22.00	9.20 [84]	No	23/12/2020	Transformer
DeiT-small-Distilled [82,83]	81.20	22.00	9.40 [84]	No	23/12/2020	Transformer
DeiT-base [82,83]	81.80	86.00	36.00 [27]	No	23/12/2020	Transformer
DeiT-base-384 [82,83]	82.90	86.00	112.00 [27]	No	23/12/2020	Transformer
BotNet-T7 [85]	84.70	75.00	92.00	No	27/01/2021	Hybrid

(*continued on next page*)

Table B.2 (*continued*).

Model	Top-1 Acc.	Params (M)	GFLOPs	Extra data	Date	Architecture
BotNet-T5 [85]	83.50	75.10	38.60	No	27/01/2021	Hybrid
T2T-ViTt-14 [84]	81.70	21.50	12.20	No	28/01/2021	Transformer
T2T-ViTt-19 [84]	82.20	39.20	19.60	No	28/01/2021	Transformer
T2T-ViTt-24 [84]	82.60	64.10	30.00	No	28/01/2021	Transformer
NFNet-F4+ [86]	89.20	527.00	734.00	JFT 300M	11/02/2021	CNN
NFNet-F0 [86]	83.60	71.50	24.76	No	11/02/2021	CNN
NFNet-F6+SAM [86]	86.50	438.40	754.56	No	11/02/2021	CNN
Swin-B 224 [87]	85.20	88.00	30.80	ImageNet 21k	25/03/2021	Transformer
Swin-B 384 [87]	86.00	88.00	94.00	ImageNet 21k	25/03/2021	Transformer
Swin-L [87]	86.40	197.00	207.80	ImageNet 21k	25/03/2021	Transformer
CrossViT-15 [88]	81.50	27.40	11.60	No	27/03/2021	Transformer
CrossViT-18 [88]	82.50	43.30	18.06	No	27/03/2021	Transformer
CaiT-S36 [89]	83.30	68.00	27.80	No	31/03/2021	Transformer
CaiT-S36 dist [89]	84.00	68.00	27.80	No	31/03/2021	Transformer
CaiT-S24-384 dist [89]	85.10	46.90	64.40	No	31/03/2021	Transformer
CaiT-M48-448 dist [89]	86.50	356.00	659.20	No	31/03/2021	Transformer
EfficientNetV2-S [27]	83.90	24.00	17.60	No	01/04/2021	CNN
EfficientNetV2-M [27]	85.10	55.00	48.00	No	01/04/2021	CNN
EfficientNetV2-L [27]	85.70	121.00	106.00	No	01/04/2021	CNN
EfficientNetV2-S [27]	85.00	24.00	17.60	ImageNet 21k	01/04/2021	CNN
EfficientNetV2-M [27]	86.10	55.00	48.00	ImageNet 21k	01/04/2021	CNN
EfficientNetV2-L [27]	86.80	121.00	106.00	ImageNet 21k	01/04/2021	CNN
ViT-G/14 [39]	90.45	1843.00	5270.00 *	JFT 3B	08/06/2021	Transformer

with 1 crop. Furthermore, the use of several crops or other kinds of repetitions is problematic, as the papers usually report the number of FLOPs for one forward pass[10] (if 10 forward passes are needed to make a single prediction, then the FLOPs should be multiplied by 10). For these reasons we only report 1-crop accuracy for all models, to make a meaningful comparison.

Note that the FLOPs also depend of the input image resolution: the higher the image resolution, the more operations (FLOPs) are required to process it. Some researchers report results with different image resolutions [38,39], and sometimes it is not clear which resolution the results are reported for. In these cases, we need to investigate until we find that information. In sum, all the collected FLOPs in this work are for a forward pass with the resolution used for inference. The selected models and their values are shown in Table B.2.

Appendix C. Methodology details for NLP models

As previously stated, for NLP models we just included all the models since 2017 for which we find inference compute estimates. Many papers do not explain how they count FLOPs (as single mathematical operations or single hardware instructions), but we ultimately found out this information explained in [40]. We compare the presented numbers with estimates in other publications (we compare the numbers for repeated and similar models) and we see that these numbers are very similar. We assume that the other authors follow this as the standard procedure to count FLOPs. In NLP, they count FLOPs as single mathematical operations and not as a single hardware instructions (like in CV). The important thing is that we use the same approach in all the NLP models, as the comparison and analysis will be intra-domain and never inter-domain.

Appendix D. Datasets

D.1. ImageNet

ImageNet is the most used dataset in the last decade for training and evaluating CV models. The full dataset consists of 14,197,122 images distributed in 21,841 classes. Researchers refer to this dataset as ImageNet21k or ImageNet22k. However, researchers commonly use a subset of the full ImageNet dataset. This subset consists of 1.2 million images for training and 50,000 images for validation distributed in 1,000 classes. This subset was released for ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) and is usually referred as ImageNet1k or just as ImageNet. In 2012 the AlexNet model [37] won the ILSVRC 2012 Image Classification with an impressive result, outperforming the other models by large margin. AlexNet was the first DNN to win this competition. Since then many other DNNs have been created for image classification.

D.2. GLUE

The General Language Understanding Evaluation (GLUE) benchmark [20] is a collection of resources for evaluating and analysing the performance of models across a diverse range of existing NLP tasks with the goal of driving “research in the development of general and robust natural language understanding systems”. The collection in GLUE consists of nine “difficult and diverse” tasks, mostly adopted from existing datasets. The tasks involve sentiment analysis, acceptability, paraphrasing, natural language inference and coreference resolution. GLUE is model-agnostic, but it incentivises sharing knowledge across tasks (using parameter sharing or other transfer learning techniques) due to the limited training data for certain tasks.

Appendix E. Hardware data compilation: floating point precision details

At the end of 2017 Nvidia launched GPUs with new features for AI acceleration (improved lower precision performance and tensor cores, which can improve low-precision calculations) [90]. For instance, many new GPUs have accelerated FP16 operations through tensor cores (DNN can operate at low precision in many calculations without problems) and combine them with FP32 precision operations when is necessary. In this way we benefit from higher performance, maintaining calculation’s precision. Nvidia specifies different FLOPS for FP16 and for tensor cores. Nowadays, frameworks as PyTorch and TensorFlow allow to train and infer with a DNN with mixed precision, i.e., taking advantage of the tensor cores, easily without practically any significant reduction in accuracy. Because of all this, we consider necessary to include the performance achieved with tensor cores in our analysis.

Theoretical FLOPS using tensor cores are very high, but this increase in FLOPS does not correspond with the gain seen in practice for deep

[10] A “forward pass” refers to calculation process, values of the output layers from the inputs data. It is traversing through all neurons from first to last layer. A loss function is calculated from the output values.

Table E.3
Throughput measures for V100, A100 and T4 GPUs on different Models. The 'speed-up' column is the speed-up achieved with respect to FP32 throughput using different precision formats. A100 speed-up is calculated with respect to V100 FP32 throughput.
Source: The data is obtained from NVIDIA NGC catalog (https://ngc.nvidia.com/catalog/resources).

Task	Model	Framework	Batch size	GPU	Precision	Throughput	Speed-up
CV	efficientnet-b0	PyTorch	256	V100 16GB	FP32	2968	1.00
	efficientnet-b0	PyTorch	256	V100 16GB	Mixed	6176	2.08
	efficientnet-b0	PyTorch	256	A100 80GB	TF32	5154	1.74
	efficientnet-b0	PyTorch	256	A100 80GB	Mixed	10239	3.45
	efficientnet-b4	PyTorch	128	V100 16GB	FP32	376	1.00
	efficientnet-b4	PyTorch	128	V100 16GB	Mixed	843	2.24
	efficientnet-b4	PyTorch	128	A100 80GB	TF32	700	1.86
	efficientnet-b4	PyTorch	128	A100 80GB	Mixed	1418	3.77
	ResNeXt101-32 × 4d	PyTorch	256	V100 16GB	FP32	533	1.00
	ResNeXt101-32 × 4d	PyTorch	256	V100 16GB	Mixed	1746	3.28
	ResNeXt101-32 × 4d	PyTorch	256	T4 16GB	FP32	161	1.00
	ResNeXt101-32 × 4d	PyTorch	256	T4 16GB	Mixed	598	3.71
	ResNet v1.5	PyTorch	256	V100 16GB	FP32	1261	1.00
	ResNet v1.5	PyTorch	256	V100 16GB	Mixed	3382	2.68
	ResNet v1.5	PyTorch	256	T4 16GB	FP32	415	1.00
	ResNet v1.5	PyTorch	256	T4 16GB	Mixed	1198	2.89
	ResNet v1.5	TensorFlow	256	V100 16GB	FP32	1348.52	1.00
	ResNet v1.5	TensorFlow	256	V100 16GB	Mixed	2742.14	2.03
	ResNet v1.5	TensorFlow	256	A100 40GB	TF32	1911.96	1.42
	ResNet v1.5	TensorFlow	256	A100 40GB	Mixed	3229.32	2.39
	ResNet v1.5	TensorFlow	256	T4 16GB	FP32	425.72	1.00
	ResNet v1.5	TensorFlow	256	T4 16GB	Mixed	993.39	2.33
	SSD v1.1	PyTorch	32	V100 16GB	FP32	271.73	1.00
	SSD v1.1	PyTorch	32	V100 16GB	Mixed	438.85	1.62
	SSD v1.1	PyTorch	32	A100 40GB	TF32	548.75	2.02
	SSD v1.1	PyTorch	32	A100 40GB	Mixed	910.17	3.35
	UNet Industrial	TensorFlow	16	V100 16GB	FP32	250.23	1.00
	UNet Industrial	TensorFlow	16	V100 16GB	Mixed	469.27	1.88
	UNet Industrial	TensorFlow	16	A100 40GB	TF32	424.57	1.70
	UNet Industrial	TensorFlow	16	A100 40GB	Mixed	823.46	3.29
	SE-ResNeXt101-32 × 4d	TensorFlow	128	V100 16GB	FP32	460.82	1.00
	SE-ResNeXt101-32 × 4d	TensorFlow	128	V100 16GB	Mixed	1102	2.39
	SE-ResNeXt101-32 × 4d	TensorFlow	128	A100 40GB	TF32	802.64	1.74
	SE-ResNeXt101-32 × 4d	TensorFlow	128	A100 40GB	Mixed	1728.27	3.75
	SE-ResNeXt101-32 × 4d	TensorFlow	128	T4 16GB	FP32	105.16	1.00
	SE-ResNeXt101-32 × 4d	TensorFlow	128	T4 16GB	Mixed	195.17	1.86
NLP	BERT-LARGE	TensorFlow	8	V100 16GB	FP32	44.03	1.00
	BERT-LARGE	TensorFlow	8	V100 16GB	Mixed	168.34	3.82
	BERT-LARGE	TensorFlow	8	A100 80GB	TF32	241.68	5.49
	BERT-LARGE	TensorFlow	8	A100 80GB	Mixed	342.22	7.77
	BERT-LARGE	TensorFlow	8	T4 16GB	FP32	16.04	1.00
	BERT-LARGE	TensorFlow	8	T4 16GB	Mixed	62.99	3.93
	BERT-Base	TensorFlow	8	V100 16GB	FP32	146.15	1.00
	BERT-Base	TensorFlow	8	V100 16GB	Mixed	504.24	3.45
	BERT-Base	TensorFlow	8	A100 80GB	TF32	645.88	4.42
	BERT-Base	TensorFlow	8	A100 80GB	Mixed	846.81	5.79
	BERT-Base	TensorFlow	8	T4 16GB	FP32	51.33	1.00
	BERT-Base	TensorFlow	8	T4 16GB	Mixed	192.61	3.75
	Transformer-XL	TensorFlow	32	V100 16GB	FP32	8555.6	1.00
	Transformer-XL	TensorFlow	32	V100 16GB	Mixed	11215.5	1.31
	Transformer-XL	TensorFlow	32	A100 40GB	TF32	19434.5	2.27
	Transformer-XL	TensorFlow	32	A100 40GB	Mixed	21854.7	2.55
	Transformer-XL	TensorFlow	32	T4 16GB	FP32	3439.1	1.00
	Transformer-XL	TensorFlow	32	T4 16GB	Mixed	6174.3	1.80
	Transformer	PyTorch	10240	V100 16GB	FP32	3782	1.00
	Transformer	PyTorch	10240	V100 16GB	Mixed	7464	1.97
	Transformer	PyTorch	10240	A100 40GB	TF32	7755	2.05
	Transformer	PyTorch	10240	A100 40GB	Mixed	9653	2.55

learning applications (maybe gaming is different). This is because it is not possible to use tensor cores for all operations. To solve the discrepancy between tensor core FLOPS and the real utilisation of these FLOPS, we calculate the speed up achieved for DNN when inference is done with mixed precision. We have looked for experimental results to adjust the tensor FP16/FP32 FLOPS to real performance improvement, the inference experimental results that we use are available in Nvidia NGC Catalog.[11] The collected data can be found in Table E.3.

[11] https://ngc.nvidia.com/catalog/resources

We do not include estimated mixed precision performance for all GPUs that support it because we have not found sufficient benchmarks for all GPUs to carry out an estimation. Also, we do not consider INT8 precision format because in many cases using this format leads to performance downgrade, and therefore the accuracy metric of the models should be adapted for a fair analysis. We perform a different estimation for CV and for NLP networks because these two kinds of networks operate in different ways and take different advantage of mixed precision. During training the speed-up from mixed precision in comparison to FP32 is usually of 2x for image models, and up to 4x for language models [91]. This is corroborated in information about some benchmarks on Nvidia blogs too [92].

Table F.4
Mixed precision speed ups from experimental results for inference.

GPU	Precision speed up	CV models	NLP models
V100	Mixed speed up ratio to V100 FP32	2.27	2.64
A100	TF32 speed up ratio to V100 FP32	1.75	3.56
	Mixed speed up ratio to V100 FP32	3.33	4.67
T4	Mixed speed up ratio to T4 FP32	2.7	3.16

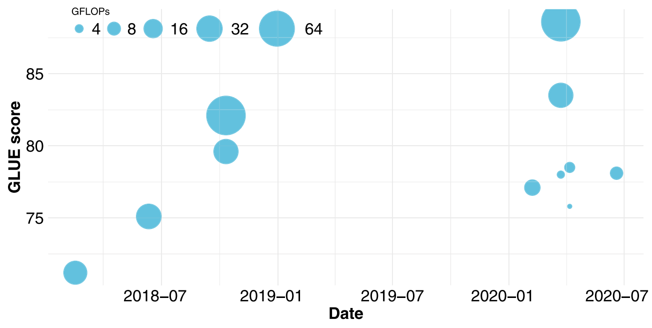

Fig. G.15. GFLOPs per token analysis for NLP models.

Appendix F. Hardware mixed precision speed-ups

As we have discussed, theoretical FLOPS for tensor cores are very high, as we can see in Fig. 7 in the main text. However, the performance for inference using tensor cores is not so high. For this reason we propose an estimate for the Nvidia GPUS: V100, A100 and T4 for CV models and for NLP models. For these calculations we collected inference data from NVIDIA NGC. The estimates for A100 are in relation to V100 because there is no data about FP32 for A100 (because FP32 is substituted by TF32,[12] which is a precision format in between of FP32 and FP16), so we estimated the speed-up to V100 FP32 FLOPS (see Table F.4).

Appendix G. Performance and compute (NLP)

We represent the improvement on the GLUE score over the years as well as models inference GFLOPs (bubbles size) in Fig. G.15. GFLOPs are for single input of length 128, which is a reasonable sequence length for many use cases, being able to fit text messages or short emails. We can observe a very similar evolution to the evolution observed in ImageNet: SOTA models require a large number of FLOPs, but in a short period of time other models appear, which require much fewer FLOPs to reach the same score.

Appendix H. FLOPS estimation for CV models

H.1. EfficientNet-based models FLOPs estimation

There are many EfficientNet variations, mostly using different input resolution or scaling. For these modifications, FLOPs are not always reported. In this work, we estimate them following the relation presented in Eq. (H.1)

$$\text{FLOPs} \propto d + w^2 + r^2 \tag{H.1}$$

for the following models:

- **NoisyStudent-L2**: Having the scale factors of the networks (Table H.5) we estimate NoisyStudent-L2 FLOPs as shown in Eq. (H.2)

[12] https://blogs.nvidia.com/blog/2020/05/14/tensorfloat-32-precision-format/

Table H.5
EfficientNet models architecture specifications obtained from [51].

Model	w	d	Test Resolution
EfficientNet-B7	2	3.1	600 × 600
EfficientNet-L2	4.3	5.3	800 × 800

Table H.6
ViT-G/14 GFLOPs from.

Model	GFLOPS	
	224 × 224	384 × 384
ViT-G/14	965.3	2859.9

$$\begin{aligned}&\text{NoisyStudent-L2 FLOPs} = \\ &= \text{EfficientNet-B7 FLOPs} \cdot d_\sigma \cdot w_\sigma^2 \cdot r_\sigma^2\end{aligned} \tag{H.2}$$

where d_σ, w_σ and r_σ are the scaled factors for, respectively, the network depth, width and input resolutions. By using the values from Table H.5, $d_\sigma = 5.3/3.1 = 1.7097$, $w_\sigma = 4.3/2 = 2.15$ and $r_\sigma = 800/600 = 1.3334$. Knowing that the GFLOPS for EfficientNet-B7 are 74, substituting in (H.2), we obtain the estimate of 74 GFLOPs $\cdot\, 1.7097 \cdot 2.15^2 \cdot 1.3334^2 \approx 1040$ GFLOPS for NoisyStudent-L2.
- **Meta Pseudo Labels L2**: We use the estimate of NoisyStudent-L2 FLOPs for Meta Pseudo Labels L2, because it is the same model and only changes the training strategy.
- **FixEfficientNet-L2**: In FixEfficientNet-L2 they use a resolution of 600 × 600 for testing, so the estimation is the same as for NoisyStudent-L2 but without taking into account the resolution scaling (r_σ). Then, the estimated GFLOPS are 74 GFLOPs $\cdot\, 1.7097 \cdot 2.15^2 \approx 585$ GFLOPS.
- **FixEfficientNet-B7**: This model is the same as EfficientNet-B7 but using a slightly different resolution (632 × 632). Therefore, $r_\sigma = 632/600 = 1.0534$ and, thus we estimate 74 GFLOPs $\cdot\, 1.0534^2 \approx 82$ GFLOPs.
- **FixEfficientNet-B0**: This model is the same as EfficientNet-B0 but using a higher resolution (320 × 320). Therefore, $r_\sigma = 320/224 = 1.4286$ and, thus we estimate 0.78 GFLOPs $\cdot\, 1.4286^2 \approx 1.6$ GFLOPs.

H.2. Vit-g/14 FLOPs estimation

In the paper [39] introducing the model, although authors provide the GFLOPs for 224 × 224 and 384 × 384 resolutions (see Table H.6), they also use 518 × 518 resolution for ViT-G finetuning, so we assume they use the same resolution for testing too. ViT-G/14 is a vision transformer model, so the scale relation presented in (H.1) do not apply for this kind of models. However, knowing the GFLOPs for 224 × 224 and 384 × 384, we may calculate how GFLOPs scale with resolution (given that $r_\sigma^2 = (384/224)^2 = 2.9388$). In this regard, we calculate the GFLOPs ratio as $2859.9/965.3 = 2.9627$ and we observe that GFLOPs scale quadratically with respect to resolution. Note, in this paper they report "real" FLOPs and not multiply-add operations. Therefore, we recalculate $r_\sigma = 518/384 = 1.3490$ and multiply the GFLOPs for 384 × 384 resolution by this scale factor estimating 2859.9 GFLOPs $\cdot\, 1.3490^2 \approx 5270$ GFLOPs for the ViT-G/14 model.

Appendix I. NLP data

Many times researchers report GLUE score without the punctuation on the WNLI task, because this task is problematic. We have marked which scores are reported without this task. Since there are 9 tasks in total, we consider that excluding one of them is not problematic for our analysis.

We did not find inference GFLOPs for the model Bert-Large, but we have ELECTRA-Large GFLOPs and this is actually the same model

Table I.7
NLP models data set. If there is a citation next to the GFLOPs value means that GFLOPs and Input Tokens values are extracted from that source, otherwise the values are from the paper (cited in the 'Model' column). The symbol ♠ means that GLUE score was calculated without punctuation on the WNLI task; the symbol ∗ means that we estimated the value and ♣ means that GLUE score is for GLUE dev set instead of test set.

Model	Input tokens	GFLOPs	Params (M)	Date	GLUE test set
Transformer [45]	512	54 [7]	65	12/06/2017	–
ELMo [93]	128	26 [40]	96	15/02/2018	71.2 [40] ♣
GPT-1 [94]	128	30 [40]	117	11/06/2018	75.1 [95] ♠
BERT Large [95]	128	79	335 ∗	11/10/2018	82.1 ♠
BERT-Small [95]	128	3.7 [40]	14	11/10/2018	–
BERT-Base [95]	128	29 [40]	110	11/10/2018	79.6 ♠
GPT-2 [96]	1024	3400 [7]	1500	14/02/2019	–
Megatron [97]	1024	18000 [7]	8300	17/09/2019	–
ALBERT-xxl [98]	512	2500 [7]	235	26/09/2019	–
ALBERT-base [98]	128	22.5 [53]	12	26/09/2019	–
Theseus 6/768 [99]	128	11.3 [53]	66	07/02/2020	77.1 [53]
Microsoft T-NLG [100]	1024	36000 [7]	17000	13/02/2020	–
ELECTRA Large [40]	128	79 [7]	335	23/03/2020	88.6 ♠
ELECTRA-Small [40]	128	3.7	14	23/03/2020	78 ♠
ELECTRA-Base [40]	128	29	110	23/03/2020	83.5 ♠
MobileBERT [52]	128	5.36	25.3	06/04/2020	78.5 ♠
MobileBERT tiny [52]	128	3.1	15.1	06/04/2020	75.8 ♠
GPT-3 [101]	2048	740000 [7]	175000	28/05/2020	–
SqueezeBERT [53]	128	7.42	51.1	19/06/2020	78.1

but following a different training strategy. In this sense, we use take ELECTRA-Large GFLOPs as BERT-Large GFLOPs. For ELMo we take GLUE "dev-set" score because we do not found the score on the test set (we assume this score should be close to the test set). Values shown in Table I.7.

Appendix J. GPU consumption data

Tables J.8 and J.9 show further technical details regarding, respectively, the GPU's theoretical characteristics (compiled from the manufacturer's specification sheet and reference manuals), and their throughput and power consumption "adapted", if necessary, to the specifics of CV or NLP tasks.

Appendix K. Energy consumption estimates

For an accurate analysis of energy consumption in inference, it would be necessary to measure the energy consumption of each network with the original hardware and software (e.g., using energy meters). However, this is not usually the case and, unfortunately, the energy required by (an) inference is rarely reported. In order to validate our model, We have scrutinised the literature extensively to try to find

Table J.8
Nvidia GPUs theoretical data recopilation.

GPU	Precision	TFLOPS	Watts	Launch date	Type	GFLOPS/Watt
GeForce GTX 580	FP32	1.58	244	09/11/2010	Desktop	6.48
GeForce GTX 590	FP32	2.49	365	24/03/2011	Desktop	6.82
GeForce GTX 680	FP32	3.09	195	22/03/2012	Desktop	15.85
GeForce GTX 690	FP32	5.62	300	29/04/2012	Desktop	18.73
GeForce GTX 780	FP32	4.16	250	23/04/2013	Desktop	16.62
GeForce GTX 780 TI	FP32	5.35	250	07/11/2013	Desktop	21.38
GeForce GTX Titan Black	FP32	5.65	250	18/02/2014	Desktop	22.58
GeForce GTX Titan Z	FP32	8.12	375	28/05/2014	Desktop	21.66
GeForce GTX 980	FP32	4.98	165	18/09/2014	Desktop	30.19
GeForce GTX 980 Ti	FP32	6.06	250	02/06/2015	Desktop	24.24
GeForce GTX TITAN X	FP32	6.69	250	17/03/2015	Desktop	26.76
GeForce GTX 1080	FP32	8.87	180	26/05/2016	Desktop	49.29
GeForce GTX 1080 Ti	FP32	11.34	250	10/03/2017	Desktop	45.36
TITAN X Pascal	FP32	10.97	250	02/08/2016	Desktop	43.88
TITAN XP	FP32	12.15	250	06/04/2017	Desktop	48.60
GeForce RTX 2080	FP32	10.07	215	20/09/2018	Desktop	46.84
GeForce RTX 2080 Ti	FP32	13.45	250	20/09/2018	Desktop	53.80
Nvidia Titan RTX	FP32	16.31	280	18/12/2018	Desktop	58.26
GeForce RTX 3080	FP32	29.80	320	01/09/2020	Desktop	93.13
GeForce RTX 3090	FP32	35.60	350	01/09/2020	Desktop	101.71
GeForce RTX 2080	FP16	20.14	215	20/09/2018	Desktop	93.67
GeForce RTX 2080 Ti	FP16	26.90	250	20/09/2018	Desktop	107.60
Nvidia Titan RTX	FP16	32.62	280	18/12/2018	Desktop	116.50
GeForce RTX 3080	FP16	29.80	320	01/09/2020	Desktop	93.13
GeForce RTX 3090	FP16	35.60	350	01/09/2020	Desktop	101.71
GeForce RTX 2080	FP16/FP32 Tensor	40.30	215	20/09/2018	Desktop	187.44
GeForce RTX 2080 Ti	FP16/FP32 Tensor	56.90	250	20/09/2018	Desktop	227.60
Nvidia Titan RTX	FP16/FP32 Tensor	130.50	280	18/12/2018	Desktop	466.07
GeForce RTX 3080	FP16/FP32 Tensor	59.50	320	01/09/2020	Desktop	185.94
GeForce RTX 3090	FP16/FP32 Tensor	71.00	350	01/09/2020	Desktop	202.86
Tesla K10	FP32	4.58	225	01/05/2012	Server	20.36
Tesla K20x	FP32	3.94	235	12/11/2012	Server	16.74
Tesla K40	FP32	5.04	235	08/10/2013	Server	21.45
Tesla K80	FP32	8.22	300	17/10/2014	Server	27.40

(continued on next page)

Table J.8 (*continued*).

GPU	Precision	TFLOPS	Watts	Launch date	Type	GFLOPS/Watt
Tesla M40	FP32	6.84	250	10/10/2015	Server	27.36
Tesla M60	FP32	9.65	300	30/08/2015	Server	32.17
Tesla P100	FP16	21.20	300	20/05/2016	Server	70.67
Tesla V100	FP16	31.40	300	27/03/2018	Server	104.67
A100	FP16	78.00	400	14/04/2020	Server	195.00
Tesla P100	FP32	10.60	300	20/05/2016	Server	35.33
Tesla V100	FP32	15.70	300	27/03/2018	Server	52.33
A100	FP32	19.50	400	14/04/2020	Server	48.75
A30	FP32	10.30	165	12/04/2021	Server	62.42
A40	FP32	37.4	300	05/10/2020	Server	124.00
Tesla V100	FP16/FP32 Tensor	125.00	300	27/03/2018	Server	416.67
A100	FP16/FP32 Tensor	312.00	400	14/04/2020	Server	780.00
A30	FP16/FP32 Tensor	165.00	165	12/04/2021	Server	1000.00
A40	FP16/FP32 Tensor	149.70	300	05/10/2020	Server	499.00
T4	FP32	8.10	70	13/09/2018	Server	115.71
T4	FP16/FP32 Tensor	65.00	70	13/09/2018	Server	928.57

Table J.9
GPUs throughput and power consumption data compilation.

Adapted	GPU	Precision	TFLOPS	Watts	Launch date	Type	GFLOPS/Watt
No	GeForce GTX 580	FP32	1.58	244	09/11/2010	Desktop	6.48
	GeForce GTX 590	FP32	2.49	365	24/03/2011	Desktop	6.82
	GeForce GTX 680	FP32	3.09	195	22/03/2012	Desktop	15.85
	GeForce GTX 690	FP32	5.62	300	29/04/2012	Desktop	18.73
	Tesla K10	FP32	4.58	225	01/05/2012	Server	20.36
	Tesla K20x	FP32	3.94	235	12/11/2012	Server	16.77
	GeForce GTX 780	FP32	4.16	250	23/04/2013	Desktop	16.64
	Tesla K40	FP32	5.04	235	08/10/2013	Server	21.45
	GeForce GTX 780 TI	FP32	5.35	250	07/11/2013	Desktop	21.40
	GeForce GTX Titan Black	FP32	5.65	250	18/02/2014	Desktop	22.60
	GeForce GTX Titan Z	FP32	8.12	375	28/05/2014	Desktop	21.65
	GeForce GTX 980	FP32	4.98	165	18/09/2014	Desktop	30.18
	Tesla K80	FP32	8.22	300	17/10/2014	Server	27.40
	GeForce GTX TITAN X	FP32	6.69	250	17/03/2015	Desktop	26.76
	GeForce GTX 980 Ti	FP32	6.06	250	02/06/2015	Desktop	24.24
	Tesla M60	FP32	9.65	300	30/08/2015	Server	32.17
	Tesla M40	FP32	6.84	250	10/10/2015	Server	27.36
	GeForce GTX 1080	FP32	8.87	180	26/05/2016	Desktop	49.28
	TITAN X Pascal	FP32	10.97	250	02/08/2016	Desktop	43.88
	GeForce GTX 1080 Ti	FP32	11.34	250	10/03/2017	Desktop	45.36
	TITAN XP	FP32	12.15	250	06/04/2017	Desktop	48.60
	Tesla V100	FP32	15.70	300	27/03/2018	Server	52.33
	Tesla T4	FP32	8.10	70	13/09/2018	Server	115.71
	GeForce RTX 2080	FP32	10.07	215	20/09/2018	Desktop	46.84
	GeForce RTX 2080 Ti	FP32	13.45	250	20/09/2018	Desktop	53.80
	Nvidia Titan RTX	FP32	16.31	280	18/12/2018	Desktop	58.25
	GeForce RTX 3080	FP32	29.80	320	01/09/2020	Desktop	93.13
	GeForce RTX 3090	FP32	35.60	350	01/09/2020	Desktop	101.71
For CNN	Tesla V100	Mixed	35.71	300	27/03/2018	Server	119.03
	Tesla T4	Mixed	21.85	70	13/09/2018	Server	312.15
	A100	TF32	27.41	400	14/04/2020	Server	68.52
	A100	Mixed	52.35	400	14/04/2020	Server	130.88
For NLP	Tesla V100	Mixed	41.44	300	27/03/2018	Server	138.13
	Tesla T4	Mixed	25.58	70	13/09/2018	Server	365.46
	A100	TF32	55.85	400	14/04/2020	Server	139.64
	A100	Mixed	73.29	400	14/04/2020	Server	183.23

Table K.10
Energy consumption of DNN inference. Reported measured values compared with our estimates.

Architecture	Joules					
	P100		V100		RTX 2080 Ti	
	NVML	Estimate	NVML	Estimate	NVML	Estimate
AlexNet	0.033	0.035	0.023	0.022	0.037	0.019
GoogleNet	0.077	0.074	0.055	0.046	0.090	0.040
ResNet50	0.179	0.189	0.132	0.116	0.190	0.102
Vgg16	0.542	0.769	0.373	0.473	0.555	0.417

papers with experimental data of energy consumption. Unfortunately, we have found very few papers where energy consumption (for inference) is reported. But with these, we show that our estimates of energy consumption are comparable to direct measurements in the literature (in those cases where such comparisons can be made).

Before that comparison, let us review some papers that do direct measurement but that are not comparable to our estimations. In [9], Li et al. report energy values for three particular architectures (AlexNet, OverFeat, VGG and GoogleNet) on two GPUs (Nvidia K20 m and TitanX), but they only analyse the energy consumption for training the models. They only provide data for single forward/backward propagation iterations. In [24], Henderson et al. provide kWh measurements for a good number of networks (such as densenet, vgg, squeezenet, etc.) evaluated in ImageNet, as well as a handful of transformers-based networks evaluated in a translation benchmark. However, they run the inference process with batches of one image. Using a batch size of this size does not take advantage of the parallelisation capabilities of the GPU, and usually leads to worse energy efficiency compared to using larger batch sizes [9] (in our estimations we assume maximum GPU usage). In [102], Cao et al. estimate the inference consumption for multiple transformers-based networks for NLP tasks. They also measure real energy consumption data with an emonPi[13] device to prove that

[13] https://guide.openenergymonitor.org/technical/emonpi/

Table K.11
Energy consumption of DNN inference for ResNet-50 with different GPUs. Reported measured values compared with our estimates.

GPU	Imgs/s FP32	Imgs/s FP16	Watts	J/img FP32	J/img FP16	J/img estimated
Nvidia Titan RTX	1617	5672	288	0.1781	0.0507	0.1026
Nvidia RTX 2080 Ti	1515	5318	266	0.1755	0.0500	0.1021
Nvidia Tesla T4	532	2195	74	0.1390	0.0337	0.0958

their estimates are quite good. They use two GPUs (gtx 1080 Ti and gtx 1070) but, in this case, they measure the energy consumed by the whole system, which includes CPU, RAM, etc., and we focus on the energy consumed by the GPU.

We now cover a couple of sources which do provide energy data comparable to our estimations. Tang et al. [103] measure the impact on the energy consumption for inference for three particular NVIDIA GPUs (P100, V100 and RTX 2080 Ti) and four DNNs (alexNet, resnet50, vgg16, and googleNet) when addressing different convolution algorithms (image processing). In this case, power consumption (for inference) is measured by the NVIDIA management library (NVML) API [104], showing the energy consumption as the average energy required by inferring an image depending on the architecture and core frequency of the GPU used.

We compare our estimates with the values measured using the higher core frequency (since we suppose peak TFLOPS and maximum energy consumption for GPUs in our analysis). Our estimates are calculated following the same procedure used to estimate the energy in the main analysis; specifically, we take the number of FLOPs for each network and divide it by the hardware energy efficiency for the launch date of the GPU. In this way we estimate how much energy is required (as per today) to run an inference.

As can be seen in Table K.10, our methodology provides a coherent estimation compared to the actual measurements of energy consumption. Measured values for P100 and V100 are very close to our estimates; in many cases are basically the same. With RTX 2080 Ti the estimates differ slightly from the measured values. This difference is because RTX 2080 Ti is a desktop GPU specialised in gaming. Consequently, it is not as efficient as server GPUs, which are optimised for AI. Since we have included various server AI GPUs, the measured values with this gaming GPU are higher than the estimated ones.

Additionally, we have found an online review [105] in which different GPUs are reviewed and tested for inference for the ResNet-50 architecture using ImageNet as a benchmark. In this case, the GPUs power consumption is measured and the results can be compared to ours. In this regard, we have calculated the energy (Joules) dividing the GPU power by the images per second for the larger batch size. We have compared the data of the benchmark with our estimates, see K.11. It can be seen that our estimated values (last column in Table K.11) are between the measured values for FP32 and FP16. This is in perfect agreement with what is expected since, in our case, we have used FP32 and FP16 performance data to create our estimation model.

Appendix L. Edge devices

In this section we want to stress the difficulty of including 'edge' AI devices (e.g., low-power accelerators) in our analysis due to the reason that they cannot be directly compared with GPUs. This is mainly because these devices usually work with different arithmetic (INT8, Fixed-point arithmetic, etc.) and the provided performance metric is not comparable to the GPUs performance metrics.

The above issues have already been confronted in the literature (e.g., [106]), and different sort of estimations had to be done to compare the different devices to each other. Moreover, some of these edge accelerators can only run a certain type of DNNs, so comparing an optimised device for a certain DNN with a GPU capable of working with all kinds of DNNs would not be fair. Furthermore, AI accelerators in the IoT field usually work with quantised models, i.e., models performing some or all of the operations with reduced precision rather than FP32. According to our investigations and further calculations, quantised models tend to have a lower accuracy because of the lower arithmetic precision used and the estimation of energy consumption based on FLOPS for these models is not reliable. This lack of reliability is due to the fact that the theoretical yield increase in TOPS for INT8 (e.g., 8x for Nvidia Titan RTX [107], 16x for Nvidia T4 [108], or 30x for Nvidia A100 [109]) does not usually correspond to the actual one, and we find many examples in the literature. For instance, when running PyTorch, INT8 computations are typically 2 to 4 times faster on average compared to FP32 compute [110]. For TensorFlow, the speedup is more than 3x [111]. Also, in [112], Kim et al. directly deployed quantised models and measured the end-to-end inference latency, showing that models such as I-BERT can achieve up to 4.00x speedup when using INT8 on a Tesla T4 GPU as compared to floating point baseline (which is far from the 16x that can be observed in the GPU specifications). It is not always possible to fully exploit the INT8 hardware optimisations. In your analysis we have included hardware optimisations in the estimation model but without considering experimental data.

Table L.12
Edge AI devices theoretical efficiency in GOPs/W. CGRA refers to Coarse-Grained Reconfigurable Architecture accelerators.

Device	Year	Type	GOPs/W
Kneron	2018	Edge AI Accelerator	434
Eyeriss (MIT)	2016	Edge AI Accelerator	302
1.42TOPS/W	2016	Edge AI Accelerator	1422
Myriad x (Intel)	2017	Edge AI Accelerator	1500
NVIDIA Tegra X1	2019	Edge AI Accelerator	142
Rockchip RK1808	2018	Edge AI Accelerator	91
Texas InstrumentsAM5729	2019	Edge AI Accelerator	18
GTI Lightspeeur SPR2801S	2019	Edge AI Accelerator	15750
Optimising FPGA-based	2015	Edge AI Accelerator	3
Google Edge TPU	2018	Edge AI Accelerator	1000
ADRES	2017	CGRA	228
VERSAT	2016	CGRA	160
FPCA	2014	CGRA	4333
SURE BasedREDEFINE	2016	CGRA	165
TRANSPIRE	2020	CGRA	239
DT-CGRA	2016	CGRA	53
Heterogeneous PULP	2018	CGRA	386
SOFTBRAIN	2017	CGRA	474
Lopes et al. 2017	2017	CGRA	774
Eyeriss v2	2016	CGRA	960
Nvidia T4	2018	GPU	929
Nvidia A100	2020	GPU	780
Nvidia A30	2021	GPU	1000

We can, however, provide some insights on the performance of this sort of accelerators. For this, we have gathered the theoretical values of GOPs/W from [106]. Results are shown in Table L.12. It can be observed that the efficiency for the different accelerators is very diverse. This can be attributed to the newness of the technology and the fact that there is still a lot of experimentation to be performed. It should also be noted, as already mentioned, that these devices are specifically optimised for certain DNNs. Still, apart from a few exceptions (such as the *GTI Lightspeeur SPR2801S* which has extremely high efficiency), it can be observed that, in general, the theoretical efficiency of edge accelerators is not very far from the theoretical efficiency of GPUs using mixed precision FP16/FP32 on tensor cores.

We have also collected experimental measurements. In L.13 we find the performance (img/s) on ResNet-50, the power of the devices and the energy consumption (which is calculated dividing power by img/s). Similar to what we can observe in the previous table, there is a big

Table L.13
Edge AI devices energy consumption per image for ResNet-50. *Auvidea X220-LC AGX Xavier 32GB* and *Auvidea JNX30 Xavier NX* img/s and power measurements are extracted from MLPerf v1.1 Inference Edge (https://mlcommons.org/en/inference-edge-11/) "Closed-Power" division. *Odroid N2+*, *TurboX EB6 Edge AI Box*, *Firefly-RK3588S* and *NVIDIA Orin* img/s and power measurements are extracted from MLPerf v2.1 Inference Edge "Closed-Power" division (https://mlcommons.org/en/inference-edge-21/). *NVIDIA Jetson Nano* power is from the manufacturer specifications and img/s are from OpenBenchmarking (https://openbenchmarking.org/result/1908087-HV-1908018HV82).

System	Launch year	ResNet-50 img/s	Power	J/img
Auvidea X220-LC AGX Xavier 32GB (MaxQ, TensorRT)	2018	1.506.53	25.24	0.017
NVIDIA Jetson Nano	2019	42.65	10.00	0.238
Odroid N2+	2020	3.84	3.89	1.013
Auvidea JNX30 Xavier NX (MaxQ, TensorRT)	2020	1,092.08	19.97	0.018
TurboX EB6 Edge AI Box	2021	7259.95	25.32	0.003
Firefly-RK3588S	2022	13.00	7.55	0.581
NVIDIA Orin (MaxQ, TensorRT)	2022	3525.91	23.06	0.007

difference between devices in terms of efficiency. Also, it is not possible to extract a clear pattern from the data. This makes the task of analysing overall trends considerably more difficult. Still, we can analyse the systems individually. As a reference we can take the value of T4 for INT16 and INT8 from the previous analysed benchmark [108]: the resulting energy is 0.017 J/img for INT8 and 0.034 J/img for FP16 (the T4 is from 2018). The INT8 value is the same as the *Auvidea X220-LC AGX Xavier* system, one of the most efficient edge systems, as can be seen in Table L.13.

References

[1] J. McDonald, B. Li, N. Frey, D. Tiwari, V. Gadepally, S. Samsi, Great power, great responsibility: Recommendations for reducing energy for training language models, 2022, arXiv preprint arXiv:2205.09646.
[2] K. Freund, Google cloud doubles down on nvidia gpus for inference, 2019, URL https://www.forbes.com/sites/moorinsights/2019/05/09/google-cloud-doubles-down-on-nvidia-gpus-for-inference.
[3] D. Hernandez, T. Brown, Ai and Efficiency, Tech. Rep., OpenAI, San Francisco, CA, USA, 2020, abs/2005.04305 arxiv.
[4] J. Barr, Amazon ec2 update-infl instances with aws inferentia chips for high performance cost-effective inferencing, 2019.
[5] G. Leopold, Aws to offer nvidia's t4 gpus for ai inferencing, 2019, URL: https://web.archive.org/web/20220309000921/ https://www.hpcwire.com/2019/03/19/aws-upgrades-its-gpu-backed-ai-inference-platform/(visited on 2022-04-19).
[6] D. Amodei, D. Hernandez, Ai and compute, 2018, https://openai.com/blog/ai-and-compute/.
[7] A. Gholami, Z. Yao, S. Kim, M.W. Mahoney, K. Keutzer, Ai and memory wall, 2021, RiseLab Medium Post.
[8] A. Canziani, A. Paszke, E. Culurciello, An analysis of deep neural network models for practical applications, 2017, arXiv:1605.07678.
[9] D. Li, X. Chen, M. Becchi, Z. Zong, Evaluating the energy efficiency of deep convolutional neural networks on cpus and gpus, in: 2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom), BDCloud-SocialCom-SustainCom, 2016, pp. 477–484, http://dx.doi.org/10.1109/BDCloud-SocialCom-SustainCom.2016.76.
[10] L.F.W. Anthony, B. Kanding, R. Selvan, Carbontracker: Tracking and predicting the carbon footprint of training deep learning models, 2020, arXiv preprint arXiv:2007.03051.
[11] N.C. Thompson, K. Greenewald, K. Lee, G.F. Manso, The computational limits of deep learning, 2020, arXiv preprint arXiv:2007.05558.
[12] C.F. Rodrigues, G. Riley, M. Luján, Synergy: An energy measurement and prediction framework for convolutional neural networks on jetson tx1, in: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA, The Steering Committee of The World Congress in Computer Science, Computer . . . , 2018, pp. 375–382.
[13] E. Cai, D.-C. Juan, D. Stamoulis, D. Marculescu, Neuralpower: Predict and deploy energy-efficient convolutional neural networks, in: Asian Conference on Machine Learning, PMLR, 2017, pp. 622–637.
[14] V.E. Balas, S.S. Roy, D. Sharma, P. Samui, Handbook of Deep Learning Applications, Vol. 136, Springer, 2019.
[15] J. Park, M. Naumov, P. Basu, S. Deng, A. Kalaiah, D. Khudia, J. Law, P. Malani, A. Malevich, S. Nadathur, J. Pino, M. Schatz, A. Sidorov, V. Sivakumar, A. Tulloch, X. Wang, Y. Wu, H. Yuen, U. Diril, D. Dzhulgakov, K. Hazelwood, B. Jia, Y. Jia, L. Qiao, V. Rao, N. Rotem, S. Yoo, M. Smelyanskiy, Deep learning inference in facebook data centers: Characterization, performance optimizations and hardware implications, 2018, arXiv:1811.09886.
[16] X. Xu, Y. Ding, S.X. Hu, M. Niemier, J. Cong, Y. Hu, Y. Shi, Scaling for edge inference of deep neural networks, Nat. Electron. 1 (4) (2018) 216–222.
[17] S. Bianco, R. Cadene, L. Celona, P. Napoletano, Benchmark analysis of representative deep neural network architectures, IEEE Access 6 (2018) 64270–64277.
[18] European Environment Information, Greenhouse gas emission intensity of electricity generation in Europe, 2021, https://www.eea.europa.eu/data-and-maps/indicators/overview-of-the-electricity-production-3/assessment-1.
[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, Imagenet large scale visual recognition challenge, 2015, arXiv:1409.0575.
[20] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, S.R. Bowman, GLUE: A multi-task benchmark and analysis platform for natural language understanding, in: The Proceedings of ICLR, 2019.
[21] D. Hernandez, T.B. Brown, Measuring the algorithmic efficiency of neural networks, 2020, arXiv:2005.04305.
[22] R. Schwartz, J. Dodge, N.A. Smith, O. and Etzioni, Green ai, 2019, arXiv:1907.10597.
[23] E. Strubell, A. Ganesh, A. McCallum, Energy and policy considerations for deep learning in nlp, 2019, arXiv:1906.02243.
[24] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, J. Pineau, Towards the systematic reporting of the energy and carbon footprints of machine learning, J. Mach. Learn. Res. 21 (248) (2020) 1–43.
[25] R. Bommasani, D.A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M.S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al., On the opportunities and risks of foundation models, 2021, arXiv:2108.07258.
[26] M. Tan, Q.V. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, 2020, arXiv:1905.11946.
[27] M. Tan, Q.V. Le, Efficientnetv2: Smaller models and faster training, 2021, arXiv:2104.00298.
[28] T. Molom-Ochir, R. Shenoy, Energy and cost considerations for GPU accelerated AI inference workloads, in: 2021 IEEE MIT Undergraduate Research Technology Conference, URTC, IEEE, 2021, pp. 1–5.
[29] A. Paszke, S. Gross, S. Chintala, G. Chanan, Torchvision models, 2016, https://pytorch.org/vision/stable/models.html.
[30] R. Cadene, Pretrained models for Pytorch, 2016, https://github.com/Cadene/pretrained-models.pytorch#torchvision.
[31] O. Sémery, Computer vision models on pytorch, 2019, https://pypi.org/project/pytorchcv/.
[32] S. Albanie, Convnet burden: Estimates of memory consumption and flop counts for various convolutional neural networks, 2016, https://github.com/albanie/convnet-burden.
[33] C. Coleman, D. Narayanan, D. Kang, T. Zhao, J. Zhang, L. Nardi, P. Bailis, K. Olukotun, C. Ré, M. Zaharia, Dawnbench: An end-to-end deep learning benchmark and competition, Training 100 (101) (2017) 102.
[34] P. Mattson, V.J. Reddi, C. Cheng, C. Coleman, G. Diamos, D. Kanter, P. Micikevicius, D. Patterson, G. Schmuelling, H. Tang, et al., Mlperf: An industry standard benchmark suite for machine learning performance, IEEE Micro 40 (2) (2020) 8–16.
[35] R. Stojnic, R. Taylor, Papers with code imagenet benchmark (image classification), 2021, https://paperswithcode.com/sota/image-classification-on-imagenet.
[36] V. Sovrasov, Flops counter for convolutional networks in pytorch framework, 2020, https://github.com/sovrasov/flops-counter.pytorch.
[37] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 25 (2012) 1097–1105.
[38] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2015, arXiv:1409.1556.
[39] X. Zhai, A. Kolesnikov, N. Houlsby, L. Beyer, Scaling vision transformers, 2021, arXiv:2106.04560.
[40] K. Clark, M.-T. Luong, Q.V. Le, C.D. Manning, Electra: Pre-training text encoders as discriminators rather than generators, 2020, arXiv:2003.10555.
[41] G. Batra, Z. Jacobson, S. Madhav, A. Queirolo, N. Santhanam, Artificial-Intelligence Hardware: New Opportunities for Semiconductor Companies, McKinsey Co., 2018.

[42] A. HajiRassouliha, A.J. Taberner, M.P. Nash, P.M. Nielsen, Suitability of recent hardware accelerators (dsps, fpgas, and gpus) for computer vision and image processing algorithms, Signal Process., Image Commun. 68 (2018) 101–119.
[43] S. Asano, T. Maruyama, Y. Yamaguchi, Performance comparison of fpga, gpu and cpu in image processing, in: 2009 International Conference on Field Programmable Logic and Applications, IEEE, 2009, pp. 126–131.
[44] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 248–255.
[45] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.
[46] M. Hollemans, How fast is my model? 2018, https://machinethink.net/blog/how-fast-is-my-model/.
[47] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, 2013, arXiv:1311.2901.
[48] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, 2014, arXiv:1409.4842.
[49] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017, arXiv:1704.04861.
[50] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, 2019, arXiv:1801.04381.
[51] Q. Xie, M.-T. Luong, E. Hovy, Q.V. Le, Self-training with noisy student improves imagenet classification, 2020, arXiv:1911.04252.
[52] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, D. Zhou, Mobilebert: a compact task-agnostic bert for resource-limited devices, 2020, arXiv:2004.02984.
[53] F.N. Iandola, A.E. Shaw, R. Krishna, K.W. Keutzer, Squeezebert: What can computer vision teach nlp about efficient neural networks? 2020, arXiv:2006.11316.
[54] H. Ritchie, M. Roser, Energy, Our World in Data, 2020, https://ourworldindata.org/energy.
[55] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. Patwary, M. Ali, Y. Yang, Y. Zhou, Deep learning scaling is predictable, empirically, 2017, arXiv preprint arXiv:1712.00409.
[56] J. Kaplan, S. McCandlish, T. Henighan, T.B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, D. Amodei, Scaling laws for neural language models, 2020, arXiv preprint arXiv:2001.08361.
[57] T. Henighan, J. Kaplan, M. Katz, M. Chen, C. Hesse, J. Jackson, H. Jun, T.B. Brown, P. Dhariwal, S. Gray, et al., Scaling laws for autoregressive generative modeling, 2020, arXiv preprint arXiv:2010.14701.
[58] C. NVIDIA, Achieved FLOPs, 2015, https://docs.nvidia.com/gameworks/content/developertools/\protect\discretionary{\char\hyphenchar\font}{}{}desktop/analysis/report/cudaexperiments/kernellevel/achievedflops.htm.
[59] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015, arXiv:1502.03167.
[60] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, 2015, arXiv:1512.00567.
[61] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, 2015, arXiv:1512.03385.
[62] K. He, X. Zhang, S. Ren, J. Sun, Deep residual networks github, 2015, https://github.com/KaimingHe/deep-residual-networks.
[63] F. Chollet, Keras applications, 2015, https://keras.io/api/applications/.
[64] C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, 2016, arXiv:1602.07261.
[65] G. Huang, Z. Liu, L. van der Maaten, K.Q. Weinberger, Densely connected convolutional networks, 2018, arXiv:1608.06993.
[66] F. Chollet, Xception: Deep learning with depthwise separable convolutions, 2017, arXiv:1610.02357.
[67] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, 2017, arXiv:1611.05431.
[68] X. Zhang, X. Zhou, M. Lin, J. Sun, Shufflenet: An extremely efficient convolutional neural network for mobile devices, 2017, arXiv:1707.01083.
[69] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, J. Feng, Dual path networks, 2017, arXiv:1707.01629.
[70] B. Zoph, V. Vasudevan, J. Shlens, Q.V. Le, Learning transferable architectures for scalable image recognition, 2018, arXiv:1707.07012.
[71] J. Hu, L. Shen, S. Albanie, G. Sun, E. Wu, Squeeze-and-excitation networks, 2019, arXiv:1709.01507.
[72] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, K. Murphy, Progressive neural architecture search, 2018, arXiv:1712.00559.
[73] E. Real, A. Aggarwal, Y. Huang, Q.V. Le, Regularized evolution for image classifier architecture search, 2019, arXiv:1802.01548.
[74] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, L. van der Maaten, Exploring the limits of weakly supervised pretraining, 2018, arXiv:1805.00932.
[75] N. Ma, X. Zhang, H.-T. Zheng, J. Sun, Shufflenet v2: Practical guidelines for efficient cnn architecture design, 2018, arXiv:1807.11164.
[76] I.Z. Yalniz, H. Jégou, K. Chen, M. Paluri, D. Mahajan, Billion-scale semi-supervised learning for image classification, 2019, arXiv:1905.00546.
[77] I.Z. Yalniz, H. Jégou, K. Chen, M. Paluri, D. Mahajan, Semi-supervised and semi-weakly supervised imagenet models github, 2019, https://github.com/facebookresearch/semi-supervised-ImageNet1K-models.
[78] H. Touvron, A. Vedaldi, M. Douze, H. Jégou, Fixing the train-test resolution discrepancy: Fixefficientnet, 2020, arXiv:2003.08237.
[79] H. Pham, Z. Dai, Q. Xie, M.-T. Luong, Q.V. Le, Meta pseudo labels, 2021, arXiv:2003.10580.
[80] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, H. Lin, Z. Zhang, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, A. Smola, Resnest: Split-attention networks, 2020, arXiv:2004.08955.
[81] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16 × 16 words: Transformers for image recognition at scale, 2021, arXiv:2010.11929.
[82] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jégou, Training data-efficient image transformers & distillation through attention, 2021, arXiv:2012.12877.
[83] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jégou, Deit: Data-efficient image transformers github, 2021, https://github.com/facebookresearch/deit.
[84] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z. Jiang, F.E. Tay, J. Feng, S. Yan, Tokens-to-token vit: Training vision transformers from scratch on imagenet, 2021, arXiv:2101.11986.
[85] A. Srinivas, T.-Y. Lin, N. Parmar, J. Shlens, P. Abbeel, A. Vaswani, Bottleneck transformers for visual recognition, 2021, arXiv:2101.11605.
[86] A. Brock, S. De, S.L. Smith, K. Simonyan, High-performance large-scale image recognition without normalization, 2021, arXiv:2102.06171.
[87] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin transformer: Hierarchical vision transformer using shifted windows, 2021, arXiv:2103.14030.
[88] C.-F. Chen, Q. Fan, R. Panda, Crossvit: Cross-attention multi-scale vision transformer for image classification, 2021, arXiv:2103.14899.
[89] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, H. Jégou, Going deeper with image transformers, 2021, arXiv:2103.17239.
[90] C. NVIDIA, Nvidia tesla v100 GPU architectur, 2017, https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf.
[91] C. Li, Openai's gpt-3 language model: A technical overview, 2020, https://lambdalabs.com/blog/demystifying-gpt-3.
[92] C. NVIDIA, Training with mixed precision, 2018, https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html.
[93] M.E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, 2018, arXiv:1802.05365.
[94] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving language understanding by generative pre-training, 2018.
[95] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2019, arXiv:1810.04805.
[96] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, 2019.
[97] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, B. Catanzaro, Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020, arXiv:1909.08053.
[98] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, Albert: A lite bert for self-supervised learning of language representations, 2020, arXiv:1909.11942.
[99] C. Xu, W. Zhou, T. Ge, F. Wei, M. Zhou, Bert-of-theseus: Compressing bert by progressive module replacing, 2020, arXiv:2002.02925.
[100] C. Rosset, Turing-nlg: A 17-billion-parameter language model by microsoft, 2020, https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/.
[101] T.B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D.M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, 2020, arXiv:2005.14165.
[102] Q. Cao, Y.K. Lal, H. Trivedi, A. Balasubramanian, N. Balasubramanian, Irene: Interpretable energy prediction for transformers, 2021, arXiv preprint arXiv:2106.01199.
[103] Z. Tang, Y. Wang, Q. Wang, X. Chu, The impact of gpu dvfs on the energy and performance of deep learning: An empirical study, in: Proceedings of the Tenth ACM International Conference on Future Energy Systems, 2019, pp. 315–325.
[104] NVIDIA, Nvidia management library, 2022, https://developer.nvidia.com/nvidia-management-library-nvml.
[105] W. Harmon, Nvidia tesla t4 ai inferencing gpu benchmarks and review, 2019.
[106] W. Lin, A. Adetomi, T. Arslan, Low-power ultra-small edge ai accelerators for image recognition with convolution neural networks: Analysis and future directions, Electronics 10 (17) (2021) 2048.

[107] NVIDIA, Titan rtx specifications, 2019, https://www.nvidia.com/content/dam/en-zz/Solutions/titan/documents/titan-rtx-for-creators-us-nvidia-1011126-r6-web.pdf.
[108] NVIDIA, Titan t4 specifications, 2019, https://www.nvidia.com/en-us/data-center/tesla-t4/.
[109] NVIDIA, Titan a100 tensor core gpu specifications, 2022, https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-nvidia-us-2188504-web.pdf.
[110] PyTorch, Pytorch: Quantization, 2022, https://pytorch.org/docs/stable/quantization.html.
[111] PyTorch, Tensorflow: Post-training quantization, 2022, https://www.tensorflow.org/lite/performance/post_training_quantization.
[112] S. Kim, A. Gholami, Z. Yao, M.W. Mahoney, K. Keutzer, I-bert: Integer-only bert quantization, in: International Conference on Machine Learning, PMLR, 2021, pp. 5506–5518.