Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Analysis of Deep Learning Inference Compute and Energy Consumption Trends

MASTER'S DEGREE FINAL WORK

Master's Degree in Informatics Engineering

Author: Radosvet Desislavov Georgiev

Tutors: José Hernández Orallo
Fernando Martínez Plumed

Academic Year 2020-2021

Resumen

En la última década, el aprendizaje profundo ha conseguido resultados espectaculares en numerosas aplicaciones. El incremento del desempeño ha venido acompañado de un importante incremento en el tamaño de estas redes, medido en número de parámetros. Pero ¿cómo afecta este incremento del tamaño de las redes a la energía consumida por éstas? En este trabajo analizamos el cómputo (número de operaciones) y energía necesaria para llevar a cabo una inferencia (o propagación hacia delante) con diferentes redes neuronales. Elegimos inferencia ya que para las redes neuronales que se despliegan para uso masivo, el cómputo necesario para realizar las inferencias acaba dominando al cómputo necesario para entrenar la red, ya que se llegan a ejecutar una gran cantidad de inferencias a lo largo del tiempo. Para llevar a cabo el estudio recogemos datos sobre una gran cantidad de redes neuronales y sobre hardware, para poder así estimar el consumo y su evolución. Centramos el análisis en el campo de la visión artificial y en el campo del procesamiento de lenguaje natural. Observamos que la evolución del consumo energético crece exponencialmente para las redes que marcan un hito a nivel investigador, pero para las redes que consolidan la innovación este crecimiento es mucho menor; mejoran sus prestaciones significativamente a lo largo de los años, incluso con consumo de energía constante.

Palabras clave: Inteligencia artificial, Análisis de datos, Aprendizaje profundo, Redes neuronales

Resum

En l'última dècada, l'aprenentatge profund ha aconseguit resultats espectaculars en nombroses aplicacions. L'increment de les prestacions ha vingut acompanyat d'un important increment en la grandària d'aquestes xarxes, mesurat en nombre de paràmetres. Però com afecta aquest increment de la mida de les xarxes a l'energia consumida per aquestes? En aquest treball analitzem el còmput (nombre d'operacions) i energia necessària per dur a terme una inferència (o propagació cap endavant) amb diferents xarxes neuronals. Triem inferència ja que per a les xarxes neuronals que es despleguen per a ús massiu, el còmput necessari per realitzar les inferències acaba dominant el còmput necessari per entrenar la xarxa, ja que s'arriben a executar una gran quantitat d'inferències al llarg del temps. Per dur a terme l'estudi recollim dades sobre una gran quantitat de xarxes neuronals i sobre hardware, per poder així estimar el consum i la seua evolució. Centrem l'anàlisi en el camp de la visió artificial i en el camp del processament de llenguatge natural. Observem que l'evolució del consum energètic creix exponencialment per a les xarxes que marquen una fita a nivell investigador, però per a les xarxes que consoliden la innovació aquest creixement és molt menor; milloren les seues prestacions significativament al llarg dels anys, fins i tot amb consum d'energia constant.

Paraules clau: Intel·ligència artificial, Anàlisi de dades, Aprenentatge profund, Xarxes neuronals

Abstract

In the last decade, deep learning has achieved spectacular results in numerous applications. The increase in performance has been accompanied by a significant increase in the size of these networks, measured in number of parameters. But how does this increase in the size of the networks affect the energy consumed by them? In this work we analyze the computation (number of operations) and required energy to carry out an inference (or forward propagation) with different neural networks. We chose inference since for neural networks that are deployed for large scale use, the computation necessary to make the inferences ends up dominating the computation necessary to train the network, since a large number of inferences are executed over time. To carry out the study we collected data on a large number of neural networks and hardware, in order to estimate consumption and its evolution. We focus the analysis in the field of computer vision and in the field of natural language processing. We observe that the evolution of energy consumption grows exponentially for networks that mark a milestone at research level, but for networks that consolidate innovation this growth is much lower; their performance improves significantly over the years, even with constant energy consumption.

Key words: Artificial Intelligence, Data Analysis, Deep Learning, Neural Networks

Contents

Appendices

List of Figures

List of Tables

CHAPTER 1
Introduction

The effect of computers on society has been constantly evaluated since the beginning of the discipline. In particular, the balance between energy consumption and computing performance has been discussed since the early computers, frequently of a size of a room and with enormous energy consumption. Today, artificial intelligence (AI) is one of the fastest-growing areas in computer science in terms of compute and usage. Many Deep Learning (DL) models are becoming very complex and computationally heavy. Consequently, it is important to analyse the trends in compute and energy consumption, in order to determine the impact and its future sustainability.

In the last decade DNNs have become very accurate and have been replacing other legacy AI systems. The term Deep Learning is applied to any machine learning algorithms over Deep Neural Networks. A DNN is an artificial neural network with many hidden layers, in comparison to more shallow networks, using only one or very few hidden layers. DNNs use multiple layers to progressively extract higher-level features from the raw input. This kind of networks can perform an extensive amount of tasks, such as image classification, object detection in images, text generation, etc. DNNs have surpassed humans in many tasks, for instance, in SuperGLUE benchmark [1]. SuperGLUE benchmark is a set of language understanding tasks, with a public leaderboard[1]. In the leaderboard there is a human performance baseline and some DNNs have surpassed it. But at which cost comes this great performance of DNNs? In this work we explore the evolution of different metrics of DNNs, paying particular attention to *inference* computational cost and inference energy consumption, which are associated with economical costs and ultimately having its carbon footprint.

Nowadays, DNNs are almost everywhere, from smartphones to automobiles. In automobiles, inference speed is critical because real time response is required, so very big models that are slow at performing inference are not suitable. In smartphones, model size is critical too because these devices do not have as much computing power and memory as a desktop computer or as a server. More importantly, they cannot consume much energy because they have important power consumption limitations because they are powered by batteries and do not have good heat dissipation (overheating occurs if they consume a lot of energy). Despite these restrictions, DNNs are being used in these and other situations. Precisely because of this, there has been a pressure to build DNNs that are not so resource demanding, even if larger DNNs usually outperform smaller ones. Alternatively to this in-device use, many DNNs run on the cloud, where DNNs could be bigger. Many services that people use daily execute inference in data centres. For instance, in social networks [2], users are not aware of this because the process is transparent for them. The servers receive a huge amount of requests to be executed, the

[1]https://super.gluebenchmark.com/leaderboard

burden and energy consumption on the server side grows significantly, especially if there are millions of requests, implying millions of inferences over the same DNN.

In our study, we clearly differentiate between training and inference. Training consists in "teaching" a DNN to perform a task. During training, data is provided to the network, and it learns from that data. Learning consists in updating the strength of the connections between the artificial neurons, connections are numeric values, called network parameters. More specifically, a forward pass takes place first, a forward pass refers to the propagation of the input data through all the layers, from the input layer to the output layer, obtaining an output result from the output layer. Next, the difference between the result and the expected value is calculated, using a loss function, and a backward pass is performed. The backward pass consists in calculating the gradient of neural network parameters using the gradient descent algorithm and the backpropagation technique from the output to the input layer. Finally, the calculated gradient is used to update the network parameters (i.e. learn). The learning process is compute intensive because a large amount of data is used for training, and the data is fed into the network many times until it achieves optimal performance.

Once the network is trained, it can be used to perform predictions about real world data. This process is called inference, which is based in performing a forward-pass alone, since there is no learning. At first look, it seems that the training cost is higher. However, for successfully deployed systems, inference costs tend to exceed training costs, because inference costs build up with time due to the usage of the system. Training is done once, but inference is done repeatedly. It is estimated that inference accounts for up to 90% of the costs in successfully deployed systems. [2]

For conducting our analysis two representative domains have been chosen: Computer Vision (CV) and Natural Language Processing (NLP). For CV we analyse image classification models, because (1) there is a great quantity of historical data in this area and (2) 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. More specifically, we analyse DNN approaches with published results for the Imagenet dataset [3]. For NLP we will analyse results on General Language Understanding Evaluation (GLUE) benchmark [4]. In both cases, we focused our analysis on inference FLOPs (Floating Point Operations). This metric represents the number of operations that are required to process one input item (image for CV models and text fragment for NLP models). We collect inference FLOPs for many different DNNs architectures and implementations following a comprehensive literature review. Due to the immense recent growth in the field of deep learning, 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 through the years, and estimate how much FLOPs can be obtained with one Joule with each chip. Having all this data we can finally estimate how much energy is needed to perform one inference with a given DNN.

1.1 Motivation

There are many studies reporting that compute used for neural networks is growing at a very fast pace the last decade [5, 6]. These studies will be discussed more extensively in section 2.1. Figure 1.1 shows how the amount of compute needed to train AI systems is increasing much faster in the last years than before. Due to this tendency we consider

[2]Percentage announced in a Amazon Web Services Online Tech Talk: https://www.youtube.com/watch?v=ET2KVe2du3Y.

important to explore this growth deeper, specially in terms of energy consumption. We have observed that there are many publications about training computation and its environmental impact but there are very few focused on inference costs and its associated energy consumption. As discussed before, for successful deployed systems inference cost is usually higher than training cost. All this motivates us to carry out an analysis about inference compute usage and energy consumption.

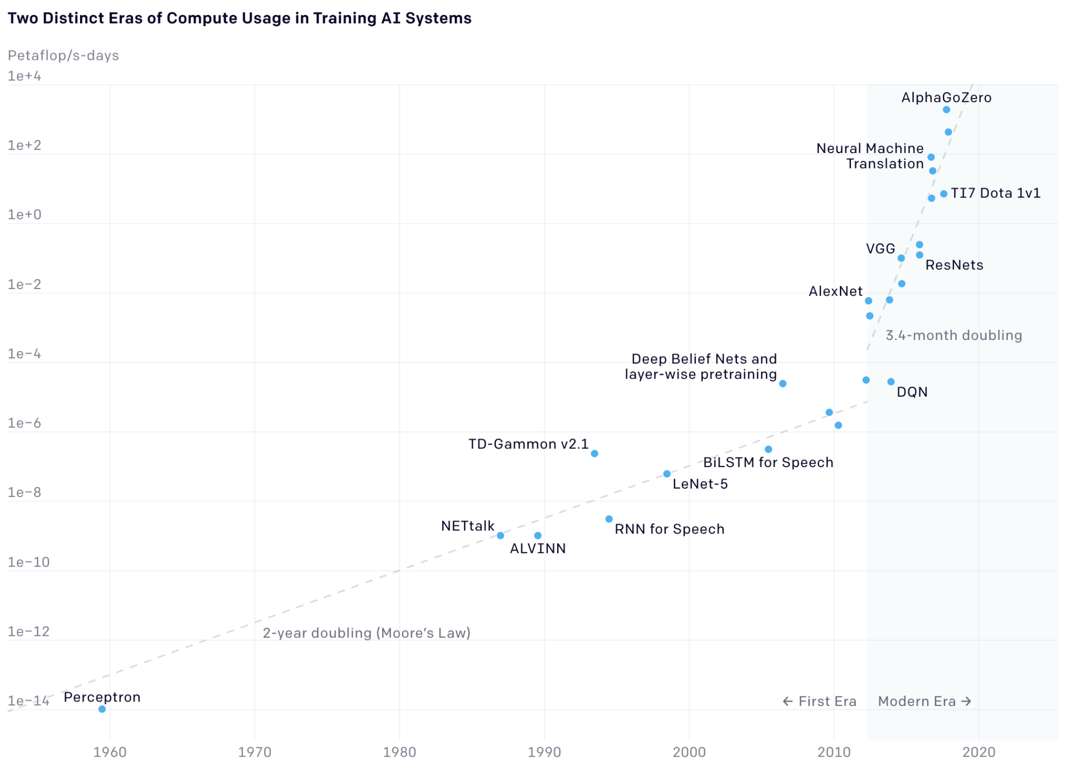


Figure 1.1: Compute usage in training AI systems evolution. Image obtained from [5].

Determining energy consumption is very important for a sustainable development. As seen in the last decades, a constant increment in CO_2 emissions is unsustainable. Although efforts are being made to reduce the utilization of polluting processes to obtain energy, they are still being an important part of the electricity mix. In the European Union in the year 2019, almost 40% of the consumed electricity came from power stations burning fossil fuels [3]. This percentage is decreasing each year, but consuming more energy still means more CO_2 emissions. For this reason it is important to evaluate energy consumption of computation systems since the use of computing devices is increasing (each time we use more and more computers, smartphones, Internet of Things devices, etc.).

Our motivation is to provide an analysis that can help to a more sustainable use of the AI. In particular, this project could help to the Sustainable Development Goals (SDGs) proposed by The United Nations. Concretely it is perfectly aligned with Goal 11: Sustainable Cities and Communities; Goal 12: Responsible Consumption and Production; and Goal 13: Climate Action.

Regarding personal motivations, I am interested in focusing my career on AI. Carrying out this analysis will help me understand some AI concepts better, as well as be-

[3] https://ec.europa.eu/eurostat/cache/infographs/energy/bloc-3b.html?lang=en
[3] https://unfoundation.org/what-we-do/issues/sustainable-development-goals/

coming more familiar with the trends in AI, its sustainability and its present and future possibilities.

1.2 Objectives

As we have seen, the required compute to train a DNN grows exponentially. However, this does not necessarily imply that the energy cost is also growing exponentially, as the energy consumption per unit of compute is decreasing. Furthermore, many studies focus on the state-of-the-art (SOTA) or the cutting-edge methods according to a given metric of performance to declare this very fast growth, but many algorithmic improvements usually come in the months or few years after a new technique has been introduced, having similar results with much lower compute requirements. All of these elements have been studied separately, but a more comprehensive analysis of all these factors together is necessary to properly evaluate whether the impact of AI on energy consumption and its carbon footprint is alarming or simply worrying.

Consequently, the objectives of this work are the following ones:

- To study the evolution of required FLOPs to perform an inference with a DNN and analyse the relation between network performance and inference FLOPs through the years.
- To study the evolution of the required energy for one inference through the years and observe if hardware efficiency progress cancels out the increment observed in FLOPs.
- To provide a comprehensive analysis based on the previous points that brings more information about whether the growth of AI models will be unsustainable soon, and discuss if this will mean the end of the progress in DL.

1.3 Expected Impact

We expect to provide a deeper analysis of DL progress. Usually in research papers or data compilations about DNNs, the required compute and energy consumption for inference is not taken into account. However, these metrics are usually more important for the industry than the usually reported values about training effort of models in research papers. The usually reported values are model's performance, which many times are only better by a tenth than the previous SOTA model, and the number of parameters. These metrics are not very informative for real applications, so we provide a more helpful analysis. Furthermore, in the analysis we include middle and small-sized models which many times do not receive much attention but are very important in terms of efficiency. We hope that this analysis could help to the adoption of efficient AI models in the industry, and give an estimation of AI models energy costs to companies and governmental organisations.

1.4 Structure

The organization of this work is as follows: first some background information is presented (chapter 2), which includes related work analysis and some background for DNNs and hardware. After that, methodology is explained (chapter 3), in this chapter we expose the used methodology for carrying out the analysis. Next comes the analysis: we present first inference compute analysis (chapter 4) and after that, energy consumption analysis (chapter 5). Finally we draw the conclusions of the work in (chapter 6).

CHAPTER 2

Background

In this chapter, first are presented some related studies to this one, after that, the history of neural networks is briefly explained. Once the history is explained, some modern deep neural network architectures are presented. Next, an review of hardware for AI is exposed, and the chapter finishes with some recent achievements performed with DL.

2.1 Related Work

In line with other areas of computer science, there is previous work that analyses compute and its cost for AI, and DNNs more specifically. Still, with the rapid increase in complexity of DNN, along with the volumes of data to be processed, there is a clear corresponding growth in demand for AI computing and numerous studies have addressed this issue. [6] reports the computational demands of several Deep Learning applications showing that progress in them is strongly reliant on increases in computing power. Certainly, this imposes a clear limit on how far we can improve performance in its current form. The same author mentioned in [7] how companies such as Google and Facebook build high-impact, cost-saving models, that go unused due to computational cost, which makes the model not viable economically. In this regard, OpenAI researchers estimated that the computing power needed to train AI models is now rising seven times faster than ever before, and AI models have doubled the computational power used every 3.4 months since 2012 [5]. There is a study [8] that reports similar scaling rates for AI training compute to [5] and they forecast that DNNs memory requirements will soon become the main limiting factor. Recently, OpenAI carried out a detailed analysis about AI efficiency [9], focusing on the compute used to train models with Imagenet dataset. Despite of their previous estimation which shows that used compute for AI training is growing very fast, their work proves that 44 times less compute was required in 2020 to train a network to reach the performance AlexNet achieved seven years before.

Regarding inference compute, [10] studies accuracy, memory footprint, parameters, operations count, inference time and power consumption of 14 Imagenet models. To measure the power consumption they execute the DNNs on a NVIDIA Jetson TX1 board[1]. A similar study [11] 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 CNNs analysed. Both publications analyse model efficiency, but they do this for very concrete cases. New algorithms and architectures such as EfficientNet [12] and EfficientNetV2 [13] aimed at reduction in

[1]https://developer.nvidia.com/embedded/jetson-tx1-developer-kit

compute. The papers include an extensive comparison of different DNNs, with particular emphasis on FLOPs and accuracy. In this work, we focus on exploring the trends over the years, we analyse a greater number of DNNs and hardware components in a longer time frame.

The accelerating growth in compute has been accompanied with increasing concern about efficiency and energy consumption. [14] estimates the energy consumption, the cost and CO_2 emissions of training various of the most popular NLP models. [15] proposes a systematic reporting of the energy and carbon footprints of machine learning, they have developed a framework that tracks energy usage, carbon emissions, and compute utilisation of a system[2]. [16] (section 5.3) seeks to identify assumptions that shape the calculus of environmental impact for foundation models. [17] analyzes training costs and proposes that researchers should put more attention on efficiency and they should always report the number of FLOPs.

2.1.1. FLOPS and FLOPs

Terminology used to count the number of floating point operations is usually confusing. We found out that the acronym FLOP may be misleading. By FLOP, we mean one floating point operation, and we use FLOPs as the plural of FLOP, 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 term FLOPS to refer to the number of operations, but we will be just use FLOPs as the plural of FLOP, never as FLOPS. There are some researchers proposing to use the term FPO [17] as total number of floating point operations to avoid confusions. Furthermore, there is the question of what is a FLOP. 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 for DNNs the predominant operations are of the type multiply-add, so counting only this operations is a good estimation [18, 19]. More specifically, we will count one fused multiply-add operation as 2 FLOPs (note the lowercase 's'). Hardware manufacturers count them in this manner [20], 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.

2.2 History of Neural Networks

Neural Networks (NN) could seem a new modern approach, but in fact NNs were proposed long time ago. The idea of NNs appeared in 1943 as a model of how neurons in the brain work proposed by Warren McCulloch and Walter Pitts [21]. This model used connected circuits to simulate intelligent behaviour. In 1958 Rosenblatt created the Perceptron, an algorithm which can learn to perform binary classification through supervised learning, a perceptron operates in similar to modern NNs. But a perceptron has a important drawback, it can only learn to separate linearly separable classes. In 1959 Bernard Widrow and Marcian Hoff developed two models called ADALINE and MADALINE. ADALINE recognises binary patterns, it can predict the next bit of a bit

[2] https://github.com/Breakend/experiment-impact-tracker

stream. MADALINE, the multi-layer version of ADALINE, was the first NN to be applied to a real problem: it was used to eliminate echoes on phone lines.

These achievements promoted the optimism about the possibilities of NN, but researchers faced many problems trying to improve and extend the networks. Finally, in 1969 the optimisms for NN came to its end with the publication of a book called "Perceptrons" by Marvin Minsky. The book argued that the single perceptron approach to neural networks could not be translated effectively into multi-layered neural networks. , The approach was said to be infeasible because the evaluation of the correct relative values of the weights of the neurons spread across layers based on the final output would take a large number of iterations and would take a very long time to be computed. This book had huge impact on the research community, and research about NNs stopped for a decade. This period is known as "the AI winter".

In 1981 the research on neural networks started out again, with some conferences about NNs being organised. In 1986 a technique called backpropagation was rediscovered and popularised. This was a great advance in the field because backpropagation allowed to train multi-layer networks more easily. Since then, backpropagation along with gradient descent is being used to train NN. Nowadays, DNN are at the spotlight, and this in part due to the advances in compute power (specially in GPUs) and distributed computing of the last decade. This progress allowed to build more complex networks and to process larger amounts of data for training.

2.3 Deep Neural Networks Architectures

Deep Neural Networks Architectures have evolved, from the basic network architecture that can be observed in Figure 2.1 to more complex architectures, like Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), etc. These more complex architectures are inspired on brain mechanisms, CNNs are inspired in visual cortex, attention mechanisms (used in transformers and RNNs) are inspired in how the brain put attention to some specific part of a sentence or image, etc.

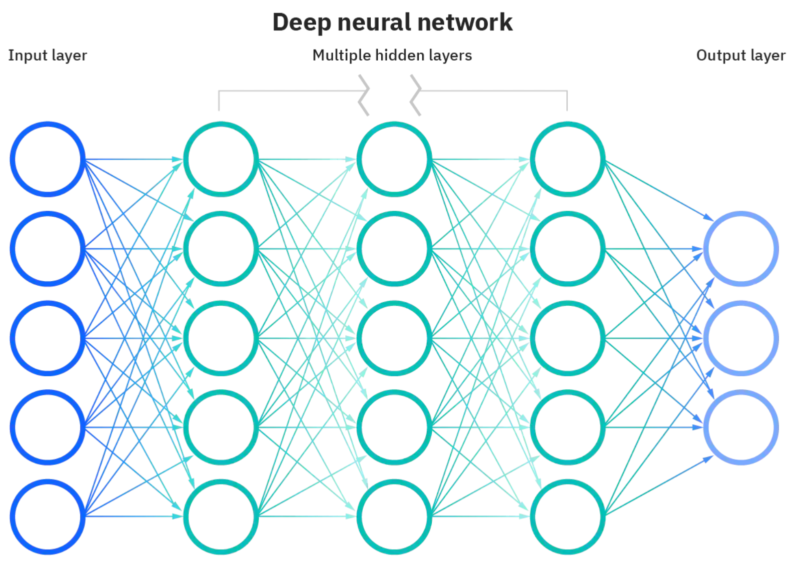


Figure 2.1: Simple deep neural network architecture. [3]

In this section two of the most used architectures for DNNs nowadays are going to be briefly explained. These are CNNs and Trasnformers. This section aims to give an idea about how these networks operate because the majority of the analysed DNN are of one of these two types. It is not pretended to give a full explanation with details, because there are a lot of details, given that these are very complex networks.

2.3.1. Convolutional Neural Networks

A kernel or convolution matrix is a matrix which is used to perform a convolution between a kernel and an image fragment (set of pixels represented as a matrix). Convolution in image processing is the process of adding each pixel of the image to its local neighbours, weighted by the kernel. This can be seen clearly in an example, see Figure 2.2, the value of 31 is obtained as shown in Equation 2.1. Applying this operation to all pixels of an image gives as a result a new image when each pixel is the result of applying the kernel to a region of the original image.

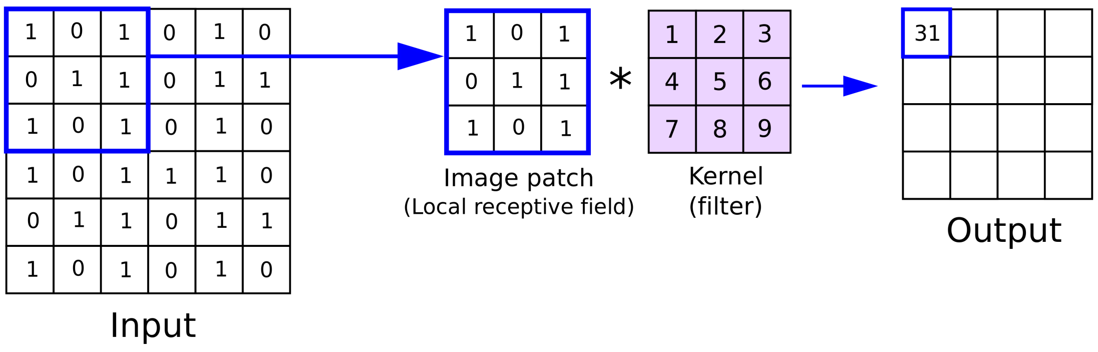


Figure 2.2: Illustration of how a convolution is performed. [4]

$$1 \cdot 1 + 0 \cdot 1 + 1 \cdot 3 + 0 \cdot 4 + 1 \cdot 5 + 1 \cdot 6 + 1 \cdot 7 + 0 \cdot 8 + 1 \cdot 9 = 31 \tag{2.1}$$

Choosing the right filter allows using the convolution operation for performing useful transformations on the image. As blurring, sharpening or performing edges detection on an image. Figure 2.3 shows a Laplacian filter

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

applied to an image (this filter is used for edge detection). In this example the image is in grey scale, but when working with RGB images there is one filter for each colour channel.

The idea beyond Convolutional Neural Networks (CNNs) is to learn these filters training the network. The filters of the first layer usually extracts basic features, such as edges. The output of the first layer is passed to the next layer which detects more complex features. As deeper into the network more complex features are detected (objects, faces, etc.).

The main layers of a CNN are:

[3]Image obtained from: https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks

[4]Image obtained from: https://anhreynolds.com/blogs/cnn.html

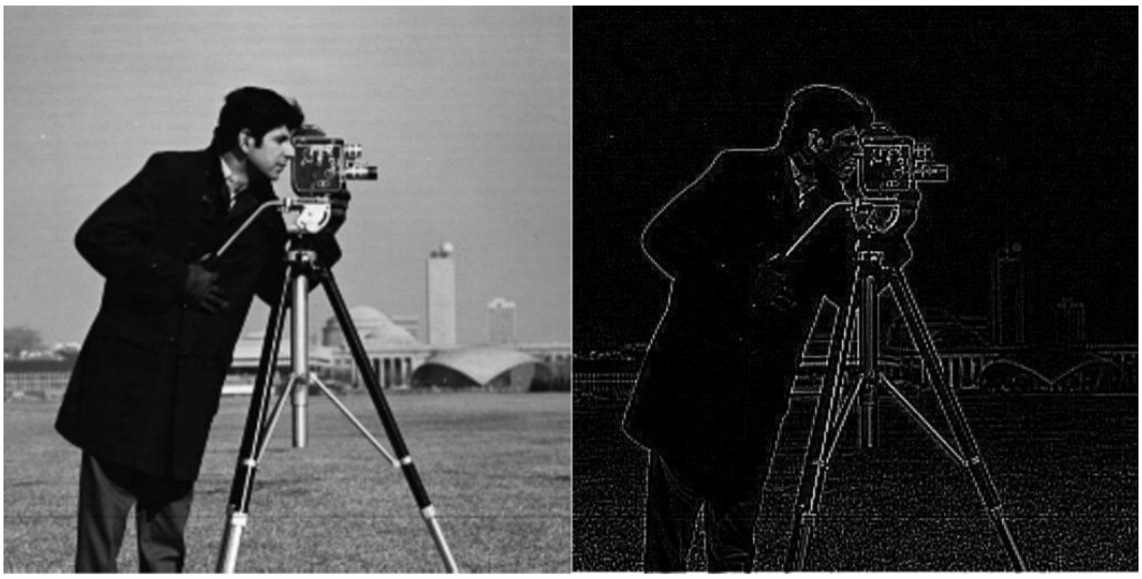


Figure 2.3: Edge detection example with Laplacian filter.

- Convolution Layer: this layers perform the previously explained convolution operations. Convolution Layers usually stack many filters, to extract many features.
- Pooling Layer: the pooling operation consists in sliding a filter over each channel of feature map and summarising the features encompassed within the region covered by the filter. This reduces required computational power to process the image because this reduces the dimensionality. Furthermore, this technique is useful for extracting dominant features which are rotational and positional invariant. There are two types pooling: max pooling (taking the max value) and average pooling (taking as value the average). A visual example of max pooling can be observed in Figure 2.4.
- Fully Connected Layers: these layers come after the block formed by the convolutional and pooling layers, fully connected layers are used to learn non-linear combinations of the high-level features, usually used to classify the image.

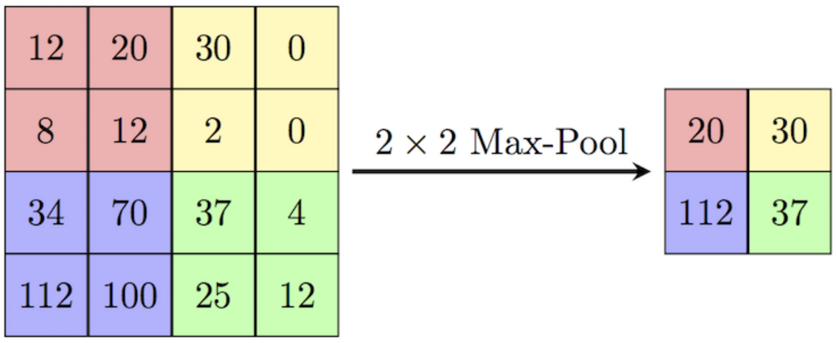


Figure 2.4: Max pooling example. [5]

Figure 2.5 shows a simple CNN sequence to classify an image.

[5]Image obtained from: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

[6]Image obtained from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

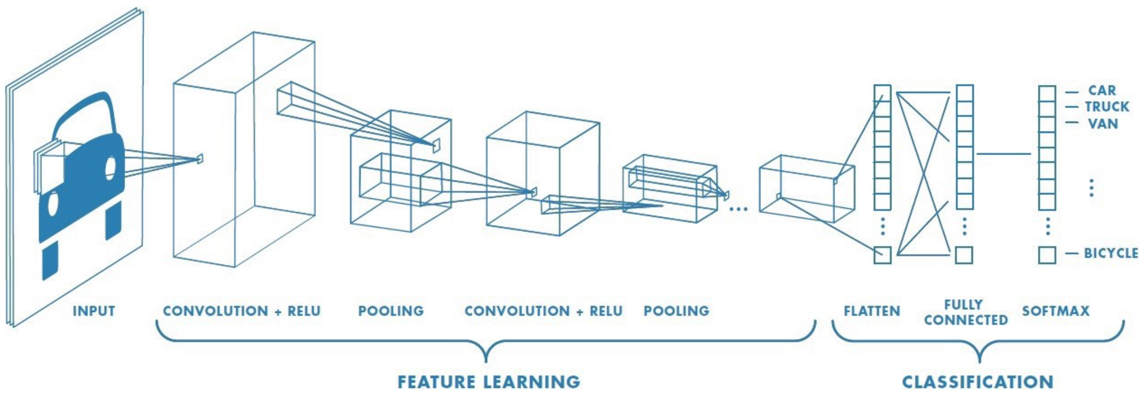


Figure 2.5: CNN example. [6]

2.3.2. Transformer

Recurrent Neural Networks (RNNs) were widely used for NLP before 2017. A RNN is a neural network where connections between nodes form a directed graph along a temporal sequence. This allows to store information of previous inputs. These networks can use information from previous inputs to influence the current input and output. They can process variable length sequences, so they are suitable for NLP, where there are variable length sentences and the context is very important (when processing a word we have to take into account the previously processed words). However this kind of networks are very slow and inefficient, especially when the input length grows.

A transformer neural network architecture was proposed in 2017 in a paper titled "Attention Is All You Need" [22]. The idea is to use encoder-decoder architecture (already used with RNN) with attention mechanism instead of using a recurrence structure. The encoder-decoder mechanism works in the following way: the encoder takes the input sequence and transforms it into a vector which represents an abstract representation of the input. That abstract vector is fed into the decoder which maps the encoded state to an output sequence. The novel part in the transformer is the implementation of the self-attention mechanism. The self-attention mechanism calculates how important other tokens of the input are for a given token. This provides context information for any position in the input sequence, and in the paper they demonstrate that with their approach it is possible to achieve SOTA results on benchmarks without using recurrence structure (so being computationally inefficient).

As transforms do not use recurrence, they do not have to process the data in order. This allows parallelisation, therefore reducing training and inference times. Since 2017, many transformer-based networks were developed for NLP, such as BERT [23] and GPT-3 [24]. Nowadays, transformers are widely used for NLP. Furthermore, the transformer architecture is being applied to more domains in the last years, such as computer vision.

2.4 Hardware for Neural Networks

Graphics Processing Units (GPUs) are the most popular DNN accelerators. A GPU is a specialised processor designed to accelerate the creation of images for output to a display device. Originally GPUs were designed to accelerate the rendering of 3D graphics. In order to render 3D graphics in real time high parallelisation is required, and GPUs began to integrate more and more cores. Note that these cores are much simpler than a CPU core. GPU cores are specialised in matrix multiplications, because rendering 3D graphics consists of matrix operations. DNNs operate with matrix too, so the GPUs, originally designed for rendering 3D graphics started to be used for DNNs because they provide higher performance and efficiency than CPUs for DNNs. Also GPUs are used for High Performance Computing (HPC).

Nvidia is one of the biggest GPU manufacturers. Nvidia GPUs are the most used GPUs for AI and HPC. They provide an API called CUDA that allows the creation of massively parallel programs. 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) [25]. With the GPUs with tensor cores it is possible to accelerate FP16 format operations through tensor cores (DNN can operate at low precision in many calculations without problems) and to combine them with FP32 precision operations if it is necessary. In this way we can benefit from higher performance, maintaining calculation precision. With each new generation the GPU's performance for AI is getting improved.

As we mentioned in the previous paragraph, DNN can operate with precision formats lower than FP32, specially for inference. Inference can be done with FP16 format in the majority of networks without the network's performance getting practically reduced [26]. Regarding training, reducing precision is more difficult, but this mixed precision could be used. This means that for the variables for which it is necessary, FP32 is used, and for the rest FP16 is used. The precision reductions for inference can go further. Making use of the quantisation technique (quantisation is the process of constraining an input from a continuous set of values to a discrete set) it is possible to use 8-bit integers instead of floating point numbers, and integer math instead of floating point math. This reduces memory and computing requirements, and therefore, energy consumption. DNNs quantisation is an efficient approach for deploying networks to embedded hardware, where high efficiency is required.

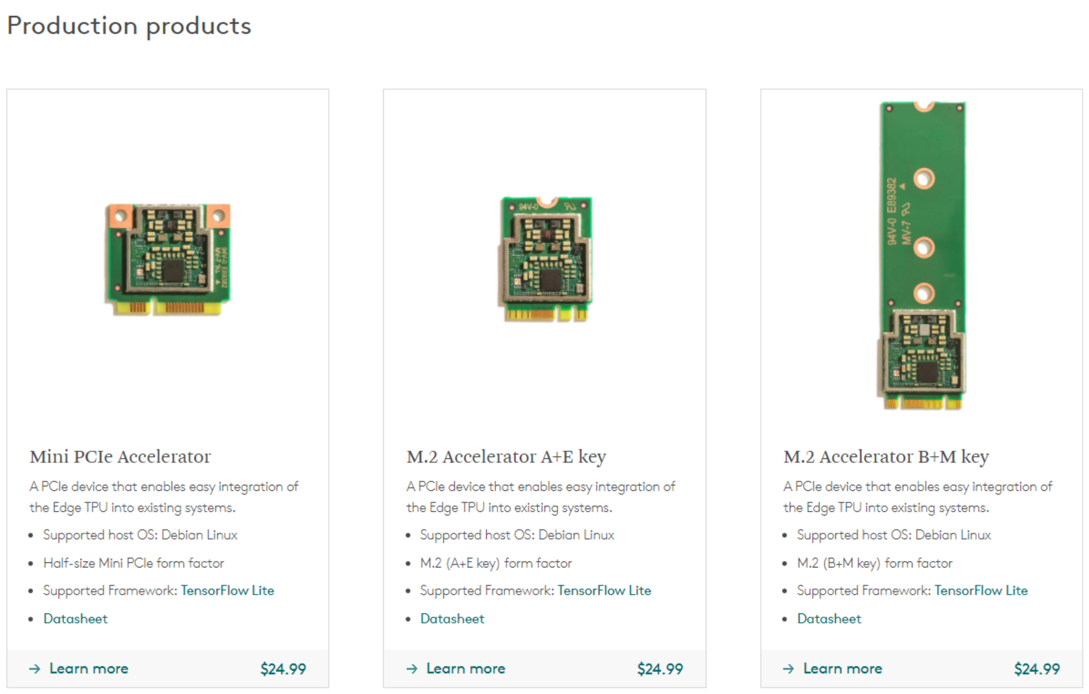
Besides GPUs, there is more hardware for AI. In fact, GPUs are not very specialised for AI, as they can perform many other HPC tasks (simulations, scientific calculations, etc), real time 3D rendering, etc. In the last years there is a lot of research about new specialiSed hardware for AI, and many application-specific integrated circuits (ASICs) have been developed by companies as Google, Amazon, Intel, Qualcomm, Nvidia and Tesla. Some examples are described below:

- Tensor Processing Unit (TPU): it is an AI accelerator ASIC developed by Google specifically for DNNs. Google began using TPUs internally in 2015, and in 2018 made them available for third party use as part of its cloud infrastructure [7]. They also offer TPU based products for edge AI. Some products are shown in Figure 2.6 (images obtained form Coral products web page [8]). As we can see there is a great variety of accelerators for edge AI. These products are designed to work with TensorFlow, an open-source library for machine learning developed by Google.

- Qualcomm Cloud AI 100 [9]: processor designed for AI inference acceleration in the cloud. The objective is to achieve high performance with low energy consumption. Also, they introduced an edge device which integrates this processor (in conjunction with a CPU and communication chip).

- AWS Inferentia Machine Learning Processor [10]: ASIC developed by Amazon Web Services (AWS) for machine learning inference. This processor is only available in AWS.

[7] https://cloud.google.com/tpu?hl=en
[8] https://coral.ai/products/
[9] https://www.qualcomm.com/products/cloud-artificial-intelligence/cloud-ai-100
[10] https://perspectives.mvdirona.com/2018/11/aws-inferentia-machine-learning-processor/



Production products

Mini PCIe Accelerator

A PCIe device that enables easy integration of the Edge TPU into existing systems.

- Supported host OS: Debian Linux
- Half-size Mini PCIe form factor
- Supported Framework: TensorFlow Lite
- Datasheet

→ Learn more $24.99

M.2 Accelerator A+E key

A PCIe device that enables easy integration of the Edge TPU into existing systems.

- Supported host OS: Debian Linux
- M.2 (A+E key) form factor
- Supported Framework: TensorFlow Lite
- Datasheet

→ Learn more $24.99

M.2 Accelerator B+M key

A PCIe device that enables easy integration of the Edge TPU into existing systems.

- Supported host OS: Debian Linux
- M.2 (B+M key) form factor
- Supported Framework: TensorFlow Lite
- Datasheet

→ Learn more $24.99

Figure 2.6: Hardware products for AI deployment.

- Smartphone processors (as Qualcomm 865 and Apple A12 Bionic) integrate AI acceleration hardware, this allows executing DNN on smartphones (usually int8 quantized networks) efficiently.

These are only a few examples, but there are many more chips for AI acceleration. These specialised chips are providing a huge improvement in performance. Overall, all this provides an improvement in hardware performance for AI much higher than the one that could be expected from the improvement of traditional CPUs.

2.5 Recent Achievements in DL

In this section we present some recent significant achievements using DNNs.

2.5.1. AlphaFold

The "protein folding problem" consists in determining a protein's 3D shape from its amino-acid sequence [27]. The 3D structure of the protein determines its functionality. That is the reason why many biology researchers are interested in studying the structure of proteins and the relationship between amino acids sequence in a protein and its 3D structure. There are experimental techniques that can show the protein 3D structure, as X-ray crystallography, but performing this experiment take a long time and is very expensive (specifically, it takes about a year and costs about $120,000 [28]). For this reason, computational methods are being studied for solving this task. For measuring the progress in this problem was created CASP [29] (Critical Assessment of Structure Prediction). CASP is a biennial community experiment to determine the state of the art in modeling protein structure. Participants are provided with amino acid sequences of target proteins, and build models of the corresponding 3D structures.

In 2018, AlhpaFold [30], an AI system developed by DeepMind, achieved in CASP13 (the 13th edition of CASP) the highest accuracy among all participants, surpassing the 2nd best result by a large margin. AlphaFold is a DL system, the main component of AlphaFold is a convolutional neural network (specifically, a ResNet based network). As a curiosity, they used 128 TPUv3 cores to train the network. Two years latter, in CASP14 DeepMind participated again with an improved version of their system, called AlhpaFold 2 [31]. The metric used for evaluating the systems is the Global Distance Test (GDT) which ranges from 0 to 100. This system achieved a median score of 92.4 GDT overall across all targets. And for the hardest protein targets (those in the free-modelling category), AlphaFold 2 achieves a median score of 87.0 GDT. A score of 90 GDT is considered to be similar to results obtained from experimental methods. The evolution of the accuracy in free-modelling category (the most difficult one) in CASP can be observed in Figure 2.7. It can be seen how AlphaFold networks represent a huge improvement.

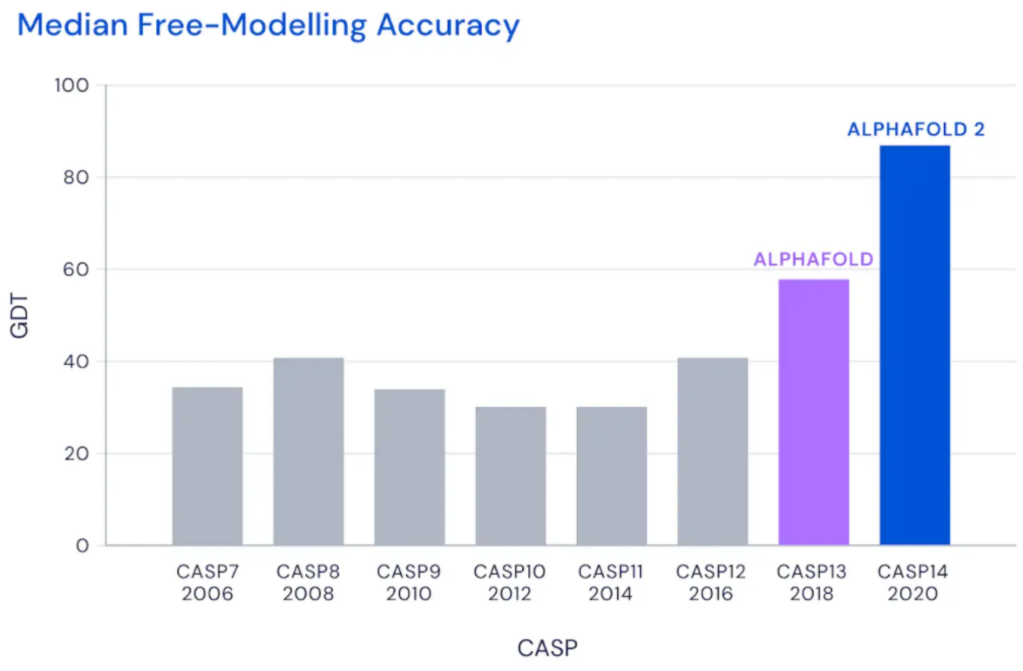


Figure 2.7: Median accuracy of predictions in the free modelling category for the best team in each CASP. [11]

Great advances in the field of biology are expected from the discovery of this system. Is expected that this approach could help to throw light on the function of the thousands of unsolved proteins in the human genome, this would help to understand many diseases and help in drug development.

2.5.2. GPT-3

GPT-3 [24] is a large transformer based DNN for NLP with 175 billion parameters. It was released in 2020 and it was the biggest non-sparse language model when released. This network does not focus on some specific NLP task (i.e., it was not fine-tuned on any dataset). It is meant to be a general NLP model which can learn to perform new NLP tasks with a few examples, which could be passed as input to the network. Humans,

[11]Extracted from: https://deepmind.com/blog/article/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology

having once acquired a general knowledge about some field, can perform a new similar tasks of that field with only a few examples or with simple instructions. GPT-3 creators try reproduce this capability for NLP, scaling up a transformed based language model. They achieved a promising result, GPT-3 is able to achieve a good performance in many different NLP tasks.

Some astonishing achievements of this model are:

- It can generate news articles which human evaluators can't distinguish from human written articles.
- Giving to GPT-3 simple instructions, it can generate code according to those instructions. We refer to giving it some simple instructions, not asking it to develop a complex application.
- It can generate a response to an email if an email and some breve indications about the response are passed as input.

CHAPTER 3
Methodology

We collect most of the information about DNNs directly from research papers that report performance 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 inspect the original paper and frequently explore 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:

- Other papers: usually the authors of a paper which introduces a new model compare it with previously existing models, giving information about these models.
- Model implementations: PyTorch [32] has a package that contains the definitions of many (pre-trained) models, and their accuracy is given. Also, there are many other projects that collect model implementations, and the model characteristics are usually reported [33] [34].
- Existing data compilations: there are some projects and public databases that collect information about deep learning architectures and their benchmarks, such as [35], [36] and [37].
- Measuring tools: if there is a public implementation of the model, we can use the ptflops library [38] or other similar tools to calculate the model's FLOPs and parameters. We use this for models for which we could not get the data, or had doubts about the veracity of the data.

Given this general methodology, we now discuss in more detail how we made the selection of models for CV, NLP. After that we discuss in detail which hardware we analyse and how we collect the data. Finally, used tools for this work are commented.

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 that the paper presenting the model is cited and how many times the model appears mentioned in other papers. We focused on model's accuracy as well because having the best models per year in terms of accuracy is necessary for analysing progress. We selected the models with higher accuracy for each year. To achieve this we used existing compilations [37] 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 (forward pass of a network is equivalent to an inference), release date and training dataset.

Accuracy and FLOPs metrics were collected carefully, taking into account that there are different sampling techniques for measuring accuracy. For instance, in the AlexNet paper [39], to classify a single image they make 10 predictions, they take 10 different crops from the original image and average the 10 predictions to get the final prediction. This is called 10-crop testing, and consists in extracting from a given image crops from four corners and the central crop plus the horizontal flipped version of these, so there are 10 crops total. The final prediction is obtained averaging the predictions made by the network on the ten patches. In Table 3.1 is possible to see the difference in accuracy between 1-crop and 10-crop strategy. While this is a useful trick, it is not fair to compare an accuracy result achieved with 10-crops with another achieved 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 (if 10 forward passes are needed to make a single prediction, then the FLOPs should be multiplied per 10). For these reasons we only report 1-crop accuracy for all models, to make a meaningful comparison.

Model	Top-1 Acc 1-crop	Top-1 Acc 10-crop
DenseNet-121 [40]	74.98	76.39
DenseNet-169 [40]	76.20	77.92
DenseNet-201 [40]	77.42	78.54
ResNet-50 [41, 42]	75.30	77.10
ResNet-101 [41, 42]	76.40	78.20
ResNet-152 [41, 42]	77.00	78.60

Table 3.1: Top-1 accuracy obtained on Imagenet with 1-crop and 10-crop

The FLOPs 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 [43] [44], and sometimes it is not clear which resolution is used for the reported result. 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 (this allows to make a meaningful analysis of how FLOPs are related to accuracy). More details about resolution are given in subsection 4.1.1.

3.2 NLP Models Data Compilation

For NLP models we noted that there is much less information about inference (inference FLOPs) and the number of models for which we can get the required information is smaller than for CV models. We looked for one benchmark that could be sufficiently representative and its value determined for a good number of architectures.

So in this case we did not make any selection of models: we just included all the models since 2017 for which we find inference compute estimation. Papers usually do not explain how they count FLOPs (as single mathematical operations or single hardware instructions), but we ultimately found out this information explained in [19]. We compare the presented numbers with estimations 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. The selected models and their values are shown in Table A.4.

3.3 Hardware Data Compilation

Regarding hardware evolution, we collected data for Nvidia GPUs. We chose Nvidia GPUs because they have been the most used GPUs for Deep Learning in the last 10 years, so we have a good temporal window for exploration and they represent one of the most efficient hardware platforms for DNN. We may have considered Google's TPUs, AWS Inferentia ASIC, and other recently developed AI accelerators for the analysis but usually there is not enough public information about them (benchmarks, power consumption), as the majority of this chips are not sold for personal use, as they are only available on the cloud. Furthermore, comparing different kind of chips performance is a difficult task, hence many accelerators are optimized for a given type of DNN or for a given framework. Because of this we choose to analyse a more general purpose accelerators, the GPUs, which still have an great efficiency for AI compared to other AI accelerators due to the new introduced features (as tensor cores). Also Nvidia GPUs are available in many cloud platforms, as in AWS [1], Microsoft Azure [2] and Google Cloud [3].

We collected GPU data for Nvidia GPUs from 2010 to 2021. The collected data is: FLOPS, power consumption (reported as Thermal Design Power, TDP) and launch date. As explained before, FLOPS is a measure of computer performance, useful in fields of scientific computations that require floating point calculations. From the FLOPS and power consumption we calculate the efficiency, dividing FLOPS by Watts, obtaining in this way FLOPS per Watt. We use TDP and peak FLOPS (the manufacturers usually report the peak FLOPS, but it is not guaranteed that in any moment these FLOPS can be achieved) 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 very common for working with DNNs.

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 learning applications. 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 DNNs 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 [4]. An example of the data of this page can be seen in Figure 3.1. We take the values of performance for the largest batch sizes,

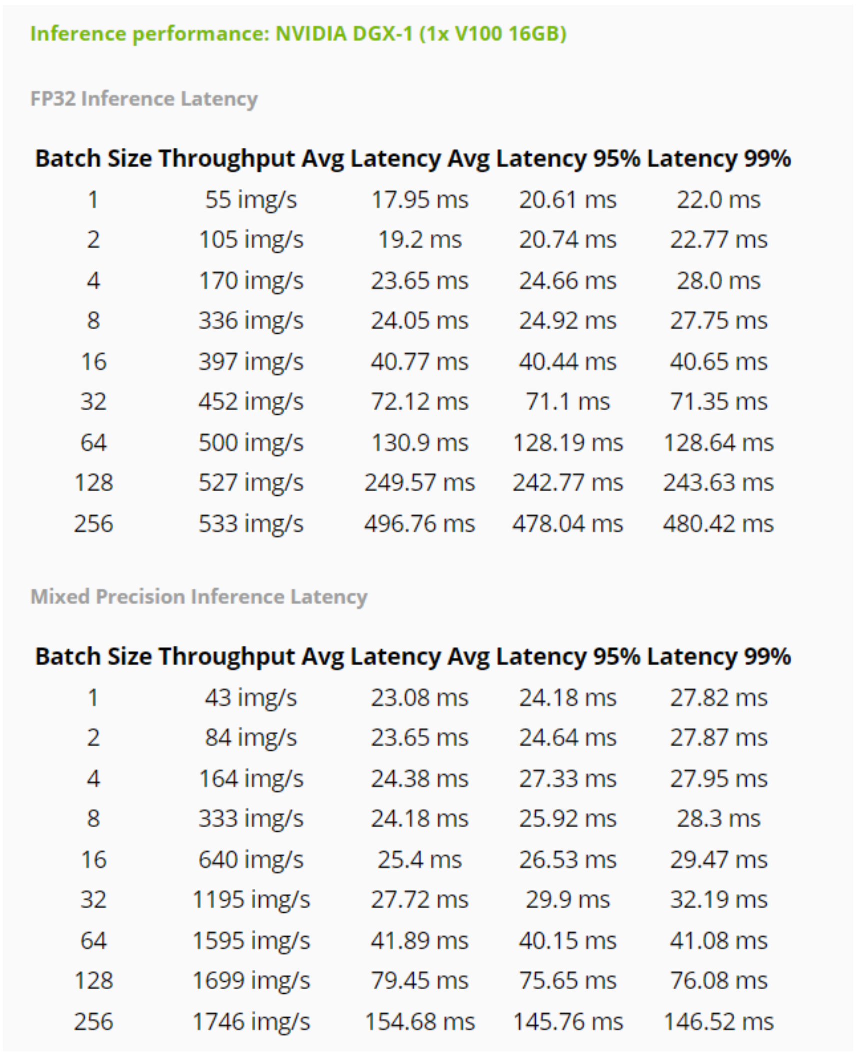
[1]Deep Learning on GPU Instances in AWS: https://aws.amazon.com/es/machine-learning/accelerate-machine-learning-P3/

[2]Inference on NVIDIA GPUs in Azure: https://techcommunity.microsoft.com/t5/azure-ai/real-time-inference-on-nvidia-gpus-in-azure-machine-learning/ba-p/1737522

[3]Google Cloud GPUs: https://cloud.google.com/gpu

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

since with small batches GPUs parallelization is not well exploited, as can be observed in throughput column.



Inference performance: NVIDIA DGX-1 (1x V100 16GB)

FP32 Inference Latency

Batch Size	Throughput	Avg Latency	Avg Latency 95%	Latency 99%
1	55 img/s	17.95 ms	20.61 ms	22.0 ms
2	105 img/s	19.2 ms	20.74 ms	22.77 ms
4	170 img/s	23.65 ms	24.66 ms	28.0 ms
8	336 img/s	24.05 ms	24.92 ms	27.75 ms
16	397 img/s	40.77 ms	40.44 ms	40.65 ms
32	452 img/s	72.12 ms	71.1 ms	71.35 ms
64	500 img/s	130.9 ms	128.19 ms	128.64 ms
128	527 img/s	249.57 ms	242.77 ms	243.63 ms
256	533 img/s	496.76 ms	478.04 ms	480.42 ms

Mixed Precision Inference Latency

Batch Size	Throughput	Avg Latency	Avg Latency 95%	Latency 99%
1	43 img/s	23.08 ms	24.18 ms	27.82 ms
2	84 img/s	23.65 ms	24.64 ms	27.87 ms
4	164 img/s	24.38 ms	27.33 ms	27.95 ms
8	333 img/s	24.18 ms	25.92 ms	28.3 ms
16	640 img/s	25.4 ms	26.53 ms	29.47 ms
32	1195 img/s	27.72 ms	29.9 ms	32.19 ms
64	1595 img/s	41.89 ms	40.15 ms	41.08 ms
128	1699 img/s	79.45 ms	75.65 ms	76.08 ms
256	1746 img/s	154.68 ms	145.76 ms	146.52 ms

Figure 3.1: Inference values example from Nvidia NGC Catalog for ResNeXt101-32x4d for PyTorch. [5]

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. We perform a different estimation for CV and for NLP networks because these two kinds of networks operate in different ways and they take different advantage of mixed precision. Experts allege that for 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 [45]. There are more benchmarks about training performance with mixed precision, as the one performed in [46] and the one available on Nvidia blogs [47]. These estimations are for training, and for this reason we perform our own estimations.

3.4 Used tools

To carry out the analysis the following tools and programming languages have been used:

[5]Extracted from: `https://ngc.nvidia.com/catalog/resources/nvidia:resnext_for_pytorch/performance`

- **R** [6]: is a language and environment for statistical computing and graphics. R provides a wide variety of statistical and graphical techniques, and is highly extensible. We use R with ggplot2 library to analyse and represent the collected data.
- **RStudio** [7]: is an Integrated Development Environment (IDE) for R.
- **Python** [8]: this programming language is used extensively for DL, we used it with some libraries as ptflops to analyse DNN, due to implementations of the networks are usually published for Python.
- **Google Colab** [9]: is a Google Research product which allows to write and execute Python code through the browser, more specifically, Colab is a hosted Jupyter notebook service. It provides an environment with many installed frameworks and packages, and we choose to execute the code there because it allows to execute the code without install all the dependencies in our local machines.

[6] https://www.r-project.org/
[7] https://www.rstudio.com/
[8] https://www.python.org/
[9] https://colab.research.google.com/

CHAPTER 4

Analysis of Inference Compute

This chapter presents an analysis of required FLOPs to perform an inference (forward-pass) for CV and NLP models.

4.1 Computer Vision

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 [39] 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. In this section, we analyse the relation between number of parameters and FLOPs and the trend of Imagenet application results (one pass) according to performance and inference compute. The collected data for the analysis presented in this section can be found in Table B.2 in the Apendix.

4.1.1. Exploring the Relation between Number of Parameters and FLOPs

The number of parameters is the most reported aspect of neural networks in literature, but in theory it is not directly proportional to the needed compute. In CNNs, convolution operations dominate the computation cost [12]. Being d network depth, w network width and r network's input resolution, the FLOPs for a CNNs grow following the relation [12]:

$$\text{FLOPs} \propto d + w^2 + r^2 \tag{4.1}$$

This means that FLOPs do not depend directly on the number of parameters. Parameters can be used to increase network depth (d) or network width (w), so distributing the same number of parameters in different ways will result in different number of FLOPs. Moreover, the resolution (r) does not depend on the number of parameters directly, because it is possible to increase the input resolution without increasing network size. Despite this, Figure 4.1 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 altogether and bigger CNNs use higher resolution. Note that recent visual transformer models [22], and these 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 0.995 (in Figure 4.1). This suggests that usually in both architectures parameters and FLOPs scale in tandem.

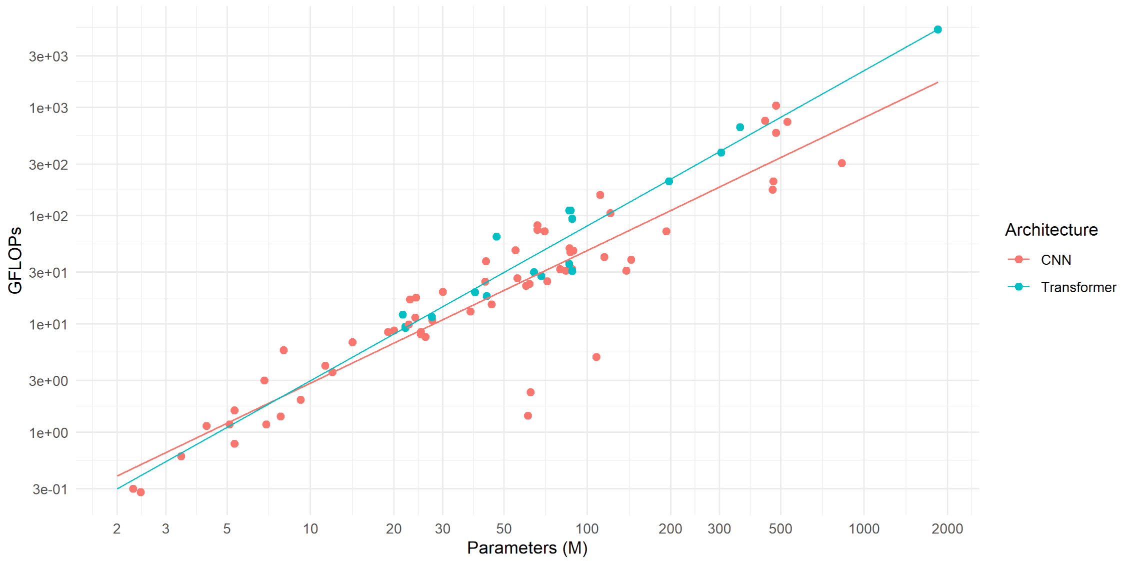


Figure 4.1: Relation between the number of parameters and FLOPs (both axes are logarithmic).

Despite the fact that FLOPs and number of parameters are highly correlated, we consider FLOPs a more useful metric for compute analysis, since FLOPs provides an estimate to the amount of compute required and this allow estimating the required energy relating hardware FLOPS with required FLOPs to perform an inference with a model.

4.1.2. Performance and Compute

Since 2012 there has been very significant progress in the accuracy achieved for Imagenet. In 2012, the AlexNet model achieved 56% Top-1 accuracy (single model, one crop). In 2021, Meta Pseudo Labels (EfficientNet-L2) model 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 model costs 1.42 GFLOPs (calculated with ptflops) while for EfficientNet-L2 we estimate that a forward pass costs 1040 GFLOPs (see subsubsection A.1.1). In Figure 4.2 we can see the evolution from 2012 to 2021 in Imagenet accuracy (with the size of the bubbles representing the FLOPs of one forward pass). We note that in recent papers some researchers began using more data than those available in Imagenet1k for training the models. The most precise models in the last years use extra data for training. Using extra data only affects training FLOPs, but does not affect the computational cost for inferring each classification (forward pass). Accordingly, we will include all these models in our study (but different symbols or colours will be used in the plots for them).

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 Figure 4.3: the dashed line represents an exponential growth (shown as a linear model 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 in the figure, which includes all points. In the same figure we see that for the same number of FLOPs we have models with increasing accuracy as time goes by. In Table 4.1 there is a collection of models with similar number of FLOPs

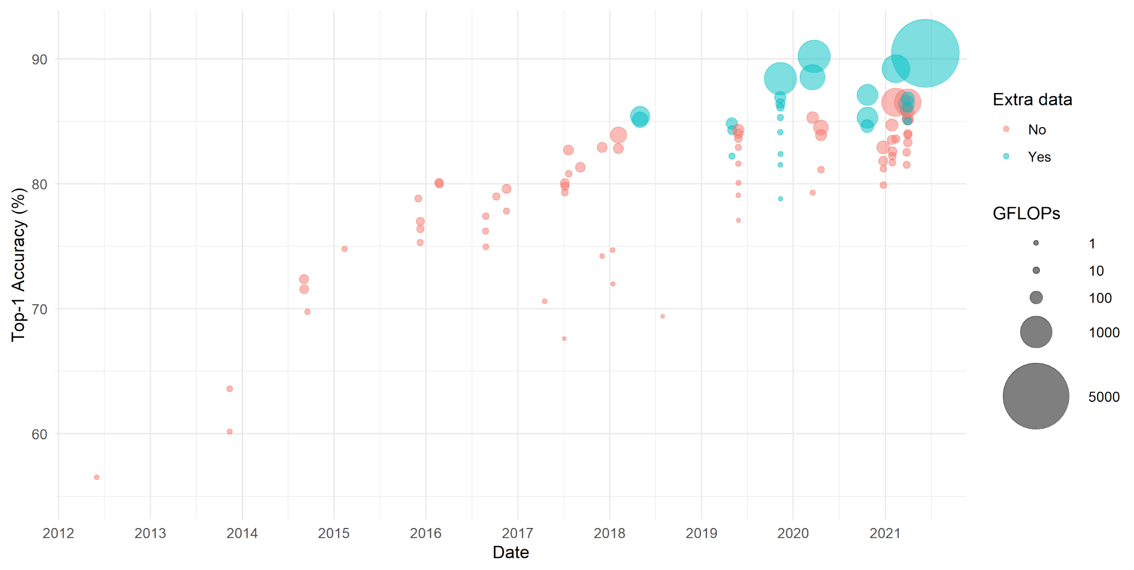


Figure 4.2: Accuracy evolution over the years. The size of the balls represent the GFLOPs of one forward pass.

to AlexNet model. In 2019 we have a model (EfficientNet-B1) with the same number of operations as AlexNet that achieves a Top-1 accuracy of 79.1% without using extra data, and a model (NoisyStudent-B1) which achieves Top-1 accuracy of 81.5% using extra data. So 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.

Model	Top-1 Accuracy	GFLOPs	Extra Data	Year
AlexNet	56.52	1.42	No	2012
ZFNet	60.21	2.34	No	2013
GoogleLeNet	69.77	3.00	No	2014
MobileNet	70.6	1.14	No	2017
MobileNetV2 1.4	74.7	1.18	No	2018
EfficientNet-B1	79.1	1.40	No	2019
NoisyStudent-B1	81.5	1.40	Yes	2019

Table 4.1: Results for several DNNs with similar number of FLOPs to AlexNet.

Figure 4.4 shows that the Pareto frontier (models with best trade-offs between high accuracy and low GFLOPs) is composed of new models (in green), whereas old models (in yellow and red) 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.

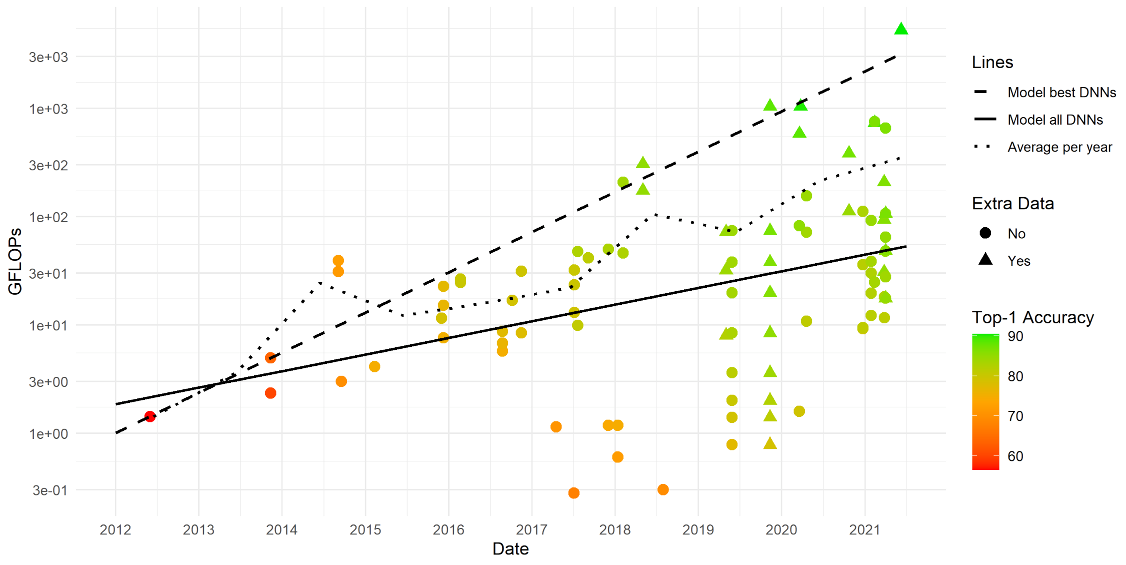


Figure 4.3: GFLOPs evolution over the years. The dashed line is a linear model (note the logarithmic y-axis) adjusted with the models with highest accuracy per year. The solid line includes all points.

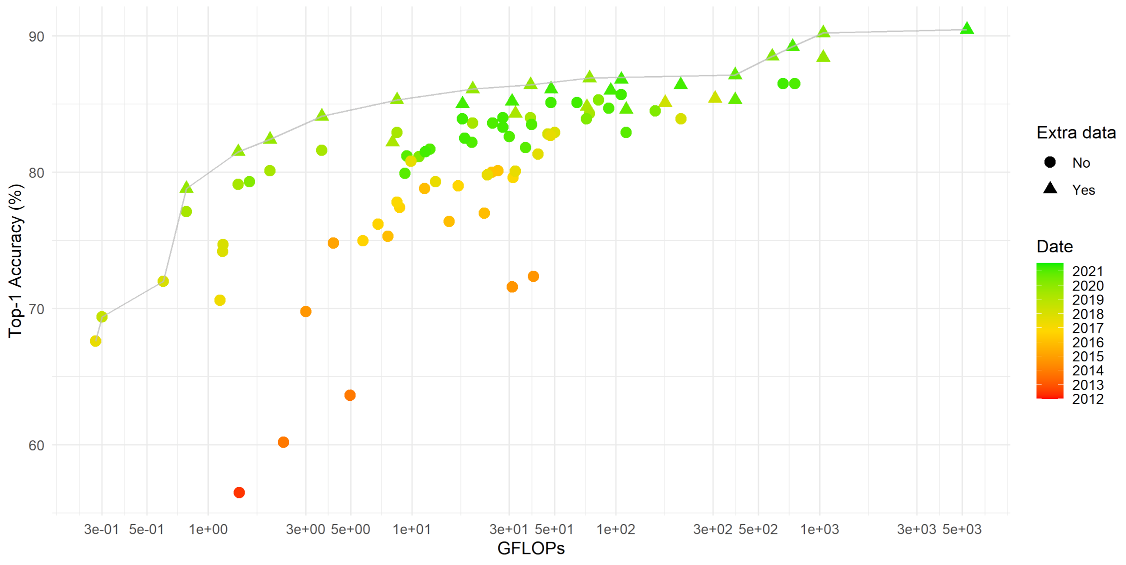


Figure 4.4: Relation between accuracy and GFLOPs.

4.1.3. Visual Transformers VS CNN

Motivated by the success of transformer architecture in the field of NLP, researchers have begun to apply transformer architectures to CV, the resultant networks are called Visual Transformers. Competitive Visual Transformers started to appear in 2020. In this section we will compare Visual Transformers and CNN in terms of accuracy and required compute. We selected the models with transformer and CNN architecture from 2019, 2020 and 2021 and we obtained the plot shown in Figure 4.5. It can be observed that CNN require less FLOPs for both datasets (Imagenet1k and Imagenet21k). However, visual transformers are a novel approach, that could be better optimized in the future.

There is another architecture for CV DNNs, usually called hybrid architecture, which, as it name suggest, is a hybrid architecture composed of both convolutional modules and self-attention modules. This architecture is showing promising results in terms of efficiency [48], it shows better efficiency than visual transformers.

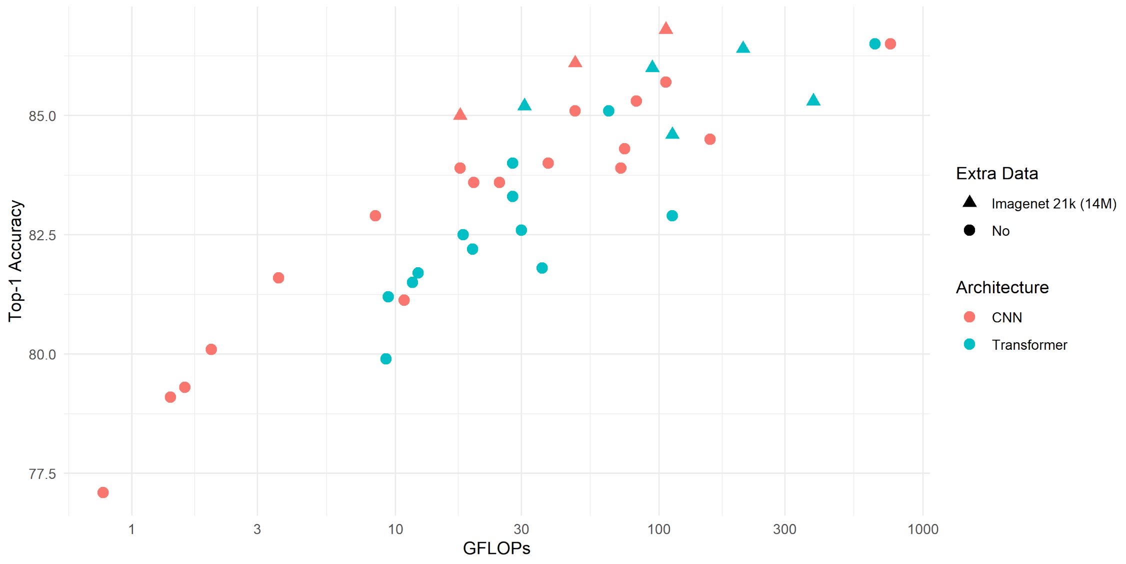


Figure 4.5: Relation between Top-1 accuracy and GFLOPs for Visual Transformer and CNN .

4.2 Natural Language Models DNNs Analysis

In this section, we analyse the trends in performance and inference compute for NLP models. There are many different tasks in NLP, such as text classification, question answering, sentiment analysis, summarisation, etc. To analyse performance we use GLUE score, which is a popular benchmark for natural language understanding, one key task in NLP. The GLUE benchmark[1] is composed of nine sentence understanding tasks, which cover a broad range of domains. The description of each task can be found in GLUE paper [4]. The collected data for the following analysis can be found in Table A.4 in the Appendixes.

4.2.1. Performance and Compute

We represent the improvement on GLUE score through the years as well as models inference GFLOPs (bubbles size) in Figure 4.6. 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.

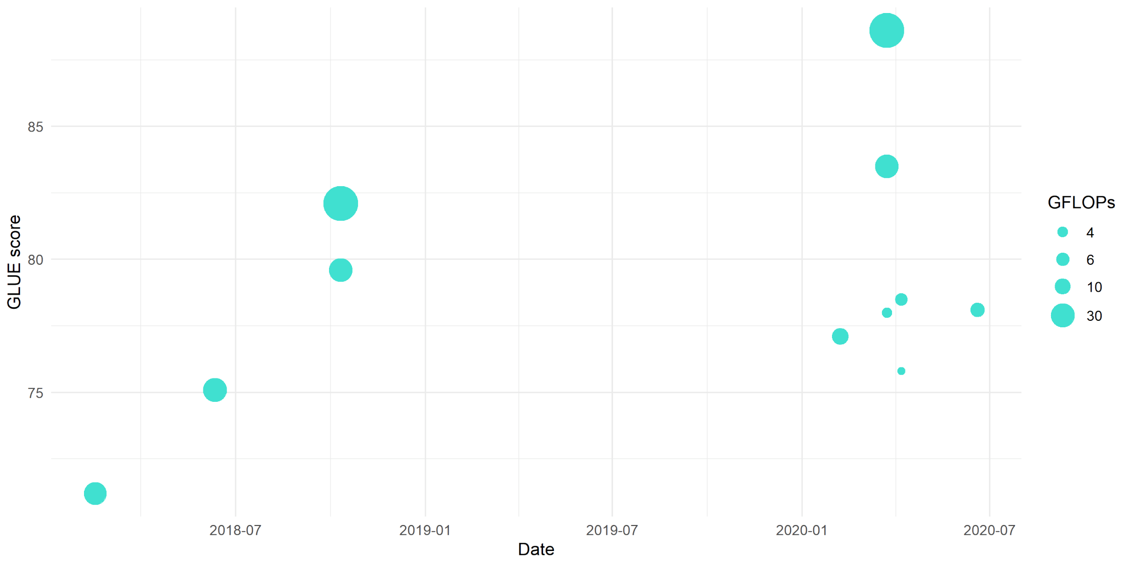


Figure 4.6: GFLOPs per token analysis for NLP models.

In Figure 4.7 we can see the GLUE score in relation to GFLOPs for an inference pass. Again, the GFLOPs are for single input of length 128. There are many models that focus on being efficient instead of reaching very high score, and this is reflected in their names too (e.g. MobileBERT and SqueezeBERT). 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.

4.2.2. Compute Trend

In Figure 4.8 we represent more models than in the previous plots. Since we do not show any benchmark punctuation, we include all models for which we found inference

[1]Many new models are evaluated on SUPERGLUE, but we choose GLUE because it allows us to have a decent temporal window for our analysis.

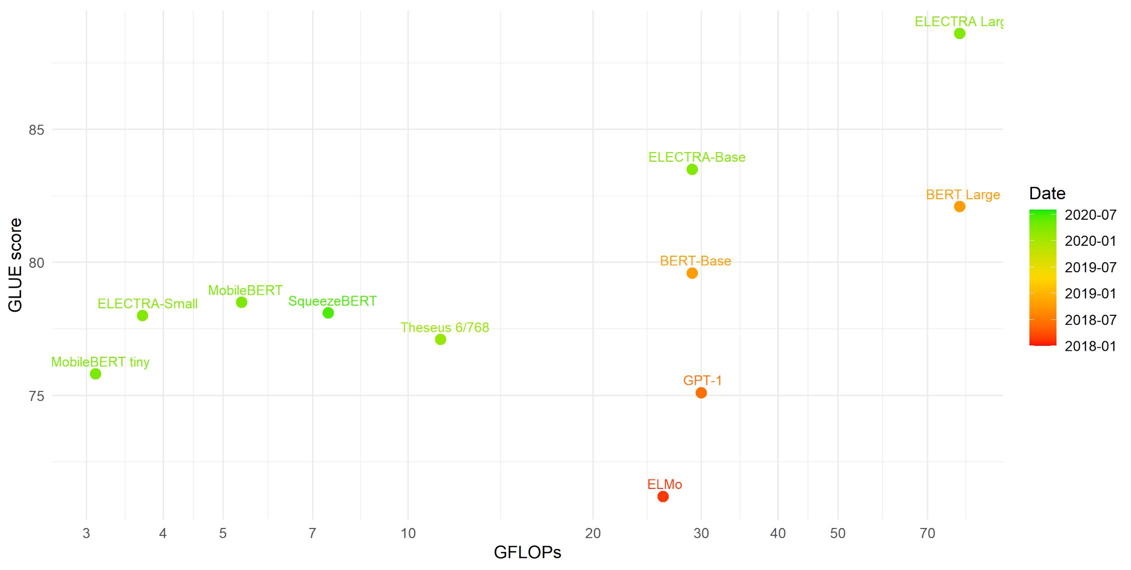


Figure 4.7: GFLOPs per token analysis for NLP models.

FLOPs estimation. The dashed line represents a model adjusted to the models with higher GFLOPs (models which when were released become the new most demanding model) and the solid line is a model adjusted to all NLP models. In this plot we indicate the input sequence length of the model, because unlike in the previous plots, in this one we represent models with different input sequence lengths. We observe a similar trend as in CV, 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.

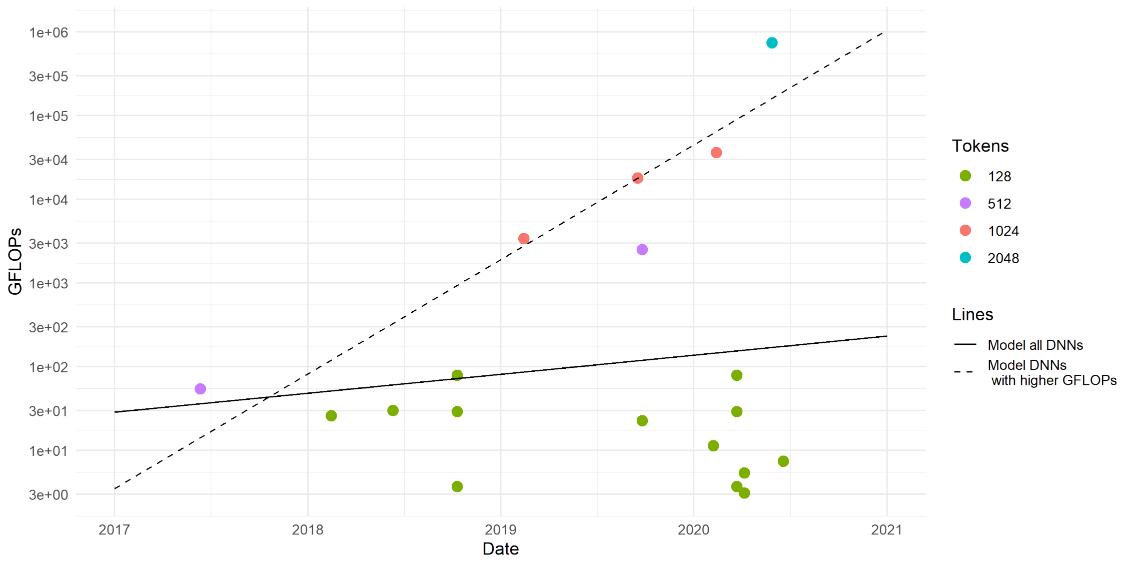


Figure 4.8: GFLOPs per token analysis for NLP models.

CHAPTER 5

Analysis of Inference Energy Consumption

In this chapter the energy consumption analysis is presented. First, hardware progress is analysed, and hardware efficiency is calculated. In the next section, the energy consumption is estimated, and the chapter finishes putting the energy consumption into context.

5.1 Hardware Progress Analysis

In this section hardware progress will be examined, we use FLOPS as a measure of hardware performance and FLOPS/Watt as a measure of hardware efficiency. We collected different precision formats performance and tensor core performance for a wide range of GPUs. The results could be observed in Figure 5.1. Note that the y-axis is in logarithmic scale. As we discussed before, theoretical FLOPS for tensor cores are very high, as can be seen in the plot. However, the performance for inference using tensor cores is not so high. For this reason we propose an estimation for three different server GPUs: V100, A100 and T4 for CV models and for NLP models. For calculating the estimations we collected inference data, the collected data can be found in the appendixes (Appendix B). The estimations for A100 are in relation to V100 because there is no data about FP32 for A100 (because FP32 is substituted by TF32, which is a precision format in between of FP32 and FP16), so we estimated the speed up to V100 FP32 FLOPS. The results of our estimation can be observed in Table 5.1.

The result of this estimations are close

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

Table 5.1: Mixed precision speed ups from experimental results for inference.

Having these estimations we fitted a linear model (with the y-axis in logarithmic scale) to each data set, one for CV and another for NLP, and we obtain good fits as shown by the solid lines in Figure 5.2. The data used for these plots can be seen in the appendixes (Table B.3). There is a point in Figure 5.2 for the year 2018 that stands out among the others by a large margin, it corresponds to the GPU T4 using mixed precision. The T4 is a

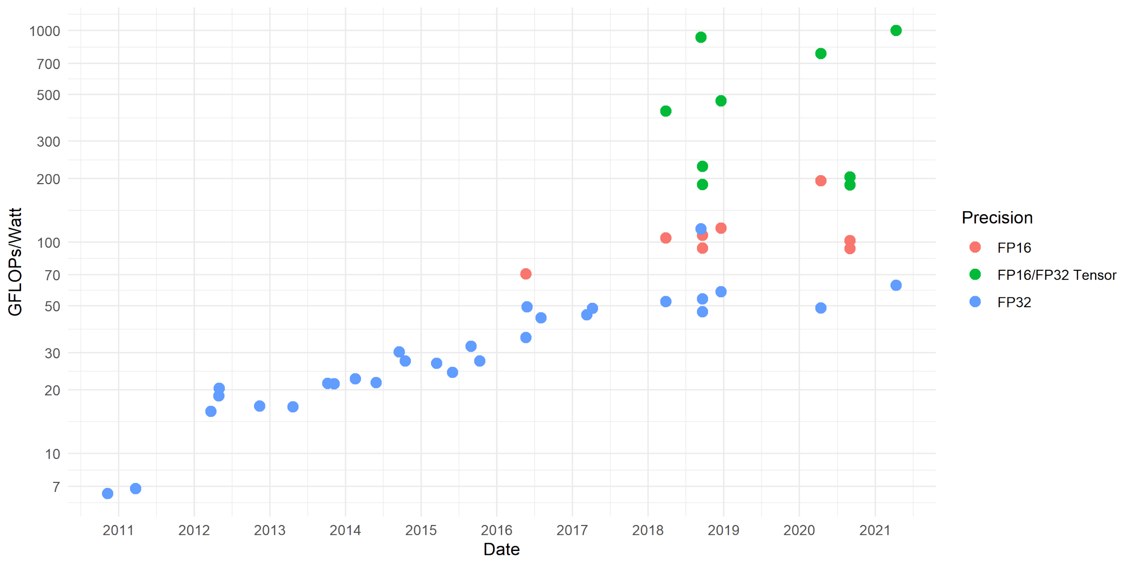


Figure 5.1: Nvidia GPUs theoretical GFLOPS per Watt.

GPU designed specially for inference, and this is the reason why it is so efficient for this task.

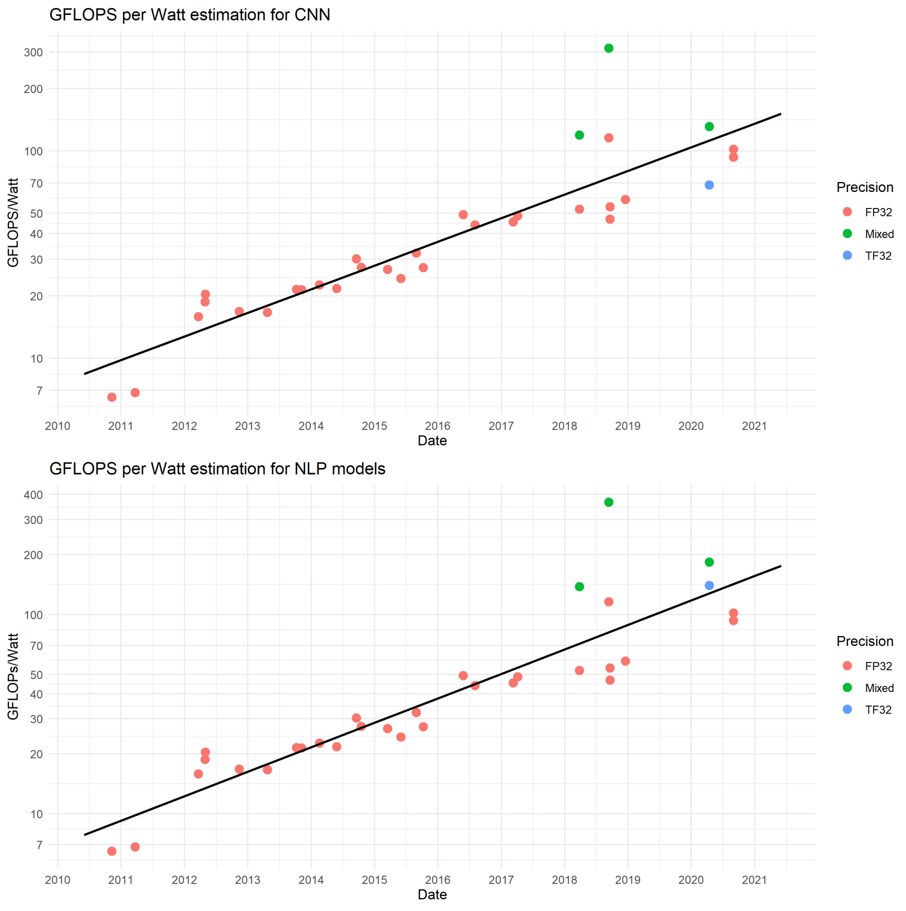


Figure 5.2: Nvidia GPUs GFLOPS per Watt adapted for CNN and NLP models.

5.2 Energy Consumption Analysis

Once we have estimated the inference FLOPs for a range of models and the GFLOPS per Watt for different GPUs, it is finally possible to estimate the energy (in Joules) consumed in one inference. To calculate this we divide 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 DNN date? In order to do this we use the models fitted to the data presented in Figure 5.2 to obtain an estimation of GLOPS per Watt *for a given date*. Namely, we use the DNN's release date to estimate the efficiency for the date when the model was released.

Knowing that 1 Watt is equal to 1 Joule per second, we can express the efficiency metric FLOPS per Watt as FLOPs per Joule, as shown in Equation 5.1. Having this equivalence we can use it to divide the FLOPs needed for a forward pass and obtain the required Joules, see Equation 5.2. Doing this operation we obtain the consumed energy in Joules.

$$\text{Efficiency} = \frac{\text{HW Perf.}}{\text{Power}} \quad \text{with units:} \quad \frac{FLOPS}{Watt} = \frac{\frac{FLOPs}{Second}}{\frac{Joules}{Second}} = \frac{FLOPs}{Joule} \tag{5.1}$$

$$\text{Energy} = \frac{\text{Fwd. Pass}}{\text{Efficiency}} \quad \text{with units:} \quad \frac{FLOPs}{\frac{FLOPs}{Joule}} = Joule \tag{5.2}$$

Applying this calculation to all collected models we obtain Figure 5.3 for CV models, the dashed line represents an exponential model (a linear fit as the axis is logarithmic), adjusted to the models with highest accuracy for each year, the solid line represents the same but for all data, and the dotted line represent the average Joules for each year, like in Figure 4.2. By comparing both plots we can see that hardware progress softens the growth observed for FLOPs in comparison with the growth for Joules, but the growth is still exponential for the models with higher 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 Figure 5.4 we do the same for NLP models and we see a similar picture.

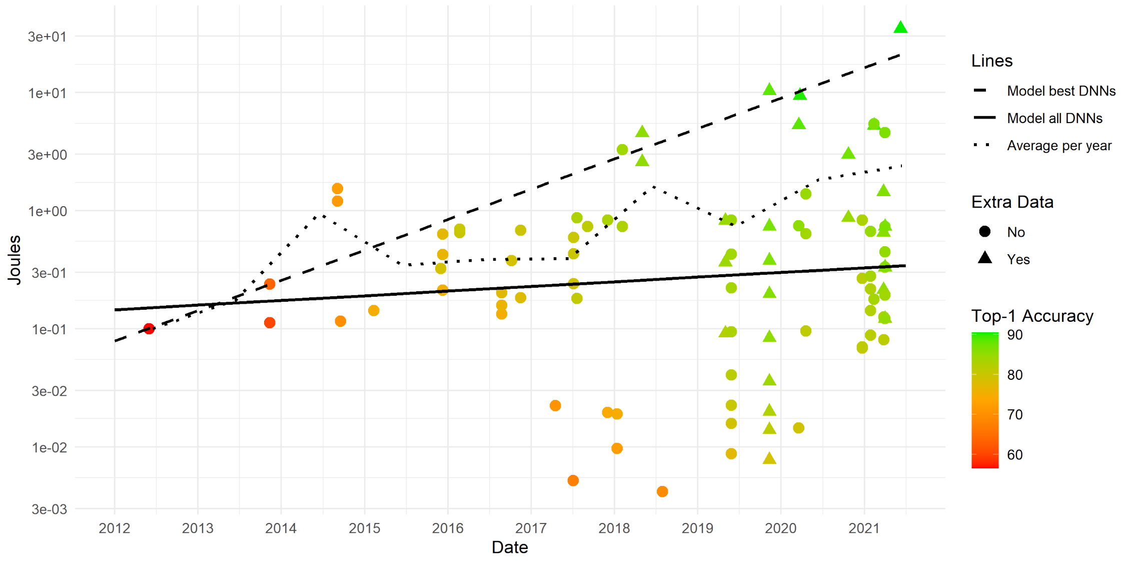


Figure 5.3: Estimated Joules of a forward pass (e.g., one prediction) for CV.

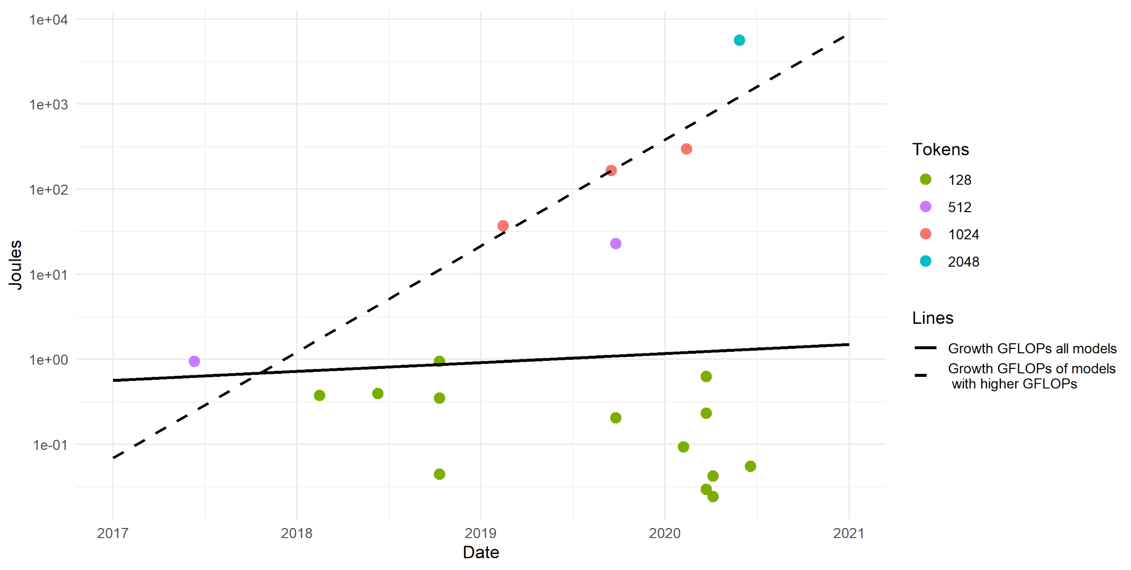


Figure 5.4: Estimated Joules of a forward pass (e.g., one prediction) for NLP.

Figure 5.5 shows the relation between Top-1 Accuracy and Joules. Joules are calculated in the same way as in Figure 5.3. The relation is similar as the observed in Figure 4.4, but in Figure 5.5 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 positio on the *y*-axis is worse than for Figure 4.4 due to the evolution in hardware. This is even more clear when we look at the corresponding plot for NLP in Figure 5.6.

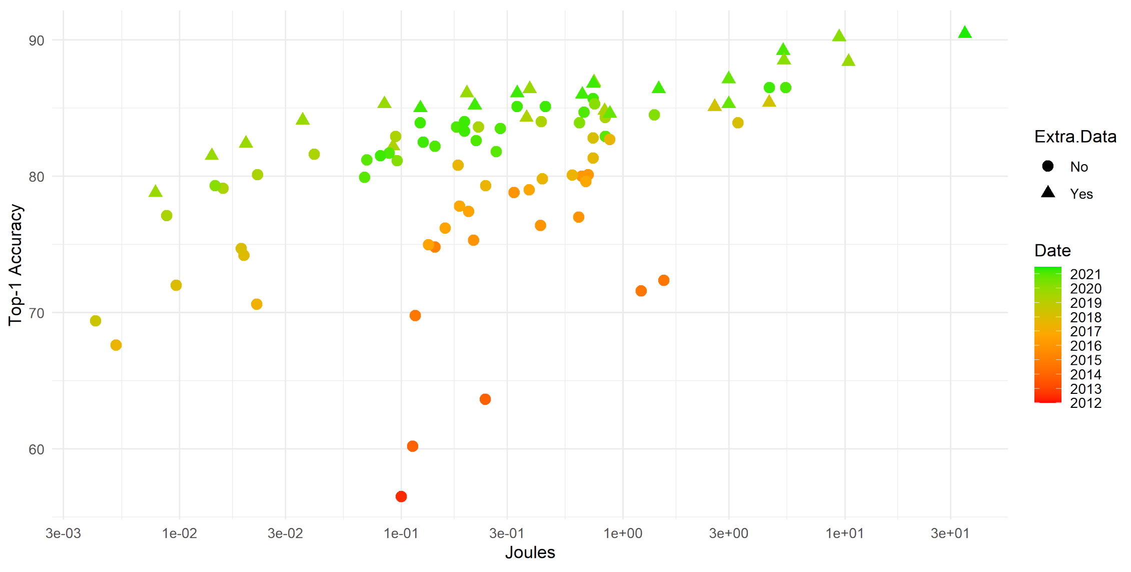


Figure 5.5: Relation between Joules and Top-1 Accuracy through the years for CV (ImageNet).

The result of adding Joules to Table 4.1 presented previously, is Table 5.2. In this table it can be observe that models with similar FLOPs to AlexNet released in 2018 and 2019 consume approximately 10 times less energy, having similar number of FLOPs. Hence, this improvement is due to hardware progress. In Table 5.3 are shown models with similar energy consumption to AlexNet, it can be observed that in 2020 and 2021 there are models with very similar energy consumption with Top-1 Accuracy above 80%. If we compare the accuracy of AlexNet with EfficientNetV2-S (note that both models are trained only with Imagenet1k, i.e., they have been trained with same amount of data),

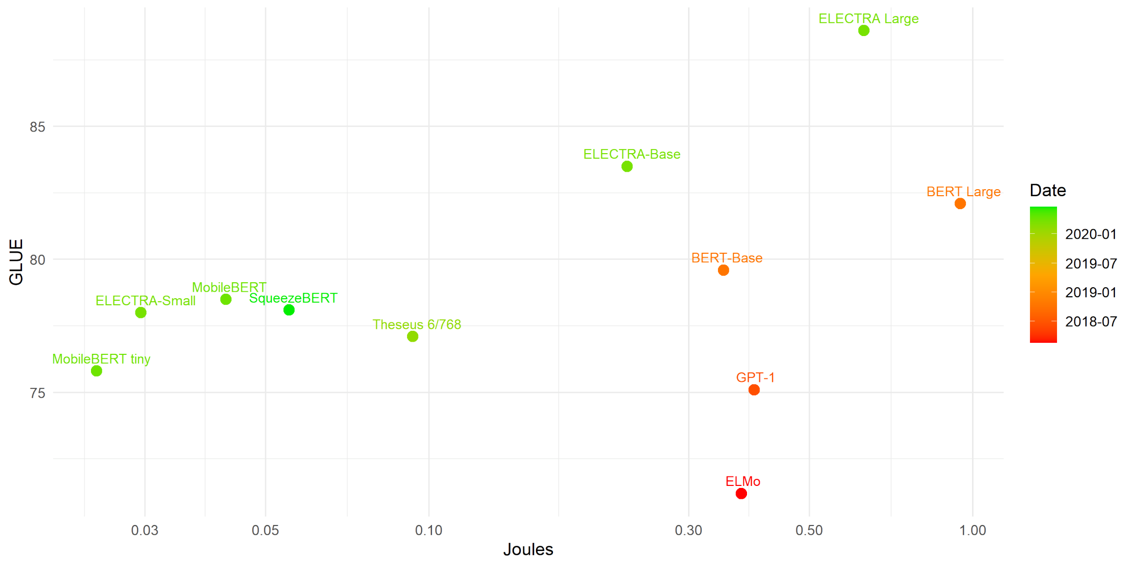


Figure 5.6: Relation between Joules and GLUE score through the years for NLP (GLUE).

we can observe an improvement of 48% in accuracy while consuming the same amount of energy.

Model	Top-1 Accuracy	GFLOPs	Joules	Extra Data	Year
AlexNet	56.52	1.42	0.100	No	2012
ZFNet	60.21	2.34	0.112	No	2013
GoogleLeNet	69.77	3.00	0.115	No	2014
MobileNet	70.6	1.14	0.022	No	2017
MobileNetV2 1.4	74.7	1.18	0.010	No	2018
EfficientNet-B1	79.1	1.40	0.016	No	2019
NoisyStudent-B1	81.5	1.40	0.014	Yes	2019

Table 5.2: Results for several DNNs with similar number of FLOPs to AlexNet.

Model	Top-1 Accuracy	Joules	Extra Data	Year
AlexNet	56.52	0.100	No	2012
ResNeXt-50 32x4d	82.20	0.092	Yes	2019
EfficientNet-B4	82.90	0.094	No	2019
NoisyStudent-B4	85.30	0.084	Yes	2019
ResNeSt-50	81.13	0.096	No	2020
T2T-ViTt-14	81.70	0.088	No	2021
CrossViT-15	81.50	0.081	No	2021
EfficientNetV2-S	83.90	0.122	No	2021

Table 5.3: Results for several DNNs with similar energy consumption to AlexNet.

5.3 Energy Consumption in Context

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 for future work. However, it is interesting to make an assumption and see how this would help to see the comparison. As an assumption, imagine that there is one inference pass of a neural network application per second per capita. What would this imply in terms of energy consumption?

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, taking this study about the USA as a reference [49]. 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 Figure 5.7. 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. Of course, this comparison is somewhat assuming that 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 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 Figure 5.7, 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 interface by human is not going to require more than the basic 128-token windows. However, many applications of language models can process data without interacting with humans as a much higher speed.

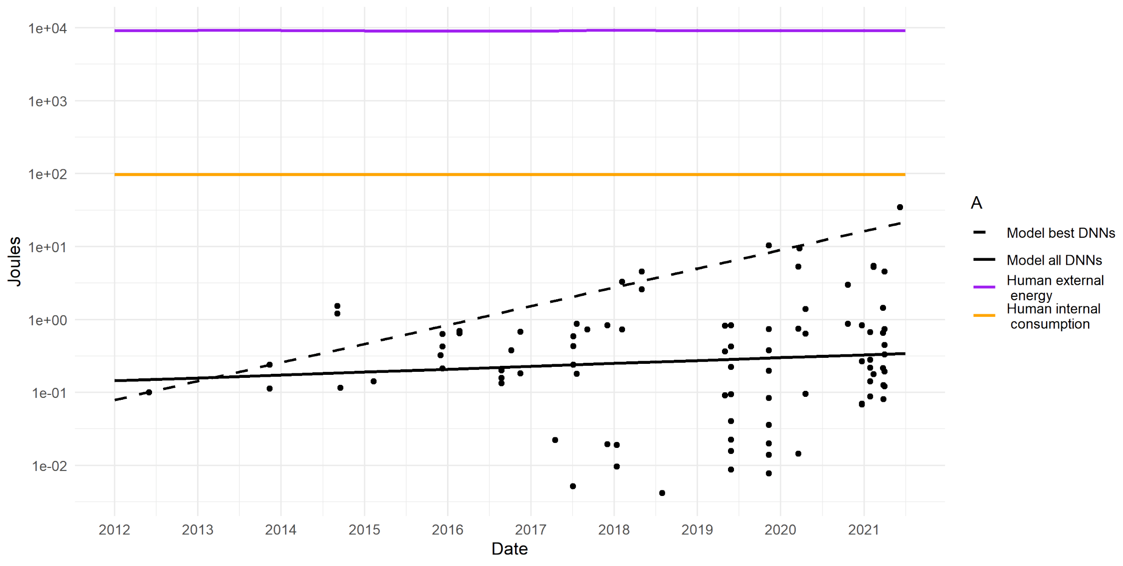


Figure 5.7: Estimated Joules per forward pass (e.g., one prediction) compared to human energy consumption in one second for CV models.

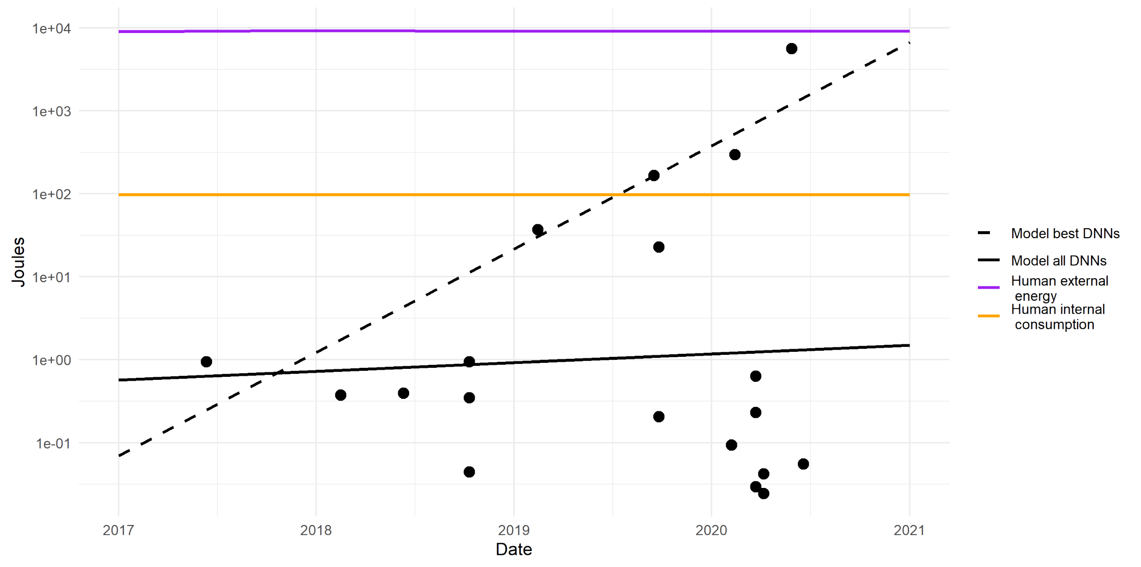


Figure 5.8: Estimated Joules per forward pass (e.g., one prediction) compared to human energy consumption in one second for NLP models.

CHAPTER 6
Conclusions

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 than the training cost. Many DNN models are trained once and applied millions of times (forward passes).

In our study we showed that inference FLOPs can be maintained constant thought the years increasing at the same time the performance, this demonstrates that better results of DNNs are in part attributable to algorithmic improvements and not only to the use of more computing power. Our main question was: "Does hardware efficiency progress cancel out the increase in inference FLOPs?". Our conclusion is that it does not cancel it out for the cutting-edge research models of each moment (which are consuming more and more energy) but this is less clear for the regular models that can be used by companies and individuals. We observe through the years that many efficient models are developed and they are much less compute and energy demanding than the cutting-edge models. As shown, DNNs as well as hardware are improving their efficiency and do not show symptoms of standstill. Consequently, despite that very large DNNs which consume a large amount of energy are build for exploring the limits of AI, the "normal sized" networks are not so energy consuming.

Finally, this work has some limitations that originate from the limited information reported in scientific papers. Many papers report the number of hyperparameters, but it is less common to have complete information about FLOPs and finding information about energy consumption is almost impossible, specially for inference. 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 old and new benchmarks, so that we can have larger windows where we can analyse the evolution of the field. We hope that future studies can build on this one and better publishing practices.

6.1 Future Work

As discussed before, although the energy consumption of DNNs could be maintained constant and still improving its performance, the total energy consumption which comes from DNNs is probably raising because of the multiplicative factor. As more and more devices use AI (locally or remotely) the energy consumption can escalate, just by means of 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 leave as future work the analysis of the increase of DNNs usage, more concretely the performed inferences with DNNs.

Also, it would be interesting to perform a wider analysis for NLP models. As we discussed, doing a comparison of NLP models is hard because NLP benchmarks are changing fast and inference FLOPs are usually not reported. We leave a deeper investigation about inference FLOPs for NLP DNNs as future work.

6.2 Relation to the Master's Degree

This project is related to several subjects of the Master's Degree. The most related subject is the one called "Data Science", in which we studied about AI and how to make use of the data to extract useful knowledge. Many concepts of data science are applied in this work (fitting models to data, representing the data, etc.). In the subject "Sistemas Inteligentes", which is translated as "Intelligent Systems" we have studied too some AI techniques. Also, the subject "Computación de Altas Prestaciones" (CAP) which can be translated as "High Performance Computing" is related to this final project. In CAP we have studied about high performance hardware and GPUs programming.

Bibliography

[1] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Superglue: A stickier benchmark for general-purpose language understanding systems," *arXiv preprint arXiv:1905.00537*, 2019.

[2] J. Park, M. Naumov, P. Basu, S. Deng, A. Kalaiah, D. Khudia, J. Law, P. Malani, A. Malevich, S. Nadathur, *et al.*, "Deep learning inference in facebook data centers: Characterization, performance optimizations and hardware implications," *arXiv preprint arXiv:1811.09886*, 2018.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[4] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," 2019. In the Proceedings of ICLR.

[5] D. Amodei and D. Hernandez, "Ai and compute.." https://openai.com/blog/ai-and-compute/, 2018.

[6] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," *arXiv preprint arXiv:2007.05558*, 2020.

[7] W. Knight, "Prepare for artificial intelligence to produce less wizardry." https://www.wired.com/story/prepare-artificial-intelligence-produce-less-wizardry/, 2020.

[8] A. Gholami, Z. Yao, S. Kim, M. W. Mahoney, and K. Keutzer, "Ai and memory wall," *RiseLab Medium Post*, 2021.

[9] D. Hernandez and T. B. Brown, "Measuring the algorithmic efficiency of neural networks," *arXiv preprint arXiv:2005.04305*, 2020.

[10] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678*, 2016.

[11] D. Li, X. Chen, M. Becchi, and 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)*, pp. 477–484, 2016.

[12] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020.

[13] M. Tan and Q. V. Le, "Efficientnetv2: Smaller models and faster training." arXiv, 2104.00298, 2021.

[14] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp." arXiv, 1906.02243, 2019.

[15] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, "Towards the systematic reporting of the energy and carbon footprints of machine learning," *Journal of Machine Learning Research*, vol. 21, no. 248, pp. 1–43, 2020.

[16] 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." arXiv, 2108.07258, 2021.

[17] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai." arXiv, 1907.10597, 2019.

[18] M. Hollemans, "How fast is my model?," 2018. https://machinethink.net/blog/how-fast-is-my-model/.

[19] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators." arXiv, 2003.10555, 2020.

[20] C. NVIDIA, *Achieved FLOPs*, 2015. https://docs.nvidia.com/gameworks/content/developertools/desktop/analysis/report/cudaexperiments/kernellevel/achievedflops.htm.

[21] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv, 1810.04805, 2019.

[24] 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, and D. Amodei, "Language models are few-shot learners." arXiv, 2005.14165, 2020.

[25] C. NVIDIA, "Nvidia tesla v100 gpu architectur," 2017. https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf.

[26] Tianzhong, "Keras float16 vs float32," 2018. https://github.com/TianzhongSong/keras-FP16-test.

[27] K. A. Dill, S. B. Ozkan, M. S. Shell, and T. R. Weikl, "The protein folding problem," *Annu. Rev. Biophys.*, vol. 37, pp. 289–316, 2008.

[28] J. Kahn, "In a major scientific breakthrough, a.i. predicts the exact shape of proteins," 2020. https://fortune.com/2020/11/30/deepmind-protein-folding-breakthrough/.

[29] A. Kryshtafovych, T. Schwede, M. Topf, K. Fidelis, and J. Moult, “Critical assessment of methods of protein structure prediction (casp)—round xiii,” *Proteins: Structure, Function, and Bioinformatics*, vol. 87, no. 12, pp. 1011–1020, 2019.

[30] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. Nelson, A. Bridgland, *et al.*, “Improved protein structure prediction using potentials from deep learning,” *Nature*, vol. 577, no. 7792, pp. 706–710, 2020.

[31] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, *et al.*, “Highly accurate protein structure prediction with alphafold,” *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[32] A. Paszke, S. Gross, S. Chintala, and G. Chanan, “Torchvision models,” 2016. https://pytorch.org/vision/stable/models.html.

[33] O. Sémery, “Computer vision models on pytorch,” 2019. https://pypi.org/project/pytorchcv/.

[34] R. Cadene, *Pretrained models for Pytorch*, 2016. https://github.com/Cadene/pretrained-models.pytorch#torchvision.

[35] S. Albanie, “Convnet burden: Estimates of memory consumption and flop counts for various convolutional neural networks.,” 2016. https://github.com/albanie/convnet-burden.

[36] A. Gholami, Z. Yao, S. Kim, M. W. Mahoney, and K. Keutzer, “Ai and memory wall,” *RiseLab Medium Post*, 2021.

[37] R. Stojnic and R. Taylor, “Papers with code imagenet benchmark (image classification),” 2021. https://paperswithcode.com/sota/image-classification-on-imagenet.

[38] V. Sovrasov, *Flops counter for convolutional networks in pytorch framework*, 2020. https://github.com/sovrasov/flops-counter.pytorch.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[40] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks.” arXiv, 1608.06993, 2018.

[41] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition.” arXiv, 1512.03385, 2015.

[42] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual networks github,” 2015. https://github.com/KaimingHe/deep-residual-networks.

[43] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition.” arXiv, 1409.1556, 2015.

[44] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, “Scaling vision transformers.” arXiv, 2106.04560, 2021.

[45] C. Li, “Openai’s gpt-3 language model: A technical overview.” https://lambdalabs.com/blog/demystifying-gpt-3, 2020.

[46] M. Balaban, “A100 vs v100 deep learning benchmarks.” https://lambdalabs.com/blog/nvidia-a100-vs-v100-benchmarks/, 2021.

[47] C. NVIDIA, "Training with mixed precision," 2018. https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html.

[48] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, *et al.*, "A survey on visual transformer," *arXiv preprint arXiv:2012.12556*, 2020.

[49] H. Ritchie and M. Roser, "Energy," *Our World in Data*, 2020. https://ourworldindata.org/energy.

[50] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10687–10698, 2020.

[51] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks." arXiv, 1311.2901, 2013.

[52] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions." arXiv, 1409.4842, 2014.

[53] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv, 1502.03167, 2015.

[54] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision." arXiv, 1512.00567, 2015.

[55] F. Chollet, "Keras applications," 2015. https://keras.io/api/applications/.

[56] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." arXiv, 1602.07261, 2016.

[57] F. Chollet, "Xception: Deep learning with depthwise separable convolutions." arXiv, 1610.02357, 2017.

[58] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks." arXiv, 1611.05431, 2017.

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

[60] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices." arXiv, 1707.01083, 2017.

[61] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks." arXiv, 1707.01629, 2017.

[62] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition." arXiv, 1707.07012, 2018.

[63] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks." arXiv, 1709.01507, 2019.

[64] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search." arXiv, 1712.00559, 2018.

[65] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks." arXiv, 1801.04381, 2019.

[66] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, “Regularized evolution for image classifier architecture search.” arXiv, 1802.01548, 2019.

[67] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, “Exploring the limits of weakly supervised pretraining.” arXiv, 1805.00932, 2018.

[68] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, “Shufflenet v2: Practical guidelines for efficient cnn architecture design.” arXiv, 1807.11164, 2018.

[69] I. Z. Yalniz, H. Jégou, K. Chen, M. Paluri, and D. Mahajan, “Billion-scale semi-supervised learning for image classification.” arXiv, 1905.00546, 2019.

[70] I. Z. Yalniz, H. Jégou, K. Chen, M. Paluri, and D. Mahajan, “Semi-supervised and semi-weakly supervised imagenet models github,” 2019. https://github.com/facebookresearch/semi-supervised-ImageNet1K-models.

[71] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou, “Fixing the train-test resolution discrepancy: Fixefficientnet.” arXiv, 2003.08237, 2020.

[72] H. Pham, Z. Dai, Q. Xie, and Q. V. Le, “Meta pseudo labels,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11557–11568, 2021.

[73] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, H. Lin, Z. Zhang, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, and A. Smola, “Resnest: Split-attention networks.” arXiv, 2004.08955, 2020.

[74] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale.” arXiv, 2010.11929, 2021.

[75] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, “Training data-efficient image transformers & distillation through attention.” arXiv, 2012.12877, 2021.

[76] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, “Deit: Data-efficient image transformers github,” 2021. https://github.com/facebookresearch/deit.

[77] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z. Jiang, F. E. Tay, J. Feng, and S. Yan, “Tokens-to-token vit: Training vision transformers from scratch on imagenet.” arXiv, 2101.11986, 2021.

[78] A. Srinivas, T.-Y. Lin, N. Parmar, J. Shlens, P. Abbeel, and A. Vaswani, “Bottleneck transformers for visual recognition.” arXiv, 2101.11605, 2021.

[79] A. Brock, S. De, S. L. Smith, and K. Simonyan, “High-performance large-scale image recognition without normalization.” arXiv, 2102.06171, 2021.

[80] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows.” arXiv, 2103.14030, 2021.

[81] C.-F. Chen, Q. Fan, and R. Panda, “Crossvit: Cross-attention multi-scale vision transformer for image classification.” arXiv, 2103.14899, 2021.

[82] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, "Going deeper with image transformers." arXiv, 2103.17239, 2021.

[83] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations." arXiv, 1802.05365, 2018.

[84] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[85] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[86] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," 2020.

[87] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," 2020.

[88] F. N. Iandola, A. E. Shaw, R. Krishna, and K. W. Keutzer, "Squeezebert: What can computer vision teach nlp about efficient neural networks?." arXiv, 2006.11316, 2020.

[89] C. Xu, W. Zhou, T. Ge, F. Wei, and M. Zhou, "Bert-of-theseus: Compressing bert by progressive module replacing." arXiv, 2002.02925, 2020.

[90] 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/.

[91] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "Mobilebert: a compact task-agnostic bert for resource-limited devices." arXiv, 2004.02984, 2020.

APPENDIX A
DNN Data Compilation

A.1 Computer Vision (CV)

A.1.1. EfficientNet Based Models FLOPs Estimation

There are many EfficientNet variations: using different input resolution or scaling further more EfficientNets. For this modifications, FLOPs many times are not reported, we estimate them following the relation presented in Equation 4.1.

NoisyStudent-L2

Having the scale factors of the networks, shown in Table A.1, we estimate NoisyStudent-L2 FLOPs as shown in Equation A.1.

Model	w	d	test res
EfficientNet-B7	2	3.1	600
EfficientNet-L2	4.3	5.3	800

Table A.1: EfficientNet models architecture specifications, obtained from [50].

$$\text{NoisyStudent-L2 FLOPs} = \text{EfficientNet-B7 FLOPs} \cdot d \text{ scaling} \cdot w \text{ scaling}^2 \cdot r \text{ scaling}^2 \tag{A.1}$$

$$d \text{ scaling} = \frac{5.3}{3.1} = 1.7097 \quad w \text{ scaling} = \frac{4.3}{2} = 2.15 \quad r \text{ scaling} = \frac{800}{600} = 1.3334 \tag{A.2}$$

Substituting the results of Equation A.2 in Equation A.1 we obtain the estimation, as shown in Equation A.3.

$$\begin{aligned} \text{NoisyStudent-L2 GFLOPs} &= 74 \text{ GFLOPs} \cdot 1.7097 \cdot 2.15^2 \cdot 1.3334^2 = \\ &= 1039.7991 \text{ GFLOPs} \approx 1040 \text{ GFLOPs} \end{aligned} \tag{A.3}$$

Meta Pseudo Labels L2

We use the estimation of NoisyStudent-L2 FLOPs for Meta Pseudo Labels L2, because it is the same model, only changes the training strategy.

FixEfficientNet-L2

In FixEfficientNet-L2 they use a resolution of 600 x 600 for testing, so the estimation is the same as for NoisyStudent-L2 but without taking into account the resolution scaling. This is represented in Equation A.4.

$$\text{FixEfficientNet-L2 GFLOPs} = 74\ \text{GFLOPs} \cdot 1.7097 \cdot 2.15^2 = 584.8285\ \text{GFLOPs} \approx 585\ \text{GFLOPs} \tag{A.4}$$

FixEfficientNet-B7

This model is the same as EfficientNet-B7 but using a slightly different resolution (632 x 632), the estimation for r scaling is calculated in Equation A.5 and the FLOPs estimation in Equation A.6.

$$r\ \text{scaling} = \frac{632}{600} = 1.0534 \tag{A.5}$$

$$\text{FixEfficientNet-B7 GFLOPs} = 74\ \text{GFLOPs} \cdot 1.0534^2 = 82.1142\ \text{GFLOPs} \approx 82\ \text{GFLOPs} \tag{A.6}$$

FixEfficientNet-B0

This model is the same as EfficientNet-B0 but using a higher resolution (320 x 320), the estimation for r scaling is calculated in Equation A.7 and the FLOPs estimation in Equation A.8.

$$r\ \text{scaling} = \frac{320}{224} = 1.4286 \tag{A.7}$$

$$\text{FixEfficientNet-B0 GFLOPs} = 0.78\ \text{GFLOPs} \cdot 1.4286^2 = 1.5919\ \text{GFLOPs} \approx 1.6\ \text{GFLOPs} \tag{A.8}$$

A.1.2. ViT-G/14 FLOPs Estimation

In the paper [44] which introduces the model, are given the GFLOPs for 224 x 224 and 384 x 384 resolutions, see Table A.2, but they comment that they use 518 × 518 resolution for ViT-G finetuning, so we suppose that they use this resolution for testing too. This is an vision transformer model, and the scale relation presented in Equation 4.1 do not apply for this kind of models, but having the GFLOPs for 224 x 224 and 384 x 384 we can calculate how GFLOPs scale with resolution, see Equation A.9. We calculate the GFLOPs ratio in Equation A.10 and we observe that GFLOPs scale quadratically with respect to resolution. Note, in this paper they report "real" FLOPs and not mult-add operations.

$$r\ \text{scaling} = \frac{384}{224} = 1.7143 \quad r\ \text{scaling}^2 = 1.7143^2 = 2.9388 \tag{A.9}$$

Model	GFLOPs	
	224 x 224	384 x 384
ViT-G/14	965.3	2859.9

Table A.2: ViT-G/14 GFLOPs.

$$r\text{ scaling} = \frac{2859.9}{965.3} = 2.9627 \tag{A.10}$$

So we calculate the r scaling in Equation A.11 and multiply the GFLOPs for 384 x 384 resolution by this scale factor (Equation A.12).

$$r\text{ scaling} = \frac{518}{384} = 1.3490 \tag{A.11}$$

$$\text{ViT-G/14 GFLOPs} = 2859.9\text{ GFLOPs} \cdot 1.3490^2 = 5269.9617\text{ GFLOPs} \approx 5270\text{ GFLOPs} \tag{A.12}$$

Model	Top-1 Acc.	Params (M)	GFLOPs	Extra Data	Date	Architecture
AlexNet [39]	56.52 [32]	61.00 †	1.42 †	No	01/06/2012	CNN
ZFNet-b [51]	63.63 [33]	107.63 [33]	4.96 [33]	No	11/11/2013	CNN
ZFNet [51]	60.21 [33]	62.36 [33]	2.34 [33]	No	12/11/2013	CNN
VGG-19 [43]	72.37 [32]	144.00	39.34 †	No	04/09/2014	CNN
VGG-16 [43]	71.59 [32]	138.00	31.00 †	No	04/09/2014	CNN
Inception V1/GoogleLeNet [52]	69.77 [32]	6.80	3.00	No	17/09/2014	CNN
Inception V2/Incepton BN [53]	74.80	11.29 [33]	4.10 [33]	No	11/02/2015	CNN
Inception V3 [54]	78.80	23.83	11.48	No	02/12/2015	CNN
ResNet-50 [41]	75.30 [42]	26.00 [55]	7.60	No	10/12/2015	CNN
ResNet-101 [41]	76.40 [42]	45.00 [55]	15.20	No	10/12/2015	CNN
ResNet-152 [41]	77.00 [42]	60.00 [55]	22.60	No	10/12/2015	CNN
Inception V4 [56]	80.00	42.68 [33]	24.60 [33]	No	23/02/2016	CNN
Inception ResNet V2 [56]	80.10	55.84 [33]	26.38 [33]	No	23/02/2016	CNN
Densenet-121 [40]	74.98	7.98 [33]	5.74 [33]	No	25/08/2016	CNN
Densenet-169 [40]	76.20	14.15 [33]	6.80 [33]	No	25/08/2016	CNN
Densenet-201 [40]	77.42	20.01 [33]	8.68 [33]	No	25/08/2016	CNN
Xception [57]	79.00	22.86	16.80 [33]	No	07/10/2016	CNN
ResNeXt-50 (32x4d) [58]	77.80	25.00	8.40	No	16/11/2016	CNN
ResNeXt-101 (64x4d) [58]	79.60	83.46	31.20 †	No	16/11/2016	CNN
MobileNet [59]	70.60	4.20	1.14	No	17/04/2017	CNN
ShuffleNet x1.0 (g=8) [60]	67.60	2.43 [33]	0.28	No	04/07/2017	CNN
DPN-131 (40 × 4d) [61]	80.07	79.50	32.00	No	06/07/2017	CNN
DPN-98 (40 × 4d) [61]	79.80	61.70	23.40	No	06/07/2017	CNN
DPN-92 (32 × 3d) [61]	79.30	37.80	13.00	No	06/07/2017	CNN
NASNet-A (6 @ 4032) [62]	82.70	88.90	47.60	No	21/07/2017	CNN
NASNet-A (7 @ 1920) [62]	80.80	22.60	9.86	No	21/07/2017	CNN
SENet-154 [63]	81.32	115.09 [33]	41.50 [33]	No	05/09/2017	CNN
PNASNet-5 (N = 4, F = 216) [64]	82.90	86.10	50.00	No	02/12/2017	CNN
PNASNet-5 (N = 3, F = 54) [63]	74.20	5.10	1.18	No	02/12/2017	CNN
MobileNetV2 [65]	72.00	3.40	0.60	No	13/01/2018	CNN
MobileNetV2 1.4 [65]	74.70	6.90	1.18	No	13/01/2018	CNN
AmoebaNet-A (N=6, F=190) [66]	82.80	86.70	46.20	No	05/02/2018	CNN
AmoebaNet-A (N=6, F=448) [66]	83.90	469.00	208.00	No	05/02/2018	CNN
ResNeXt-101 32×32d [67]	85.10	466.00	174.00	Instagram 940M	02/05/2018	CNN
ResNeXt-101 32×48d [67]	85.40	829.00	306.00	Instagram 940M	02/05/2018	CNN
ShuffleNetV2 x1.0 [68]	69.40	2.28 [33]	0.30	No	30/07/2018	CNN
ResNeXt-101 32x16d [69, 70]	84.80	193.00	72.00	Custom 940M	02/05/2019	CNN
ResNeXt-101 32x8d [69, 70]	84.30	88.00	32.00	Custom 940M	02/05/2019	CNN
ResNeXt-50 32x4d [69, 70]	82.20	25.00	8.00	Custom 940M	02/05/2019	CNN
EfficientNet-B0 [12]	77.10	5.30	0.78	No	28/05/2019	CNN
EfficientNet-B1 [12]	79.10	7.80	1.40	No	28/05/2019	CNN
EfficientNet-B2 [12]	80.10	9.20	2.00	No	28/05/2019	CNN
EfficientNet-B3 [12]	81.60	12.00	3.60	No	28/05/2019	CNN
EfficientNet-B4 [12]	82.90	19.00	8.40	No	28/05/2019	CNN
EfficientNet-B5 [12]	83.60	30.00	19.80	No	28/05/2019	CNN
EfficientNet-B6 [12]	84.00	43.00	38.00	No	28/05/2019	CNN
EfficientNet-B7 [12]	84.30	66.00	74.00	No	28/05/2019	CNN
NoisyStudent-B0 [50]	78.80	5.30	0.78	JFT 300M	11/11/2019	CNN
NoisyStudent-B1 [50]	81.50	7.80	1.40	JFT 300M	11/11/2019	CNN
NoisyStudent-B2 [50]	82.40	9.20	2.00	JFT 300M	11/11/2019	CNN
NoisyStudent-B3 [50]	84.10	12.00	3.60	JFT 300M	11/11/2019	CNN
NoisyStudent-B4 [50]	85.30	19.00	8.40	JFT 300M	11/11/2019	CNN
NoisyStudent-B5 [50]	86.10	30.00	19.80	JFT 300M	11/11/2019	CNN
NoisyStudent-B6 [50]	86.40	43.00	38.00	JFT 300M	11/11/2019	CNN
NoisyStudent-B7 [50]	86.90	66.00	74.00	JFT 300M	11/11/2019	CNN
NoisyStudent-L2 [50]	88.40	480.00	1040.00 ∗	JFT 300M	11/11/2019	CNN
FixEfficientNet-L2 [71]	88.50	480.00	585.00 ∗	JFT 300M	18/03/2020	CNN
FixEfficientNet-B7 [71]	85.30	66.00	82.00 ∗	No	18/03/2020	CNN
FixEfficientNet-B0 [71]	79.30	5.30	1.60 ∗	No	18/03/2020	CNN
Meta Pseudo Labels L2 [72]	90.20	480.00	1040.00 ∗	JFT 300M	23/03/2020	CNN
ResNeSt-269 [73]	84.50	111.00	155.8 †	No	19/04/2020	CNN
ResNeSt-200 [73]	83.90	70.00	71.56 †	No	19/04/2020	CNN
ResNeSt-50 [73]	81.13	27.50	10.78	No	19/04/2020	CNN
ViT-L/16 [74]	85.30	304.00 [13]	384.00 [13]	Imagenet 21k	22/10/2020	Transformer
ViT-L/16 [74]	87.12	304.00 [13]	384.00 [13]	JFT 300M	22/10/2020	Transformer
ViT-B/16 [74]	84.60 [13]	87.00 [13]	112.00 [13]	Imagenet 21k	22/10/2020	Transformer
DeiT-small [75, 76]	79.90	22.00	9.20 [77]	No	23/12/2020	Transformer
DeiT-small-Distilled [75, 76]	81.20	22.00	9.40 [77]	No	23/12/2020	Transformer
DeiT-base [75, 76]	81.80	86.00	36.00 [13]	No	23/12/2020	Transformer
DeiT-base-384 [75, 76]	82.90	86.00	112.00 [13]	No	23/12/2020	Transformer
BotNet-T7 [78]	84.70	75.00	92.00	No	27/01/2021	Hybrid
BotNet-T5 [78]	83.50	75.10	38.60	No	27/01/2021	Hybrid
T2T-ViTt-14 [77]	81.70	21.50	12.20	No	28/01/2021	Transformer
T2T-ViTt-19 [77]	82.20	39.20	19.60	No	28/01/2021	Transformer
T2T-ViTt-24 [77]	82.60	64.10	30.00	No	28/01/2021	Transformer
NFNet-F4+ [79]	89.20	527.00	734.00	JFT 300M	11/02/2021	CNN
NFNet-F0 [79]	83.60	71.50	24.76	No	11/02/2021	CNN
NFNet-F6+SAM [79]	86.50	438.40	754.56	No	11/02/2021	CNN
Swin-B 224 [80]	85.20	88.00	30.80	Imagenet 21k	25/03/2021	Transformer
Swin-B 384 [80]	86.00	88.00	94.00	Imagenet 21k	25/03/2021	Transformer
Swin-L [80]	86.40	197.00	207.80	Imagenet 21k	25/03/2021	Transformer
CrossViT-15 [81]	81.50	27.40	11.60	No	27/03/2021	Transformer
CrossViT-18 [81]	82.50	43.30	18.06	No	27/03/2021	Transformer
CaiT-S36 [82]	83.30	68.00	27.80	No	31/03/2021	Transformer
CaiT-S36 dist [82]	84.00	68.00	27.80	No	31/03/2021	Transformer
CaiT-S24-384 dist [82]	85.10	46.90	64.40	No	31/03/2021	Transformer
CaiT-M48-448 dist [82]	86.50	356.00	659.20	No	31/03/2021	Transformer
EfficientNetV2-S [13]	83.90	24.00	17.60	No	01/04/2021	CNN
EfficientNetV2-M [13]	85.10	55.00	48.00	No	01/04/2021	CNN
EfficientNetV2-L [13]	85.70	121.00	106.00	No	01/04/2021	CNN
EfficientNetV2-S [13]	85.00	24.00	17.60	Imagenet 21k	01/04/2021	CNN
EfficientNetV2-M [13]	86.10	55.00	48.00	Imagenet 21k	01/04/2021	CNN
EfficientNetV2-L [13]	86.80	121.00	106.00	Imagenet 21k	01/04/2021	CNN
ViT-G/14 [44]	90.45	1843.00	5270.00 ∗	JFT 3B	08/06/2021	Transformer

Table A.3: CV models data set. If there is 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.

A.2 Natural Language Processing (NLP)

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

We do not found BERT Large inference GFLOPs, but we have ELECTRA Large GFLOPs and this is the same model but trained with another strategy, so we use ELECTRA Large GFLOPs as BERT Large GFLOPs. For ELMo we take GLUE dev set score because we do not found the score on test set, but this score should be close to the score with test set.

Model	Input Tokens	GFLOPs	Params (M)	Date	GLUE test set
Transformer [22]	512	54 [36]	65	12/06/2017	-
ELMo [83]	128	26 [19]	96	15/02/2018	71.2 [19] ♣
GPT-1 [84]	128	30 [19]	117	11/06/2018	75.1 [23] ♠
BERT Large [23]	128	79	335 $*$	11/10/2018	82.1 ♠
BERT-Small [23]	128	3.7 [19]	14	11/10/2018	-
BERT-Base [23]	128	29 [19]	110	11/10/2018	79.6 ♠
GPT-2 [85]	1024	3400 [36]	1500	14/02/2019	-
Megatron [86]	1024	18000 [36]	8300	17/09/2019	-
ALBERT-xxl [87]	512	2500 [36]	235	26/09/2019	-
ALBERT-base [87]	128	22.5 [88]	12	26/09/2019	-
Theseus 6/768 [89]	128	11.3 [88]	66	07/02/2020	77.1 [88]
Microsoft T-NLG [90]	1024	36000 [36]	17000	13/02/2020	-
ELECTRA Large [19]	128	79 [36]	335	23/03/2020	88.6 ♠
ELECTRA-Small [19]	128	3.7	14	23/03/2020	78 ♠
ELECTRA-Base [19]	128	29	110	23/03/2020	83.5 ♠
MobileBERT [91]	128	5.36	25.3	06/04/2020	78.5 ♠
MobileBERT tiny [91]	128	3.1	15.1	06/04/2020	75.8 ♠
GPT-3 [24]	2048	740000 [36]	175000	28/05/2020	-
SqueezeBERT [88]	128	7.42	51.1	19/06/2020	78.1

Table A.4: NLP models data set. If there is a citation next to GFLOPs value means that GFLOPs and Input Tokens values are extracted from that source, otherwise the values are from the paper (cited in 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.

APPENDIX B

GPUs Data Compilation

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
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
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
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 B.1: GPUs specification compilation with GFLOPS per Watt calculation.

The data in Table B.2 is obtained from NVIDIA NGC catalog [1], the data can be found in this web page searching for models name.

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

Task	Model	Framework	Batch size	GPU	Presicion	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
	ResNeXt101-32x4d	PyTorch	256	V100 16GB	FP32	533	1.00
	ResNeXt101-32x4d	PyTorch	256	V100 16GB	Mixed	1746	3.28
	ResNeXt101-32x4d	PyTorch	256	T4 16GB	FP32	161	1.00
	ResNeXt101-32x4d	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-ResNeXt101-32x4d	TensorFlow	128	V100 16GB	FP32	460.82	1.00
	SE-ResNeXt101-32x4d	TensorFlow	128	V100 16GB	Mixed	1102	2.39
	SE-ResNeXt101-32x4d	TensorFlow	128	A100 40GB	TF32	802.64	1.74
	SE-ResNeXt101-32x4d	TensorFlow	128	A100 40GB	Mixed	1728.27	3.75
	SE-ResNeXt101-32x4d	TensorFlow	128	T4 16GB	FP32	105.16	1.00
	SE-ResNeXt101-32x4d	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

Table B.2: Throughput measures for V100, A100 and T4 GPUs on different models. 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.

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 B.3: GPUs throughput and power consumption data compilation.