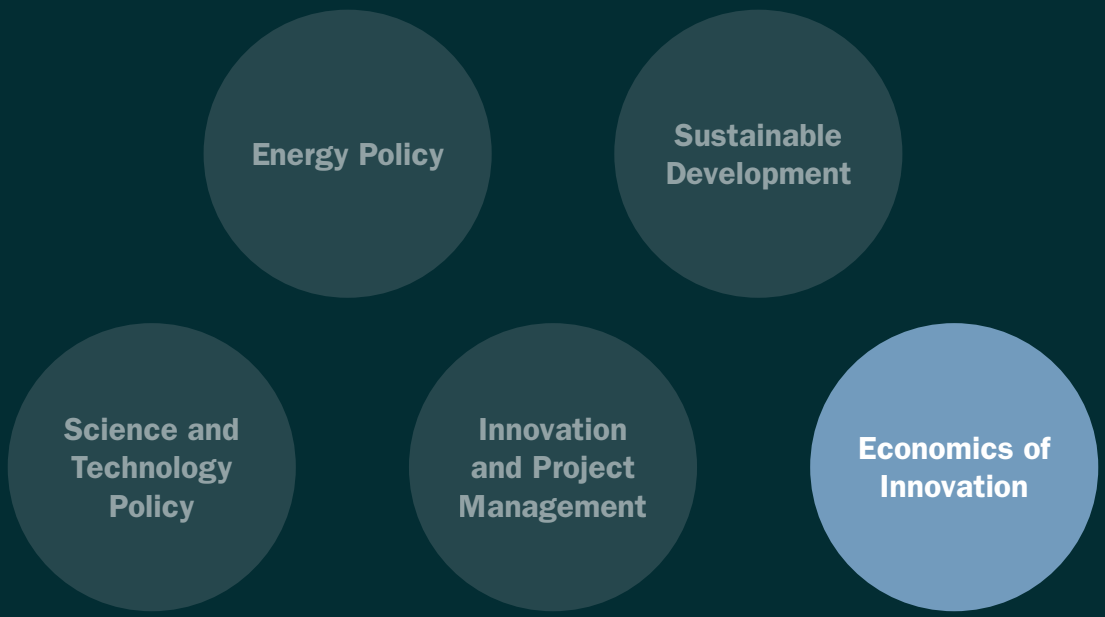
SCIENCE POLICY RESEARCH UNIT

SPRU Working Paper Series

SWPS 2020-01 (February)

On the Basis of Brain: Neural-Network-Inspired Change in General Purpose Chips

Ekaterina Prytkova and Simone Vannuccini



UNIVERSITY OF SUSSEX | BUSINESS SCHOOL

SPRU Working Paper Series (ISSN 2057-6668)

The SPRU Working Paper Series aims to accelerate the public availability of the research undertaken by SPRU-associated people, and other research that is of considerable interest within SPRU, providing access to early copies of SPRU research.

Guidelines for authors

Papers should be submitted to swps@sussex.ac.uk as a PDF or Word file. The first page should include: title, abstract, keywords, and authors' names and affiliations. The paper will be considered for publication by an Associate Editor, who may ask two referees to provide a light review. We aim to send referee reports within three weeks from submission. Authors may be requested to submit a revised version of the paper with a reply to the referees' comments to swps@sussex.ac.uk. The Editors make the final decision on the inclusion of the paper in the series. When submitting, the authors should indicate if the paper has already undergone peer-review (in other series, journals, or books), in which case the Editors may decide to skip the review process. Once the paper is included in the SWPS, the authors maintain the copyright.

Websites

UoS: **www.sussex.ac.uk/spru/research/swps**
SSRN: **www.ssrn.com/link/SPRU-RES.html**
IDEAS: **ideas.repec.org/s/sru/ssewps.html**

On the Basis of Brain: Neural–Network–Inspired Changes in General Purpose Chips

Ekaterina Prytkova[1] and Simone Vannuccini[2]

[1]Friedrich Schiller University Jena, Department of Economics and Business Administration
[2]Science Policy Research Unit, University of Sussex Business School, University of Sussex
[1]e.prytkova@uni-jena.de
[2]S.Vannuccini@sussex.ac.uk

January 24, 2021

Abstract

In this paper, we disentangle the changes that the rise of Artificial Intelligence Technologies (AITs) is inducing in the semiconductor industry. The prevailing von Neumann architecture at the core of the established technological trajectory of chip production is struggling to improve product performance for a wide range of applications, and the novel AI segment only aggravates this challenge. The revealed inefficiency of the von Neumann architecture in the execution of AI–based solutions launched innovation efforts from hardware producers in two directions: (i) to design a novel processor architecture for the needs of AI and (ii) to integrate this processor onto system–on–a–chip (SoC). Chip development is guided by three essential characteristics of chips performance — processing speed, flexibility, and energy efficiency — that at the same time are the focal points of demand's interest. We argue that flexibility, hardware's ability to execute a variety of software, remains a crucial factor of a chip's performance; an ability that novel processors for AI lack. Understanding the importance of flexibility, producers already invest in the second direction of creating heterogeneous computing systems that comprise classic and novel processors. We rationalise the unfolding situation in the semiconductor industry with a simple model formalising the mechanism of demand distribution based on a chip's characteristics. Two possible scenarios emerge from the model: (i) the emergence of a new *platform chip* appealing to a major share of demand; (ii) the failure to address a broader set of application markets with one chip leading to the fragmentation of the semiconductor industry into submarkets with dedicated chips. The convergence to one of the proposed scenarios is conditional on (i) technological progress, (ii) advances in the software domain and its compatibility with hardware, and (iii) the structure of demand's preferences.

Keywords: semiconductor industry; Artificial Intelligence; architecture; model of computation; technological trajectory

JEL Classification: L63; O31; O33

We would like to express our most sincere gratitude to Prof Edward Steinmueller who provided not only valuable insight and expertise but also relentless support that greatly improved the manuscript. We are also grateful to the anonymous reviewers from this journal, the Scientific Committee of EMAEE 2019 Conference and SPRU Working Papers Series for their positive evaluation and generous comments.

1 Introduction

The semiconductor industry has been an upstream supplier of computing devices to a wide range of market segments and during its history has faced various crises. Despite being not new to hurdles, the industry is now facing a novel, fundamental challenge: chipmakers are exploring new ways of organising computation on a chip to respond to recent breakthroughs in Artificial Intelligence (AI); AI creates a demand for computing devices directly and indirectly induces other markets that adopt AI solutions to demand changes in chips as well. A profound discrepancy resides in the mismatch between the nature of modern AI algorithms and the organisation logic of conventional hardware. Over the decades, and since its establishment as a solid field in the 1950s (McCarthy et al., 1955), AI has been developing mostly as a scientific experiment with its own successes and failures rather than a commercial technology with large potential. For this reason, AI had a small weight as an application segment for the semiconductor industry, and the discrepancy recently exposed appeared at such large scale for the first time. Nevertheless, the status of AI is changing, AI managed to gain traction and now is being experimented with in numerous markets (for example, (Agrawal et al., 2019)) so that its developers are winning the so–called 'hardware lottery' (Hooker, 2020); there occurred a swarm of new chips that embody alternative architectures capable of executing AI algorithms. Such exogenous 'shock' exerts pressure over the established technological trajectory and is poised to introduce changes in the semiconductor industry. In this paper, we analyse this technological discontinuity and how it layers up on the mechanisms and forces at work in the semiconductor industry. On the basis of this analysis we address the question of which product configuration might characterise the next phase in the semiconductor industry life cycle as a result of this shock.

Our study contributes to a number of literature strands. A first one is the nascent economics of AI (Goldfarb et al., 2019), as we analyse the impact of AI on product design and innovation–related decision making in a particular industry. A second contribution is to the literature on the economics of technological change, industrial dynamics, and systems of innovations, as we study the forces that support and contest the technological trajectory (Dosi, 1982) of chip production (Steinmueller, 1992) and the factors driving the evolution of the semiconductor industry (Malerba et al., 2008; Brown & Linden, 2011; Adams et al., 2013). A third domain we build upon is the research on platform products (Baldwin & Clark, 2000), in particular that focused on the computer industry (Bresnahan & Greenstein, 1999) and on the strategic management of semiconductor firms (Burgelman, 2002; Gawer & Henderson, 2007). A fourth strand we contribute to is the economics of network products and software as a supporting service (Church & Gandal, 1992; Chou & Shy, 1993). We build our line of argument drawing from the AI and computer science literature (Russell, 2019; Hooker, 2020) as well as from that on computation theory and integrated circuits design (Borkar & Chien, 2011).

To expose the discrepancy currently forming between capabilities of chips and their required performance, in Section 2 we put together alternative ways of organising computation in a program and the corresponding logic of hardware. This creates a framework that allows understanding the hardware and software domains and their interrelation and helps to highlight the radically different nature of Artificial Neural Networks (ANNs). In Section 3, we proceed with an overview of established and novel chip architectures and highlight their strengths and

disadvantages in application to different tasks. Comparing different architectures, the important characteristics of a chip's performance become evident: (i) processing speed, (ii) flexibility, and (iii) energy efficiency. Together, these characteristics form a trilateral technological frontier that serves as a benchmark for a chip's performance and guide design decisions. We briefly discuss several directions that chip producers can act upon by introducing improvements in chips design, and the trade–offs that might occur. We conclude that the AI shock at the moment induced two kinds of innovation efforts: (i) the design of a novel processor architecture for the needs of modern AI (especially ANNs), and (ii) the integration of this processor inside a computing system. Section 4 rationalises the unfolding situation accounting for the technological and economic factors that affect product development in the semiconductor industry. First, in Section 4.1 we introduce a stylised model of demand distribution based on the elasticity of demand with respect to hardware's flexibility (approximated with the variety of supported software) and processing speed and energy efficiency combined. Building on the analysis conducted in the previous parts, Section 4.2 outlines two scenarios for the evolution of chips and provides some arguments in support of each of them. In Section 5, we place our analysis in context by discussing how the forces and tensions we unpacked in our study align with (or differ from) those identified in related literature. Finally, Section 6 concludes.

2 The Computation Framework for Neural Networks

We perform a continuum of tasks with the help of computers. In the words of Baldwin & Clark (2000), "Computers are fascinating, interesting, and delightful to human beings because they are complex. Most of us are not especially intrigued by their raw speed or low cost. It is the many things computers do, and the many different ways they can be configured, that makes them interesting and useful. And it is the ability of computers to fulfill idiosyncratic, even whimsical desires [...] that causes these artifacts to surprise and delight us." In less than a century, computers have gotten firmly entwined with our lives, and computing became an ubiquitous activity. Any program that performs a task has an algorithm that in a structured manner leads to the achievement of a goal. In general, any program is a *virtual machine* that is ran on a physical machine — a computer. Basically, what computers facilitated people to do is the translation of regular tasks and activities into algorithms. Thus, if the performance of a task is a problem, an algorithm is its solution, regardless of the nature of a task — being it writing a document, 3D–modelling or calculating a celestial trajectory. There exist many ways of performing a task, and so do many algorithms. As a solution for a task, an algorithm can be characterized by the level of efficiency with which it achieves the goal. A first intuition would suggest time and probably memory use as inputs that an algorithm needs to deliver the result. However, to get a measure of the efficiency "it is necessary to have at hand a method of measuring the complexity of calculating devices..." (McCarthy et al., 1955, p.2). In other words, the efficiency of a task's solution should be assessed based on joint performance of an algorithm (software) and the device on which the computation occurs (hardware); the design of hardware can take over part of task's complexity so that algorithm remains simple or vice versa. The efficiency issue applies to any algorithm–device tandem and its importance grows together with complexity of a task. This fundamental complementarity between the hardware

and software domains is key to understanding the impact that Artificial Intelligence Technologies (AITs) can have on chips.

Programming Paradigms. Algorithms can approach a task in different manners, called programming paradigms. A paradigm conveys the organisation logic of computations and their execution. There are many programming paradigms — probabilistic, event–driven, automata–based, etc. — but in our analysis we employ two of them as they represent higher–level abstraction approaches to achieving a given task's goal. The first one is the *imperative* or *procedural* programming paradigm, that is concerned with the control over the flow of algorithmic instructions that lead to a desirable outcome. Thereby, an imperative algorithm is an explicit algorithm. The second programming paradigm is *declarative*, that specifies the desirable outcome but not the procedure that leads to it; hence, the algorithm can be implicit. The two approaches exhibit different level of efficiency when applied to different tasks. To illustrate this statement, we consider two examples: the first one is a simple arithmetic task of the kind 'get 8 using only 2s and basic arithmetic operations'; the second is a task of object detection in an image.

In the first task, when the arithmetic rules are well–defined the correct solution can be obtained easily with an explicit imperative algorithm. Now let's imagine that the arithmetic rules are unknown and hence an explicit algorithm as well. Thus, a program can, for example, add before multiplying. In this case there are multiple answers (and the more numbers involved, the more answers are possible). A declarative approach to the task by setting a specific number as an answer would deem other answers incorrect and hence narrow down the set of solutions (i.e. algorithms) to the ones that lead to the correct answer. This approach won't necessarily infer arithmetic rules but can approximate them. Obviously, for this task the imperative approach is much more efficient than the declarative one, as it provides a unique and correct answer in explicit steps.

Now consider a problem of object detection in an image in the context of autonomous driving. To classify an object, for example, as a pedestrian, it is necessary to identify a minimum set of features that characterises it, codify these features and their variation, and write an algorithm that evaluates the correspondence and 'decides' upon classification. In the simplest case when one feature would unilaterally identify one object, the minimum number of features to pre–program would be equal to number of objects that must be classified. Sometimes, classification can be reduced to the effective minimum of categories to distinguish by making the categories broader, for example, living creatures, mobile non–living obstacles, immobile non–living obstacles. However, the broader the category the larger the variance within a particular feature; if the feature used to classify an object as a living creature is 'presence of a head', the variety of heads' shapes, sizes and textures must be accounted to avoid misclassification into other categories. Depending on the task, the number of objects and their features can vary: more fine–grained classification is required for a high stake loss function (Russell, 2019) such as in autonomous driving. As the number of objects or/and features grows, the task of object detection quickly becomes impractical or even intractable to approach with an explicit, imperative algorithm. In contrast, the declarative approach that allows for implicit algorithms can handle this problem much better as it doesn't need to specify features and their correspondence to objects; instead, it can check whether or not the classification of an object is correct.

The comparison of the two programming paradigms on these example tasks shows two important aspects: (i) efficiency varies between approaches depending on the task to be executed, and (ii) the construction of an explicit algorithm requires some degree of certainty[1] that decreases with the complexity of a task. This is what concerns the algorithms' part of efficiency and overall computability. As pointed out earlier, the way in which computation is organized is fundamentally bound to the design of the computing hardware, and the two have implications for one another.

Models of Computation. The efficiency of a given computing technique should be estimated in connection with the device that performs it. Therefore, it is necessary to consider how the structure of a given physical device has been designed to optimise the joint performance of the device and the number of virtual machines executed on it. The theoretical concept of model of computation precedes the physical implementation of a computing device. In the theory of computation, a *model of computation* is the conceptual framework that describes how the result of an algorithm is computed given the available components of a computing device and their possible interactions. Not surprisingly, there are several models of computation implemented in hardware.

The first and dominant model implemented in the vast majority of computing devices is the *sequential model of computation*, initially proposed by Alan Turing and named after him as Turing machine. Turing's automatic machine performs computations by scanning one symbol per unit of time from an infinite tape and applying one of its finite configurations (operations) (Turing, 1937, p.231). This organisation of computation mirrors the imperative programming paradigm: control flow programs are sequences of machine instructions with tags indicating which data is needed to perform the respective instruction in a sequence. On the one hand, sequential execution allows for an immense flexibility of manipulations over data, making feasible the performance of complex algorithms, a property which Turing called *universality*. On the other hand, performing a highly complex algorithm in a sequential manner might lead to an impractically long time of execution.

The physical architecture of a computing machine corresponding to the sequential model of computation is so–called *von Neumann architecture*. Due to the property of universality the von Neumann architecture reproduces, this architecture implemented in a processor proved to be fit for the execution of vast amount of virtual machines, allowing to address a large set of tasks where the control flow logic of an algorithm is capable of achieving the goal. Put simply, explicit algorithms with stepwise instructions resemble the way humans reason, which served as inspiration for early computers. During the following decades, due to the positive reinforcement loop in optimising the design of hardware and software, the set of tasks performed on the sequential model of computation kept growing and chips with the von Neumann architecture at the core gained a foothold as the dominant design (Suárez & Utterback, 1995). The development of computers allowed applying them to increasingly complex tasks, pushing the frontiers of chips performance to keep up with speed, memory, energy efficiency and computability requirements. In the same way with programming paradigms we discussed earlier on, an identical problem can be solved on different models of computation with different efficiency up to the extreme case when

[1]Here the notion of certainty refers to the size of search space in terms of the number of (i) laws or rules that a task is subject to and (ii) objects that matter for a task.

one model of computation cannot ensure that an algorithm will convergence to the answer. When an algorithm is implicit and hardly can be expressed in the form of instructions flow sequentially changing the program's state, the efficiency of computing such algorithm on a Turing machine can decrease until it almost disappears. In this case, another model of computation can be more appropriate.

Concurrent models of computation as alternative to sequential models of computation do not focus on the order of instructions; instead, the focus is shifted to other properties of algorithm execution such as timing, parallelism or concurrency (Lee & Neuendorffer, 2005). This class of computation models is a good candidate for tasks where the algorithm is not a linear sequence of instructions but a more distributed one, for example, various instantiations of embedded software[2]. The problem with the concurrent class of models of computation is that it does not have a universal abstraction, a sort of common denominator for this class, unlike the von Neumann architecture for control flow, sequential class of models. Software that implements the concurrent computation model is an ad hoc solution for a specific hardware as opposed to the prevailing general purpose, imperative software that can be installed on any machine. This implies that chip design for the concurrent model of computation supports lower universality (heterogeneity of tasks it can execute), and initial attempts to design such circuits can be tailor–made to a specific family of algorithms and vice versa. To design and manufacture a circuit entails high costs; hence, to return the investments there should be demand from the application markets. Thus, the start of the development process of new circuits that implement the concurrent computation model depends on (i) the technical feasibility of a common abstraction, (ii) the size or/and number of markets that benefit from such hardware. For a long time concurrent models remained at the fringe of programming and the semiconductor industry, serving specialised niches like avionics and the automotive industry or functions scattered across different industries like signal processing or system modelling.

Neural Networks and AI. Everything changed when Artificial Neural Networks (ANNs) re–entered the toolkit of AI techniques.[3] Representing the dataflow programming paradigm (a subclass of declarative programming) and the eponymous model of computation (a subclass of concurrent models of computation), ANN became a revolution as it is the first program[4] that can operate as embedded software as well as conventional application software while having many distinct uses. In terms of algorithm organisation, ANNs differ significantly from classical programs. An ANN is a multi–layered directed graph. Every layer consists of nodes — instructions represented by some operations over data such as arithmetic functions, e.g. multiply–sum. Connections between nodes in different layers are dependencies between the respective instructions: every possible path in a network is, in a sense, a sequence of instructions. This logic of

[2] "Abstractions that can be used include the event–based model of Java Beans, semaphores based on Dijkstra's P/V systems [29], guarded communication [30], rendezvous, synchronous message passing, active messages [31], asynchronous message passing, streams (as in Kahn process networks[32]), dataflow (commonly used in signal and image processing), synchronous/reactive systems [6], Linda [33], and many others." (Lee, 2002)

[3] The birth of the connectionist approach to AI centered around ANNs dates back to the 1950s, with ground work of McCulloch and Pitts on neuron–like structures capable of calculations (McCulloch & Pitts, 1943) and Hebb's theory of cell–assembly formation (Hebb, 2005).

[4] We refer to a *program* as a *virtual machine*. Boden (2016, p.4) defines virtual machine as "the *information–processing system* that the programmer has in mind when writing a program". Thus, in this paper we use the term *program* in a broad sense to keep the text simple.

organising and executing computations describes a dataflow programming paradigm, where the flow of data defines which instructions to perform; when the data required for the execution of an instruction is ready, this instruction can be initiated without waiting for other independent instructions. Differently from the control flow logic realised in von Neumann architectures, where data is stable and a sequence of instructions is applied to it, in dataflow computation models instructions are stable and data floats among the instructions. There are several implications for circuit design that can be derived from this description that we discuss in the next Section.

Being inherently parallel and distributed, ANNs represent implicit algorithms where the initial network is a template and Deep Learning (DL) is the tool to establish the ANN's structure; for traditional programs, the algorithm is a sequence of instructions while for ANNs it is a network's structure of connections. The nature of ANNs perfectly fits into the declarative paradigm given the strong goal orientation and the absence of an explicit order of instructions. As already discussed, a potentially large number of tasks is hard to approach with explicit algorithms either because these algorithms are yet unclear or even if known they can be extremely inefficient solutions; the booming number of ANNs' applications confirms this statement, showing the potential of implicit algorithms. It is worth highlighting that the value of ANNs is twofold: for some tasks it increases processing speed resulting in a tremendous reduction in execution time, while for other tasks this is the only computable algorithm that can deliver result. Parallelism is necessarily present in all ANNs solutions primary as a requirement for obtaining the result rather than as an advantage in processing speed.

In general, AI is endemic to the declarative programming paradigm, with strong goal orientation and implicit, exploratory algorithms to achieve it. For example, two important instances of AI algorithms, the Logic Theorist (Newell & Simon, 1956) and ANNs both belong to the declarative paradigm despite representing two distinct approaches to AI — symbolic and connectionist respectively. The difference between the two resides in the strategy used to reduce the search space of options to converge to a goal: Logic Theorist used the rules of propositional logic to cut off irrelevant steps and navigate the convergence towards its goal — the proof of a theorem; ANNs instead use purely data–driven optimisation of a loss function. In both cases, the algorithms are exploratory on the side of *how* to achieve the specified goal. However, the guiding tool of the convergence path for Logic Theorist is logic, a formalisation of explicit reasoning, hence the inference that Logic Theorist emulated is also explicit. Indeed, the program was an attempt to prove theorems whose proof have been previously found through human explicit reasoning. Using logic as a guiding tool has its advantages, but the main problem is that "[l]ogic requires certainly, and the real world simply doesn't provide it" (Russell, 2019, p.40).

Taking stock of the discussion so far, given their many uses, ANNs have the potential to draw enough attention to the concurrent class of computational models and consequently to trigger and accelerate the development of its physical implementations. The increasing availability of data contributes to the growing applicability of modern AI. As this viable alternative to traditional programs gains traction, so does the exploration of the economic activities and new business models employing or centered around AITs. One particular transformation that is the focus of this paper is rejuvenation of the semiconductor industry technological opportunities with the arrival of ANNs. This transformation starts not only with the challenge of addressing the

technical properties of ANNs into hardware but also with the need for the industry to engage in technological and strategic foresight estimating the costs and benefits of different product configurations and industry scenarios.

3 Computational Models Shaping Hardware Architectures

As discussed in the previous Section, the architecture of a circuit is fundamentally linked to the chosen model of computation. Despite the fact that several models of computation existed on paper, the sequential one implemented in von Neumann architecture prevailed due to its universality (flexibility) and correspondence to the imperative programming paradigm. The dominance was preserved through eight technical and economic crises that forced the semiconductor industry to come up with and implement both incremental and radical innovations (Brown & Linden, 2011). The slowing down of Moore's law as the main roadmap for the industry (Flamm, 2018), as well as rising costs of design and fabrication have already influenced the industry in the past but now seem to come back. Atop of these recurring crises, the novel dataflow architecture is on the rise due to breakthroughs in AITs and threatens to fork the established technological trajectory with von Neumann architecture at the core. Instead of catering the needs of instructions flow mainly concerned with the speed up of computation, the emphasis in dataflow architectures shifts to the energy–proportional and agile data routing. By design, the two architectures have inherent advantages and disadvantages which we shall discuss in the next paragraphs where we compare types of processors that implement these architectures. The first two types — scalar and vector processors — are earlier products that represent the sequential model of computation while the last two — array and neuromorphic processors — are recent implementations of the dataflow model of computation.

3.1 An Overview of Architectures' Variety

Scalar Processors. This type of processors performs one instruction over a scalar per one clock cycle. It calls the data one scalar at a time to supply it for an instruction; then results are recorded into memory after every instruction. This architecture is realised in Central Processing Units (CPUs) and represents a physical implementation of the Turing machine with both advantages and limitations. Communication with memory for every instruction allows for the realisation of Turing's universality principle: having heterogeneous instructions (divide, multiply, AND, OR, etc.) in a sequence. However, the same feature creates the so–called von Neumann bottleneck: the transfer of data back and forth from memory for every instruction slows down processing speed (the movement along the instructions' sequence), depends on the bandwidth of the connecting channel, and significantly contributes to the energy consumption of a chip. The true concurrency or parallelism is not implemented in this architecture and can be only simulated through pipelining, a technique that allows performing concurrently a small number of instructions by processing them in a cascade (so–called instruction level parallelism).

Vector Processors. The idea of realising parallelism in computation in order to increase computing power was, however, already around since the 1970s. For example, the products of Cray Research exploited the so–called *vector processors*. A vector processor consists of a large

number of cores that are simpler than the few but more complex cores of a scalar processor. A single instruction uses a vector as a unit operand (a batch of data): an instruction fetches a vector from memory and assigns each vector component (scalar) to one of numerous cores to execute this one instruction in parallel (data level parallelism).[5] Today's Graphics Processing Units (GPUs) embody the same principle. Until mid–2000s vector processors haven't been widespread elsewhere except in supercomputers performing complex computations with large arrays of data, and later in 1990s, with the rise of computer games for the purpose of graphical rendering. GPUs consists of hundreds and even thousands of cores, however less complex and independent than CPU cores. Clearly, the coordinated work of a larger number of cores demands a higher energy consumption and the reorganisation of the computation process according to a specific programming logic; it has to deliver a low idle rate, otherwise the usage of so many cores is not justified (see Amdahl law[6]). Technically, vector processors are still von Neumann machines replicating the same principle for each of its numerous cores: instructions in a sequence can be different but this requires communication with memory for every one of them. This gives GPU the same property of universality (although significantly less than CPU) but the von Neumann bottleneck problem as well. For this reason, GPUs are very well–suited for massively parallel repetitive computations.

The conventional use of GPUs was as a discrete device on a motherboard for graphical rendering in computer games; however, with the rise of ANNs chipmakers started the integration process of GPUs into a chip's system in order to exploit GPUs as a new functional module for a broader set of calculations. The set of functional modules that include CPU, co–processor(s) like GPU, memory system, input and output units placed on a single silicon substrate constitutes a so–called System–on–a–Chip (SoC). The integration of GPUs into SoC opened up a potential to effectively include GPUs in computing processes for more general calculations, rather than just operating with graphical data. Using GPUs for general calculations was dubbed General–Purpose Computing on Graphics Processing Units (GPGPU). Bundled together in a SoC, the CPU performs orchestrating work while the GPU executes massively parallel calculations. The main principle of CPU–GPU co–working can be generically described as follows: the CPU dispatches an instruction and points at the related data to the GPU, and the GPU distributes the data across its cores in order to then perform calculations over the data in every core at the same time. The way of computing involving processors with different architectures is called *heterogeneous computing.*

Competing producers developed their own frameworks[7] allowing for heterogeneous computing. Examples are Nvidia's Compute Unified Device Architecture (CUDA) and Fusion System Architecture (FSA), started by AMD, that later transformed in Heterogeneous System Architecture (HSA) by the HSA Foundation, a consortium of companies including AMD, ARM, Qual-

[5] As an example, the multiplication of two vectors of length 20 in a scalar processor occupies roughly twenty instructions to be executed in a pipelined manner while in vector processor the same multiplication executed in parallel manner takes one instruction.

[6] Amdahl's law describes potential speedup in performance as a function of number of PUs involved (Amdahl, 1967). The exploitation of larger number of PUs represents an "enhanced" or "faster" mode of execution. Thus, said differently,"Amdahl's Law states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of time the faster mode can be used"(Hennessy & Patterson, 2011, p.39).

[7] The notion 'framework' refers to a complex of hardware, programming language and instruction set library.

comm, Samsung, etc. In 2007, when Nvidia launched CUDA for GPGPU, it was not received with much enthusiasm, certainly not enough to immediately induce a shift from conventional CPU programming to GPU programming. However, Raina et al. (2009) demonstrated the potential of GPU exploitation for ANNs' applications. They showed a method of involving GPU hardware into the training of an ANN which surpasses CPU performance in terms of time by a factor from 12 to 72 depending on the ANN's complexity. The joint success of ANNs and GPUs arrived in 2012 from the field of image recognition through the ImageNet Large Scale Visual Recognition Competition. The ANN AlexNet reached on average a 15.3% error rate in classifying 1.2 million images into 1000 categories (Krizhevsky et al., 2012). Since then, ANNs achieved substantial progress, performing some function above human level capabilities (see Eckersley et al. (2017)). Overall, the use of diverse learning techniques with ANNs opens up a path to the affordable automation of non–routine tasks or improves performance in already automated, routinized ones. Being tightly interconnected, the hardware domain has to respond to accommodate and effectively support this breakthrough in software domain that has multiple and heterogeneous applications.

In sum, comparing CPUs and GPUs, GPUs compute in parallel but are more fit for massive, regular, less sophisticated computations, consume more energy per calculation and are less conventional to program in comparison with CPUs.

Array Processors. With the advent of ANNs, GPUs acquired a number of new markets; GPUs were available at the time of ANNs' development and contributed to the breakthrough itself. It didn't take long before chip development went further. The first serious challenge for Nvidia's GPU came from Google's Tensor Processing Unit (TPU) in 2016. Google's TPU is a *matrix* (or *array*) processor which removes the von Neumann bottleneck from its cores by creating a systolic array of Data Processing Units (DPUs). A systolic array of DPUs represents a hard–wired network of homogeneous calculating units — meaning that every DPU implements the same set of operations. Once data is uploaded from the memory it travels among DPUs and is processed upon arrival within a DPU without being recorded intermediately into memory. Thus, the TPU emulates the dataflow architecture[8] introduced in Section 2. This makes TPUs faster in processing than CPUs and GPUs, performing hundreds of thousands of operations per clock cycle, but allows the execution of only regular instructions such as multiply–accumulate in ANNs (Jouppi et al., 2017).

A TPU is more energy efficient and faster in processing due to its dataflow architecture but it has the shortcoming of being less flexible or universal in computation. Furthermore, data–level parallelism realised in both GPUs and TPUs requires the representation of information in regular form of vector, matrix or array in order to effectively run programs (or parts of them) on this hardware. Google improves its TPUs continuously, issuing a new generation twice as powerful as the previous one roughly every year. However, the company has also refrained from the commercial sale of TPUs, using them only in internal services and providing access to customers through the cloud. In contrast, in early 2019, Intel unveiled Intel Nervana Neural Network Processor (NNP) containing both CPU and tensor cores.[9] More products from competitors

[8] Dataflow architecture is inherently parallel but not all parallel systems belong to the class of dataflow machines (Veen, 1986).

[9] Nervana NNP is a modified 10^{th} generation Intel Core processor (CPU) with, among other changes, replace-

followed: Eyeriss 2.0 jointly designed by MIT and Amazon (Chen et al., 2019), Hanguang 800 from Alibaba Group, Inferentia and Trainium by Amazon and many others[10]. Lastly, Huawei's Ascend 910 chip comprises all three types of processors — scalar, vector and matrix — as its functional cores within one SoC.

Neuromorphic Processors. Neuromorphic chips represent a radically different development direction of computing devices. This novel architecture mimics the impulse–based synaptic activity between neurons in the brain. Transistors wired together emulate a network of neurons with electrical synaptic connections. Information is encoded as analog signal and flows across an array of simulated neurons in form of electrical pulses. In effect, the neuromorphic architecture is an analog version of the dataflow architecture of an array processor. Examples of neuromorphic chips are TrueNorth, a joint venture of IBM and DARPA under SyNAPSE program (Merolla et al., 2014), the experimental neuromorphic chip Loihi from Intel (Davies et al., 2018) and the hybrid (CPU and network of neurons) chip Akida from BrainChip. The distinctive feature of such chips is extreme energy efficiency, which makes the neuromorphic architecture a promising competitor.

In sum, scalar processors have nearly exhausted their technological opportunities and have lost their exclusive position in computation. Nevertheless, they remain an inalienable component of a computing system. In turn, computing system are experiencing an upgrade through experimenting with novel architectures for co–processor on a SoC. At the moment, specialised processors alone do not deliver end–to–end solutions; they lack generality or flexibility and/or commonly supported and well–developed frameworks to be used for general purpose programming. CUDA for GPUs or TensorFlow for TPUs are examples of such frameworks, but they are immensely smaller than the encompassing and versatile framework created over the decades for CPUs. Overall, chips with different underlying architectures — controlflow sequential and dataflow concurrent — do not seem to be competing technologies (Arthur, 1989) but rather complements for a SoC (Baldwin & Clark, 2000). The development of a new component, its subsequent integration in and reorganisation of the computation process induces architectural innovations[11] (Henderson & Clark, 1990) in a SoC. Examples of architectural innovations include techniques of interconnection among processing units (PUs) and memory (e.g. Nvidia's NVlink network–on–a–chip and hierarchical mesh of MIT's Eyeriss) (Borkar & Chien, 2011; Winter et al., 2010; Chen et al., 2019) as well as packaging techniques of chiplets (e.g. Intel's EMIB used in Nervana NNP, TSMC's CoWoS used in Nvidia's GPUs, Swarm communication fabric from Cerebras) (Shao et al., 2019; Lie, 2019). Overall, the direction of innovation efforts seem to "expand beyond the processor core, into the whole platform on a chip, optimising the cores as well as the network and other subsystems" (Borkar & Chien, 2011, p.75)

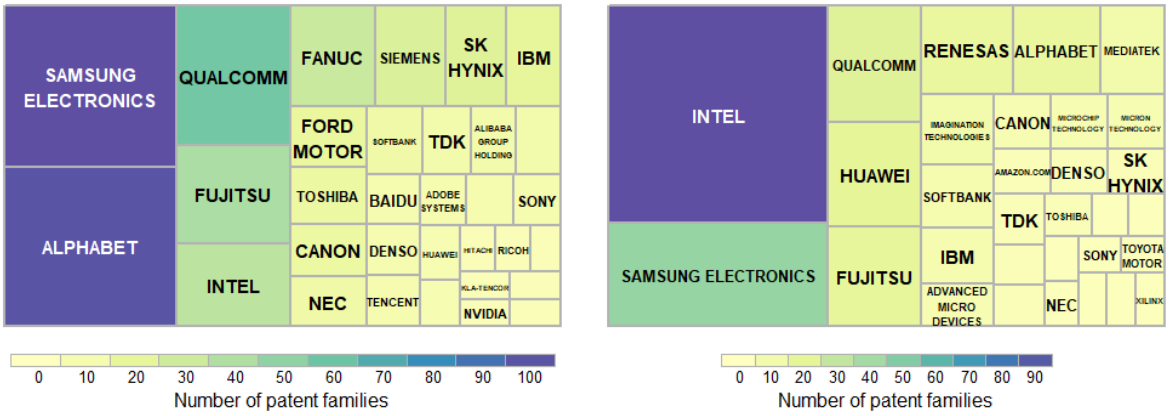
Currently, the efforts of the semiconductor industry are directed at two targets: (i) to design a novel processor architecture for the needs of AI which emerges from the class of concurrent computational models, and (ii) to integrate this processor onto a SoC. Given that (i) is in development, the first SoC resulting from (ii), though capable of supporting AI applications, are

ment of GPU cores by tensor cores.

[10] https://github.com/basicmi/AI-Chip

[11] Here we refer to the term *architectural innovation* suggested by Henderson & Clark (1990), that denotes a specific type of innovation when the core function behind a technology is preserved while the components and linkages among them undergo changes.

still early and expensive versions whose components yet fall short with respect to the performance characteristics that we discuss in the next Section 4.



(a) Dataflow **(b)** Controlflow

Figure 1: Top–30 global holders of patent families on chip's architectures 2014–2016
Data: COR&DIP©v.2 IPC classes: (a) G06N 3/02-10, (b) G06F 15/76-82

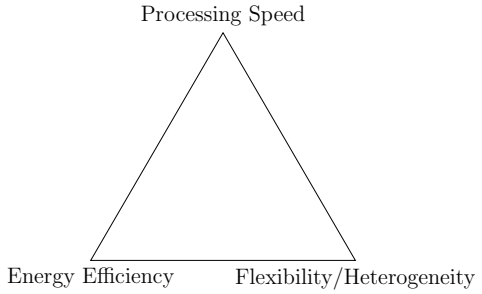
The efforts in the development of new processors for the needs of AI are already visible in the IPR system. Figure 1 shows the patenting activity of top–30 proprietors by type of architecture over the period 2014–2016. The number of patent families filed globally on dataflow architectures (Figure 1a) overcame controlflow architectures (Turing machines; Figure 1b) with a total of 786 patent families by 115 companies versus 405 from 77 firms respectively. Not surprisingly, in both categories the top positions of the ranking are occupied by large international companies such as Samsung, Alphabet, Intel, Qualcomm and Fujitsu, however with a long tail of smaller companies especially in the case of the dataflow architectures. The intensive exploration of the rich technological opportunities for processors embedding dataflow architectures by both incumbents and startup companies creates a swarm of novel and distinct products. For instance, Figure 1a includes *photonic* chips that use light to encode and transmit information instead of electricity (i.e. IPC class G06N 3/067) such as one of the programmable nanophotonic processor from Lightelligence (Shen et al., 2017).

However, there are already early signs of search for a more general or flexible dataflow architecture that might lead to a shake–out in product variety and to the emergence of a dominant design in this class of processors. Sze et al. (2020) discuss in detail the challenges and criteria for the design of Deep ANN (DNN) processors, claiming that "it is increasingly important that DNN processors support a wide range of DNN models and tasks. We can define *support* in two tiers. The first tier requires only that the hardware needs to be able to functionally support different DNN models (i.e., the DNN model can run on the hardware). The second tier requires that the hardware also maintain efficiency (i.e., high throughput and energy efficiency) across different DNN models." In sum, this statement calls for higher "flexibility to cater to a wide and rapidly changing range of workloads" along with speed and energy efficiency, navigating innovation efforts in architectures' design. Moreover, while in general welcoming novel architectures to help advancing AI, Hooker (2020) goes even further in her discussion raising concerns about the premature and costly specialization of novel hardware on ANNs. The dynamic nature of the software domain manifests itself brightly in such an experimenting field as AI. Indeed,

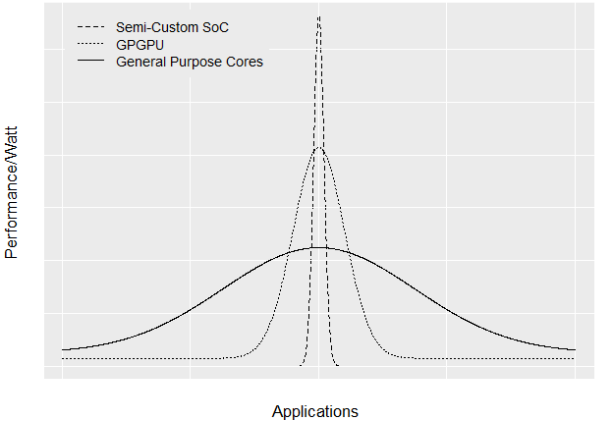
"[i]t is an ongoing, open debate within the machine learning community about how much future algorithms will differ from models like deep neural networks"; however, "[h]ardware design has prioritized delivering on commercial use cases, while built–in flexibility to accommodate the next generation of research ideas remains a distant secondary consideration" (Hooker, 2020, p.7).

3.2 The Trilateral Technological Frontier

The two architectures we analysed mirror each other in their strong sides and disadvantages: the controlflow architecture is concerned with speed of performance in first place and consumes most of its energy on data movement from and to memory, while the dataflow architecture suffers from lack of flexibility. From the supply side, these three characteristics of a chip's performance — *speed, energy efficiency, flexibility* — form a trilateral technological frontier for the chipmakers, guiding their innovation efforts. From the demand side, these characteristics constitute a chip's value and represent criteria of consumers' choice. Figure 2a graphically represents the trilateral frontier.



(a) The trilateral frontier of chip production



(b) AMD's representation of the frontier

Figure 2: Different representations of the trilateral frontier

The von Neumann architecture at the core of the majority of chips provided a sufficient level of flexibility for many applications over 40 years, up until approximately year 2010. During this period, the pursuit of miniaturisation strategy through scaling down the size of transistors and, hence, doubling their number provided a 40% increase in speed while keeping the energy consumption constant.[12] In other words, it was possible with one move (transistor miniaturisation) to achieve improvements in two out of three characteristics, speed and energy efficiency, without harming the third one, heterogeneity. However, approaching atomic scale, transistor size reduction cannot continue at the same pace. Introduced by Intel in 2011, 3D transistor (Auth et al., 2012) instead of planar ones extended the technological trajectory of increasing

[12]Here we refer to Dennard scaling rather than to Moore's law. Moore's law is an empirical regularity relating time and feasible density of elements on a circuit at minimum cost. Dennard scaling is a scaling law based on formal physical principles (Dennard et al., 1974). It states that (i) reduction of a transistor's dimensions by 30% (to 70% of initial size) allows shrinking its overall area by 50% ($0.7^2 = 0.49$) hence twice as many transistors fit in the same area on a die; (ii) consequently, the transistor's channel length and interconnections reduce as well by 30%, reducing the time of switching and transmission of current across the circuit — "0.7x delay reduction, or 1.4x frequency increase"; (iii) this allows lowering the voltage and hence energy consumption (Borkar & Chien, 2011, p.68)

processing speed due to higher density of elements on a die approximately until 2025. Miniaturisation of elements as a strategy concerns all architectures but eventually will not be further possible, and producers have to decouple speed and energy and to look for other techniques to push the frontier forward.

Relentlessly pressured by demand's needs, the von Neumann architecture evolved in complexity to encompass numerous functions in one chip. Miniaturisation provided more space on a chip to implement not only more but also diverse elements, for example, heterogeneous logic cores, on–die memory (cache), connecting channels. Heterogeneity and multiplicity of elements made the architecture capable of performing a wide range of computations and, hence, algorithms. However, the more complex the architecture the more it is flexible but also the higher the costs of fabrication and the harder the management of its energy consumption. This sets the flexibility aspect at odds with energy efficiency. Indirectly, and returning to the discussion on algorithms, flexibility makes it possible to run sophisticated algorithms, but it is not necessarily associated with high speed of processing them; another emerging tension is therefore between flexibility and processing speed. Implementing the principle of universality, i.e. flexibility, controlflow architectures mostly concentrated on improvement of processing speed and on keeping energy consumption under a constant envelope by miniaturizing elements and adding new modules on a chip (Borkar & Chien, 2011). With the advent of AI, limits of flexibility of the von Neumann architecture started to be seen: it can still run ANNs, but it is poorly–suited for that. Instead, the dataflow architectures are fit for ANNs, but for the moment lack flexibility both within (Sze et al., 2020) and beyond this type of algorithms e.g. embedded software. In the meantime, a growing number of businesses (startups as well as incumbents) experiment and adopt AI–based solutions, expanding the *number of markets* for AITs (see Sections 4.2 and 4.3 in Perrault et al. (2019)). Thus, flexibility is an emerging factor of the frontier and becomes a vital concern of chip producers for either type of architecture as well as for the whole SoC.

In Figure 2 we put side by side two representations of the technological frontier. The first one in panel 2a, constructed for this paper, and the second one in panel 2b is reproduced from a 2019 AMD's keynote address at a symposium on high performance chips held at Stanford University (Su, 2019). AMD's representation additionally places examples of chip types into the same framework; number of applications on the horizontal axis represent flexibility or generality, and performance/Watt on the vertical axis combines processing speed and energy efficiency. As expected, General Purpose Cores (CPU) have the widest support of applications but in terms of performance/Watt falls behind GPGPU and Semi–Custom SoC (highly specialised circuits tailor–made for a particular application, also called ASIC, e.g. TPU).

In sum, the trilateral frontier is a coordination mechanism between supply and demand for the development of computing devices, both processors and SoC. On the one hand, each product is characterised by these three metrics. On the other hand, there is a sheer number of application markets that place different weights on each of the frontier's characteristics. Therefore, the size of demand for a particular product can be estimated as a share of markets for whom a product matches the most with consumers' preferences with regard to these characteristics.

4 The Future of Chips: Fragmentation vs Platform

In previous Sections we discussed the changing equilibrium between models of computation pulled by AITs and how this reverberates on the choice–set of chip producers for what concerns the architecture of their products. In this section, we construct an economic framework for the development process of a computing device. In this part under computing device we mean both types of products — a processor and a SoC — referring to both as *chip* for convenience. We proceed further in the formalisation of the dynamics unfolding in the semiconductor industry and present a model of demand distribution driven by the value of a chip composed of the frontier's characteristics. The model stresses and illustrates the role of flexibility as a recently aggravated criterion of consumers' choice, and endogenises it through the software environment. Finally, drawing on the analysis of the technological and economic factors and mechanisms, we derive two scenarios for the evolution of the semiconductor industry that differ by the product form at the core of each trajectory and point out issues for further discussion.

4.1 Modelling Chips' Flexibility and Demand Distribution

We start with our novel point on the emergence of flexibility as a criterion of chips' performance. To do so, we include the software domain into the model. First, this allows for an indirect modelling of hardware's flexibility, approximated with the variety of programs a chip can support. Second, this modelling choice reproduces the feedback loop between the software domain and the semiconductor industry, in line with the argumentation of the supporting services approach. The supporting services approach is also referred to as indirect network externalities, where consumers are indifferent to the number of users of a product but are interested in the variety of services that this product gives access to. We ground our model on the framework provided in Shy (2011) and modify it for the case of chips by (i) modeling consumers' utility through the value of a chip, (ii) considering partial compatibility (Chou & Shy, 1993) and (iii) deepening the interpretation of some parameters due to industry–specific features. In what follows, first we outline the model and then we comment it, supporting the results with evidences from the marketplace.

Demand Side. We assume that consumers are uniformly distributed over a unit interval and indexed with x. Consumers can be considered as individuals or application markets. They buy only one chip each, making a choice between a chip i and a chip j. Each of the chips can address the frontier's characteristics to some extent by employing different techniques we discussed in previous Sections. For example, chip i can be either solely based on the controlflow architecture with many homogeneous (scalar) cores and implement quasi–parallelism with the help of software frameworks, or chip i can be mainly based on the controlflow architecture with some additional tensor cores, like Intel's Nervana, or it can represent a highly heterogeneous SoC comprising several architectures such as Huawei's Ascend 910. The parameter δ is usually interpreted as degree of product differentiation. In the case of chips, the parameter δ measures the disutility from purchasing a chip that does not completely match with the type of computation it is bought for by a consumer. For example, if a consumer needs a chip for mainly controlflow–organised computations and only for a small share of dataflow computations, the parameter δ reflects the

reduction in utility from buying a non–perfect match to the consumer's needs.

$$U_x = \begin{cases} V^i - \delta x - p^i & \text{if buys chip } i \\ V^j - \delta(1-x) - p^j & \text{if buys chip } j \end{cases} \tag{1}$$

Equation (1) represents the utility of a consumer from buying one of the alternatives, where the chips' values V^i and V^j are described as:

$$V^i = E^i S^i \tag{2.1}$$

$$V^j = kE^i S^j \tag{2.2}$$

The value of a chip relates to the frontier's characteristics described in Section 3.2. The rationale behind the E component is as follows: the processing speed or performance of a chip is measured in operations per second. The energy efficiency of a chip is basically its energy consumption per unit of time, expressed in Watts (W).[13] Thus, we introduce the combined efficiency measure E obtained by dividing performance (in operations/s) over energy efficiency (in W), merging two of the frontier's characteristics into one. Note that the higher the energy efficiency, the smaller its measure in W. This or similar measures are indeed used in the industry. For example, the recent analysis of Open AI on the amount of compute shown by modern AI systems uses FLOPS/W[14] as performance measure which, it is argued, is also correlated with FLOPS/$ (Amodei et al., 2019). Any of these measures fit into the model's logic. The parameter k is a scaling parameter that helps to express one chip's efficiency in terms of the other chip's efficiency, i.e. $E^i = k \times E^j$. For example, if $k = 2$, it means that chip i is twice as good as chip j by either performing twice as many calculations with the same energy consumption or consuming twice less energy to perform the same amount of calculations.

The remaining frontier's characteristic, flexibility of computation, is more subtle to model. From the discussion in Section 2 we know that programs can be addressed through either model of computation (sequential or concurrent) and hence performed on any type of chip, however with a sheer difference in time and/or energy endowment. Therefore, there is some degree of interchangeability between chip types that can be expressed in terms of software. Note though, that execution of some programs is so inefficient on a particular chip (e.g. for ANNs parallelism as a requirement to converge to a solution in reasonable time) because of yet absence of developed software environment or framework that would allow efficient execution on another model of computation. In practice, what matters is how many programs can be performed using a specific chip within a reasonable span of time and energy envelope. Therefore, the flexibility

[13] More precisely, under energy consumption we mean the amount of energy required to move an electric charge, expressed in joules (J). Thus, energy efficiency can be measured in joules per second which is equal to Watts: $W = \frac{J}{s}$.

[14] FLOPS stands for floating point operations per second.

of a chip is modeled as follows:

$$S^i = s^i + \rho^i s^j \quad \rho^i \in [0;1] \tag{3.1}$$
$$S^j = s^j + \rho^j s^i \quad \rho^j \in [0;1] \tag{3.2}$$
$$s^i + s^j = 1 \tag{3.3}$$

The total amount of software that can be run on, for example, a chip i is S^i which consists of two components: (i) the amount of chip–specific software s^i, (ii) a share of software written for the other chip that can be interchangeably run on both chips $\rho^i s^j$. The total amount of programs to be performed is normalised to 1.[15] The parameter ρ^i reflects software's *partial* ($\rho^i < 1$) and in most cases *asymmetric* ($\rho^i \neq \rho^j$) interchangeability between chips. In sum, the magnitude of S^i approximates the flexibility of computation that chip i provides.

The difference in values of the two chips can be written down explicitly:

$$\begin{aligned} V^i - V^j &= E^i S^i - kE^i S^j \\ &= E^i(s^i(1 - k\rho^j) + s^j(\rho^i - k)) \end{aligned} \tag{4}$$

To analyse the comparative statics of the value difference shown in equation (4), we simply take partial derivatives with respect to each variable in the expression.

$$\frac{\partial(V^i - V^j)}{\partial k} = -E^i(s^i\rho^j + s^j) = -E^i S^j < 0 \tag{5.1}$$
$$\frac{\partial(V^i - V^j)}{\partial \rho^j} = -E^i s^i k < 0 \tag{5.2}$$
$$\frac{\partial(V^i - V^j)}{\partial \rho^i} = E^i s^j > 0 \tag{5.3}$$

Equation (5.1) shows that the more efficient (higher k) the chip j in terms of combined performance and energy efficiency in comparison with the chip i, the smaller the gap between the chips' values, other things equal. An increasing capability ρ^j of the chip j to run software s^i written for the chip i also reduces the gap between the chips' values in favour of the chip j. Contrariwise, increasing ρ^i leads to growth of the gap in favor of the chip i, other things equal. The two derivatives with respect to the ρ parameters can be interpreted as the attempts of producers to invest in architectural improvements in order to incorporate the functionality of the competing chip, and therefore to manipulate the indirect network externalities.

The remaining two variables s^i and s^j can be interchangeably expressed according to (3.3), hence $s^i = 1 - s^j$ and $s^j = 1 - s^i$. Using this substitution and taking partial derivatives, we obtain the following:

$$\frac{\partial(V^i - V^j)}{\partial s^i} = E^i k(1 - \rho^j) + E^i(1 - \rho^i) > 0 \tag{6.1}$$
$$\frac{\partial(V^i - V^j)}{\partial s^j} = -E^i k(1 - \rho^j) - E^i(1 - \rho^i) < 0 \tag{6.2}$$

[15] We purposefully do not model a law of motion for s^i, as we are interested in understanding the allocation of users across systems given a set of available supporting software and their degree of 'multihoming' captured by the ρ parameters.

These equations show that with a growing amount of software s^i written for the chip i the gap $(V^i - V^j)$ increases, while the opposite holds for s^j. In general, the technical superiority of a chip i in terms of performance per Watt $E^i = kE^j$, along with more tasks supported on this chip s^i increases its value V^i and hence increases the gap $(V^i - V^j)$. If the software interchangeability parameter of a chip ρ^i would be equal to 1, that would mean that chip i is capable of performing all the tasks that chip j does. In other words, if $\rho^i = 1$, the chip i would support the highest possible flexibility of computation. However, precisely the imperfect interchangeability of chips, captured by $\rho^i < 1$, doesn't allow for completely dismantling either of the chips. Lastly, it is worth stressing that here we want to analyse the dynamics of the values gap when varying each of the component. Therefore, while the expression $(V^i - V^j)$ can grow along, for example, s^i, the absolute value of the gap can be any, positive or negative, namely $(V^i - V^j) \gtreqless 0$.

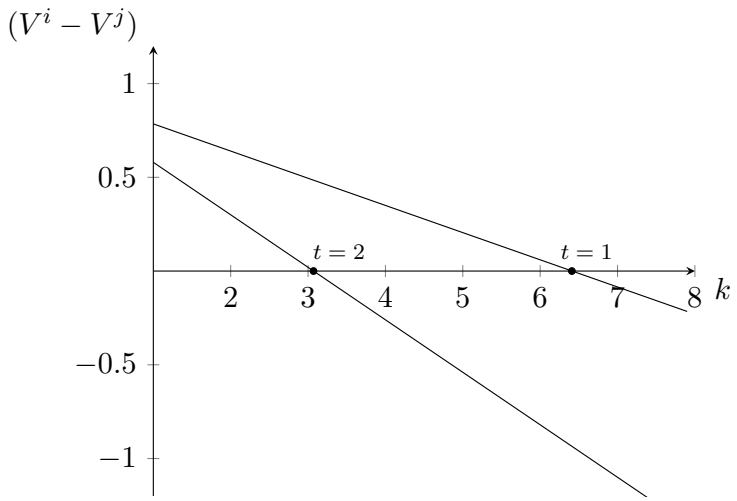


Figure 3: Effect of the efficiency multiplier k on the difference between chips' values $(V^i - V^j)$ varying flexibility parameters s^i, s^j, ρ^j

$t = 1$: $s^i = 0.9, s^j = 0.1, \rho^j = 0.05$;
$t = 2$: $s^i = 0.8, s^j = 0.2, \rho^j = 0.1$
$E^i = 1, \rho^i = 0.3$ for both $t = 1, 2$

As highlighted by the construction of the trilateral frontier, the superiority of one chip over the other is based on three factors combined together. For example, it is not sufficient for a chip to exhibit the lowest energy consumption if the processing speed and flexibility available are low. Moreover, even any pair–wise superiority can be outweighed by a deep enough inferiority with respect to the remaining third characteristic. In order to illustrate how the superiority of a chip is reached through the balancing of all the three frontier's characteristics, we constructed a stylised example, visualised in Figure 3. In this example, in period $t = 1$ the amount of software performed on chip j is only 10%, $s^j_{t=1} = 0.1$, and only 5% of programs performed on chip i are interchangeably executable on the chip j, $\rho^j_{t=1} = 0.05$. In period $t = 2$, both parameters are doubled, namely $s^j_{t=2} = 0.2$ and $\rho^j_{t=2} = 0.1$. Note that according to (3.3) if $s^j_{t=1} = 0.1$ hence $s^i_{t=1} = 0.9$, then if in the second period s^j increased to 0.2 hence $s^i_{t=2} = 0.8$. This doesn't necessarily mean that the absolute amount of tasks performed by the chip i has shrunk; it might simply mean that the amount of tasks performed by the chip j expanded without taking over

tasks from the chip i; this can be the case of ANNs. Thus, in period $t = 1$, given the setting, in order to have equal values, $(V^i - V^j) = 0$, for the chip j it is required to be almost 6.5 times more efficient in terms of combined efficiency E (performance per Watt) than the chip i, $k_{t=1} = 6.41$. In the second period $t = 2$, when the amount of software that can be run on the chip j reaches 20%, $s^j = 0.2$, and interchangeability doubles from 5% to 10%, $\rho^j_{t=2} = 0.1$, in order to have equal values chip j has to be only 3 times more efficient, $k_{t=2} = 3.07$. This numerical example displays the mechanism at work, but might not represent an accurate calibration of the parameters. However, it illustrates the trade–offs and balance through which superiority can be achieved.

Finally, it is extremely important to note that, despite the fact that the numerical expressions of improvements of s^j and ρ^j are incremental from one time period to another, it can be technically rather hard to achieve such improvements, which might also take a substantial amount of time; time periods used in the example are not specified but could be one year or a couple of years, resembling the timing of each next technological node in the industry.

Supply Side. On the supply side we assume a duopoly with price competition. Thus, we are searching for a Nash–Bertrand equilibrium. By equalising utilities from equation (1) we obtain the indifferent consumer:

$$\hat{x} = \frac{V^i - V^j + p^j - p^i + \delta}{2\delta} \tag{7}$$

Each firm has a profit function:

$$\pi^i = p^i q^i = p^i \hat{x}; \quad \pi^j = p^j q^j = p^j (1 - \hat{x}) \tag{8}$$

Maximising profit with respect to price, we derive the Nash–Bertrand pair:

$$p^i_{NB} = \frac{V^i - V^j + 3\delta}{3}; \quad p^j_{NB} = \frac{V^j - V^i + 3\delta}{3} \tag{9}$$

For equilibrium prices to be non–negative the condition $-3\delta \leq V^j - V^i \leq 3\delta$ has to be fulfilled. Finally, plugging the results of equation (9) into (7) we obtain the final formula for the indifferent consumer:

$$\hat{x} = \frac{V^i - V^j + 3\delta}{6\delta} \tag{10}$$

The position of the indifferent consumer is defined by two factors. First, the difference in chips' values $(V^i - V^j)$ whose analysis was shown in equations (4)–(6) can be employed here as well. Second, the disutility δ from the level of mismatch between computations used by a consumer and computations available on the purchased chip. Deepening further the interpretation, this means that the parameter δ reflects a degree of application specialisation or computational convergence. Let's imagine for simplicity that each application market needs a chip to perform one task. One corner case would be when every task is performed by a single algorithm establishing an unequivocal correspondence between the two; in that case δ is the highest because every task is a distinct type of computation. The second corner case would imply that every task consists of all possible types of computation; in that case δ must be low because all tasks are composite. From an economic perspective, in the first case an application market runs only

one type of computation, hence it has a strong preference for a chip that runs this calculation better. More generally, if demand consists of consumers each employing highly homogeneous and distinct computation, δ for the chip–making industry would be very high. The realistic case is none of the extreme ones, with demand consisting of consumers each using its own mixture of algorithms and only few consumers representing extreme cases each using either purely homogeneous or purely heterogeneous calculations. That is why δ can be interpreted as a measure of mismatch between the variety of software supported on a chip and the variety of computations used by a consumer.

The model does not contain a cost variable; however, implicitly a higher value is associated with higher costs to achieve it. For example, according to the financial statements of the ASML Holding (the leading company in the market of photolithography systems), the price of an average system sold in the first half of 2019 is in the range of 36–38 millions of euros (ASML Holding, 2019). Such equipment is highly standardised and it would only account for the initial investments to establish the production process. Leaving aside the formal mechanism of cost formation, in our model we deal with its final instantiation — the price. From an economic viewpoint, the purpose of our model is to show how the shares of demand are driven by the frontier's characteristics and prices as a touch–point of supply and demand. Thus, costs are involved implicitly through the cost of production of a chip with a particular value and its improvement with respect to the frontier's characteristics. It is beyond our analysis to explain *how* a particular value of a chip is achieved, while extensive technological insights regarding chips' characteristics and directions of their improvement currently under exploration in the industry were provided in previous Sections. Here we simply assume that every firm estimates its fixed costs to produce a chip and the quantity demanded in order to understand whether or not the production of a chip will be profitable, exploiting either a high price at low quantity or economies of scale at a low price. If a firm estimates that costs might overweight revenues, it doesn't enter the market.

Parallels in the Marketplace. The frontier's characteristics are operational leverages on which a semiconductor company can act in order to improve its product; producers choose technical approach and the degree of addressing these characteristics based on cost–benefit analysis, aspiring to create a product that appeals to a larger share of demand. In the previous Sections we provided examples of innovations in architecture, elements, materials and techniques that target processing speed, energy efficiency, and flexibility. Hierarchical networks instead of bus interconnect, experimentation with wafer size, new materials and signal types, 3D instead of planar transistors, die stacking, in–memory computing, array and neuromorphic processors, and heterogeneous SoC, all illustrate producers' actions undertaken to act on different segments of the trilateral frontier. Their decision results in the next generation of chips with different values offered to consumers. The real–world example of such behavior is Intel's Process–Architecture–Optimization strategy that was implemented in 2016, replacing the so–called Tick–Tock strategy.

Provided that we model flexibility through the software domain, this implies that producers can allocate their effort to increase the flexibility supported by their products in two ways: (i) introducing changes in hardware to expand functionality and through that encompass more of the existing software from the competitor; in other words, increase ρ^i, (ii) investing in the expansion

of the software set written specifically for a producer's own chip, which means increasing s^i. The first way, the introduction of hardware changes, is discussed at length in previous Sections, therefore we now focus on the second way, software–related changes. As mentioned in Section 3 regarding GPGPU, Nvidia developed the CUDA framework to support its products; the consortium Khronos Group works in the same direction of heterogeneous computing with its OpenCL framework designed by Apple. Other open–source platforms like Google's TensorFlow and Microsoft's CNTK are aimed at the collaborative development of dataflow software solutions to run on chips that can support them, such as TPU or CPU–GPU tandem. By adapting the existing software and writing programs that can effectively run on its product, a firm i increases the value of its chip i targeting precisely the s^i component. However, producers of the competing chip j can counteract by developing instruction set architecture (ISA) extensions.[16] Modifying ISA by including additional packages of new commands allows the competing chip j to encompass some functions performed on the chip i. In terms of our model, such effort affects ρ^j. As an example, we can mention Advanced Vector Extensions (AVX) and its further extension Vector Neural Network Instructions (VNNI) from Intel for x86 ISA, Vector Multimedia Extension (VMX also known as AltiVec) by IBM for Power ISA and NEON technology from ARM Holdings for its eponymous ARM ISA.

In sum, our model reveals the mechanism driving the distribution of demand based on chips' technical characteristics, available software and how well overall a chip meets the computational needs of consumers. A better–developed software environment and compatibility indicate higher flexibility of a chip, which can appeal to a larger share of demand. In turn, demand is characterised by degree of differentiation with regard to the frontier's characteristics: the higher the differentiation the more precise features of a chip are required by each application market. In general, a chip can exhibit either (i) Pareto improvements with respect to any of the frontier's characteristics gaining more applications or (ii) a trade–off between each couple of characteristics shifting the set of applications.

4.2 The Industry at a Crossroad: Alternative Scenarios

From the discussion so far we set out a collection of mechanisms and forces shaping from the outside and within the evolution of the semiconductor industry. Exogenous challenge that arrived from the AI segment tests the robustness of the established technological trajectory. In fact, this time the challenge lies at the fundamental level of the computational model on which chips are built on. Residing in declarative programming paradigm that is inefficiently executed on established sequential model of computation, AITs triggered a wave of innovation efforts that resulted in numerous novel products with the dataflow architecture at their core. It is becoming clear that the simple speedup race between competing chips is not the central issue for the future of the semiconductor industry; rather, the more profound issues of organizing the logic of computation and variety of algorithms that a chip can support in order to appeal to a sufficient share of demand are key. The question now is how chips will evolve this time. Considering all the factors at play, we derived two scenarios on which the industry can converge.

[16] In essence, ISA modifications are on the borderline between hardware and software (programmatic) changes. Given ISA's undeniable programmatic element, here we employ example of ISA modifications that, similarly to hardware changes, impact ρ^j.

Scenario I Under the relentless pressure of economic factors within the semiconductor industry and the continuous but siloed pull from the downstream markets for market–specific improvements, producers might decide to pursue trajectories tailor–made to subsets of downstream markets, grouped around specialised chips that accurately address needs within given submarkets. A *customisation strategy and hence the fragmentation of the semiconductor industry* might occur.

Scenario II Aspiring to address larger shares of demand associated with greater but probably delayed payoffs, chip producers can make long–term investments at the system level, aimed at the creation of a *platform chip comprising heterogeneous cores*. To achieve that, the overarching architecture must reproduce a composition of components on a chip that ensures scalable, heterogeneous and energy–proportional computing. Developed in response to the call of one segment, the platform chip can diffuse over time among other downstream markets with decreasing cost of production and, hence, price.

Arguments 'pro' and 'against' exists for each of the scenarios. According to our model, if the demand is significantly differentiated it is harder to acquire a large share of consumers (see 10), other things equal. The smaller the size of potential demand aggregated over application markets, the harder it is to return high costs of design and fabrication of an heterogeneous chip. Thus, naturally, if differentiation is high, the viable strategy is that of fragmentation of the semiconductor industry's offer into several distinct chips, each characterised by unique performance with respect to the frontier's characteristics; application markets decide to purchase either one or a set of chips based on their needs.

Thompson & Spanuth (2018) advocate for the first scenario by linking the future dynamics of chips production to the dual–inducement mechanism typical of General Purpose Technologies (GPTs) (Bresnahan & Trajtenberg, 1995). They develop a model of choice between universal and specialised processors based on relative speed up factor and identify a cut–off point from which the specialised processors become more appealing than the universal ones. As more and more downstream markets switch to specialised processors, this leads to the halt of the dual–inducement mechanism for universal processors. Thus, they expect the end of the GPT paradigm of universal processors and envisage a situation of application–based market fragmentation with specialised computing evolving in more compartmentalised domains. This prediction rests on (i) a view of the processor as the singleton GPT technology and (ii) the assumption that processing speed is the sole criterion of the choice of a processor. Concerning (i), from this perspective, processors can be considered as pure competing alternatives. However, we also need to consider the possibility that it is the SoC the candidate for the role of GPT, while processors are complementary blocks. As for (ii), we acknowledge that processing speed is an important factor and included it among the frontier's characteristics. However, we argue that it is not the sole criterion for all applications and might not be the primary one for some share of applications. For the development of AI itself, "[f]ocusing on raw computing power misses the point entirely. Speed alone won't give us AI. Running a poorly designed algorithm on a faster computer doesn't make the algorithm better; it just means you get the wrong answer more quickly. (And with more data there are more opportunities for wrong answers!) The principal

effect of faster machines has been to make the time for experimentation shorter, so that research can progress more quickly. It's not hardware that is holding AI back; it's software." (Russell, 2019). As we pointed out earlier in Section 3.1, even at the level of processor for AI there is an ongoing search for a more flexible architecture and "TOPS/W alone considered harmful" (Sze et al., 2020). In general, the software domain is dynamic and evolves faster than hardware due to lower costs inherent to information products (Goldfarb & Tucker, 2019), hence the variety of software solutions that a consumer (market, firm or individual) can employ increases over time. Optimization of hardware for a specific software for the sake of faster processing might result in a very limited set of application markets and in a shorter product life cycle. Instead, flexibility is a more sustainable strategy for a chip producer.

Building on the aspect of flexibility, there are arguments in favor of the platform chip scenario. Given that hardware flexibility can be approximated with the amount of software that is effectively run on a chip, the presence of indirect network externalities does have significant implications for the semiconductor industry. This approach suggests that consumers' decision upon which hardware system to buy is affected by complementary products or supporting services, in this case software, available for each system. In particular, Church & Gandal (1992) model the effect of the decision of software firms upon software provision on the market share and the number of hardware systems that will exists in equilibrium. Their analysis shows that when consumers' preferences on *software variety* are relatively high[17], this leads to the exclusive adoption of one of the hardware systems if a critical minimum amount of software is provided. Furthermore, in the case in which two hardware systems exist, total surplus would be higher under many parameters' values if a standard (a single hardware system) was mandated. Thus, strong preference for software variety is associated with the choice of one hardware system. By translating software variety into hardware's flexibility, in our model flexibility is a variable that characterises the chip and producers can act upon it by either writing software supported on their chips or by increasing compatibility with the existing one.

In our analysis we covered the technical performance of a chip and demand's preferences as factors that shape the technological trajectory and steer the development of the semiconductor industry. The last factor that can tip the balance in favor of one or the other scenario is concentration of market power among chip producers. There is a number of big players in the semiconductor industry even at the global level; some of them we already named, such as Alphabet, Amazon, Alibaba, Huawei, Samsung, Nvidia, Intel. Many of these companies are cross–industry actors that comprise a diverse portfolio of assets that they built to pursue internally established goals. The dimension that matters in the context of the semiconductor industry is *edge versus cloud computing*. Some of these big players are cloud–oriented like Google and Amazon and they already direct their innovative effort to develop in–house chips to support AI through their cloud services. The primary focus of such chips would be concurrency and speed to support numerous users working concurrently by providing low latency. The already mentioned TPU of Google and Trainium and Inferentia chips of Amazon serve exactly this purpose, being fast, highly parallel and energy efficient however non–flexible. Thus, cloud computing would rather benefit from a set of dedicated chips combined together to deliver

[17]The benefit from high software variety available on a chosen hardware system has to overweight the disutility from spending on the purchase of various software, the price of hardware and the degree of its differentiation.

state–of–the–art performance with respect to each frontier's characteristic. Contrariwise, chip producers that place their bid on edge computing lean toward more independent and capable devices and, hence, direct their innovation efforts in the direction of the platform chip. The already named Huawei's Ascend 910 is one example. Another prominent example is Apple's M1 chip that comprises CPUs, GPUs and Neural Engine cores in one SoC.[18] Apple stresses the edge–oriented application of its chip with high performance, low energy consumption and flexibility achieved through integration of heterogeneous cores. In line with our reasoning, Apple acts on hardware's flexibility through software variety as well providing Core ML software framework for programming, and optimising its Big Sur operating system to work with the M1 chip. In sum, there are pieces of evidence suggesting that a dominant design for the dataflow processor is on the way while at the same time there is ongoing experimentation with the configuration of the platform chip.

5 Related Literature and Discussion

The analysis offered in this paper relates to a number of contributions in the literature. We review the works most related to our study and emphasize similarities and differences. To organise the review, we highlight the dimensions shared by our study and the discussed works with respect to, for example, the level of analysis, the industry considered, the economic and strategic or technological arguments provided, and the role played by demand and supply. We start with a focus on the semiconductor industry to identify the forces and mechanisms shaping its technological trajectory and innovation; then, we progressively move to the computer industry to discuss the similar dynamics produced by the introduction of new products. Finally, we take a more fine–grained perspective centred on the design of platform products, whose rules apply to computers as well as to chips.

Steinmueller (1992) focuses on the semiconductor industry and provides a supply–side analysis of the economic arguments — in particular production economies and dis–economies — that have contributed to maintain chip production for decades on a stable technological trajectory tied to the von Neumann architecture and to miniaturisation as its main innovation direction. The paper outlines the trade–off between specialisation and standardisation that characterises the industry. Economies of scope fuelling product variety (and, thus, specialisation) and economies of scale fuelling production expansion (and, thus, standardisation) are at odds with each other, and the semiconductor industry has mostly pursued economies of scale. The reason for this is that chip production is characterised by the so–called *capacity races* — the incentive to engage in mass production in order to amortise large costs of equipment capable of little flexibility. The unprecedentedly big chip produced by Cerebras, which is fabricated on conventional photolithographic equipment is recent evidence of the persistent importance of equipment cost as economic factor. Capacity races produce two economic effects; the first one concerns the incentives for firms to be first movers in innovation by pre–empting competitors in the introduction of new generations of chips in order to earn additional payoffs out of the investment in capacity. The second effect is the incentive to push for cost reduction in order to hasten the scaling up

[18] https://www.apple.com/newsroom/2020/11/apple-unleashes-m1/

of production. While Steinmueller advocates for the exploration of flexible chip manufacturing technologies to make product variety economically viable at low output volumes, the forces he highlighted have mostly prevailed and kept the industry on a well–defined trajectory of standardisation and mass production. The same dis–economies of scope that enabled the production at scale of processors embodying the von Neumann architecture can push chipmakers to produce a platform chip. The success of ANNs and the demand for AI applications has drawn chipmakers' attention to concurrent models of computation, but the forces pushing for standardisation can again create a strong incentive to the integration of heterogeneous processors into a single product that can exploit economies of scale.

Adams et al. (2013) study innovative activities in the semiconductor industry in the 80s and 90s. They take a sectoral systems of innovation perspective to highlight the role played by intermediate users' demand in innovation. From the supply–side, a series of technological changes (i.e. the development of Electronic Design Automation tools) allowed for the dis–integration of the chipmaking supply chain. With weaker ties between the design and manufacturing phases, entry barriers for specialised firms (e.g. the so–called 'fabless' firms) at different points of the chain lowered. The new actors could partner with foundries to offer specialised designs to specific market niches and co–exist next to integrated producers, as the latter focus on more systemic innovation that require superior coordination efforts (Kapoor, 2013). From the demand–side, an increasing amount of the market niches started to emerge with the opening of new applications for integrated circuits — in particular wireless communication and mass consumer products; these niches are characterised by the demand for tailor–made chips. The combination of more fragmented production processes and differentiated final demand increased the importance of application knowledge, and thus induced co–innovation by semiconductor and user firms. While Steinmueller's capacity races have confined chip production within a well–defined technological trajectory shaped by economies of scale, the supply chain dis–integration illustrated by Adams and co–authors has allowed an increase in product variety through the production of specialised chips for market niches. However, this dynamics relaxed but not dismantled the dominance of classic von Neumann chips.

Malerba et al. (2008) analyse to the joint structure of the semiconductor and computer industry and use a 'history–friendly' model to reproduce the discontinuities that technological innovations in semiconductor devices induced on the industries. The paper provides a simulation of economic and technological mechanisms on both the supply and demand side to map the co–evolution of two industries' market structures. For example, the authors discuss how the introduction of integrated circuit in the 60s allowed IBM to control both the development of semiconductor devices and their implementation in mainframe systems such as the IBM System/360. The microprocessor, introduced by Intel in the 70s, challenged IBM vertically–integrated production, dis–integrated the supply chain and lead Intel to dominate the semiconductor industry. Each new class of semiconductor devices triggered changes in the industry's structure. The latest technological discontinuity we describe in our paper — the embedding of the dataflow model of computation into chips as a result of ANNs 'shock' success — will also reverberate into changes in the industry's organisation. The current turbulence characterised by exploration of product designs and entry by companies dominating in adjacent markets (e.g. Nvidia, Amazon, or Ap-

ple) and startups (e.g. Cerebras or Graphcore) might turn into a new structural equilibrium as soon as production economies and dis–economies will play out.

In general, we can consider the problem of producing a new chip capable to integrate sequential and concurrent models of computation as a problem of bundling features into the overall configuration of a product that combines heterogeneous components. Such configuration is an instantiation of a 'platform product' in the context of the semiconductor industry. Platform products have been extensively studied in the context of the computer industry. The industry has been producing computer platforms (an innovation inaugurated by the IBM System/360 — see Baldwin & Clark (2000)) integrating different components — chips being a core and often the highest–value one. A platform chip and a computer platform are two different types of product; however, the mechanisms at work shaping the configuration of a platform are essentially the same at different levels. Any platform product is subject to dynamic tensions of both economic and technological kind, as the relationship among its components needs to accommodate both innovation and degrees of (backward) compatibility. These tensions emerge at many levels, from the industry to the firm and product level. For this reason, while the focus of our paper is a very fine–grained one — the platform chip — we can refer to findings from the literature on computing platforms.

Bresnahan & Greenstein (1999) take an economic perspective on computer platforms, as they study the evolution of technological competition and market structure in the computer industry. Focusing on the industry, their scope of analysis is broader than ours as they consider in detail the industry and segment–wise convergence to equilibrium. However, the key points they make applies to our case as well: first is the need to focus on (platform) products rather than firms as unit of analysis. In fact, computer platforms (such as the IBM System/360, or Apple Macintosh) have been the point of interaction between supply and demand in the computer industry. A second key element is the role of endogenous sunk costs and demand (reflected in the market segments served by the industry) in shaping which platform product gains dominance and persistence. A third important element regards the nature of competition in the industry after what they label the 'competitive crash' of the 90s: platform competition within the same market segment was the result of indirect entry, with new computer platforms first entering a novel market segment with specialised (usually technical) users and then moving to established segments (business and then consumer users — a dynamics illustrated also in Bresnahan & Yin (2010)). Bresnahan and Greenstein's account of the computer industry's evolution around platforms illustrates how industry–wide and within–segment equilibrium are related. "Equilibrium in each segment of the computer industry obeys its own logic of concentration and persistence, determined by buyer/seller interactions" around a platform; indirect entry allows segment dynamics to channel change to the industry level. This dynamics resulted in a 'divided technical leadership' in the computer industry. Our model is a snapshot of this very mechanism at work in the context of semiconductor industry, where the matching of chips with demand structure (with user needs approximating market segments) determines the industry–wide equilibrium split among alternative technologies. In particular, we can apply their framework to our case by considering AI–users a novel market segment for the semiconductor industry. Through indirect entry, new chips can occupy the segment of AI–users first and then move to compete with estab-

lished products in other segments. As in the case of the computer industry, the success of direct entry or the insulation of a new platform within a segment depends on several factors — the differentiation of demand's needs in different segments, as well as the technological features of the new chip. A platform chip integrating classic and AI–specialised components could induce a competitive crash with dominant products in many segments served by the semiconductor industry.

Taking stock, our paper shares with Malerba et al. (2008) the interest on technological discontinuities. Instead of focusing on the varying structure of the semiconductor and computer industries' supply chains, we are interested in how the push to introduce chips capable of supporting AI applications as a new discontinuity will influence chipmakers product design. Our focus is on how producers will bundle AI–related computations into the functionality supported on a chip, considering that their production choices are influenced by technical feasibility, design and fabrication costs, the matching between product characteristics and end–users demand. Taken together, Steinmueller (1992), Adams et al. (2013) and Bresnahan & Greenstein (1999) provide us with a useful framework to understand the channels through which such new product can emerge and whether it can appeal a major share of market segments (and demand needs), as suggested by our Scenario II. Adams and co–authors and Bresnahan and Greenstein show how the structure of demand (and its participation in innovation) is a potential source of product variety. Steinmueller shows how dis–economies of scope drive the industry back towards standardisation. The current moment is a crossroad. One the one hand, the standard over which the industry settle can emerge through the process of indirect entry and a new competitive crash. In our case, the commercial use of AI and ANNs has indeed induced the exploration of new product space; the successful design of a platform chip integrating AI and non–AI components can enter a specific market niche first and from there diffuse and emerge as a dominant design for the whole industry. On the other hand, high costs and the appeal to an insufficient share of demand can fragment the product space resulting in non–overlapping demand clusters served by custom chips. All these mechanisms can be rationalised by our model by considering the tension between high differentiation in the structure of demand and at the same time one of the competing systems displaying high flexibility so to serve at scale a large share of the markets.

Cusumano & Gawer (2002), Gawer & Henderson (2007), and Burgelman (2002) shift the analysis from the industry to the firm and product level. As the product platform they study are microprocessors, their work is proximate to ours in terms of the level of analysis. These studies provide a more find–grained analysis of the tensions emerging inside the leading platform sponsor, Intel. The tensions they describe relate to strategic management choices but capture mechanisms at work in our case as well. Cusumano & Gawer (2002) focus on the platform level and discuss the tensions emerging in the process of balancing the relationship between the platform owner (the firm controlling the core architecture of the system) and its complementors. In a platform product, owner and complementors are linked in an ecosystem that displays non–generic and supermodular complementarities (Jacobides et al., 2018). Chipmakers such as Intel experience 'platform dependency' as — to quote the Director of Intel Architecture Lab mentioned by the authors, "(w)e are tied to innovations by others to make our innovation valuable". Intel's strategy with respect to complements is explored in depth in Gawer & Henderson

(2007). They study the incentives for the platform owner to enter complementors' markets; the tensions highlighted in this case are prevalently those internal to the organization. In fact, the strategy to expand the demand for microprocessor implied growing the whole computer platform (microprocessors plus complements) and, thus, to allow complementors to grow profits as well. However, a balanced growth of the whole platform contrasted the need of Intel to enter and 'squeeze' profits from complementary markets (for example motherboards, online services, PC peripherals and accessories). To address this tension, Intel entered complementary markets only when the company matched the capabilities of the competitors and prevalently when the complementary markets were 'connector' markets, those producing products that embodied interfaces to the core technology (e.g. chipsets, or motherboards). Burgelman (2002) zooms further into organizational mechanisms to illustrate the role of leadership in resolving the tensions occurring within Intel, in particular in the period surrounding Andrew Grove's tenure as CEO (1985–1998). Burgelman stresses how innovation taking place at the level of the platform product tied Intel (the producer of the highest value component of computers, microprocessors) to its complementors. This tying resulted in a co–evolutionary lock–in (the platform dependency of Gawer and Cusumano), which has strongly impacted Intel's strategy: to maintain market power, Intel needed to align the pace its technological advances to that of the whole system. The relevance of co–evolutionary lock–in becomes evident in Burgelman's recounting of the resolution of Intel's 'internal battle' between the i860 and the x86 microprocessor architecture (and, respectively tied to them, between the so–called RISC and CISC instruction set). Intel opted for the x86 architecture, especially as it decided to follow the strategy vector leading to focus on the personal computer (PC) market segment; in fact, Intel "increasingly tied its strategic direction and economic fortunes to the evolution of the PC market segment". Co–evolutionary lock–in has been an additional force, this time technology–driven and occurring at the product and firm level rather than emerging from dis–economies of scope and applying to the whole industry, to conserve the semiconductor industry persistent technological trajectory. As dis–economies of scope can push the current state of the semiconductor industry towards our scenario II, the same holds with co–evolutionary lock–in: chipmakers can explore alternative product design, but their economic fortunes are tied to their complementors. As long as uncertainty characterises the future applications and evolution of AI algorithms, the development of a platform chip design would ease coordination among actors and provide a safer bet regarding the future evolvability of the system.

In line with our paper, the studies we reviewed highlight tensions and different mechanisms shaping a platform product. However, they all tell a story placed within the established technological trajectory with the sequential model of computation at its core; instead, we look at the impact of a more profound technological discontinuity — the exhaustion of the capability of the sequential model of computation to address an increasing variety of algorithms and the rise of concurrent model of computation. Despite AI being an active field since the 1950s, this discontinuity did not take place before because AI as an application segment did not have a big weight and only recently entered a commercial phase at scale. Before the current AI commercial boom, the majority of other applications requiring computing devices could get by with the ever–improving von Neumann architecture. The competitive crash among computer platforms

described by Bresnahan and Greenstein and the changes in the structure of the semiconductor industry supply chain discussed by Adams and co–authors occurred in response to discontinuities in technology and changes in demand, but were not contesting the organisation logic of chips. Instead, the increasing demand for AI–related computing that we unpacked starts to exert a stronger pressure than the prevailing architecture could accommodate, launching a wave of radical and architectural innovations, respectively developing specialised components and experimenting with the design of a platform chip.

6 Conclusion

In this paper, we investigated how a technological discontinuity can impact product design and production strategies in a highly technological industry. The industry we focus on is the semiconductor industry, and the technological discontinuity is introduced by the novel application segment of AITs that grows rapidly. In turn, the use of AITs for a growing variety of applications produced an increasing demand for compute. This has triggered a search for the hardware (chips) capable of executing AI algorithms such as ANNs more efficiently. The chips on which the semiconductor industry has built its success and that dominated its technological trajectory for decades are built around the von Neumann architecture, that is ill–suited to execute modern AI algorithms and ANNs in particular. In fact, the commercial boom of AI has shed light on the limitations of this classic architecture despite its continuous performance improvements over time. For this reason, chip producers are shifting attention to alternative architectures to implement in their chips. The prospective candidate is the so–called dataflow architecture. This is the hardware implementation of the concurrent model of computation, a different model compared to the sequential one at the core of the von Neumann architecture. The properties of the dataflow architecture match better the organisation of computation underlying AI algorithms. This technological discontinuity with respect to the industry's established technological trajectory represents the challenge the chipmakers are currently facing. Thus, we studied the nature of this discontinuity and how forces and mechanisms at work in the semiconductor industry might steer its further development in one of two potential scenarios.

As our study deals with technological innovation in a highly technological industry and stresses the systemic relationship between hardware and software, it is a novel contribution to several strands of literature, from the economics of AI to the study of platform products in the context of the economics and strategic management of the semiconductor and computer industry, as well as to the literature on technological trajectories. In the analysis, we combined insights and perspectives from different fields such as AI, engineering and computer science with modelling approaches from the economics of software and system products. In order to assess the direction in which the AI discontinuity is steering the design of chips, we started our study by overviewing the computational framework for ANNs. We highlighted how ANNs are endemic to the so–called declarative programming paradigm in virtue of their organisational logic as algorithms, and how the concurrent model of computation, as opposed to the sequential one, matches this logic. Given that, we reviewed how models of computation are implemented in hardware architectures and explored the difference between scalar and vector processors embodying the sequential model and novel architectures such as array and neuromorphic ones embodying the concurrent model

of computation.

When designing a chip, producers can opt for one or the other architecture or for an integration of them into one SoC. The performance of the resulting chip is measured with respect to the three fundamental characteristics — speed, flexibility, and energy efficiency — that constitute a trilateral technological frontier. The frontier serves as a benchmark for producers, guiding their design decisions. However, the market success of a new chip depends on the demand's preferences with regard to these frontier's characteristics. We captured this mechanism with an analytical model determining the distribution of demand between two alternative chips based on hardware's flexibility approximated with the software variety available for a chip and an efficiency metric that combines processing speed and energy efficiency. In stylised terms, the model represents the current state of the semiconductor industry, with AI applications expanding demand variety (directly through itself and indirectly through AI–using segments) and the difference among competing chips reflecting the experimentation surrounding the design challenge.

All the forces and tensions we described have derailed the established technological trajectory of the industry and injected uncertainty regarding the novel track on which it will settle. We summarised the outcomes to which the future of chip can converge in two scenarios. In the first, the demand from the AI segment lead to the development of specialised chips but does not induce changes to the industry–level equilibrium — chip production becomes siloed and fragmented. In the second, in response to the increasing variety of algorithms with the advent of modern AI and under the pressure of production economies, chip producers allocate innovative efforts to the flexibility of their products creating a novel platform chip. Such chip would encompass different architectures onto a single substrate and come to serve most of the industry's demand segments. Both scenarios can emerge out of the current technological turbulence in the industry, and we lay out arguments in favour and against them. However, we stress that within the AI field the pace of progress is high, as well as software domain is more dynamic than hardware. The growing variety of (competing) AI techniques and algorithms raises a valid concern with over–specialisation; given the high costs to develop and produce novel semiconductor devices and the high inertia of these processes, the decision to fork the production of chips with specialised products tailored to current AI algorithms while other approaches are yet in their infancy can be a risky bet for chipmakers. Directing innovative efforts towards flexibility and, thus, a platform chip might result in higher pay–offs in the long run. Quoting Marvin Minsky, "[t]he power of intelligence stems from our vast diversity, not from any single, perfect principle"; the same holds for the potential appeal of a platform chip capable to harness the diversity of computations.

References

Adams, P., Fontana, R., & Malerba, F. (2013). The magnitude of innovation by demand in a sectoral system: The role of industrial users in semiconductors. *Research Policy*, 42(1), 1–14.

Agrawal, A., Gans, J. S., & Goldfarb, A. (2019). Artificial intelligence: the ambiguous labor market impact of automating prediction. *Journal of Economic Perspectives*, 33(2), 31–50.

Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference* (pp. 483–485).: ACM.

Amodei, D., Hernandez, D., Sastry, G., Clark, J., Brockman, G., & Sutskever, I. (2019). Ai and compute.

Arthur, W. B. (1989). Competing technologies, increasing returns, and lock-in by historical events. *The economic journal*, 99(394), 116–131.

ASML Holding, N. (2019). Financial statements us gaap q2 2019.

Auth, C., Allen, C., Blattner, A., Bergstrom, D., Brazier, M., Bost, M., Buehler, M., Chikarmane, V., Ghani, T., Glassman, T., et al. (2012). A 22nm high performance and low-power cmos technology featuring fully-depleted tri-gate transistors, self-aligned contacts and high density mim capacitors. In *2012 Symposium on VLSI Technology (VLSIT)* (pp. 131–132).: IEEE.

Baldwin, C. Y. & Clark, K. B. (2000). Design rules, volume 1: The power of modularity. *MIT Press Books*, 1.

Boden, M. A. (2016). *AI: Its nature and future*. Oxford University Press.

Borkar, S. & Chien, A. A. (2011). The future of microprocessors. *Communications of the ACM*, 54(5), 67–77.

Bresnahan, T. & Yin, P.-L. (2010). Reallocating innovative resources around growth bottlenecks. *Industrial and Corporate Change*, 19(5), 1589–1627.

Bresnahan, T. F. & Greenstein, S. (1999). Technological competition and the structure of the computer industry. *The Journal of Industrial Economics*, 47(1), 1–40.

Bresnahan, T. F. & Trajtenberg, M. (1995). General purpose technologies 'engines of growth'? *Journal of econometrics*, 65(1), 83–108.

Brown, C. & Linden, G. (2011). *Chips and change: how crisis reshapes the semiconductor industry*. MIT Press.

Burgelman, R. A. (2002). Strategy as vector and the inertia of coevolutionary lock-in. *Administrative science quarterly*, 47(2), 325–357.

Chen, Y.-H., Yang, T.-J., Emer, J., & Sze, V. (2019). Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*.

Chou, C.-f. & Shy, O. (1993). Partial compatibility and supporting services. *Economics letters*, 41(2), 193–197.

Church, J. & Gandal, N. (1992). Network effects, software provision, and standardization. *The journal of industrial economics*, (pp. 85–103).

Cusumano, M. A. & Gawer, A. (2002). The elements of platform leadership. *MIT Sloan management review*, 43(3), 51.

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1), 82–99.

Dennard, R. H., Gaensslen, F. H., Rideout, V. L., Bassous, E., & LeBlanc, A. R. (1974). Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5), 256–268.

Dosi, G. (1982). Technological paradigms and technological trajectories: a suggested interpretation of the determinants and directions of technical change. *Research policy*, 11(3), 147–162.

Eckersley, P., Nasser, Y., et al. (2017). Eff ai progress measurement project.

Flamm, K. (2018). *Measuring moore's law: Evidence from price, cost, and quality indexes*. Technical report, National Bureau of Economic Research.

Gawer, A. & Henderson, R. (2007). Platform owner entry and innovation in complementary markets: Evidence from intel. *Journal of Economics & Management Strategy*, 16(1), 1–34.

Goldfarb, A., Gans, J., & Agrawal, A. (2019). *The Economics of Artificial Intelligence: An Agenda*. University of Chicago Press.

Goldfarb, A. & Tucker, C. (2019). Digital economics. *Journal of Economic Literature*, 57(1), 3–43.

Hebb, D. O. (2005). *The organization of behavior: A neuropsychological theory*. Psychology Press.

Henderson, R. M. & Clark, K. B. (1990). Architectural innovation: The reconfiguration of existing. *Administrative science quarterly*, 35(1), 9–30.

Hennessy, J. L. & Patterson, D. A. (2011). *Computer architecture: a quantitative approach*. Elsevier.

Hooker, S. (2020). The hardware lottery. *arXiv preprint arXiv:2009.06489*.

Jacobides, M. G., Cennamo, C., & Gawer, A. (2018). Towards a theory of ecosystems. *Strategic Management Journal*, 39(8), 2255–2276.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al. (2017). In-datacenter performance analysis of a tensor processing unit. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)* (pp. 1–12).: IEEE.

Kapoor, R. (2013). Persistence of integration in the face of specialization: How firms navigated the winds of disintegration and shaped the architecture of the semiconductor industry. *Organization Science*, 24(4), 1195–1213.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Lee, E. A. (2002). Embedded software. In *Advances in computers*, volume 56 (pp. 55–95). Elsevier.

Lee, E. A. & Neuendorffer, S. (2005). Concurrent models of computation for embedded software. *IEE Proceedings-Computers and Digital Techniques*, 152(2), 239–250.

Lie, S. (2019). Wafer scale deep learning. In *IEEE Hot Chips 31. Symposium (HCS 2019)*.

Malerba, F., Nelson, R., Orsenigo, L., & Winter, S. (2008). Vertical integration and disintegration of computer firms: a history-friendly model of the coevolution of the computer and semiconductor industries. *Industrial and Corporate Change*, 17(2), 197–231.

McCarthy, J., Minsky, M., Rochester, N., & Shannon, C. (1955). Proposal for the 1956 dartmouth summer research project on artificial intelligence, dartmouth college.

McCulloch, W. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 115–133.

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., Jackson, B. L., Imam, N., Guo, C., Nakamura, Y., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197), 668–673.

Newell, A. & Simon, H. (1956). The logic theory machine–a complex information processing system. *IRE Transactions on information theory*, 2(3), 61–79.

Perrault, R., Shoham, Y., Brynjolfsson, E., Clark, J., Etchemendy, J., Grosz, B., Lyons, T., Manyika, J., Mishra, S., & Niebles, J. C. (2019). *The AI Index 2019 Annual Report*. Technical report, AI Index Steering Committee, Human-Centered AI Institute, Stanford University.

Raina, R., Madhavan, A., & Ng, A. Y. (2009). Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning* (pp. 873–880).: ACM.

Russell, S. (2019). *Human Compatible: Artificial Intelligence and the Problem of Control*. VIKING.

Shao, Y. S., Clemons, J., Venkatesan, R., Zimmer, B., Fojtik, M., Jiang, N., Keller, B., Klinefelter, A., Pinckney, N., Raina, P., et al. (2019). Simba: Scaling deep-learning inference with multi-chip-module-based architecture. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture* (pp. 14–27).

Shen, Y., Harris, N. C., Skirlo, S., Prabhu, M., Baehr-Jones, T., Hochberg, M., Sun, X., Zhao, S., Larochelle, H., Englund, D., et al. (2017). Deep learning with coherent nanophotonic circuits. *Nature Photonics*, 11(7), 441.

Shy, O. (2011). A short survey of network economics. *Review of Industrial Organization*, 38(2), 119–149.

Steinmueller, W. E. (1992). The economics of flexible integrated circuit manufacturing technology. *Review of Industrial Organization*, 7(3-4), 327–349.

Su, L. (2019). Delivering the future of high-performance computing with system, software and silicon co-optimization, keynote address at hot chips: A symposium on high performance chips, edition 31.

Suárez, F. F. & Utterback, J. M. (1995). Dominant designs and the survival of firms. *Strategic management journal*, 16(6), 415–430.

Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2020). How to evaluate deep neural network processors: Tops/w (alone) considered harmful. *IEEE Solid-State Circuits Magazine*, 12(3), 28–41.

Thompson, N. & Spanuth, S. (2018). The decline of computers as a general purpose technology: Why deep learning and the end of moore's law are fragmenting computing. *SSRN 3287769*.

Turing, A. M. (1937). On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London mathematical society*, 2(1), 230–265.

Veen, A. H. (1986). Dataflow machine architecture. *ACM Computing Surveys (CSUR)*, 18(4), 365–396.

Winter, M., Prusseit, S., & Gerhard, P. F. (2010). Hierarchical routing architectures in clustered 2d-mesh networks-on-chip. In *2010 International SoC Design Conference* (pp. 388–391).: IEEE.

Recent papers in the SPRUWorking Paper Series:

December

2019-25. Mobilizing the Transformative Power of the Research System for Achieving the Sustainable Development Goals. *Matias Ramirez, Oscar Romero, Johan Schot and Felber Arroyave*

November

2019-24. Job Composition and Its Effect on UK Firms in the Digital Era. *Mabel Sánchez Barrioluengo*

2019-23. Start-up Subsidies: Does the Policy Instrument Matter? *Hanna Hottenrott and Robert Richstein*

2019-22. Organised Crime and Technology. *Mustafa Caglayan, Alessandro Flamini and Babak Jahanshahi*

October

2019-21. The Value of Data: Towards a Framework to Redistribute It. *Maria Savona*

September

2019-20. Teaming up with Large R&D Investors: Good or Bad for Knowledge Production and Diffusion? *Sara Amoroso and Simone Vannuccini*

2019-19. Experimental Innovation Policy. *Albert Bravo-Biosca*

August

2019-18. Relating Financial Systems to Sustainability Transitions: Challenges, Demands and Dimensions. *Chantal P. Naidoo*

Suggested citation:
Ekaterina Prytkova and Simone Vannuccini (2020). On the Basis of Brain: Neural-Network-Inspired Change in General Purpose Chips. SPRU Working Paper Series (SWPS), 2020-01: 1-34. ISSN 2057-6668. Available at: www.sussex.ac.uk/spru/swps2020-01

Science Policy Research Unit
University of Sussex, Falmer
Brighton BN1 9SL
United Kingdom

SPRU website: **www.sussex.ac.uk/business-school/spru**
SWPS website: **www.sussex.ac.uk/business-school/spru/research/swps**
Twitter: **@spru**

UNIVERSITY OF SUSSEX | BUSINESS SCHOOL