DISCUSSION PAPER SERIES

DP16267

Machine Learning the Carbon Footprint of Bitcoin Mining

Hector Calvo Pardo, Jose Olmo and Tullio Mancini

PUBLIC ECONOMICS

CEPR

ISSN 0265-8003

Machine Learning the Carbon Footprint of Bitcoin Mining

Hector Calvo Pardo, Jose Olmo and Tullio Mancini

Discussion Paper DP16267
Published 16 June 2021
Submitted 14 July 2020

Centre for Economic Policy Research
33 Great Sutton Street, London EC1V 0DX, UK
Tel: +44 (0)20 7183 8801
www.cepr.org

This Discussion Paper is issued under the auspices of the Centre's research programmes:

- Public Economics

Machine Learning the Carbon Footprint of Bitcoin Mining

Abstract

Building on an economic model of rational Bitcoin mining, we measure the carbon footprint of Bitcoin mining power consumption using feedforward neural networks. After reviewing the literature on deep learning methods, we find associated carbon footprints of 3.8038, 23.8313 and 19.83472 MtCOe for 2017, 2018 and 2019, which conform with recent estimates, lie within the economic model bounds while delivering much narrower confidence intervals, and yet raise alarming concerns, given recent evidence from climate-weather integrated models. We demonstrate how machine learning methods can contribute to non-for-profit pressing societal issues, like global warming, where data complexity and availability can be overcome.

JEL Classification: Q47, Q54, C45, C55, F55, F64

Keywords: Machine Learning, carbon footprint, cryptocurrencies, Nowcasting, Feed- forward Neural Networks, climate change

Hector Calvo Pardo - h.f.calvo-pardo@soton.ac.uk
University of Southampton, CPC

Jose Olmo - j.b.olmo@soton.ac.uk
University of Southampton, Universidad de Zaragoza

Tullio Mancini - t.mancini@soton.ac.uk
University of Southampton

Acknowledgements
Tullio Mancini acknowledges financial support from the University of Southampton Presidential Scholarship and Jose Olmo from 'Fundación Agencia Aragonesa para la Investigación y el Desarrollo'. The authors are grateful to José Miguel Hernández-Lobato for insightful comments on a previous version.

Machine Learning the Carbon Footprint of Bitcoin Mining*

Hector Calvo-Pardo† Tullio Mancini‡ Jose Olmo§

July 9, 2020

Abstract

Building on an economic model of rational Bitcoin mining, we measure the carbon footprint of Bitcoin mining power consumption using feedforward neural networks. After reviewing the literature on deep learning methods, we find associated carbon footprints of 3.8038, 23.8313 and 19.83472 MtCOe for 2017, 2018 and 2019, which conform with recent estimates, lie within the economic model bounds while delivering much narrower confidence intervals, and yet raise alarming concerns, given recent evidence from climate-weather integrated models. We demonstrate how machine learning methods can contribute to non-for-profit pressing societal issues, like global warming, where data complexity and availability can be overcome.

Keywords: Machine Learning, Carbon Footprint, Cryptocurrencies, Nowcasting, Feedforward Neural Networks, Climate Change.

JEL Codes: Q47, Q54, C45, C55, F55, F64.

*Corresponding author: Hector F. Calvo-Pardo, Department of Economics, University of Southampton. Highfield Campus, SO17 1BJ, Southampton. UK. E-mail: calvo@soton.ac.uk. Tullio Mancini acknowledges financial support from the University of Southampton Presidential Scholarship and Jose Olmo from 'Fundación Agencia Aragonesa para la Investigación y el Desarrollo'. The authors are grateful to José Miguel Hernández-Lobato for insightful comments on a previous version.

†University of Southampton, CPC and CEPR

‡University of Southampton.

§University of Southampton and Universidad de Zaragoza.

1 Introduction

Does Bitcoin mining contribute to climate change? Participation in the Bitcoin blockchain validation process[1] requires specialized hardware and vast amounts of electricity, translating into a significant carbon footprint. Mora et al. (2018) estimate that the 2017 carbon footprint of Bitcoin was 69 Mt of CO_2-equivalent ($MtCO_2e$), forecasting a violation of the Paris COP21 UNFCC agreement –limiting GHG emissions to keep temperatures within 2°C of pre-industrial levels– by 2040 due to Bitcoin cumulative emissions alone. At the heart of the controversy sparked, with various contributions revising downward Mora et al.'s (2018) projections (e.g. Houy, 2019; Masanet et al., 2019 or Stoll et al., 2019), is the difficulty in measuring the power consumption of the Bitcoin mining network (De Vries, 2018). Bitcoin miners are globally geo-located, facing very different energy costs, and employ hardware with unknown energy intensities. To overcome the significant constraints in estimating the daily power consumption associated with Bitcoin's blockchain, here we use machine learning (ML) methods, demonstrating their usefulness for pressing societal issues, like climate change.

A subset of ML methods, feedforward neural networks, are becoming increasingly popular due to their unrivaled performance in prediction tasks (LeCun et al., 2015). Feedforward neural networks, also called multilayer perceptrons (MLPs), have been developed since the mid-twentieth century, relying on joint advances from computer science, applied mathematics and information and probability theory. Their recent success stems from their theoretical ability to approximate unknown data generating processes (*Universal Approximation Theorem* and its variants), while handling large and complex datasets. They approximate or learn some unknown function of the data (or inputs) that generates an output, like the Bitcoin network energy consumption, assuming that information 'feeds forward' from the input, through the unknown function, to the output.[2] They are called neural networks (NN) because they are composed of many functions connected in a chain, where each link is called a layer, each of which consists of an array of nodes (or units). By adding layers and nodes within each layer, feedforward NNs (or deep neural networks, DNN) can approximate functions of increasing complexity. CO_2 emissions are complex to forecast, but having a reliable general purpose method to do so in a timely manner can inform progress towards keeping global temperatures from rising above 2°C, in addition to net-zero carbon emissions. Our main contribution is to provide a robust measure of the carbon

[1] The revolutionary element of Bitcoin is the underlying 'blockchain' technology. Instead of a trusted third party, incentivized network participants validate transactions and ensure the integrity of the network via the decentralized administration of a data protocol (also called 'proof-of-work'). The distributed ledger protocol created has since then been called the 'first blockchain'.

[2] This is in contrast to recurrent neural networks, where information is allowed to feed-back from the output to the model itself.

footprint associated with producing increasingly popular cryptocurrencies, like Bitcoin (BTC), as well as of the uncertainty associated with that measure.

To estimate a realistic level of daily electricity consumption to produce Bitcoins, we first calculate a lower and an upper limit based on Hayes' (2015) economic model of rational Bitcoin mining decisions. The lower limit corresponds to the lowest marginal cost for mining Bitcoins, as defined by a scenario in which all miners use the most efficient available hardware. The upper limit obtains when the least efficient technology for mining Bitcoins is employed instead. Based on IPO filings of major hardware manufacturers, insights on mining facility operations, and mining pool compositions, our DNN adopts as target output the market share weighted average of the daily energy efficiency deployed by operating miners, identified by their IP addresses. Our estimated level of electricity consumption is thus a conservative one, closely tracking Hayes'(2015) lower limit. As inputs, our DNN admits a comprehensive range of factors previously found to drive Bitcoin prices in different currencies, like (i) fundamental factors advocated by monetary economics (e.g. its usage in trade, money supply or price level); (ii) factors driving investors' interest in/attention to the crypto-currency (e.g. speculation or Bitcoin's role as safe haven); (iii) exchange rate hedging motives (see Kristoufek, 2015; Liu and Tsivinsky, 2018; McNally et al., 2018, or Jang and Lee, 2018), together with (iv) novel supply-side factors for both Bitcoin and ASIC mining chips producers, related to for-profit mining decisions, but excluding those employed in the construction of the upper and lower limits. Aggregated at the yearly frequency, we find Bitcoin mining energy consumption's ranging between 5.2384 and 43.1218 TWh in 2017, between 25.0786 and 80.4240 TWh in 2018, and between 27.0537 and 80.3026 TWh in 2019. Obtaining mean point estimates of daily power consumption within those economically meaningful limits provides substantial gains in accuracy relative to recent contributions in the literature, while externally validating our ML approach.

The carbon intensity associated with Bitcoin mining obtains from multiplying the estimated daily electricity consumption by the average emission factor of power generation, using the geolocation of IP-addresses, as Stoll et al. (2019) do.[3] Crucially, our novel approach also enables the construction of confidence intervals (CIs) around the estimated carbon footprint of Bitcoin mining, substantially narrowing down the associated uncertainty –currently measured by the difference between the carbon footprint of the upper and lower bounds, corresponding to the expected marginal revenue and marginal cost of Bitcoin network operating miners. When aggregated at a yearly frequency, the corresponding CO_2 estimates [and associated 0.95 CIs]

[3]As of November 2018, and relative to Stoll et al. (2019) who find a Bitcoin mining annual energy consumption level of 45.8 TWh and associated carbon emissions ranging between 22 and 22.9 $MtCO_2$, we find instead 48.2 TWh of electricity consumption, with annual carbon emissions ranging between 23.6 and 28.8 $MtCO_2$.

are, for the year 2017, 3.8038 $MtCO_2e$ [3.2151, 4.3925] $MtCO_2e$; for 2018, 23.8313 $MtCO_2e$ [22.1055, 25.5572] $MtCO_2e$; and for 2019, 19.83472 $MtCO_2e$ [18.4852, 21.1842] $MtCO_2e$. To provide an order of magnitude, the Bitcoin mining estimated fossil fuels emissions for the year 2018 are higher than the annual levels of fossil fuel emissions of (i) the US states of Maine (15.63 $MtCO_2e$), New Hampshire (13.55 $MtCO_2e$), Rhode Island (10.13 $MtCO_2e$) or South Dakota (14.6 $MtCO_2e$), of (ii) more than half the cumulative CO_2 flux from the Earth's 91 most actively degassing subaerial volcanoes (a 2005-2015 yearly average of 38.7 ± 2.9 CO_2e from Aiuppa et al., 2019), or of (iii) those of smaller countries, like Bolivia, Sudan or Lebanon (*Global Carbon Atlas*). A measure of the magnitude of the economic problem can be obtained from adopting the *social cost of carbon* (SCC) estimate of 62 USD per metric ton of CO_2 equivalent (Interagency Working Group, IWG, 2016) in 2007 USD: yearly, the Bitcoin mining SCC reliably lies between [\$199, 336, 200; \$272, 335, 000] for 2017; in [\$1, 370, 541, 000; \$1, 584, 546, 400] for 2018; and in [\$1, 146, 082, 400; \$1, 313, 420, 400] for 2019.

Relative to the aforementioned literature, the obtained point estimates and confidence intervals also represent a downward revision of the results reported by Mora et al. (2018), and are broadly in line with figures from Foteinis (2018), reporting global emissions for Bitcoin and Ethereum for 2017 of 43.9 $MtCO_2$ or from Stoll et al. (2019), reporting annual carbon emissions for Bitcoin mining in 2018 in the range 22.0 to 22.9 $MtCO_2$. Our estimates further revise downward the 2017 estimates provided by Houy (2019) or Dittmar and Praktiknjo (2019), who criticized Mora et al.'s (2018) inclusion of energy inefficient (unprofitable) mining rigs, reporting 15.5 $MtCO_2e$ for 2017, or those from Masanet et al. (2019), who report for 2017 an estimate of 15.7 $MtCO_2e$ arguing that Mora et al.'s (2018) projected tecnology adoption rates did not resemble the growth rates of Bitcoin usage. What makes them nevertheless extremely worrying is recent evidence from integrated weather-climate models (CMIP6), feeding into the Sixth Assessment Report of the Intergovernmental Panel on Climate Change (IPCC) 2021, reported in Williams et al. (2020). According to them, global temperatures may rise as much as 5°C, prompting the recent global call to urgent policy measures by IMF's Chief Economist Gita Gopinath in Davos (Switzerland, 2020).

The topic is extremely important considering the interest of national governments on cryptocurrencies (e.g. China), the possibility of issuing financial instruments solely on blockchain technologies (e.g. Bank of Australia and World Bank *bond-i*), while respecting the Paris Agreement. The social cost of carbon (SCC), associated with proof-of-work protocols, and the associated carbon tax, are important aspects that policy makers should consider when implementing blockchain technologies. Besides the gains in accuracy, we argue that ML methods present the additional significant advantages for enabling timeless public decision making regarding pressing complex social issues, just as they do in private sector for-profit decisions, e.g. business analytics,

new technology design, improvement or product adaptation and/or marketing. Being able to process bigger and increasingly complex data in raw form, ML techniques return tailored solutions in an automated manner. The significant 'entry cost' in terms of conceptual difficulty and computational time has significantly decreased over the last ten years, thanks to advancements in computational capacity, user-friendly software and increasing resources devoted to training and technology adoption, rendering their use commonplace. Although our main goal here is on prediction tasks, an incipient strand within the economics/econometrics literature focuses on interpretation tasks, seeking conditions under which valid (two-step) inference can be conducted after applying ML methods. Athey and Imbens (2019) report on progress made and challenges remaining in interpretation tasks.

To further reduce readers' entry cost, Section 2 introduces and quickly surveys the recent ML literature, focusing on deep learning, and describing our novel approach. The rest of the paper proceeds as follows: Section 3 reports the methodology and the data used to estimate Hayes' (2015) lower and upper bounds of Bitcoin mining power consumption and associated emission bounds. Section 4 demonstrates the usefulness for predicting the carbon footprint associated with Bitcoin mining of our deep learning approach ('optimized ReLu DNN'), delivering substantially narrower bounds that increase the reliability of the provided estimates. We also show that our approach outperforms when benchmarked against state-of-the-art ML methods. Finally, Section 5 concludes.

2 Literature Review

Machine learning (ML) technology is widespread nowadays: from web searches to content filtering on social networks to recommendations on e-commerce websites. ML identifies objects in images, transcribes speech into text, matches news items, posts or products with users' interests, and selects relevant results of search, making use of a class of techniques called deep learning. Deep learning allows computational models that are composed of multiple processing layers to learn representations of big complex datasets, uncovering intricate structure within them. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics, being increasingly present in consumer products such as cameras, smartphones or computerized personal assistants. For example, Apple's Siri, Amazon's Alexa, Google Now or Microsoft's Cortana employ deep neural networks to recognize, understand and answer human questions. But so far, they have not been applied to solving societal pressing issues, like quantifying the effects of greenhouse emissions on climate change.

This section briefly reviews the literature on machine learning (ML), and places deep learning

within it.[4] Emphasis is put on 'shallow' versus 'deep' neural network (NN) architectures, closing with a brief description of our novel approach to architecture optimization and construction of confidence intervals, applied in the following section to estimate the carbon footprint of Bitcoin network production.

2.1 Machine Learning Basics

ML aims to develop a computational relationship (formula/algorithm) between P inputs (predictors, features, explanatory or independent variables), $\mathbf{X} = \{...x_p...\}$, and K outputs (dependent or response variables), $\mathbf{y} = \{...y_k...\}$, for determining/predicting/estimating values for $\mathbf{y}$ given only the values of $\mathbf{X}$, in the presence of U unobserved/uncontrolled quantities $\mathbf{z} = \{...z_u...\}$:

$$y_k = g_k(...x_p...; ...z_u...), \forall k$$

To reflect the uncertainty associated with the unobserved inputs **z,** the above relationship is replaced by a statistical model:

$$y_k = f_k(...x_p...) + \varepsilon_k : \varepsilon_k \sim F_\varepsilon(\varepsilon_k), \mathbb{E}[\varepsilon_k|...x_p...] = 0, \forall k$$

where $f_k(...x_p...) = \mathbb{E}_\varepsilon[y_k|...x_p...]$. For simplicity, we drop the k subscript, indicating that we are assuming that there are separate models for each output k, ignoring that they depend on the same set of input variables[5]:

$$y = f(\mathbf{X}) + \varepsilon : f(\mathbf{X}) = \mathbb{E}_\varepsilon[y|\mathbf{X}] \tag{1}$$

i.e. to the extent that the error term ε is a random variable, the output variable y becomes a random variable. Specifying a set of observed input values $\mathbf{X}$, specifies a distribution of output $y-$values the mean of which is the target function $f(\mathbf{X})$. Input and output variables can be real or categorical, but categories can be always converted into 'indicators' or 'dummies' that are real-valued. An example of an output variable y is the carbon footprint of Bitcoin mining, input variables $\mathbf{X}$ of which are electricity prices, the energy efficiency of available mining hardware, drivers of Bitcoin prices, foreign currencies exchange rates against the USD or the country-specific carbon intensities of electricity consumed, among others. Finally examples of unobserved inputs $\mathbf{z}$ are the actual energy efficiency of mining hardware or the carbon intensities of different sources of electricity effectively employed.

ML algorithms can be broadly categorized as unsupervised or supervised. Unsupervised learning algorithms aim at uncovering useful properties of the structure of the input dataset,

[4] Friedman (1994) provides an early unifying review across the relevant disciplines (applied mathematics, statistics, engineering, artificial intelligence and connectionism), LeCun et al. (2015) provide a general overview of deep learning while Goodfellow et al. (2016) provide a thorough textbook treatment.

[5] In practice, strategies that treat the K outputs as a joint system often improve accuracy.

i.e. there is no y, and given that the true data generating process (DGP) $p_{\text{data}}(\mathbf{X})$ is unknown, the goal is to learn $p_{\text{data}}(\mathbf{X})$, or some useful properties of it, from a random sample of $i = 1...N$ realizations of input data only, $\{\mathbf{X}_i\}$, on the basis of which the empirical distribution $\widehat{p}_{\text{data}}(\mathbf{X})$ obtains. Letting $p_{\text{model}}(\mathbf{X};\boldsymbol{\theta})$ be a parametric family of probability distributions indexed by $\boldsymbol{\theta}$ that estimates the unknown true $p_{\text{data}}(\mathbf{X})$, unsupervised learning corresponds to finding the parameter vector $\boldsymbol{\theta}$ that minimizes the dissimilarity/distance between $p_{\text{model}}(\mathbf{X};\boldsymbol{\theta})$ and $\widehat{p}_{\text{data}}(\mathbf{X})$:[6]

$$\boldsymbol{\theta}_{ML} \in \arg\min_{\boldsymbol{\theta}} D_{KL}(\widehat{p}_{\text{data}}||p_{\text{model}}) = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{X}\sim\widehat{p}_{\text{data}}}[\log \widehat{p}_{\text{data}}(\mathbf{X}) - \log p_{\text{model}}(\mathbf{X};\boldsymbol{\theta})] \tag{2}$$

noticing that $\boldsymbol{\theta}_{ML}$ *is* the maximum likelihood estimator[7], and $D_{KL}(\widehat{p}_{\text{data}}||p_{\text{model}})$ denotes the Kullback-Leibler divergence. The cross-entropy is then simply $-\mathbb{E}_{\mathbf{X}\sim\widehat{p}_{\text{data}}} \log p_{\text{model}}(\mathbf{X};\boldsymbol{\theta})$: since $\log \widehat{p}_{\text{data}}(\mathbf{X})$ does not depend on $\boldsymbol{\theta}$, minimizing D_{KL} is equivalent to minimizing the cross-entropy, or 'empirical risk minimization', e.g. the mean-squared error is the cross-entropy between the empirical distribution and a Gaussian model. In ML, the cross-entropy is called 'cost function', $J(\boldsymbol{\theta})$, while in statistics it is called the 'loss function', $l(\boldsymbol{\theta}) \equiv L[\widehat{p}_{\text{data}}(\mathbf{X}), p_{\text{model}}(\mathbf{X};\boldsymbol{\theta})]$.

Instead, supervised learning algorithms aim to obtain a useful approximation $\widehat{f}(\mathbf{X})$ to the true (unknown) 'target' function $f(\mathbf{X})$ in (1), by modifying (under constraints) the input/output relationship $\widehat{f}(\mathbf{X})$ that it produces, in response to differences $\{y_i - \widehat{y}_i\}$ (errors) between the predicted $\widehat{y}_i = \widehat{f}(\mathbf{X}_i)$ and real y_i system outputs[8]:

$$\widehat{f}(\mathbf{X}) \in \arg\min_{g(X)} \frac{1}{N} \sum_{i=1}^{N} L[y_i, g(\mathbf{X}_i)] \tag{3}$$

where $L(.,.)$ is the 'loss function', or a measure of distance (error) between y_i and $\widehat{y}_i = \widehat{f}(\mathbf{X}_i)$. Common examples are $L[y_i, \widehat{y}_i] = |y_i - \widehat{y}_i|$ which plugged into (3) corresponds to selecting the $\widehat{f}(\mathbf{X}) = Med_{y,\mathbf{X}\sim\widehat{p}_{\text{data}}}[y|\mathbf{X}]$ that minimizes the Mean Absolute Error (MAE), or $L[y_i, \widehat{y}_i] = [y_i - \widehat{y}_i]^2$ which selects the $\widehat{f}(\mathbf{X}) = \mathbb{E}_{y,\mathbf{X}\sim\widehat{p}_{\text{data}}}[y|\mathbf{X}]$ that minimizes the Mean Squared Error (MSE) in (3). Alternatively stated, consider a random sample of $i = 1...N$ realizations, $\{y_i, \mathbf{X}_i\}$, constituting the empirical distribution $\widehat{p}_{\text{data}}(y, \mathbf{X})$, the goal of supervised learning is

[6] Other popular unsupervised deep learning models, not necessarily parametric, are *k-means clustering*, *auto-encoders* and *generative adversarial networks* (GANs).

[7] Since $\boldsymbol{\theta}_{ML} = \arg\max_{\boldsymbol{\theta}} p_{\text{model}}(\mathbf{X};\boldsymbol{\theta})$ and $p_{\text{model}}(\mathbf{X};\boldsymbol{\theta}) = \prod_{i=1}^{N} p_{\text{model}}(\mathbf{X}_i;\boldsymbol{\theta})$ which, after taking logs and dividing by N, is equivalent to

$$\boldsymbol{\theta}_{ML} = \arg\max_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \log p_{\text{model}}(\mathbf{X}_i;\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{X}\sim\widehat{p}_{\text{data}}}[\log p_{\text{model}}(\mathbf{X};\boldsymbol{\theta})],$$

by the analogy principle. Maximum likelihood is the preferred estimator for machine learning (ML) because of its theoretical properties of consistency and efficiency. In cases where these cannot be guaranteed, a different estimator is adopted instead.

[8] Formally, this is a variational problem in that the argument of the optimization is a function of functions, and requires calculus of variations to solve it. See Goodfellow et al. (2016) for a textbook treatment, and Cover and Thomas (2006) for a more advanced treatment.

to learn to predict y from $\mathbf{X}$, estimating $p(y|\mathbf{X})$. Letting $p_{\text{model}}(y|\mathbf{X};\boldsymbol{\theta})$ be a parametric family of probability distributions indexed by $\boldsymbol{\theta}$ that estimates the unknown true $p(y|\mathbf{X})$, supervised learning corresponds to finding the parameter vector $\boldsymbol{\theta}$ that minimizes the dissimilarity/distance between $p_{\text{model}}(y|\mathbf{X};\boldsymbol{\theta})$ and $\widehat{p}_{\text{data}}(y|\mathbf{X})$:[9]

$$\boldsymbol{\theta}_{ML} \in \arg\min_{\boldsymbol{\theta}} D_{KL}(\widehat{p}_{\text{data}}||p_{\text{model}}) = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{y,\mathbf{X}\sim\widehat{p}_{\text{data}}}[\log \widehat{p}_{\text{data}}(y|\mathbf{X}) - \log p_{\text{model}}(y|\mathbf{X};\boldsymbol{\theta})] \quad (4)$$

and again, solving (4) is equivalent to cross-entropy minimization, $\min_{\boldsymbol{\theta}} -\mathbb{E}_{y,\mathbf{X}\sim\widehat{p}_{\text{data}}} \log p_{\text{model}}(y|\mathbf{X};\boldsymbol{\theta})$. As an example, notice that if we set $p_{\text{model}}(y|\mathbf{X};\boldsymbol{\theta}) = N(g(\mathbf{X};\boldsymbol{\theta}),\sigma^2)$ in (4) we obtain:

$$\begin{aligned}
\min_{\boldsymbol{\theta}} -\mathbb{E}_{y,\mathbf{X}\sim\widehat{p}_{\text{data}}} \log p_{\text{model}}(y|\mathbf{X};\boldsymbol{\theta}) &= \min_{\boldsymbol{\theta}} -\frac{1}{N}\textstyle\sum_{i=1}^{N} \log p_{\text{model}}(y_i|\mathbf{X}_i;\boldsymbol{\theta}) \\
&= \min_{\boldsymbol{\theta}} \left\{ \log(\sigma[2\pi])^{1/2} + \left[2\sigma^2\right]^{-1} \underbrace{\frac{1}{N}\sum_{i=1}^{N}[y_i - g(\mathbf{X}_i;\boldsymbol{\theta})]^2}_{\equiv MSE(\boldsymbol{\theta})} \right\}
\end{aligned}$$

and therefore, cross-entropy minimization corresponds to mean squared error (MSE) minimization when the model is hypothesized to be Gaussian with mean $g(\mathbf{X};\boldsymbol{\theta})$. In addition, this example shows that optimally choosing the parameter vector $\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta}_{ML}$, characterizing $\widehat{f}(\mathbf{X}) = g(\mathbf{X};\widehat{\boldsymbol{\theta}})$, is equivalent to solving (3) when $L[y_i,\widehat{y}_i] = [y_i - \widehat{y}_i]^2$:

$$\widehat{f}(\mathbf{X}) \in \arg\min_{g(X)} \mathbb{E}_{y,\mathbf{X}\sim\widehat{p}_{\text{data}}}[y - g(\mathbf{X})]^2 = \arg\min_{g(X)} \frac{1}{N}\textstyle\sum_{i=1}^{N}[y_i - g(\mathbf{X}_i)]^2$$

Therefore, approximating/learning the unknown function $f(\mathbf{X})$ corresponds to estimating the unknown true conditional probability $p(y|\mathbf{X})$, once we conjecture a parameterization $p_{\text{model}}(y|\mathbf{X};\boldsymbol{\theta})$ for it.

Notice that (3) is the available sample $\{y_i,\mathbf{X}_i\}$ analog to solving for the global prediction error in (1):

$$\widehat{f} \in \arg\min_{g(X)} \int \{\mathbb{E}_{\varepsilon} L[f(\mathbf{X}) + \varepsilon, g(\mathbf{X})]\}\, p_{\text{data}}(\mathbf{X}) d\mathbf{X} \quad (5)$$

where $p_{\text{data}}(\mathbf{X})$ is the unknown true data generating process. Problem (5) defines the target performance measure for prediction in supervised learning/function approximation[10]: as future new

[9] Popular supervised deep learning models, not necessarily parametric, are *support vector machines* (SVMs) based on kernel methods, *k-nearest neighbor regression* or *decision trees*. See more below.

[10] As an example, replace $L[y_i,\widehat{y}_i] = [y_i - \widehat{y}_i]^2$ in (5) to obtain the standard expressions for the bias-variance trade-off in the Mean Squared Error (MSE):

$$\begin{aligned}
\widehat{f} &\in \arg\min_{g(X)} \int \{\mathbb{E}_{\varepsilon}[f(\mathbf{X}) + \varepsilon - g(\mathbf{X})]^2\}\, p_{\text{data}}(\mathbf{X}) d\mathbf{X} \\
&= \arg\min_{g(\mathbf{X})} \underbrace{\int [f(\mathbf{X}) - g(\mathbf{X})]^2 p_{\text{data}}(\mathbf{X}) d\mathbf{X}}_{\text{MSE}(\widehat{f})} \\
&\quad + \underbrace{\int \{\mathbb{E}_{\varepsilon}[\varepsilon^2|\mathbf{X}]\}\, p_{\text{data}}(\mathbf{X}) d\mathbf{X}}_{\text{Variance of the noise } \varepsilon}
\end{aligned}$$

input only observations become available, collected in a prediction or test sample '$\top$', $\{y_i, \mathbf{X}_i\}_{i=1}^{N^\top}$, we want to predict (estimate) a likely output value using $\widehat{f}(\mathbf{X}_i)$, $\widehat{y}_i = \widehat{f}(\mathbf{X}_i)$, where $\widehat{f}(\mathbf{X})$ was obtained from (3) exploiting the available sample, $\{y_i, \mathbf{X}_i\}_{i=1}^{N}$. Computing then $\frac{1}{N^\top}\sum_{i=1}^{N^\top} L[y_i, \widehat{y}_i]$ allows the researcher to evaluate the out-of-sample performance of the algorithm/function approximation $\widehat{f}(\mathbf{X})$, showing that accurate approximation and future prediction are one and the same objective[11]. As future data is unavailable, the standard practice is to divide the available sample $\{y_i, \mathbf{X}_i\}_{i=1}^{N}$ into two disjoint parts: a training/learning sample '$\llcorner$' $\{y_i, \mathbf{X}_i\}_{i=1}^{N^\llcorner}$ in (3) where $\widehat{f}(\mathbf{X})$ obtains, and a prediction/test sample $\{y_i, \mathbf{X}_i\}_{i=1}^{N^\top}$ where the out-of-sample predictive performance of $\widehat{f}(\mathbf{X})$ is evaluated, so that $N = N^\llcorner + N^\top$. More complex forms of the unknown target function $f(\mathbf{X})$ naturally call for bigger training samples $N^\llcorner$ to obtain better representations/approximations $\widehat{f}(\mathbf{X})$. However, this comes at the expense of increasing the chances of $\widehat{f}(\mathbf{X})$ 'overfitting'. Overfitting happens when a model that represents the training data very well, represents very poorly unseen data $N^\top$ in the 'prediction/test phase'.[12] The reason lies on the 'curse-of-dimensionality' that the complexity of the unknown target function creates: as the number of input variables P upon which $f(\mathbf{X})$ depends increases, the necessary sample size to accurately approximate $f(\mathbf{X})$ grows exponentially, i.e. at a rate $N^{1/P}$, rendering all training

---

where the MSE($\widehat{f}$) denotes the MSE$\widehat{f}(\mathbf{X})$ averaged over all training samples of size N that could be realized from the system with probabilities governed by $p_{\text{data}}(\mathbf{X})$ and $F_\varepsilon(\varepsilon)$. It can be further decomposed as:

$$MSE(\widehat{f}) \equiv \int MSE[\widehat{f}(\mathbf{X})]p_{\text{data}}(\mathbf{X})d\mathbf{X} = \int Var[\widehat{f}(\mathbf{X})]p_{\text{data}}(\mathbf{X})d\mathbf{X} + \int Bias^2[\widehat{f}(\mathbf{X})]p_{\text{data}}(\mathbf{X})d\mathbf{X}$$

where $Bias^2[\widehat{f}(\mathbf{X})] = \{f(\mathbf{X}) - \mathbb{E}_\varepsilon[\widehat{f}(\mathbf{X})]\}^2$ measures the square of the difference between the target function $f(\mathbf{X})$ and the average approximation value at a particular sample $\mathbf{X}$, $\mathbb{E}_\varepsilon[\widehat{f}(\mathbf{X})]$.

[11] Yet another goal of supervised learning is interpretation, as opposed to prediction: there, interest lies in the structural form of the approximating function obtained from (3) to understand the mechanism that produced the data. Identification of the input variables that are most relevant to explain the variation in output, or the nature of that dependence and how it changes with changes in other inputs are instead the primary objectives, and the aim is to understand how does the system work.

[12] An intuitive way to understand why is as follows. Suppose that we have a sample of size N with which we are trying to approximate a function of N variables $f(x_1, ..., x_N)$. If Kolmogorov's conjecture was right, we could instead approximate a degree N polynomial function of just one variable, say x_1, $f(x_1, ..., x_N) = g(x_1) = \sum_{i=1}^{N} a_i x_1^i$ and problem (3) would reduce to a parametric least squares (OLS) solution:

$$\widehat{f}(\mathbf{X}) = f(\mathbf{X};\widehat{\mathbf{a}}) \in \arg\min_{\{a_i\}} \frac{1}{N}\sum_{i=1}^{N}[y_i - \sum_{i=1}^{N} a_i x_1^i]^2$$

Since there are N normal equations (one for each) in N unknowns (sample observations), we would obtain a unique solution $\widehat{\mathbf{a}}$, corresponding to a 'perfect fit' of the sample/training data. If then one more sample observation was collected, $N^\top = \{y^{N+1}, x_1^{N+1}\}$, and we wanted to test the predictive ability of $\widehat{f}(\mathbf{X}) = \sum_{i=1}^{N} \widehat{a}_i x_1^i$, almost with probability one $y^{N+1} \neq \widehat{y}^{N+1} = \sum_{i=1}^{N} \widehat{a}_i (x_1^{N+1})^i$, i.e. the prediction error $[y^{N+1} - \widehat{y}^{N+1}]^2$ will be very big, indicating 'overfitting'. In big data problems, where $P > N$ (or is close to N), overfitting means that the approximation obtained from (3) will almost surely perform poorly in unseen data, i.e. in (5).

samples very sparsely populated.[13]

Because $N^{\llcorner}$ is finite, problem (3) does not have a unique solution[14]. One must therefore restrict the set of admissible functions to a smaller set $\mathcal{G}$ than the set of all possible functions $g(\mathbf{X})$. To see the effect of restricting the class of admissible functions in (3), denote by $f^*(\mathbf{X}) \in \arg\min_{g(X)} \frac{1}{N^{\top}} \sum_{i=1}^{N^{\top}} L[y_i, g(\mathbf{X}_i)]$ and by $f^*_{\mathcal{G}}(\mathbf{X}) \in \arg\min_{g(X)\in G} \frac{1}{N^{\top}} \sum_{i=1}^{N^{\top}} L[y_i, g(\mathbf{X}_i)]$ the best approximation in the unrestricted and restricted classes of functions respectively, both in terms of out-of-sample performance, $N^{\top}$. The difference in out-of-sample performance between the solution from (3) and $f^*(\mathbf{X})$ ('excess test error' $\mathcal{E}$) can then be decomposed as follows:

$$\begin{aligned} \mathcal{E} &\equiv \frac{1}{N^{\top}} \sum_{i=1}^{N^{\top}} L[y_i, \widehat{f}(\mathbf{X}_i)] - \frac{1}{N^{\top}} \sum_{i=1}^{N^{\top}} L[y_i, f^*(\mathbf{X}_i)] \\ &= \frac{1}{N^{\top}} \sum_{i=1}^{N^{\top}} \underbrace{\{L[y_i, \widehat{f}(\mathbf{X}_i)] - L[y_i, f^*_{\mathcal{G}}(\mathbf{X}_i)]\}}_{\text{Estimation error}} + \underbrace{\{L[y_i, f^*_{\mathcal{G}}(\mathbf{X}_i)] - L[y_i, f^*(\mathbf{X}_i)]\}}_{\text{Approximation error}}. \end{aligned}$$

The *approximation error* increases the more restrictive the class of functions $\mathcal{G}$ is, unless the true unknown target function $f(\mathbf{X})$ happens to belong to $\mathcal{G}$, in which case $f^*_{\mathcal{G}}(\mathbf{X}) = f^*(\mathbf{X})$. The *estimation error* depends on how good the algorithm/approximation $\widehat{f}(\mathbf{X})$ is (1st term) as well as on how well the selected class of functions $\mathcal{G}$ can best represent the complexity of the unknown target function $f(\mathbf{X})$ (2nd term). 'Universal approximators' for the class of all continuous target functions $f(\mathbf{X})$ are classes of functions $\mathcal{G} = \{g(\mathbf{X}) : g(\mathbf{X}) = \sum_{z=1}^{Z} a_z b(\mathbf{X}|\boldsymbol{\gamma}_z), \boldsymbol{\gamma}_z \in \mathbb{R}^q\}$ that *could* exactly represent $f(\mathbf{X})$ if the sample size was not finite, i.e. $f(\mathbf{X}) = \sum_{z=1}^{\infty} a^*_z b(\mathbf{X}|\boldsymbol{\gamma}_z)$ for some set of expansion coefficient values $\{a^*_z\}_{z=1}^{\infty}$. Therefore, universal approximators minimize the approximation error and the estimation error, minimizing the out-of-sample performance difference $\mathcal{E}$ between the solution from (3) and $f^*(\mathbf{X})$, i.e. if the training sample size was infinite, $\lim_{N^{\llcorner}\to\infty} \widehat{f}(\mathbf{X}) = f(\mathbf{X}; \widehat{\boldsymbol{\theta}}) = \sum_{z=1}^{\infty} \widehat{a}_z b(\mathbf{X}|\widehat{\boldsymbol{\gamma}}_z) = \sum_{z=1}^{\infty} a^*_z b(\mathbf{X}|\boldsymbol{\gamma}_z) = f(\mathbf{X})$ with $\widehat{\boldsymbol{\theta}} = \widehat{\boldsymbol{\theta}}_{ML} = \{\widehat{a}_z, \widehat{\boldsymbol{\gamma}}_z\}_{z=1}^{\infty}$, and therefore $\lim_{N^{\top}\to\infty} \frac{1}{N^{\top}} \sum_{i=1}^{N^{\top}} L[y_i, \widehat{f}(\mathbf{X}_i)] = 0$ ('Oracle property'). But because the training sample size is finite, $Z < \infty$ and $\frac{1}{N^{\top}} \sum_{i=1}^{N^{\top}} L[y_i, \widehat{f}(\mathbf{X}_i)] > 0$. Choosing Z corresponds then to 'model selection': as entries $\{a_z\}_{z=1}^{Z}$ are added, the approximation is able to better fit the training data, increasing the variance component of (5) but decreasing the bias. The bias decreases because adding entries enlarges the function space spanned by the approximation $\widehat{f}(\mathbf{X})$. With a finite sample size, the goal is to choose a small Z that keeps the variance and the bias small, so that (5) can be expected to remain small.

In general, the choice of the set of admissible functions $\mathcal{G}$ is based on considerations outside

[13] Note that this is the case even if we set $\varepsilon = 0$ in (1), converting (3) into an interpolation problem, i.e. reducing the MSPE to an MSE-only problem still requires a large enough training sample for the approximation to be accurate.

[14] If $N^{\llcorner} = +\infty$ (and with an infinitely fast computer) we would directly compute $f(\mathbf{X})$ from (1) predicting the mean of y for each value of $\mathbf{X}$.

the data and is usually done by the choice of a learning method.[15] Choosing a learning method can be modeled as adding a penalty term $\lambda\Omega[g(\mathbf{X})]$ to restrict solutions to (3):

$$\widehat{f}(\mathbf{X};\lambda) \in \arg\min_{g(X)} \frac{1}{N^{\llcorner}} \sum_{i=1}^{N^{\llcorner}} L[y_i, g(\mathbf{X}_i)] + \lambda\Omega[g(\mathbf{X})] \tag{6}$$

where λ ('regularization parameter') modulates the strength of the penalty functional $\Omega[.]$ over all possible functions $g(\mathbf{X})$.[16] For example, restricting $g(\mathbf{X}) \in \mathcal{G}$ as above can be achieved by setting $\Omega[g(\mathbf{X})] = H\{bias^2[g(\mathbf{X})]\}$ with $H\{h\} = 0 \cdot \mathbf{1}_{\{h=0\}} + \infty \cdot \mathbf{1}_{\{h\neq 0\}}$ (with the convention that $\infty \cdot 0 = 0$), since when $h = 0 = bias^2[g(\mathbf{X})] \Leftrightarrow g(\mathbf{X};\widehat{\boldsymbol{\theta}}) = \sum_{z=1}^{Z} \widehat{a}_z b(\mathbf{X}|\widehat{\boldsymbol{\gamma}}_z)$, i.e. learning $\widehat{f}(\mathbf{X};\lambda)$ in (3) reduces to parameter learning, $\widehat{f}(\mathbf{X};\lambda) = g(\mathbf{X};\widehat{\boldsymbol{\theta}},\lambda)$ where $\boldsymbol{\theta} = \{a_z, \boldsymbol{\gamma}_z\}_{z=1}^{Z}$.

Additional parametric or non-parametric penalty terms can be added to (6), with the result of further restricting the solutions in the approximation subspace of $\mathcal{G}$ that respect that particular penalty. By the addition of a penalty term (or 'regularization') the aim is to improve the out-of-sample performance of the approximation $\widehat{f}(\mathbf{X};\lambda)$, reducing its chances to 'overfit', without affecting its training error. Non-parametric penalties can be of the form $\Omega[g(\mathbf{X})] = \int |Dg(\mathbf{X})|^2 d\mathbf{X}$ where, for example, $|Dg(\mathbf{X})|^2 = \sum_{j=1}^{n} (\frac{\partial g}{\partial x_j})^2$ is the norm of the gradient of the functions in the class, with larger values of λ penalizing functions that oscillate more (i.e. that are 'less smooth').

[15] The class of functions $g(\mathbf{X}) = \sum_{m=1}^{M} a_m b(\mathbf{X}|\boldsymbol{\gamma}_m), \boldsymbol{\gamma}_m \in \mathbb{R}^q$ are commonly known as 'dictionaries'. The choice of a learning method selects a particular dictionary. Examples of dictionaries that are universal approximators are feed-forward neural networks, radial basis functions, recursive partitioning tree-structured methods and tensor product methods. See Friedman (1994) for additional details.

[16] The choice of a penalty functional is made on the basis of 'outside the data information' about the unknown target $f(\mathbf{X})$, e.g. on the basis of a prior over the class of models $g(\mathbf{X})$, $\Pr[g(\mathbf{X})]$. A natural choice for $\widehat{f}(\mathbf{X})$ would then be the function that is most probable given the data:

$$\widehat{f}(\mathbf{X}) \in \arg\max_{g(X)} \Pr[g(\mathbf{X})|\{y_i, \mathbf{X}_i\}] \tag{7}$$

which is known as maximum a posteriori probability (MAP) estimate. According to Bayes' theorem, the probability of a model given the training data is proportional to the likelihood that the training data has been generated by the model (and is therefore a model $F_\varepsilon(\varepsilon)$ for the error ε) times the probability of the model:

$$\Pr[g(\mathbf{X})|\{y_i, \mathbf{X}_i\}] \sim \Pr[\{y_i, \mathbf{X}_i\}|g(\mathbf{X})] \Pr[g(\mathbf{X})], \tag{8}$$

If $F_\varepsilon(\varepsilon) = N(0, \sigma^2)$ then (1) implies that

$$\Pr[\{y_i, \mathbf{X}_i\}|g(\mathbf{X})] = \Pr[\{\mathbf{X}_i\}] \prod_{i=1}^{N^{\llcorner}} (2\pi\sigma)^{-1} \exp\{-\varepsilon_i^2/2\sigma^2\}$$

with $\varepsilon_i = y_i - g(\mathbf{X}_i)$. Substituting the above expression into (8), taking logs and discarding terms not involving $g(\mathbf{X})$ yields an equivalent expression to (7):

$$\widehat{f}(\mathbf{X}) \in \arg\min_{g(X)} \frac{1}{\sigma^2} \sum_{i=1}^{N^{\llcorner}} [y_i - g(\mathbf{X}_i)]^2 - 2\log\Pr[g(\mathbf{X})]$$

which coincides with (6) if $L(.,.)$ is the quadratic loss function and $\lambda\Omega[g(\mathbf{X})] = -2\sigma^2 \log\Pr[g(\mathbf{X})]$. $\lambda\Omega[g(\mathbf{X})]$ naturally captures that reductions in the noise variance σ^2 lead to increasing weight on the training data part $\Pr[\{y_i, \mathbf{X}_i\}|g(\mathbf{X})]$ in determining the approximation $\widehat{f}(\mathbf{X})$, relative to the prior $\Pr[g(\mathbf{X})]$.

Parametric penalties would instead penalize functions $g(\mathbf{X})$ not in a particular parametric family $k(\mathbf{X}|\boldsymbol{\theta})$ **:** $g(\mathbf{X}) \notin \{k(\mathbf{X}|\boldsymbol{\theta}), \boldsymbol{\theta} \in \mathbb{R}^q\} \Longrightarrow \Omega[g(\mathbf{X})] = \infty$, transforming (3) into an equivalent parameter estimation problem:

$$\widehat{\boldsymbol{\theta}}_\lambda \in \arg\min_{\boldsymbol{\theta}} \frac{1}{N^{\llcorner}} \sum_{i=1}^{N^{\llcorner}} L[y_i, k(\mathbf{X}_i|\boldsymbol{\theta})] + \lambda \varpi[\boldsymbol{\theta}] \tag{9}$$

where different forms form $\varpi[\boldsymbol{\theta}]$ admit commonly studied cases, like (i) 'ridge' (L^2 regularization): $\varpi[\boldsymbol{\theta}] = \sum_{j=1}^{q} \theta_j^2$, penalizing approximations with large parameter values[17]; (ii) 'subset selection': $\varpi[\boldsymbol{\theta}] = \sum_{j=1}^{q} \mathbf{1}_{\{\theta_j \neq 0\}}$, which penalizes approximations with a large number of parameters (requiring combinatorial optimization); (iii) 'bridge': $\varpi_v[\boldsymbol{\theta}] = \sum_{j=1}^{q} |\theta_j|^v$, which coincides with 'ridge' when $v = 2$ and is a continuous approximation of the subset selection penalty as $v \to 0$. When $v = 1$, L^1 regularization obtains, akin to the 'least absolute shrinkage and selection operator', LASSO[18]; (iv) 'weight decay': $\varpi_w[\boldsymbol{\theta}] = \sum_{j=1}^{q} \frac{(\theta_j/w)^2}{1+(\theta_j/w)^2}$ approaches 'ridge' as $w \to \infty$ and subset selection as $w \to 0$. Smaller values of v and w privilege approximations with a small number of parameters. (v) '(Stochastic) Gradient descent': $\varpi[\boldsymbol{\theta}] = \frac{1}{N^{\llcorner}} \sum_{i=1}^{N^{\llcorner}} L[y_i, k(\mathbf{X}_i|\boldsymbol{\theta})]$**,** which penalizes 'paths' that do not follow the 'steepest descent', $\nabla_{\boldsymbol{\theta}} \varpi[\boldsymbol{\theta}] = \frac{1}{N^{\llcorner}} \sum_{i=1}^{N^{\llcorner}} \nabla_{\boldsymbol{\theta}} L[y_i, k(\mathbf{X}_i|\boldsymbol{\theta})]$**,** when searching for the value $\widehat{\boldsymbol{\theta}}_\lambda$ that minimizes (9) with $\widehat{f}(\mathbf{X}; \lambda) = k(\mathbf{X}|\widehat{\boldsymbol{\theta}})$, i.e. a high value of λ privileges '$\tau-$paths' $\boldsymbol{\theta}_{\tau+1} = \boldsymbol{\theta}_\tau - \epsilon \nabla_{\boldsymbol{\theta}} \varpi[\boldsymbol{\theta}_\tau]$ that reach $\widehat{\boldsymbol{\theta}}_\lambda$ taking the least possible number of steps τ, each of which depends on ϵ or 'learning rate'. Since ϵ governs the strength of the gradient $\nabla_{\boldsymbol{\theta}} \varpi[\boldsymbol{\theta}_\tau]$ in the updating of $\boldsymbol{\theta}_\tau$, choosing λ is equivalent to the choice of ϵ, a free hyperparameter to be 'fine tuned' or optimized during training. When instead of using all available $N^{\llcorner}$ observations in the training sample, we subsample randomly from $\{y_i, \mathbf{X}_i\}$ and form a 'minibatch' with $B < N^{\llcorner}$ observations, $\varpi[\boldsymbol{\theta}] = \frac{1}{B} \sum_{i=1}^{B} L[y_i, k(\mathbf{X}_i|\boldsymbol{\theta})]$ is called a 'stochastic gradient descent (SGD) penalty'. SGD can be combined with 'momentum', where the size of the updating step depends on how large an exponentially decaying moving average sequence of past gradients is, α **:** $\boldsymbol{\theta}_{\tau+1} = \boldsymbol{\theta}_\tau - \frac{\epsilon}{1-\alpha} \nabla_{\boldsymbol{\theta}} \varpi[\boldsymbol{\theta}_\tau]$. Momentum adds then another hyperparameter α, with larger values of $\alpha \in (0, 1)$ corresponding to a higher reliance on previous gradients, leading to a larger step size when updating. Current optimization methods like AdaGrad, RMSProp or Adam, supplement SGD (with or without 'momentum') to allow the learning rate ϵ to 'adapt', shrinking or expanding according to the entire history (e.g. Adam[19])

[17] 'Early stopping' the number of training iterations ('epochs') over the learning sample once the out-of-sample performance of the approximation starts to increase, can be shown to be equivalent to L^2 regularization (Goodfellow et al., 2016). Similarly, 'dropout' when applied to neural network (NN) methods, has been shown to be equivalent to L^2 regularization with a penalty strength parameter λ inversely proportional to the precision of the prior of a deep Gaussian process characterizing the NN parameters (Gal and Ghahramani, 2016).

[18] Just as we saw in the previous F.N. that L^2 regularization is equivalent to MAP Bayesian inference with a Gaussian prior over $\boldsymbol{\theta}$**,** L^1 regularization is equivalent to MAP Bayesian inference with an isotropic Laplace distribution prior $p(\boldsymbol{\theta})$ over $\boldsymbol{\theta}$**,** $\log p(\boldsymbol{\theta}) = \sum_j \log Laplace(\theta_j; 0, \frac{1}{\lambda}) = -\lambda \sum_{j=1}^{q} |\theta_j| + n \log \lambda - n \log 2$.

[19] Adam combines RSMProp and momentum, which is directly incorporated with exponential decay rates,

or to an exponentially decaying average (e.g. RMSProp[20]) of the squared gradient, so that the updating can converge even faster. Back-propagation is the method to compute the gradient of the cost function in (9), $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{N^{\llcorner}} \sum_{i=1}^{N^{\llcorner}} \nabla_{\boldsymbol{\theta}} L[y_i, k(\mathbf{X}_i|\boldsymbol{\theta})] + \lambda \nabla_{\boldsymbol{\theta}} \varpi[\boldsymbol{\theta}]$, which itself is a function of the gradients of the loss function and penalty terms. Those gradients are computed 'backwards' as dictated by the 'chain rule of calculus', since they are compositions of functions of the parameters $\boldsymbol{\theta}$. Once those gradients are computed, SGD or other optimization algorithms are used to perform the learning/approximation exploiting them.

Finally 'bagging' ('bootstrap aggregating') is also a powerful regularization method that can combine parametric and non-parametric penalties. It involves creating k different datasets from the training sample $N^{\llcorner}$ by sampling with replacement $N^{\mathrm{k}} = N^{\llcorner}$ observations, and solving (6) on each of the k different training datasets, $\widehat{f}_{\mathrm{k}}(\mathbf{X}; \lambda)$. The out-of-sample performance of the k-ensemble predictor is then $\frac{1}{N^{\top}} \sum_{i=1}^{N^{\top}} \frac{1}{\mathrm{k}} L[y_i, \widehat{f}_{\mathrm{k}}(\mathbf{X}_i; \lambda)]$. Since sampling is done with replacement, each dataset k is missing some of the observations from the original dataset $N^{\llcorner}$ with high probability, resulting in different approximations $\widehat{f}_{\mathrm{k}}(\mathbf{X}; \lambda)$ which make different errors in the test sample $N^{\top}$. Those errors will tend to cancel out if sampling is random, improving the out-of-sample performance of the k-ensemble model relative to its members.

How is λ determined? Since choosing the strength of the penalty λ determines the solution approximation $\widehat{f}(\mathbf{X}; \lambda)$ to (6) –and hence (9)–, this is referred to as 'model selection'. Ideally one would like to choose the λ that maximizes the out-of-sample performance of $\widehat{f}(\mathbf{X}; \lambda)$:

$$\widehat{\lambda} \in \arg \underset{\lambda}{min} \frac{1}{N^{\top}} \sum_{i=1}^{N^{\top}} L[y_i, \widehat{f}(\mathbf{X}_i; \lambda)]. \tag{10}$$

But different 'splittings' of the available sample into complementary learning and test subsamples, $N = N^{\llcorner} + N^{\top}$, are going to provide different values of $\widehat{\lambda}$. To avoid the computational burden associated with computing $\widehat{\lambda}$ for all possible assignments $\binom{N}{N^{\llcorner}}$ and then minimizing the average over these replications, this process is instead approximated by dividing the learning sample into K disjoint subsamples of approximately equal size, $N^K : N^{\llcorner} = KN^K$, each of which is used as 'test sample', $N^{\top} = N^K$, in (10). The complement sample, $N - N^K = N^{\llcorner K}$, is used as training sample in (6) to obtain K different approximations $\widehat{f}_K(\mathbf{X}; \lambda)$ each of which is evaluated once on the test sample N^K. Averaging the results over K in (10), $\frac{1}{K}\{\frac{1}{N^K} \sum_{i=1}^{N^K} L[y_i, \widehat{f}_K(\mathbf{X}_i; \lambda)]$ and solving for $\widehat{\lambda}$ returns $\widehat{\lambda}_K$ as determined by 'K-fold' cross validation.

$\rho_1, \rho_2 \in [0, 1)$, for the first two moment estimates, s_1 and s_2, of the gradient $\nabla_{\boldsymbol{\theta}} \varpi[\boldsymbol{\theta}_\tau]$, initialized at the origin, $s_1 = s_2 = 0$. Then, the bias-corrected updates of the first and second moments, $\widehat{s}_1 = \frac{\rho_1 s_1 + (1-\rho_1) \nabla_{\boldsymbol{\theta}} \varpi[\boldsymbol{\theta}_\tau]}{1-\rho_1^\tau}$ and $\widehat{s}_2 = \frac{\rho_2 s_2 + (1-\rho_2) [\nabla_{\boldsymbol{\theta}} \varpi[\boldsymbol{\theta}_\tau]]' \nabla_{\boldsymbol{\theta}} \varpi[\boldsymbol{\theta}_\tau]}{1-\rho_2^\tau}$, are used to update the parameters: $\boldsymbol{\theta}_{\tau+1} - \boldsymbol{\theta}_\tau = -\epsilon \frac{\widehat{s}_1}{\sqrt{\widehat{s}_2 + \delta}}$.

[20] RSMProp uses an exponentially decaying average with decay rate $\rho \in [0, 1)$ that discards history from the extreme past and employs the squared gradient, initializing at the origin, $s = 0$. Then, the update of s, $\widehat{s} = \rho s + (1 - \rho) [\nabla_{\boldsymbol{\theta}} \varpi[\boldsymbol{\theta}_\tau]]' \nabla_{\boldsymbol{\theta}} \varpi[\boldsymbol{\theta}_\tau]$, is used to update the parameters: $\boldsymbol{\theta}_{\tau+1} - \boldsymbol{\theta}_\tau = -\frac{\epsilon}{\sqrt{\widehat{s} + \delta}} \nabla_{\boldsymbol{\theta}} \varpi[\boldsymbol{\theta}_\tau]$.

2.2 Deep Learning Basics

To build a ML algorithm one thus needs a dataset, a cost or loss function, an optimization procedure and a model/approximation method. Deep learning builds on feed-forward neural networks (NN) or multi-layer perceptrons (MLPs) to learn unknown target functions of increasing complexity. MLPs are then compositions of single-layer/shallow NNs, each hidden unit of which (or 'neuron') is fully connected to the hidden units of the subsequent layer, to capture the fact that information flows *forward* from the inputs $\mathbf{X}$ to the output y. Artificial neural networks, or MLPs, are thus similar to biological neural networks: they are collections of connected units called neurons. An artificial neuron receives inputs from other neurons, computes the weighted sum of the inputs, and maps the sum via an activation function to the neurons in the next layer, and so on until it reaches the last layer or output. Accordingly, the network is free of cycles or feedback connections that pass information backwards.[21]

Single-layer/shallow NNs are universal approximators (Hornik et al., 1989; Cybenko, 1989) and have dictionaries of functions of the form $\{b(\mathbf{X}|\boldsymbol{\gamma}_1) = s(\mathbf{W}_1'\mathbf{X}+\mathbf{b}_1) : \boldsymbol{\gamma}_1 = (\mathbf{b}_1, \mathbf{W}_1), \mathbf{W}_1'\mathbf{X} = [...\sum_{p=1}^{P} w_{zp}x_p...]' \in \mathbb{R}^{Z_1}\}$ where $s(.) : \mathbb{R}^{Z_1} \to \mathbb{R}^{Z_1}$ is a vector-valued 'activation function' (i.e. applied unit-wise), mapping the output from the single hidden layer $\mathbf{h}_1 = \mathbf{W}_1'\mathbf{X} + \mathbf{b}_1 \in \mathbb{R}^{Z_1}$ and the bias of each hidden unit $z \in \mathbb{R}^{Z_1}$ in the single hidden layer, $\mathbf{b}_1 \in \mathbb{R}^{Z_1}$, into the output, $\widehat{y} = \sum_{z=1}^{Z_1} w_{2z}s_z(\mathbf{W}_1'\mathbf{X} + \mathbf{b}_1) + b_{2z} \equiv \widehat{f}(\mathbf{X}; \boldsymbol{\theta}_1)$, with the weights $\mathbf{w}_2 \in \mathbb{R}^{Z_1}$ and bias $b_2 \in \mathbb{R}$ being the parameters $\{a_z\}_{z=1}^{Z_1}$ of the function class $\mathcal{G}$ defined above, i.e. $\boldsymbol{\theta}_1 = (\mathbf{w}_2, b_2; \mathbf{b}_1, \mathbf{W}_1) \equiv (\mathbf{a}; \boldsymbol{\gamma}_1)$. Popular choices for the activation function include: (i) Rectified linear units (ReLu), $s(h) = \max\{0, h\}$; (ii) Softplus, $s(h) = \log(1 + e^h)$; (iii) Hard tanh, $s(h) = \max\{-1, \min\{1, h\}\}$; (iv) Sigmoid or 'logistic', $s(h) = (1+e^{-h})^{-1}$; or (v) Maxout, $s(h) = \underset{j \in G^i}{max} h_j$ where the number of hidden units z in layer l, Z_l, is divided into groups of k values, $\{(z_1, ..., z_k), ..., (z_{Z_l-k+1}, ..., z_{Z_l})\}$, and $\mathbb{G}^i = \{(i-1)k + 1, ..., ik\}$ is the set of indices into the inputs for group i. All activation functions $s(.)$ have in common that a certain threshold must be overcome for information to be passed forward, much alike neurons in the human brain, that need to receive a certain amount of stimuli in order to be activated. The threshold hurdle creates a non-linearity that allows artificial NNs to learn non-linear and non-convex unknown target functions $f(\mathbf{X})$.

Single-layer NNs are also known as 'three-layer' networks, where the inputs $\mathbf{X}$ form the first, the second or 'hidden' layer $\mathbf{h}_1$ is comprised of $(\mathbf{b}_1, \mathbf{W}_1, s(.)) : \mathbf{h}_1 = s(\mathbf{W}_1'\mathbf{X} + \mathbf{b}_1)$, and the third corresponds to the output layer, $\widehat{y} = \mathbf{w}_2's(\mathbf{h}_1) + b_2 \in \mathbb{R}$. A deep NN (DNN) is constructed by adding hidden layers, each subsequent one taking as inputs the output of the previous ones. For example, a 'four-layer' NN that adds one hidden layer to a 'three-layer' NN

[21] MLPs that allow information to flow *backwards* are called *recurrent neural networks* and are discussed in Goodfellow et al. (2016).

(or shallow/single-layer NN), rather than simply taking the linear combination of the dictionary entries of single-layered NNs, $\{b(\mathbf{X}|\boldsymbol{\gamma}_1)\}$, would result in the collection of functions represented by the dictionary $\{b(\mathbf{X}|\boldsymbol{\gamma}_2) = s(\mathbf{W}_2' s(\mathbf{W}_1'\mathbf{X} + \mathbf{b}_1) + \mathbf{b}_2) : \boldsymbol{\gamma}_2 = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{W}_1, \mathbf{W}_2), \mathbf{W}_1'\mathbf{X} = [...\sum_{p=1}^{P} w_{zp}x_p...]' \in \mathbb{R}^{Z_1}, \mathbf{W}_2 \in \mathbb{R}^{Z_1 \times Z_2}\}$. Adding hidden layers results then in parameter addition, increasing the variance and reducing the bias. The overall effect on performance (i.e. on generalization/test error) will depend on how well the resulting dictionary matches the unknown target function $f(\mathbf{X})$. And although it is an open question in the deep learning literature why do over-parameterized DNNs perform well in terms of generalization/test error, original contributions due to Pascanu et al. (2013) and Montufar et al. (2014) show that deeper ReLu architectures have more flexibility to express the behavior of the unknown target function, relative to equally sized single-layer/shallow architectures.[22]

Generally, a DNN approximation $\widehat{f}(.) : \mathbb{R}^P \to \mathbb{R}$ of size $Z = \sum_{l=1}^{L} Z_l$ with $L \in \mathbb{N}$ hidden layers and $Z_l \in \mathbb{N}$ nodes per layer l, is of the form:

$$\begin{aligned} \widehat{f}(\mathbf{X}) &\equiv f(\mathbf{X}; \boldsymbol{\Lambda}_L) = \mathbf{w}_{L+1}' s(\mathbf{W}_L' \mathbf{h}_{L-1} + \mathbf{b}_L) + b_{L+1} \\ &= \underbrace{f \circ f \circ ... \circ f}_{L-\text{composition}}(\mathbf{X}; \boldsymbol{\Lambda}_1) \end{aligned}$$

where $s(.) : \mathbb{R}^{Z_{L-1}} \to \mathbb{R}^{Z_L}$ is the vector-valued activation function that maps the output from the previous hidden layer $\mathbf{h}_{L-1} = s(\mathbf{W}_{L-1}'\mathbf{h}_{L-2} + \mathbf{b}_{L-1}) \in \mathbb{R}^{Z_{L-1}}$ and the bias of each hidden unit $z \in \mathbb{R}^{Z_L}$ in the last hidden layer L, $\mathbf{b}_L \in \mathbb{R}^{Z_L}$, into the output layer $l = L+1$, with weights $\mathbf{w}_{L+1} \in \mathbb{R}^{Z_L}$ and bias unit $b_{L+1} \in \mathbb{R}$. The matrices $\mathbf{W}_l = [\mathbf{w}_1 ... \mathbf{w}_{Z_l}] \in \mathbb{R}^{Z_{l-1} \times Z_l}$ contain the weights $\mathbf{w}_z \in \mathbb{R}^{Z_{l-1}}$ of each hidden unit $z = 1...Z_l$ for each hidden layer $l = 1...L$, with $Z_0 = P$ the dimension of the input vector $\mathbf{X} \in \mathbb{R}^P$. $\boldsymbol{\Lambda}_L \equiv [\boldsymbol{\theta}_L; Z, L, \{Z_l\}_{l=1}^{L}; \epsilon, \lambda, \alpha]$ is the collection of parameters $\boldsymbol{\theta}_L = [(\mathbf{w}_{L+1}, b_{L+1})...(\mathbf{W}_1, \mathbf{b}_1)]$ and hyperparameters $[Z, L, \{Z_l\}_{l=1}^{L}]$ and $[\epsilon, \lambda, \alpha]$ to be learned and/or 'fined tuned' by the optimization algorithm, e.g. if the DNN is trained with SGD combined with momentum α and a parametric penalty term $\lambda\varpi[\boldsymbol{\theta}]$, for a specific choice of the activation function $s(.)$. The last equality simply conveys that a DNN can be expressed as the composition of L-single layer NNs where $\boldsymbol{\Lambda}_1 \equiv [[\boldsymbol{\theta}_1; Z, 1, \{Z_l\}_{l=1}^{1}; \epsilon, \lambda, \alpha]$ and $Z_1 = Z - \sum_{l=2}^{L} Z_l$ denotes the hidden units remaining from the allocation of Z to the subsequent layers $l = 2, ..., L$.

Approximating the unknown target function $f(\mathbf{X})$ with a DNN is then equivalent to parameter estimation:

$$\widehat{\boldsymbol{\Lambda}}_L \in \arg\min_{\Lambda_L} \frac{1}{N^{\llcorner}} \sum_{i=1}^{N^{\llcorner}} L[y_i, f(\mathbf{X}_i; \boldsymbol{\Lambda}_L)] + \lambda\varpi[\boldsymbol{\theta}] \tag{11}$$

[22] An incipient strand of the literature (e.g. Arora et al., 2019; Allen-Zhu et al., 2020) building instead on the Rademacher complexity of both the function class being approximated and of the dataset, shows that the dictionaries of deeper architectures can better capture interactions between the units of different layers through the composition of functions they can represent.

where it is standard practice to 'cross-validate' the choice of hyperparameters $[Z, L, \{Z_l\}_{l=1}^L]$ and $[\epsilon, \lambda, \alpha]$ before estimating the parameters that characterize the restricted class of functions/models represented by the dictionary $\{b(\mathbf{X}|\boldsymbol{\gamma}_L) : \boldsymbol{\gamma}_L = (\mathbf{b}_1, ..., \mathbf{b}_L, \mathbf{W}_1, ..., \mathbf{W}_L)\}$ augmented by the output layer weights and bias, $(\mathbf{w}_{L+1}, b_{L+1})$, $\boldsymbol{\theta}_L = [(\mathbf{w}_{L+1}, b_{L+1})...(\mathbf{W}_1, \mathbf{b}_1)]$, that solve the 'empirical risk minimization' problem (11). In deep learning, standard choices are: (i) a cross-entropy cost/loss function, $L[.,.]$; (ii) a ReLu activation function $s(.)$, which naturally leads to sparse settings whereby a large portion of hidden units are not activated, thus having zero output (LeCun et al., 2015); (iii) a SGD penalty $\varpi[\boldsymbol{\theta}]$, usually combined with momentum α, as optimization method; and (iv) network architecture size, depth and nodes per layer, $[Z, L, \{Z_l\}_{l=1}^L]$, as well as learning rate, ϵ, that depend on the characteristics of the dataset, $\{y_i, \mathbf{X}_i\}_{i=1}^N$. Performance is then assessed on the test sample, from evaluating $\frac{1}{N^\top}\sum_{i=1}^{N^\top} L[y_i, f(\mathbf{X}_i; \widehat{\boldsymbol{\Lambda}}_L)]$.

In practice, 'tuning' or optimizing the hyperparameters is a daunting task in terms of processing time and computational capacity, e.g. only determining the optimal depth (number of layers L) and nodes per layer $(\{Z_l\}_{l=1}^L)$ for architectures of a given size Z involves solving an NP-hard combinatorial optimization problem because $L, \{Z_l\}_l \in \mathbb{N}$, i.e. are integer values (Judd, 1990). Yet in Calvo-Pardo et al. (2020a) we show that recent advances in combinatorial optimization software (RStudio) can be exploited to optimally allocate hidden units $(\{Z_l\}_{l=1}^L)$ within ('width') and across ('depth', L) layers in deep architectures of a given size $Z = \sum_{l=1}^L Z_l$. Adopting Montufar et al.'s (2014) lower bound on the maximal number of linear regions that ReLu DNNs can approximate as maximization criterion, $LB(L, \{Z_l\}_{l=1}^{L-1}; P) \equiv \left(\prod_{l=1}^{L-1} \left\lfloor \frac{Z_l}{P} \right\rfloor^P\right) \sum_{r=0}^P \binom{Z - \sum_{l=1}^{L-1} Z_l}{r}$,[23] we effectively solve (11) in two-stages:

$$(\widehat{L}, \{\widehat{Z}_l\}_{l=1}^{\widehat{L}}) \in \arg \max_{(L, \{Z_l\}_{l=1}^{L-1})} LB(L, \{Z_l\}_{l=1}^{L-1}; P) \tag{12}$$

$$\widehat{\boldsymbol{\Lambda}}_L(\widehat{L}, \{\widehat{Z}_l\}_{l=1}^{\widehat{L}}) \in \arg \min_{\Lambda_L(\widehat{L}, \{\widehat{Z}_l\}_{l=1}^{\widehat{L}})} \frac{1}{N^\llcorner} \sum_{i=1}^{N^\llcorner} L[y_i, f(\mathbf{X}_i; \boldsymbol{\Lambda}_L)] + \lambda \varpi[\boldsymbol{\theta}] \tag{13}$$

The first stage optimization (12) solves for the optimal depth $\widehat{L}$ and number of hidden

[23] Upper bounds, or maximal number of linear regions of a function approximated by a network architecture with rectified linear units of size Z, have recently been characterized by Raghu et al.'s (2017) Theorem 1 to equal

$$UB(L, \{Z_l\}_{l=1}^L; P) = O\left(\left[\frac{Z}{L}\right]^{ZP}\right)$$

from which they conclude that the maximal number of regions approximated by a shallow ReLu NN, $UB(1, Z; P)$, is always smaller than the maximal number of regions approximated by an equally-sized deep ReLu NN, $UB(L, \{Z_l\}_{l=1}^L; P) : \sum_{l=1}^L Z_l = Z$:

$$UB(1, Z; P) < UB(2, \frac{Z}{2}; P) < ... < UB(L, \frac{Z}{L}; P)$$

units per layer (or optimal width, layer-wise) $\{\widehat{Z}_l\}_{l=1}^{\widehat{L}}$ given the network architecture size, $Z = \sum_{l=1}^{L} Z_l$.[24] The outcome of the first stage is an optimal deep network architecture in the sense of maximizing the expressive power of the approximation $f(\mathbf{X}; \mathbf{\Lambda}_L)$ within the restricted class of functions generated by the dictionary $\{b(\mathbf{X}|\boldsymbol{\gamma}_L) : \boldsymbol{\gamma}_L = (\mathbf{b}_1, ..., \mathbf{b}_L, \mathbf{W}_1, ..., \mathbf{W}_L)\}$. The second stage optimization (13) proceeds just as in (11) but takes as given the optimal values of the hyperparameters $(\widehat{L}, \{\widehat{Z}_l\}_{l=1}^{L})$ from the first stage (12), i.e. $\mathbf{\Lambda}_L(\widehat{L}, \{\widehat{Z}_l\}_{l=1}^{\widehat{L}}) = [\boldsymbol{\theta}_L; Z, (\widehat{L}, \{\widehat{Z}_l\}_{l=1}^{\widehat{L}}); \epsilon, \lambda, \alpha]$. Rather than engaging into time and computer intensive 'fine tuning' of the whole set of hyperparameters $[Z, L, \{Z_l\}_{l=1}^{L}; \epsilon, \lambda, \alpha]$ while training the deep architecture to estimate/learn $\boldsymbol{\theta}_L$ as in (11), proceeding in two-stages considerably saves on runtime and memory while improving performance, as we show in the next section. Finally notice that being the first stage conditional on the architecture size, bigger and more complex datasets $\{y_i, \mathbf{X}_i\}_{i=1}^{N}$ will naturally summon architectures with more hidden units, Z.

Deep neural networks have become so powerful because of (i) the availability of large datasets, necessary to 'train' them[25], and because of the rapid improvements in (ii) computational power[26] and in (iii) optimization algorithms and software. The backpropagation optimization algorithm informs the machine how it should change the internal parameters used to compute the representation in each layer from the representation in the previous layer. Software optimization methods (e.g, Adam, Adagrad, RMSprop) that implement SGD or any of its variants, allow substantial gains in the necessary time and computational power when training models with millions of parameters, and is nowadays often paired with step size 'adaptive regularization'. It is also now standard practice to do regularization while optimizing (e.g. via 'weight decay', 'dropout' or 'batch normalization') to prevent overfitting and improve the performance of DNNs 'out-of-sample'.

'Batch normalization' (Ioffe & Szegedy, 2015; not to be mistaken with 'minibatch regular-

[24] The first stage optimization (12) is a constrained combinatorial optimization problem:

$$(\widehat{L}, \{\widehat{Z}_l\}_{l=1}^{\widehat{L}}, \{\mu_l\}_{l=1}^{\widehat{L}}) \in \arg \max_{(L, \{Z_l\}_{l=1}^{L-1}, \{\mu_l\}_{l=1}^{L})} LB(L, \{Z_l\}_{l=1}^{L-1}; P) + \sum_{l=1}^{L-1} \mu_l (P - Z_l) + \mu_L(-L)$$

where $\{\mu_l\}_{l=1}^{\widehat{L}} \in \mathbb{R}^L$ is the collection of L Lagrange multipliers associated with the $L-1$ constraints, $Z_l \geq P, l = 1...L-1$, and with the constraint $L > 0$, because the constraint on the architecture size $Z = \sum_{l=1}^{L} Z_l$ is incorporated into the maximand. Since $L, \{Z_l\}_{l=1}^{L} \in \mathbb{N}$, we also solve in two stages to reduce the computational burden. In the first stage of (12), the number of hidden units are optimally allocated for a given depth, $\left\{L, \{\widehat{Z}_l\}_{l=1}^{L}\right\}$, while in the second stage of (12), the optimal depth is sought after for a given allocation of hidden units, $\left\{\widehat{L}, \{Z_l\}_{l=1}^{\widehat{L}}\right\}$.

[25] Deep neural networks are characterized by a large number of parameters that need to be 'optimized' during 'training'. This is called 'fine-tuning' or 'optimally fitting a neural network' to the 'training sample'.

[26] Particularly, of graphics processing units (GPUs), suited to perform the linear algebra operations at the root of 'fitting' neural networks, e.g. Google DeepMind optimized a deep neural network using 176 GPUs for 40 days to beat the best human players in the game Go.

ization') is a method of adaptive reparameterization best suited for training very deep models that involve the composition of several functions or layers. By normalizing the output of each layer before forwarding it as input to the next layer[27], the unexpected effect of many functions being composed together changing simultaneously is removed, allowing the gradient to update the parameters under the assumption that the other layers do not change. As a result, it allows the use of higher learning rates, ϵ, which are less sensible to the initialization of parameters.

'Dropout' discards a small but random portion of the neurons during each iteration of training to prevent neurons from co-adapting to the same features (or predictors, P), providing a powerful regularization method (Srivastava et al., 2014). The intuition is that since several neurons are likely to model the same non-linear relationship simultaneously, discarding a random fraction of them forces them to perform well regardless of which other hidden units are in the model. With dropout, each input and hidden unit z in layer $l = 1...L$, h_{zl}, is pre-multiplied by a random variable $r_{zl} \sim F(r_{zl})$, $\overline{h}_{zl} = r_{zl} \cdot h_{zl}, \forall (z,l)$, prior to being fed forward to the activation function of the next layer, $h_{zl+1} = s_z(\sum_{z=1}^{Z_l} w_{zl+1}\overline{h}_{zl} + b_{zl+1}), \forall z = 1...Z_{l+1}$. For any layer l, $\mathbf{r}_l$ is then a vector of independent random variables, $\mathbf{r}_l = [r_{1l}, ..., r_{Z_l l}] \in \mathbb{R}^{Z_l}$. Standard choices for the probability distribution $F(\mathbf{r}_l)$ are (i) the Normal, i.e. $F(\mathbf{r}_l) = N(\mathbf{1}, \mathbf{I})$, or (ii) the Bernoulli, in which case each r_{zl} has probability p of being 1 (and $1-p$ of being 0). The vector $\mathbf{r}_l$ is then sampled and multiplied element-wise with the outputs of that layer, h_{zl}, to create the thinned outputs, $\overline{h}_{zl}$, which are then used as input to the next layer, h_{zl+1}. When this process is applied at each layer $l = 1...L$, this amounts to sampling a sub-network from a larger network. In the ML literature, common choices for p are 0.8 for the input layer, $l = 1$, and 0.5 for the units in hidden layers, in $l = 2...L$. During learning, the derivatives of the loss function are backpropagated through the sub-network. At test time, the weights are scaled down as $\overline{\mathbf{W}}_l = p\mathbf{W}_l, l = 1...L$, resulting in a DNN (without dropout) that allows the conduct of approximate inference[28]. This efficient test time procedure is an approximate model combination that (i) scales down the weights of the trained neural network, (ii) works well with other distributed representation models, e.g. restricted Boltzmann machines, and (iii) acts as a regularizer. Beyond the MLPs discussed, an array of alternative architectures have been proposed, including convolutional and recurrent NNs, that target specific data structures, like vision tasks and sequential data handling

[27] Concretely, the normalization involves computing:

$$\overline{h}_{zl} = \frac{1}{\sigma}(h_{zl} - \mu), z \in B : \mu = \frac{1}{|B|}\sum_{z \in B} h_{zl}, \sigma = \sqrt{\delta + \frac{1}{|B|}\sum_{z \in B}(h_{zl} - \mu)^2}$$

with $\delta \approx 10^{-8}$ set to prevent the undefined value $\sqrt{0}$, and B denoting a minibatch of output units h_{zl} in layer $l = 1...L$.

[28] It is actually exact for many classes of models that do not have nonlinear hidden units, like the softmax regression classifier, regression networks with conditionally normal outputs or deep networks with hidden layers without nonlinearities.

respectively. See Goodfellow et al. (2016) for a detailed textbook treatment.

2.3 Uncertainty and Deep Learning

Despite their unrivaled success in prediction and forecasting tasks, deep learning models struggle in conveying the uncertainty or degree of statistical confidence/reliability associated with those forecasts. Some recent contributions in the ML literature have made progress in the provision of confidence intervals for the point forecasts provided by deep learning models trained with dropout. For example, Gal and Ghahramani (2016) show that a NN with arbitrary depth and non-linearities, with dropout applied before every weight layer and a parametric L^2 penalty $\varpi[\boldsymbol{\theta}] = \sum_{l=1}^{L}\left\{\|\mathbf{W}_l\|_2^2 + \|\mathbf{b}_l\|_2^2\right\}$, minimises the Kullback–Leibler divergence between an approximate (variational) distribution, $q(\boldsymbol{\theta})$ –over matrices $\boldsymbol{\theta} = (\mathbf{W}_1, ..., \mathbf{W}_L)$ with columns randomly set to zero, $\mathbf{W}_l = \mathbf{M}_l diag[r_{zl}]_{z=1}^{Z_l}, r_{zl} \sim Bernoulli(p_l), l = 1, ..., L, z = 1, ..., Z_l$ – and the posterior of a deep Gaussian process, $p(\boldsymbol{\theta}|y, \mathbf{X})$, which is intractable:

$$\begin{aligned} & -\sum_{i=1}^{N}\int q(\boldsymbol{\theta})\log p(y_i|\mathbf{X}_i,\boldsymbol{\theta})d\boldsymbol{\theta} + D_{KL}(q(\boldsymbol{\theta})||p(\boldsymbol{\theta})) \\ \propto \quad & -\sum_{i=1}^{N}\frac{\log p(y_i|\mathbf{X}_i,\widehat{\boldsymbol{\theta}})}{\tau N} + \sum_{l=1}^{L}\left\{\frac{p_l l^2}{2\tau N}\|\mathbf{M}_l\|_2^2 + \frac{l^2}{2\tau N}\|\mathbf{b}_l\|_2^2\right\} \end{aligned}$$

where the first and second terms in the sum are approximated. In the first term, each term in the sum over N is approximated by Monte Carlo integration with a single sample $\widehat{\boldsymbol{\theta}}^t \sim q(\boldsymbol{\theta})$ to get an unbiased estimate of $\log p(y_i|\mathbf{X}_i, \widehat{\boldsymbol{\theta}})$. In the second, l denotes prior length-scale, and τ model precision, i.e. $p(y|\mathbf{X}, \boldsymbol{\theta}) = N(\widehat{y}(\mathbf{X}, \boldsymbol{\theta}), \frac{1}{\tau}\mathbf{I}) : \widehat{y}(\mathbf{X}, \boldsymbol{\theta}) = \sqrt[-2]{Z_L}\mathbf{W}_L s(... \sqrt[-2]{Z_1}\mathbf{W}_2 s(\mathbf{W}_1\mathbf{X} + \mathbf{b}_1)...)$ and variance-covariance matrix $\frac{1}{\tau}\mathbf{I}$. The sampled $\widehat{\boldsymbol{\theta}}^t$ result in realizations from the Bernoulli distribution $[\mathbf{r}_l^t]$ equivalent to the binary variables in the dropout case, i.e. sampling T sets of vectors of realizations from the Bernoulli distribution $\{[\mathbf{r}_l^t]\}_{t=1}^T$ with $[\mathbf{r}_l^t] = [r_{zl}^t]_{z=1}^{Z_l}$, giving $\{\mathbf{W}_1^t, ..., \mathbf{W}_L^t\}_{t=1}^T$, with which the first two moments of the predictive distribution $p(y_i|\mathbf{X}_i, \widehat{\boldsymbol{\theta}})$ are estimated (by moment-matching). The first moment, $\frac{1}{T}\sum_{t=1}^{T}\widehat{y}(\mathbf{X}, \mathbf{W}_1^t, ..., \mathbf{W}_L^t)$, is known as *Monte Carlo (MC) dropout*, and in practice it corresponds to performing T stochastic forward passes through the NN and averaging the results (model averaging). The second moment, $\frac{1}{\tau}\mathbf{I} + \frac{1}{T}\sum_{t=1}^{T}\widehat{y}(\mathbf{X}, \mathbf{W}_1^t, ..., \mathbf{W}_L^t)'\widehat{y}(\mathbf{X}, \mathbf{W}_1^t, ..., \mathbf{W}_L^t)$, equals the sample variance of T stochastic forward passes through the NN plus the inverse model precision, providing a measure of the uncertainty attached to the deep NN point forecast.

In Calvo-Pardo et al.(2020b) we build upon the previous Bayesian interpretation of dropout in deep NNs to advance a different measure of model uncertainty, and compare it to MC dropout. There we perform *inverted dropout* (using Keras from RStudio), where during training we instead scale-up weights as $\overline{\mathbf{W}}_l = (1/p)\mathbf{W}_l, l = 1...L$, and then perform T stochastic forward passes through the DNN with weights $\overline{\mathbf{W}}_l, l = 1...L$, producing a sample $\{\widehat{y}(\mathbf{X}, \overline{\mathbf{W}}_1^t, ..., \overline{\mathbf{W}}_L^t)\}_{t=1}^T$ from sampling T sets of vectors of realizations from the Bernoulli distribution $\{[\mathbf{r}_l^t]\}_{t=1}^T$. Confidence

intervals can thus be provided for a deep NN point estimate, as we illustrate in the application below, representing a substantial narrowing down of those provided in the literature based on Hayes' (2015) model of rational Bitcoin mining.

Finally, attention within the deep learning literature has more recently focused on their interpretability. The goal of interpretation tasks is to use the structural form of the approximating function $\widehat{f}(\mathbf{X})$ to try to understand the mechanism that produced the data $\{y_i, \mathbf{X}_i\}_{i=1}^N$. Interest lies then in identification of those input variables that are most relevant to the variation in the output, the nature of the dependence of the output on the most relevant inputs or how that dependence changes with changes in the values of other inputs. Conducting valid inference rests on the amount of correct information learned about the system (i.e. minimizing the bias at the expense of increasing the variance), rather than just prediction accuracy (where some bias is optimally traded-off against the resulting reduction in the variance). Although both are often in conflict, limiting the inferential abilities of ML methods, it is not always the case. Athey and Imbens (2019) note that one way to perform valid (causal) inference would be to adapt the 'out-of-sample' performance objective in ML cost/loss functions to control for confounders or for discovering treatment effect heterogeneity, as standard in the model-based statistics and econometrics literatures. Allen-Zhu et al. (2018) within the ML literature, and Farrell et al. (2018) within the econometrics literature, obtain nonasymptotic bounds. Based on Farrell (2015), the latter obtain conditions for valid two-step causal inference after first-step deep learning estimation.[29]

We do not detail the conditions for valid causal inference in deep learning as here we are concerned with showcasing how ML methods broadly, and deep learning in particular, can also be applied to broader societal issues of rising global concern like the pollution content of economic activity and its implications for global warming. The next section uncovers a novel application of deep learning where interest lies in measuring/predicting the carbon footprint associated with Bitcoin network mining and the uncertainty surrounding those estimates.

3 CO_2 Emissions from Bitcoin Mining

There are three primary ways one can obtain Bitcoins (BTC), the most popular and widely accepted of the so-called cryptocurrencies: buy them outright, accept them in exchange, or produce them by 'mining'. 'Mining' for Bitcoins requires computer hardware and software specifically designed to solve the cryptographic algorithm underlying the bitcoin protocol, and

[29] A survey about the differences between the two literatures, promises and recent progresses made along integrating both are provided in Athey and Imbens (2019).

such computational effort mainly consumes electricity.[30] Since at any point in time different miners operate hardware and software with varying levels of energy efficiency, measuring the overall network power consumption involved in Bitcoin production remains a challenge to date.[31] To overcome it, we propose a novel ML approach.

To estimate a realistic level of daily electricity consumption to produce Bitcoins based on a feed forward neural network, we first calculate a lower and an upper limit –based on Hayes (2015)– within which our mean predicted electricity consumption must 'travel' between the 01/01/2017 and the 01/01/ 2020, the considered period. The lower limit corresponds to the lowest marginal cost for mining Bitcoins, and is defined by a scenario in which all miners use the most efficient available hardware. The upper limit obtains when instead the least efficient technology for mining Bitcoins is employed, i.e. the break-even point of mining revenues and electricity costs. Obtaining mean point estimates of daily power consumption within those economically meaningful limits provides substantial gains in accuracy relative to recent contributions in the literature, while externally validating our ML approach.

Our feed forward deep neural network (DNN) is a supervised ML algorithm that adopts as target output the market share weighted average[32] of the daily energy efficiency deployed by operating miners[33]. Our target level of electricity consumption is a conservative one in that it follows the approach of the lower limit, and is based on the anticipated energy efficiency of the network and on hardware sales, but ignores auxiliary losses[34]. As inputs, our DNN admits a comprehensive range of factors previously found to drive Bitcoin prices in different currencies, like (i) standard fundamental factors advocated by monetary economics and the quantity theory of money (e.g. its usage in trade, money supply or price level); (ii) factors driving investors' interest in/attention to the crypto-currency, like speculation or the role of Bitcoin as a safe haven; (iii) exchange rate hedging motives, like the tight connection between the USD and the CNY markets

[30] Each unit of mining effort has a fixed sunk cost involved in the purchase, transportation and installation of the mining hardware. De Vries (2018) reports different prices of available models of mining hardware, such as the Antminer S9. Mining effort also has a variable cost which is the direct expense of electricity consumption. See Hayes (2015) for further details.

[31] As an example, De Vries (2018) notes that 'A hashrate of 14 terahashes per second (TH/s) can either come from a single Antminer S9 running on just 1,372 W, or more than half a million PlayStation-3 devices running on 40 MW (as a single PlayStation-3 device has a hashrate of 21 megahashesper second and a power use of 60 W).'

[32] The market shares are computed in terms of either computational power or revenues, and are available from IPO filings disclosed in 2018 by Bitmain, and in 2019 by Canaan, retrieved from Bloomberg terminals.

[33] We obtain the computational power (usually provided in terahashes per second, TH/s) and the electricity consumed (in Watts per second, W/s) by ASIC chips used for Bitcoin mining from *AsicIndex*. Only mining chips that perform the *SHA-256* algorithm are considered.

[34] These are energy losses associated with cooling and investment in new IT equipment, which are instead included by Stoll et al. (2019).

(see Kristoufek, 2015; Liu and Tsivinsky, 2018; McNally et al., 2018, or Jang and Lee, 2018)[35], together with (iv) novel supply-side factors for both Bitcoin and ASIC mining chips producers, related to for-profit mining decisions, but excluding those employed in the construction of the upper and lower limits.[36] The carbon intensity associated with Bitcoin mining obtains then from multiplying the estimated electricity consumption by the average emission factor of power generation. Crucially, our novel approach also enables the construction of confidence intervals around the estimated carbon footprint of Bitcoin mining, substantially narrowing down the associated uncertainty, as currently measured in the literature by the difference between the carbon footprint of the upper and lower bounds. The latter broadly represent, respectively, the expected marginal revenue and marginal cost of Bitcoin network operating miners (Hayes, 2015; Stoll et al., 2019).

3.1 Power Bounds in Bitcoin Production

Hayes (2015) argues that Bitcoin production resembles a competitive market, where risk-neutral rational miners will produce until their marginal costs equal the value of their expected marginal products. To produce bitcoins, a miner directs computational effort at solving a difficult cryptologic 'puzzle' in competition with other miners in the network, to confirm and validate transactions. And computational effort mainly consumes electrical power, measured in Watts, W.[37] The marginal cost (MC) of producing Bitcoins per day (in USD/day) depends on the cost of electricity (price p_e in USD per kWh, or $10^{-3} \times p_e$ in USD per Wh) and the energy efficiency of mining (denoted by e and measured in W per unit of 'mining effort', or 'hashing power' ρ)[38]:

$$\underset{[\text{USD/day per } \rho=1{,}000\text{GH/s}]}{MC} = (10^{-3} \times p_e \cdot 24 \cdot e) \cdot \left(\frac{1000GH}{1000}\right) \tag{14}$$

In return for their work of validating the blockchain, miners are rewarded with a block of 'coins', or 'block reward' (measured in BTC per block, β)[39]. Per day, miners can then expect

[35] These are collected at/or converted into daily frequencies from Bloomberg, from the Federal Reserve Bank of St. Luis, from Blockchain.com, from asic-dex.com, and from the IPO filings of Canaan and Bitmain.

[36] The Bitcoin exchange rates with other cryptocurrencies such as Ethereum, Ripple, and Litecoin are not studied as these 'altcoins' were issued from 2018 onward.

[37] Recall that one Watt equals one Joule per second, i.e. 1 W=1 J/s.

[38] Mining effort or 'hashing power' or 'hashrate' is measured in gigahashes per second (GH/s), and refers to the computational effort applied by miners to obtain bitcoins over a given time interval, typically one day. The hashrate, or number of hashes per second can be thought of as somewhat analogous to the cycles per second (hertz) of computer processors: the higher the hashrate, the more likely it is to successfully mine bitcoins per day. See Hayes (2015).

[39] When analyzing the reward obtained from mining, it is important to consider the phenomenon of *halvening* (Bitcoin halving) where the reward from mining Bitcoins is halved. *Halvening* occurs every 210,000 blocks (every four years). The last *halvening* happened in 09/07/2016 with the mining revenue halved from 2,396,656

to earn an amount of Bitcoins (BTC/day), or expected marginal product ($\mathbb{E}$MP), the value of which depends on the market price of bitcoin (p_b in USD per BTC), the block reward β, the hashing power ρ employed by a miner (that is normalized at ρ =1,000 GH/s =1 TH/s for conformity with the MC units), and the 'difficulty' of mining (denoted by δ) which captures how much aggregate effort other operating miners are putting[40]:

$$\underbrace{\underset{[\text{USD/BTC}]}{p_b} \cdot \underset{[\text{BTC/day per } \rho=1\text{TH/s}]}{EMP}}_{[\text{Value of Expected MP}]} = p_b \cdot \left[\underbrace{\left(\frac{1}{\delta \cdot 2^{32}}\right)}_{[\text{Reward probability}]} \underbrace{(\beta \cdot \rho)\,(24 \cdot 3{,}600)}_{[\text{Daily reward per unit of effort } \rho]} \right] \tag{15}$$

where $s = 3,600$ is the number of seconds in one hour, $h = 24$ is the number of hours in a day, and 2^{-32} is the normalized probability of a single hash solving a block, given that the mining algorithm is the *SHA-256* algorithm.

Given the market price of Bitcoin p_b, a rational miner would produce Bitcoins until when $MC = p_b \cdot \mathbb{E}MP$ if mining for Bitcoins is competitive. Since the actual energy efficiency e of the Bitcoin network miners is unknown, the theoretical relationship $p_b = MC/\mathbb{E}MP$ can be used to obtain the break-even level of energy efficiency $\underline{e}$ below which the marginal cost of mining is above the market value of the marginal product, $e \leq \underline{e} \Longrightarrow MC(e) \geq MC(\underline{e}) = p_b \cdot \mathbb{E}MP$, driving rational miners out of business. Hence:

$$\underset{[\text{J/GH per } \rho=1{,}000\text{GH/s}]}{\underline{e}} = p_b \cdot \left(\frac{\beta \cdot \rho}{\delta \cdot 2^{32}}\right)(24 \cdot 3{,}600)\left[(10^{-3} \times p_e \cdot 24)\right]^{-1} \tag{16}$$

denotes the break-even daily energy efficiency production of Bitcoins, which characterizes the *upper limit of daily electricity consumption* $\overline{E}$ of the Bitcoin network when multiplied by the overall network hash rate H (measured in hashes per second, H/s, corresponding to 10^{-9} GH/s or 10^{-12} per 1TH/s):

USD to 1,208,034 USD. The *halvening* is an important event not only for determining the Bitcoin price (reduction of Bitcoin supply, with unchanged demand) and the break-even energy efficiency level of mining production, but also because it produces a *jump* or discontinuity in the historical observations at hand. The time interval considered 2017 - 2019 ensures that there are no observed *halvening*s. Starting from 09/07/2016, the *block reward* is 12.5 Bitcoin per block.

[40]The 'difficulty' of mining refers to the difficulty level of the algorithm when mining is undertaken. It specifies how hard in terms of computational effort it is to find a bitcoin during a given time interval, and is therefore measured in gigahashes per 'block' of bitcoins, GH/block. The bitcoin network automatically adjusts the difficulty variable so that one block of bitcoins is found, on average, every ten minutes. As more aggregate computational effort is added to mining bitcoins, the time between blocks will tend to decrease below ten minutes, and the network will automatically adjust the difficulty upwards to maintain the ten minute interval. And conversely, is less aggregate computational effort is added, adjusting the difficulty downwards.

$$\underset{\text{[W per day, per TH/s]}}{\overline{E}} = \underline{e} \cdot H \times 10^{-12} \tag{17}$$

Daily data for the Bitcoin network *difficulty* δ and network *hash rate* H are retrieved using the public available API from *blockchain.com,*[41] and reported together with their distributions, in Figure 1, as well as for the daily Bitcoin price p_b and the daily value in USD of the number of Bitcoins obtained by the overall network from mining (BTC/USD), as defined in equation (15).[42] Notice that although the network *hash rate* and the network *difficulty* are strongly positively correlated, they nevertheless correspond to two different variables relevant for Bitcoin mining.

Similarly, it is possible to define the *lower limit of daily electricity consumption* $\underline{E}$ of the Bitcoin network, assuming that all miners operate instead with the most energy efficient $\overline{e}$ hardware:

$$\underset{\text{[W per day, per TH/s]}}{\underline{E}} = \overline{e} \cdot H \times 10^{-12} \cdot 24 \tag{18}$$

To date, the most energy efficient dedicated computer hardware embeds application specific integrated circuit (ASIC) chips[43]. Monthly data about the mining chips' efficiency, measured (in J/GH) as the ratio between the energy used by the ASIC chip (in Joules, J) and the number of iterations performed by the *SHA-256* algorithm (in gigahashes per second, GH/s), for different mining rigs is displayed in Figure 2, between 01/01/2017 and 01/01/2020.[44] $\overline{e}$ then corresponds to the lowest monthly energy efficiency of ASIC chips (in red), which as time passes tends to decrease –except for a few outliers– due to an increase in the network hash rate and thus in the difficulty in producing new Bitcoins.

Figure 2 reports the number of Bitcoins mined per day by the network (i.e. the average $\mathbb{E}MP$ in equation (15), excluding the Bitcoin price p_b)[45], and the associated upper $\overline{E}$ and lower

[41] Since for the time interval 18/07/2018 - 03/08/2018 those network statistics are missing, they are imputed using the MissForest algorithm (Stekhoven, 2013), with a maximum number of trees to be grown in each forest equal to 500, a maximum number of nodes per tree equal to 100, and a maximum number of iterations of 50. The MissForest algorithm is agnostic about the distribution of the variables, estimating the missing values by fitting a random forest trained on the observed values. The Out-Of-Bag (OOB) estimates of the imputation error in terms of Normalized Root Mean Squared Error (NRMSE) is 0.04831 and convergence is achieved after 4 iterations.

[42] For example by the end of 2019, the lower left panel of Figure 1 reports the USD value of the number of bitcoins one can expect per day (in BTC/USD) to be approximately 0.0003. We can obtain the actual number from equation (15), when employing ρ =1,000 GH/s of mining effort with a difficulty $\delta = 4 \times 10^{12}$ (lower right panel of Figure 1) at a price $p_b = 4,890$ (upper left panel of Figure 1): $4890 \cdot (\frac{12.5 \cdot 1000}{2^{32} \cdot 4 \times 10^{12}}) \cdot 24 \cdot 3600 = 0.0003074$ BTC/USD.per day.

[43] Murray (2018) argues that the higher computational power associated to ASIC chips and the increase in the mining difficulty of Bitcoins led to the disappearance of CPU and GPU mining chips.

[44] The data can be retrieved online from https://asic-dex.com.

[45] Notice that the reported average number of Bitcoins mined per day over the period is similar to the one

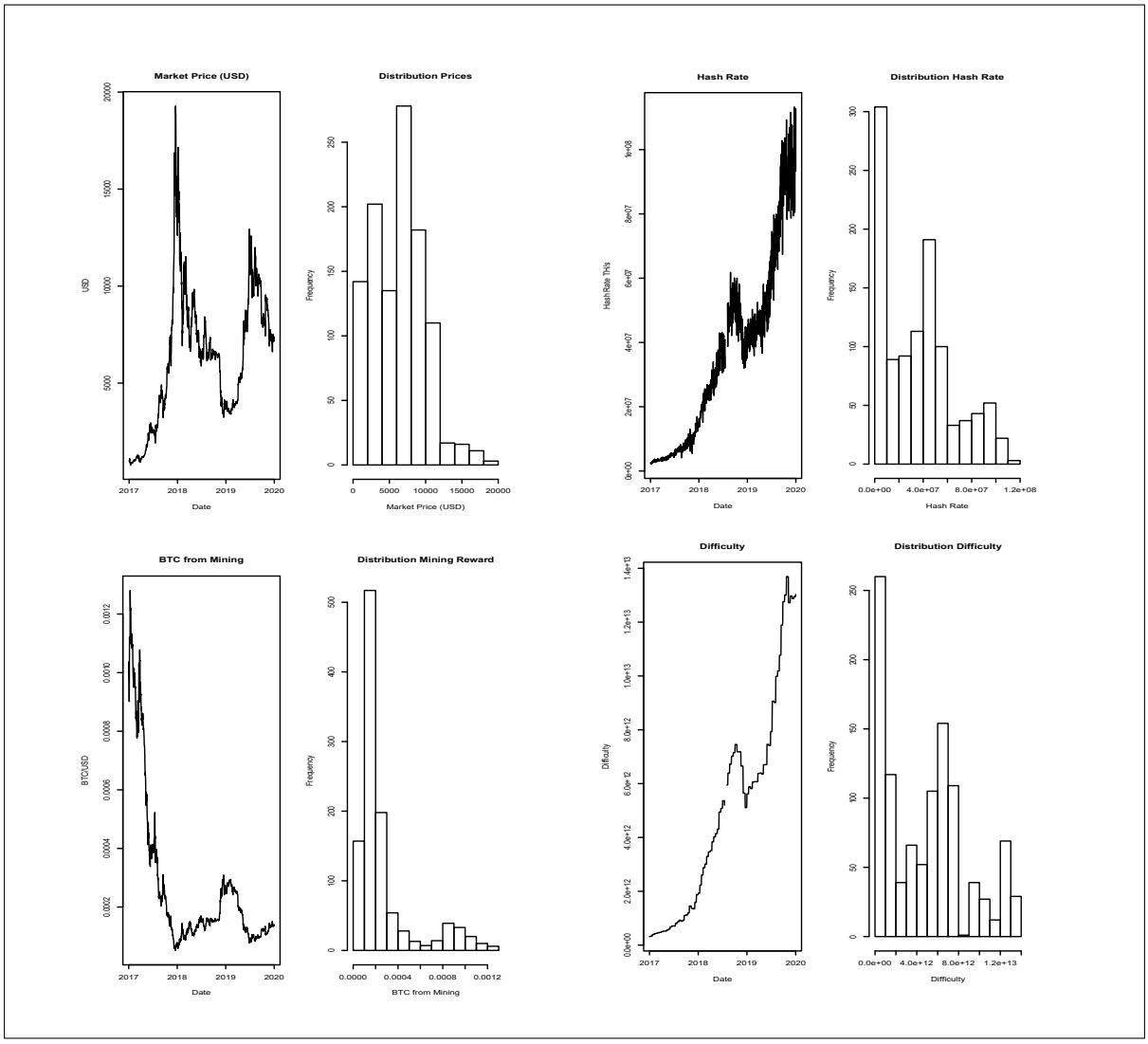

Figure 1: The Figure reports the difficulty in terms of hashing power employed by the network miners, the hash rate in terms of estimated number of tera hashes per second the Bitcoin network is performing, the average USD market price across major bitcoin exchanges, and the mining reward in terms of Bitcoin

$\underline{E}$ limits of daily electricity consumption obtained from equations (17) and (18) after multiplying them by 10^{-6} (to convert them into mega Watts, MW), respectively. Although the upper limit of daily power consumption is more volatile as it follows the market price of Bitcoin, the lower limit is more stable, being defined by hardware efficiency and network hash rate. The difference between the upper and lower limits measures the uncertainty associated with the actual daily hardware efficiency in electricity consumption deployed by the Bitcoin production network of miners. The annual electricity consumption corresponding to the lower and upper bounds $\underline{E}$ and $\overline{E}$ is obtained by summing the daily electricity consumption over the year of interest: for 2017, it ranges between 5.2384 and 43.1218 TWh, for 2018 between 25.0786 and 80.4240 TWh, and for 2019, between 27.0537 and 80.3026 TWh.

Notice from Figure 2 the decreasing gap between $\overline{E}$ and $\underline{E}$, converging to a point of almost equality in 2019: miners with less efficient ASIC chips were then mining at a loss as a result of the significant decrease in Bitcoin prices that can be observed in the upper left panel of Figure 1. One would expect the same narrowing in the difference between the two daily limits as we get closer to 05/2020, when the *halvening* of the 'block reward' is expected to happen. By then, miners will have to run twice the number of computations to mine the same amount of Bitcoins, doubling their electricity usage. This will reduce the break-even level of energy efficiency $\underline{e}$, reducing $\overline{E}$, until when new and more efficient ASIC chips are introduced.

Relative to Stoll et al. (2019), our upper limit behaves similarly to theirs but is lower: the difference stems from the different calculation of the break-even level of energy efficiency $\underline{e}$, equation (16), from which we exclude the transaction fees accrued to successful miners from the $\mathbb{E}MP$ component, as well as from using different electricity prices p_e. We compute electricity prices, p_e, as a weighted average of the annual electricity prices in the countries were Bitcoin miners are located, using as weights the share of miners located in each country. We exploit the Internet of Things (IoT)-search engine *Shodan* to locate the geographic area of the Bitcoin miners IP addresses over the period examined[46]:

Figure 3 reports the countries with the highest number of miners: Venezuela (91), China (162), Russia (158), Iran (122), USA (75). Venezuela, Iran, Russia and (some regions of) China are the countries with the lowest electricity prices in the World (in USD per kWh). We collect historical data on electricity prices for the USA, China, and Russia from Bloomberg Terminal up to 2018, and the electricity prices for 2019 from *GlobalPetrolPrices.com*. Figure 4 reports the evolution of the yearly electricity prices for different usages (residential, industrial and other) in

that obtains instead from the supply side: dividing equation (14) by the Bitcoin price p_b, we get $\frac{MC}{p_b} \simeq 1{,}800$ BTC/day.

[46] Being *antminer* the primary tool for Bitcoin mining, by mapping the instances *Digest real="antMiner Configuration"* we were able to map the IP addresses of the Bitcoin miners.

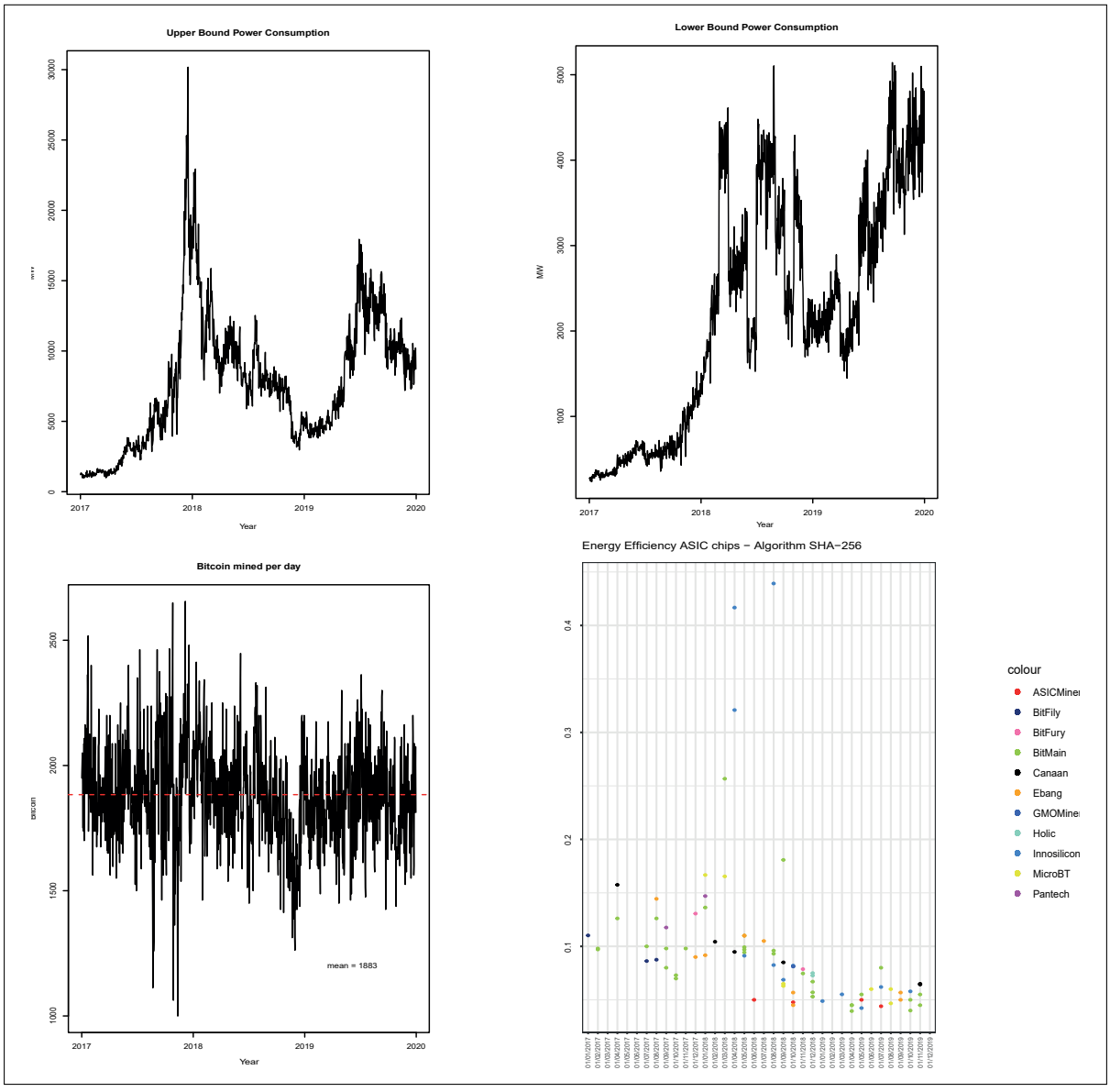

Figure 2: The Figure reports the number of Bitcoin mined per day, the upper and lower bounds of the energy consumption associated to Bitcoin mining, and the energy efficiency in terms of J/Gh of the ASIC mining chips that use the SHA-256 Algorithm

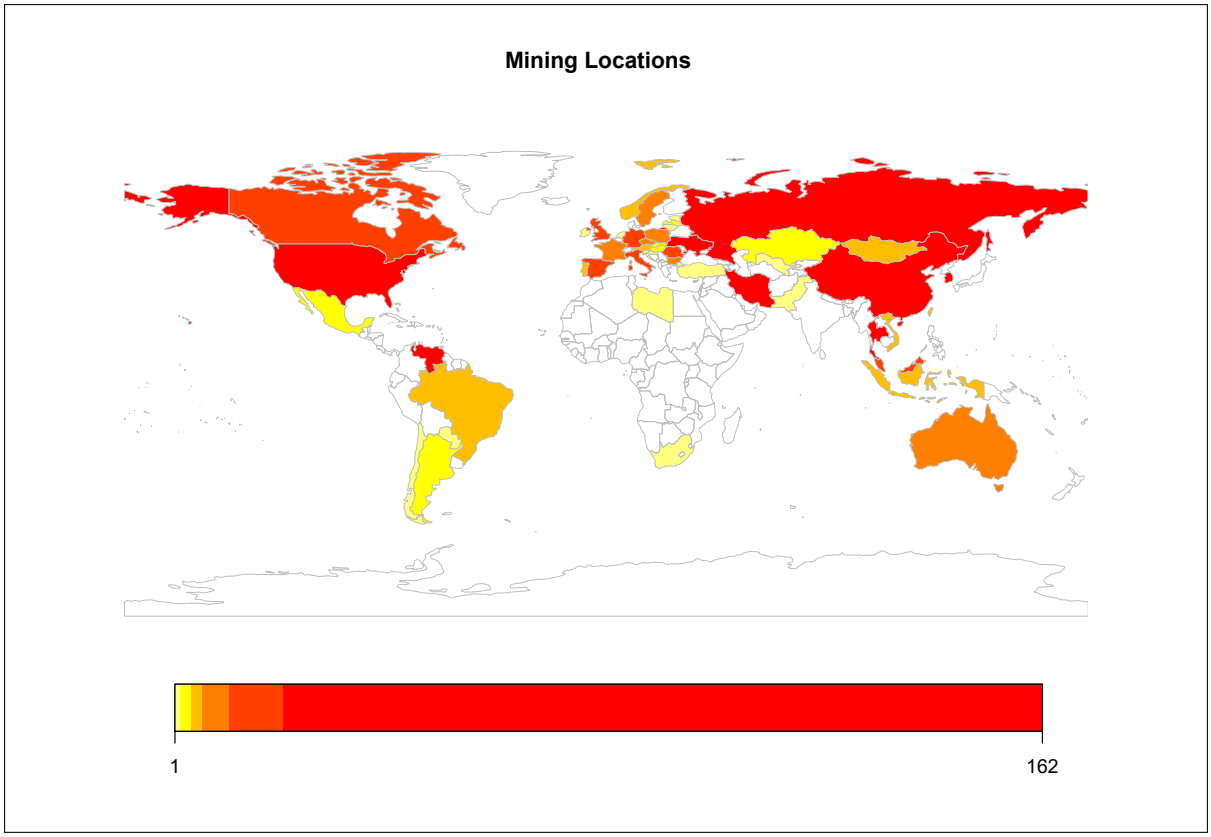

Figure 3: Location Bitcoin Miners 31/01/2020

China, the United States, and Russia[47].

For Venezuela and Iran, it was not possible to collect historical prices: since electricity prices (approximated to two digits) are generally constant over a three year horizon, we apply the 2019 electricity price over the three-year time window examined. The household electricity price in Iran is 0.008 USD/kWh; for Venezuela, the Business electricity price is 0.1284 USD/kWh (1.283 VEF/kWh). Figure 4 reports the employed electricity price p_e, computed as a weighted average of the electricity prices in the United States, China, Russia, Venezuela, and Iran, where the weights are determined by the proportion of Antminer IP addresses of Bitcoin miners located in those countries.[48]

[47] When available and clearly indicated, we only consider the residential electricity price. When unavailable, or unclear (e.g. China), we compute the average of the electricity prices corresponding to the different levels of usage.

[48] 39% of the IP addresses operating in the Bitcoin network are attributed to the remaining 44 countries.

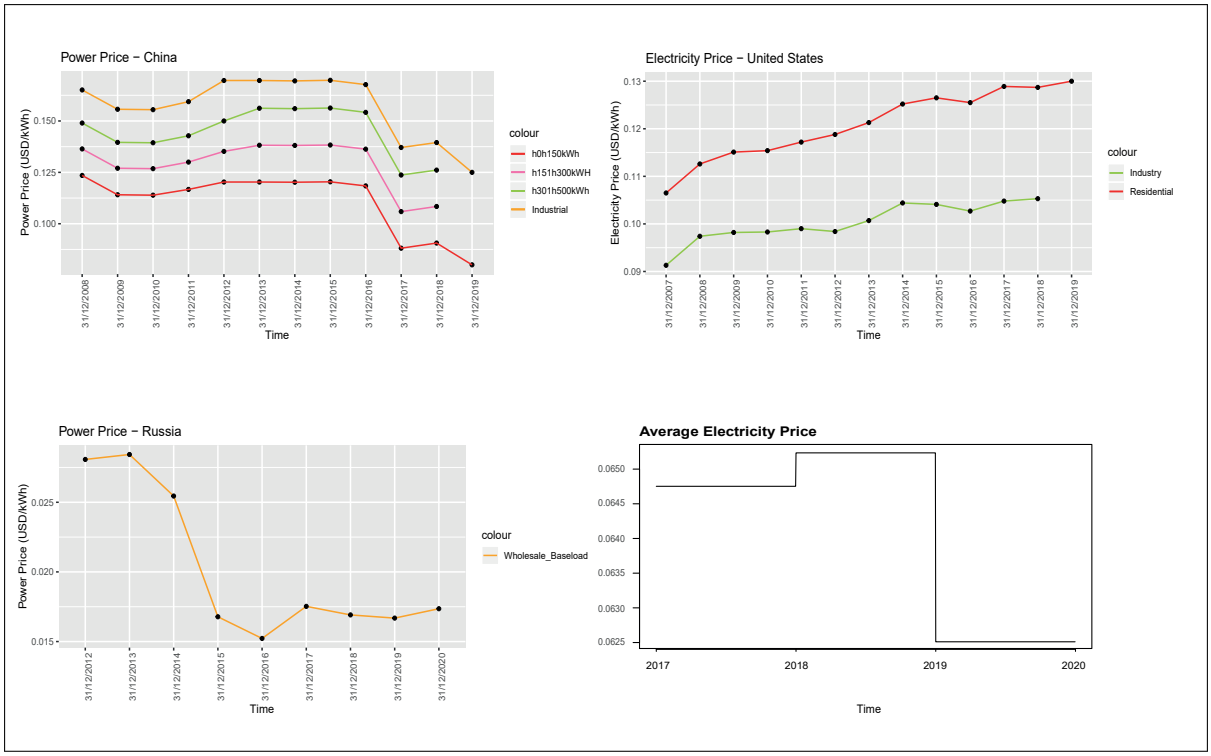

Figure 4: The Figure reports the energy prices (USD/kWh) for the countries United States, China, and Russia, and the weighted average of the energy prices (USD/kWh) across the countries United States, China, Russia, Venezuela, and Iran.

3.2 The Carbon Footprint of Power Bounds in Bitcoin Production

We compute the CO_2 upper ($\overline{CO_2}$) and lower ($\underline{CO_2}$) limits of the Bitcoin network daily emissions (measured in ktCO_2e), associated with the daily electricity consumption upper and lower limits, $\overline{E}$ and $\underline{E}$, from equations (17) and (18) respectively, as follows:

$$\overline{CO_2} = \overline{E} \times 10^{-3} \cdot I \times 24 \times 10^{-6} \tag{19}$$

$$\underline{CO_2} = \underline{E} \times 10^{-3} \cdot I \times 24 \times 10^{-6} \tag{20}$$

where I is the average emission factor, or carbon intensity, of power generation (measured in kgCO_2 per kWh)[49], which obtains from weighting the C country-specific emission factors, I_c, by the computing power share, s_c, of Bitcoin miners' IP addresses located in each country c, $I = \sum_{c=1}^{C} s_c I_c$.[50] A weighted average carbon intensity of $I = 0.6183$ is returned, from country-

[49] Notice that the expressions in (19) and (20) are in ktCO_2 units per day, while $\overline{E}$ and $\underline{E}$ are in Watts per per day (per unit of mining effort, 1GH/s) and I is in kgCO_2 per kWh. To conform, we need to multiply $\overline{E}$ and $\underline{E}$ by 10^{-3} (KWh per Watts) per day, and I by 24 (hours per day), resulting in a product that will then be in units of kgCO_2 per day. Multiplying then by 10^{-6} we obtain ktCO_2 per day. After simplifying, expressions (19) and (20) obtain as reported.

[50] We follow as much as we possibly can the methodology reported in Volume 2 of the 2006 *IPCC Guidelines for National Greenhouse Gas Inventories*, which states that the emission of greenhouse gas (CO_2) from station-

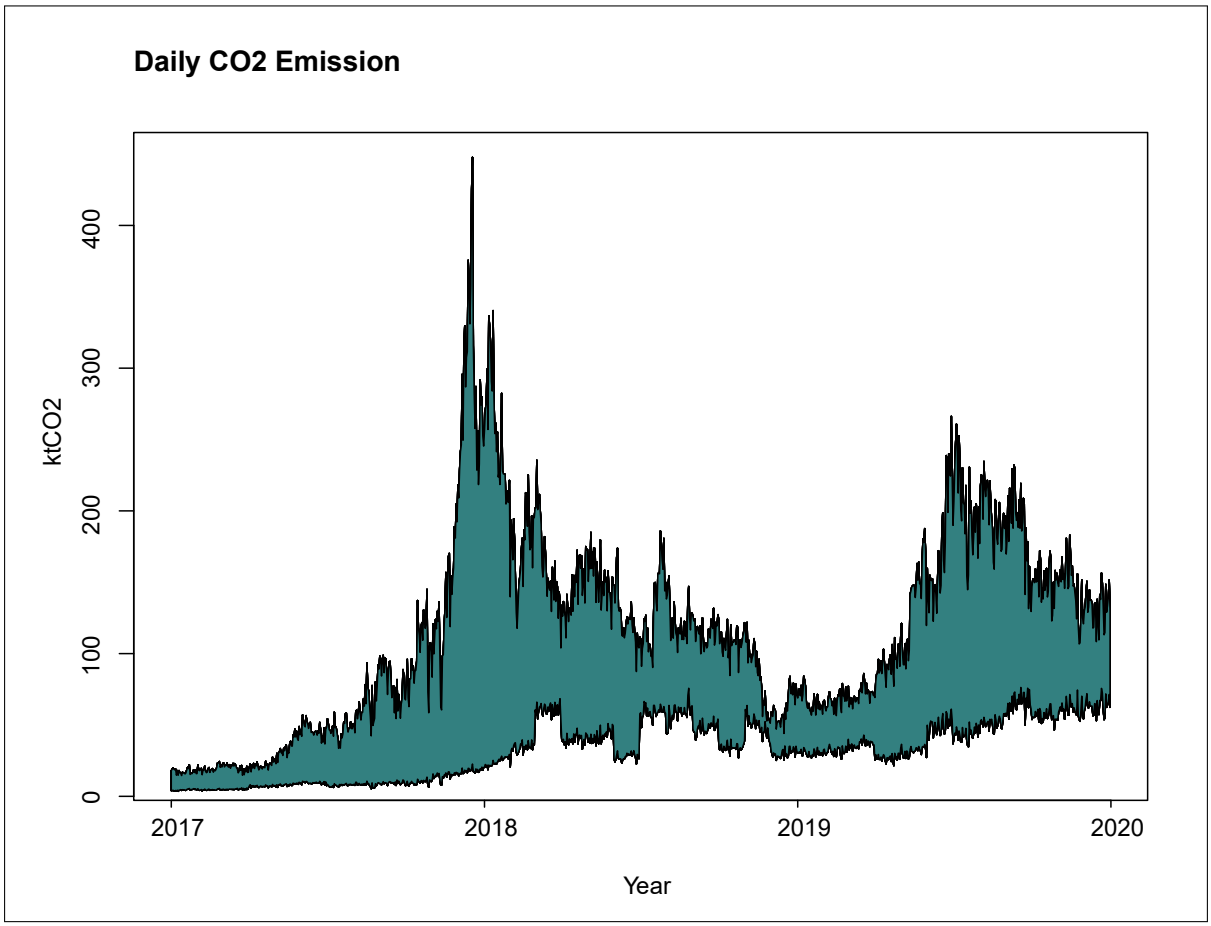

Figure 5: The Figure reports the lower and upper bound for the daily CO_2 estimates.

level electricity emission factors of 0.97463 for China, 0.63111 for Iran, 0.5132 for Russia, 0.5471 for the United States, and of 0.2081 for Venezuela (Brander et al., 2011). Figure 5 displays the evolution of the upper and lower limits of the Bitcoin network daily carbon footprint (measured in $ktCO_2$) over the 2017-2019 period.

The annual Bitcoin network carbon footprint lower $\underline{CO_2}$ and upper $\overline{CO_2}$ limits obtain from adding the corresponding daily CO_2 emissions over the year, for each year considered, reported in million tons of CO_2, $MtCO_2$: between 3.2390 and 26.6627 $MtCO_2$ for 2017, between 15.5064 and 49.7271 $MtCO_2$ for 2018, and between 16.7276 and 49.6521 $MtCO_2$ for 2019.

These CO_2 emissions are consistent with the estimates provided by (i) Foteins (2018) who finds that the annual carbon emissions for Bitcoin and Ethereum for the year 2017 are 43.9 $MtCO_2$, and by (ii) Stoll et al. (2019), ranging between 22.0 (device IP method) and 22.9 $MtCO_2$ (pool IP method) for 2018. From the *Global Carbon Atlas*, the estimated fossil fuels emissions as of 2018 are 44 $MtCO_2$ for Norway, 41 $MtCO_2$ Sweden, 47 $MtCO_2$ for Finland, and 35 $MtCO_2$ for New Zeland, showing that the *maximum* level of CO_2 that can be produced by Bitcoin mining is higher than (i) the emissions of these countries, than (ii) those of US states like Connecticut, Maryland, Nebraska, New Mexico or Oregon, and than (iii) those of all Earth's 91 subaerial volcanoes (i.e. not under water), with average yearly emissions of 38.7$\pm$ 2.9 $MtCO_2$ between 2005 and 2015 (Aiuppa et al., 2019). Focusing on the lower bound, for example, *minimum* total Bitcoin CO_2 emissions for the year 2018 are higher than the levels of annual fossil fuel emissions from smaller countries, like Bolivia, Sudan or Lebanon.[51]

4 ML-based Carbon Footprint in Bitcoin Production

The most salient fact about Figure 5, displaying the upper and lower limits for the carbon footprint associated with Bitcoin network mining based on Hayes (2015) and Stoll et al. (2019), is the uncertainty surrounding the actual CO_2 emissions generated by Bitcoin production. This uncertainty stems from the difficulty in estimating the actual power consumption involved in

ary sources (Electricity and Power consumption) should be calculated by multiplying the source consumption by the corresponding emission factor (IPCC, 2006). Since we are interested in the carbon footprint of the Bitcoin network mining electricity consumption, which spans many different countries, we would need to construct for each country the upper and lower limits of electricity consumption, which would require us to know what is the contribution of the miners located in each country to the overall network hash rate, something that to the best of our knowledge cannot be done. In view of that, the best we can do is to compute the average carbon intensity of power production as a weighted average of the country-specific carbon intensities, weighted by each country's share of miners' Antminer IP addresses, just as Stoll et al. (2019) do.

[51] The $34GtCO_2$ estimated minimum increase in global emissions between 1900 and 2020 appears to correspond to a global average temperature increase of 1.2°C, although the underlying relation is non-linear and the models are not able to differentiate between natural and anthropogenic effects.

Bitcoin mining. In this section we exploit supervised machine learning (ML) methods to narrow down that uncertainty and provide a more accurate quantitative prediction.

Concretely, we deploy a deep neural network with a rectified linear unit activation function, or ReLu DNN, which will have as target output y, the 'realistic' CO_2 emissions, CO_2^r, from the Bitcoin network daily electricity consumption E^r associated with a 'realistic' energy efficiency use of hardware, e^r.[52] It is therefore a conservative approach, which will closely 'track' the lower CO_2 emission limit, $\underline{CO_2}$ in (20):

$$\underset{[\text{ktCO}_2 \text{ per day, per TH/s}]}{CO_2^r} = E^r \cdot I = e^r \cdot H \cdot I \times 10^{-9} \tag{21}$$

where I is the average carbon intensity and H is the daily Bitcoin network hash rate. The 'realistic' energy efficiency $e^r = \sum_{m=1}^{M} s_m^{ASIC} \cdot e_m^r$ obtains as a weighted mean of the *average* energy efficiency of all the reported ASIC mining chips at a given date, e_m^r, as displayed in Figure 2. In particular, considering M rational miners operating in the network, it is assumed that when a new mining chip is available, miner m will invest in updating the hardware. Therefore, the computational power of a particular mining chip at a given date is considered indicative of the energy efficiency of the ASIC producer m, until the release of a new chip. The weights associated with each ASIC mining chip producer, s_m^{ASIC}, are identified by the market share in terms of either computing power or revenue, and are obtained from the IPO filings disclosed in 2018 by Bitmain, and in 2019 by Canaan. For 2017, Frost & Sullivan report that Bitmain accounted for 74.5% of the revenue of the global ASIC mining hardware, Company E for 6.2%, and Company F for 4.5% (E and F's companies names were undisclosed). As of November 2018, Stoll et al. (2019) report that Bitmain accounts for 76% of the network computing power, and Canaan and Ebang account for 12%. Finally, looking at the IPO filings disclosed in November 2019 by Canaan, Frost & Sullivan report that as of July 2019, Bitmain accounts for 65.2% of the computing power of the market, Canaan for 21.9%, and Ebang 7.9%. Based on these estimates, Figure 6 reports the actual weights, s_m^{ASIC}, between 2017 and 2020, assuming that they are constant during a given calendar year. Based on equation (21), Figure 7 displays the daily evolution of the 'realistic' level of CO_2 emissions, CO_2^r, from Bitcoin miners operating in the network over the period. It constitutes the target output y to be learned by our supervised ML ReLu DNN, on the basis of the collected input data $\mathbf{X}$.

Because Bitcoin is a cryptocurrency based on a fundamentally new technology not fully understood –'blockchain'– while performing similar functions as other, more traditional assets, one key advantage of our ML-based approach is that it can handle big and complex input data in raw form, $\mathbf{X} = \{...\mathbf{x}_p...\}$. Our ReLu DNN will admit a very comprehensive set of

[52] This is similar to Stoll et al.'s (2019) 'best guess' power consumption, although we ignore losses from cooling and IT equipment (or 'power usage effectiveness').

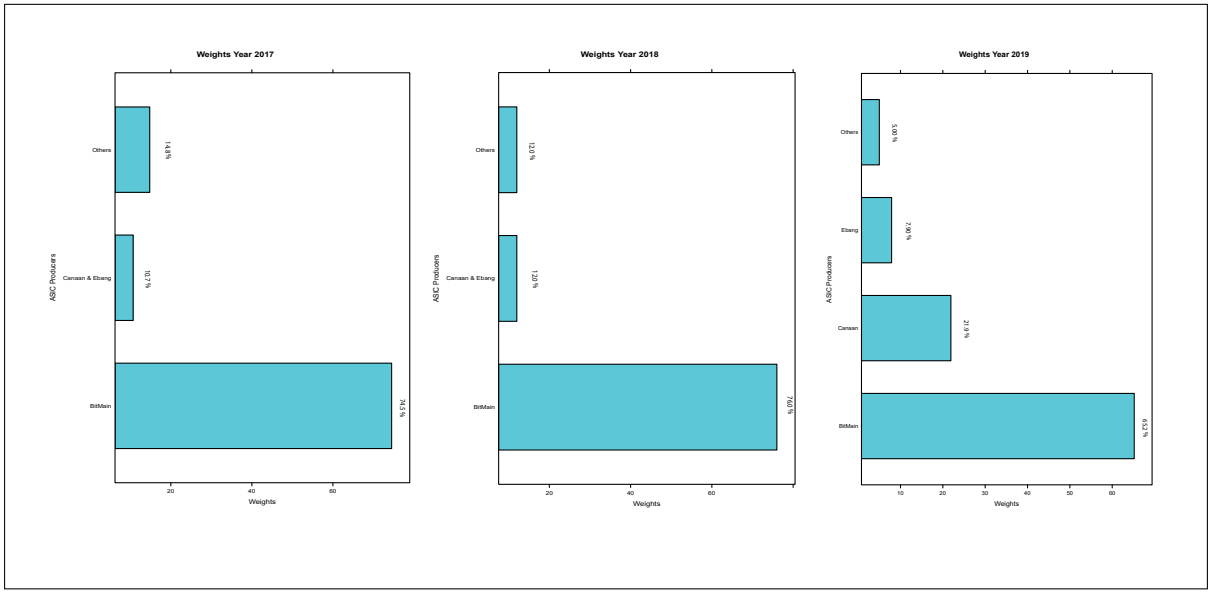

Figure 6: The Figure reports the weights assigned from 2017 until 2020 to the weighted mean of the energy efficiency of the different ASIC mining hardware.

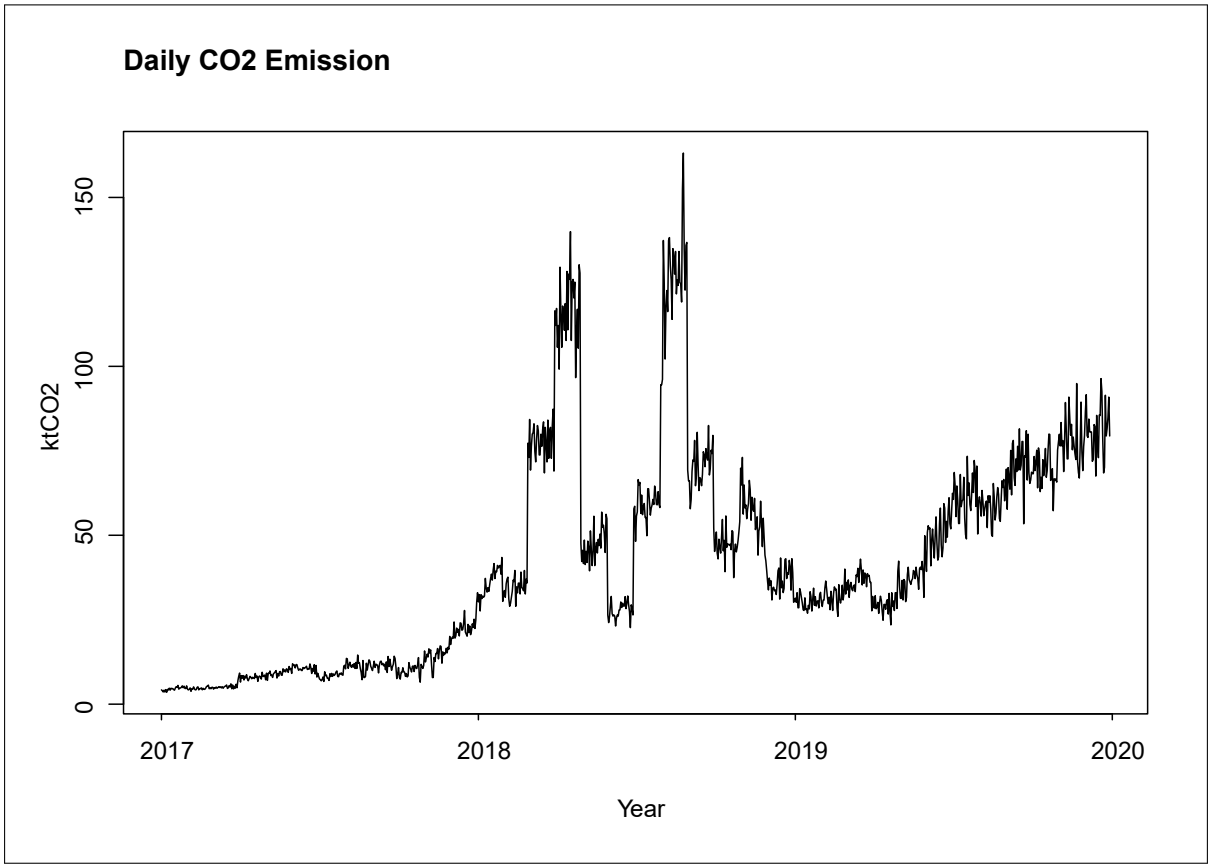

Figure 7: The Figure reports the realistic daily CO_2 emission in $ktCO_2$.

$p = 1...P$ factors, excluding those necessary to compute the carbon footprint lower $\underline{CO_2}$ and upper $\overline{CO_2}$ limits, derived in the previous section. The rationale behind this exclusion is to test whether our ML-based CO_2 emission mean predictions lie within the bounds that obtain from basic economic principles, externally validating our ML approach. In addition we provide (95%) confidence intervals around our ML-CO_2 predictions which are substantially narrower than the economics-based bounds, contributing methodologically to the ML and climate change literatures.

4.1 Input Data

The factors considered as input data range from (i) standard fundamental factors advocated by monetary economics and the quantity theory of money, like predictors of the Bitcoin price level; (ii) factors driving investors' interest in/attention to the crypto-currency, like speculation or the role of Bitcoin as a safe haven; (iii) exchange rates with other currencies, to capture investors' hedging motives, e.g. the tight connection between the USD and the CNY markets; or (iv) supply-side factors for the costs incurred by Bitcoin and ASIC mining chips producers, related to rational for-profit mining decisions. Factors associated with the Blockchain network operation, like the network hash rate, difficulty or block reward, are excluded as they enter the definition of either the Bitcoin carbon footprint upper and lower bounds, or of our ReLu DNN target output. The resulting novel input dataset for the period 01/01/2017 - 31/12/2019 covers a comprehensive set of factors as reported in Kristoufek (2015), Liu and Tsivinsky (2018), McNally et al. (2018) and Jang and Lee (2018), adding some novel ones. Data are collected at different frequencies, and converted into daily ones using either simple imputations or Random Forest algorithms.

Because Bitcoin prices determine the upper limit of CO_2 emissions generated by the break-even electricity consumption of rational Bitcoin network miners, we start with the predictors of Bitcoin prices identified in the literature:

1. Commodity prices of Gold, platinum and crude oil are included because of the common traits shared with cryptocurrencies such as limited supply and high price volatility, but also because it is believed that Bitcoin could serve as an alternative to these commodities either as a store of value or as a hedging instrument (Dyhrberg, 2016). The daily future price of crude oil, and the spot prices of platinum (USD/ounce) and gold (USD/ounce) are obtained from Bloomberg;

2. Macroeconomic factors in different markets, such as consumption, production, and personal income growth (in USD), measure the extent to which Bitcoin is perceived as a traditional financial asset, like the stock market. The $CAIPMOM$, $UKIPIMOM$,

IPCHNG, *JNIPMOM*, and *SIIPMOM* indices, measuring the volume of output in the industries of mining and quarrying, manufacturing and public utilities (electricity, gas, and water supply) for the USA, the UK, China, Japan and Singapore, as well as the indices *PITL* and *PITLCHNG*, measuring the income received by households including wages and salaries, investment income, rental income, and transfer payments in the USA and China, are included. Finally, the *PCEMOM* index quantifying the price changes for goods and services purchased by consumers in the USA is also considered;

3. Relative asset market performance measures capture the extent to which Bitcoin is similarly exposed to factors driving the returns of traditional assets. Based on Figure 3, we include the major stock market indices of the countries most relevant for Bitcoin mining: the USA, China, Venezuela, and Europe. For this reason, the indices S&P 500, Dow Jones, Nasdaq, Euro Stoxx 50, Shanghai Stock Exchange (SSE), Nikkei 225, FTSE 100, Caracas Stock Exchange (IBVC), and SHASHR will be considered as predictors;

4. Investor attention, measured by "Bitcoin" word Google searches. Liu and Tsyvinki (2018), Garcia et al. (2014) or Bouoiyour et al. (2014) empirically show that only cryptocurrency market specific factors –momentum and the proxies for investor attention– consistently explain the variations of cryptocurrency returns, suggesting that investors do not perceive them as traditional assets. Figure 8 reports the geographic location of daily data returned from Google Trends search queries for the word "Bitcoin", which quantifies the interest in the form of an index between 0 and 100. A value of 100 corresponds to peak popularity, and of 0 to insufficient data for Google to quantify any interest in the term "Bitcoin". With the exception of Nigeria, the country that receives the highest interest index, one could notice the similarity with Figure 3, where the geographical location of Bitcoin miners' IP addresses from the IoT search engine *Shodan* can be visualized, suggesting that a high value of the interest index is associated with Bitcoin mining activities.

5. Exchange Rates are included because of the popular belief that Bitcoin, if sufficiently adopted, may replace existing fiat currencies as a medium of exchange. Exposure of the cryptocurrency returns to major currencies is captured by the inclusion of the spot exchange rates between the USD and units of foreign currency, for the Australian Dollar, the Euro, the British Pound, the Canadian Dollar, the Singapore Dollar, the Swiss Franc, the Japanese Yen, the Chinese Yuan Renminbi (CNH), and the Chinese Yuan (CNY), all collected from Bloomberg. Being the Bitcoin price denominated in USD, Ciaian et al. (2015) notice that an appreciation of the USD against the above currencies could result in an appreciation against the Bitcoin and thereby, affect mining decisions through the

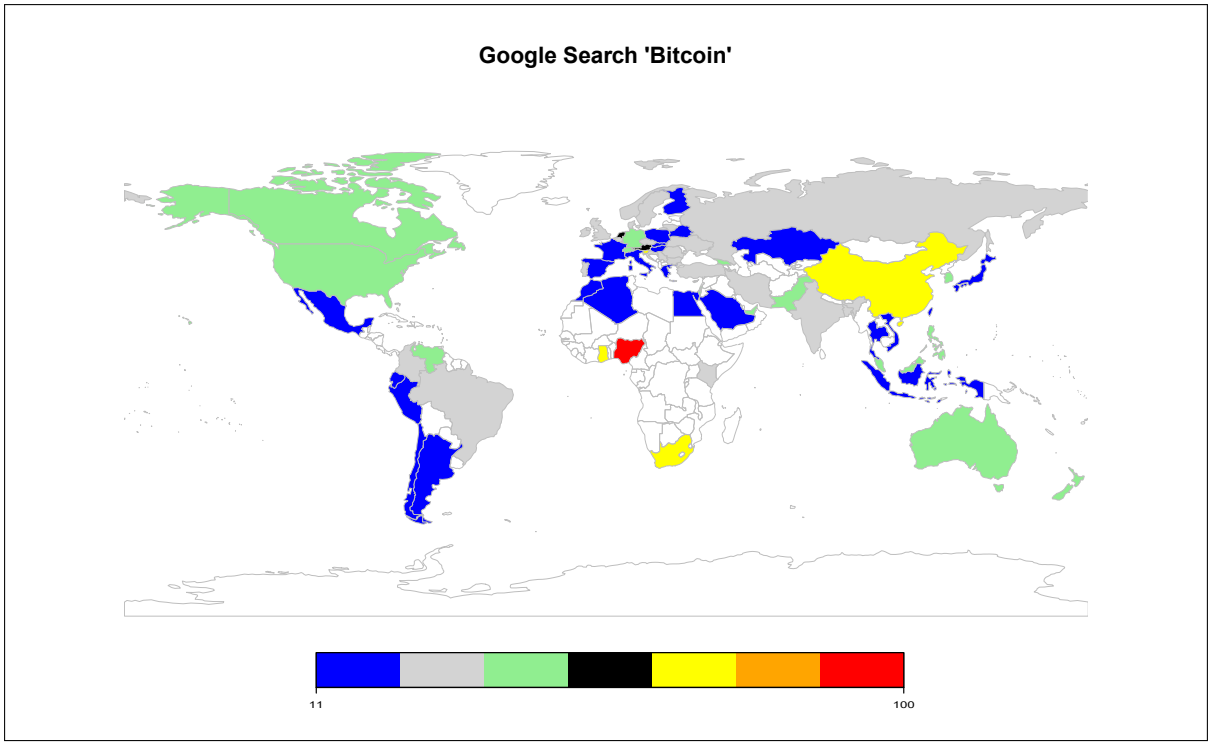

Figure 8: The Figure reports the Google search "Bitcoin" using 100 as reference for the maximum interest.

reduction in the price of the cryptocurrency.[53]

6. The FED financial stress index (FSI) is a popular measure of financial uncertainty. Its inclusion is intended to capture the possibility that Bitcoin is perceived as a safe haven, following Kristoufek (2015). The weekly series is provided by the Federal Reserve Bank of St. Luis (2016), and it is built from 18 different series of data at a weekly frequency: seven interest rate series, six yield spreads and five other indicators, each of which capturing a different aspect of 'financial stress'. The FSI is centered around 0 ('normal financial stress'), with negative values indicating unusual calmness and positive ones 'abnormally high' levels of financial uncertainty.

Finally, supply factors that proxy for the costs of Bitcoin mining and ASIC mining chips producers are also included:

7. ASIC mining chips producers offer mining hardwares (e.g. Antminers) the profitability of which is directly related to the marginal costs that can be expected from Bitcoin mining. Being electricity the most important input in mining for Bitcoins, we follow Liu and

[53] We exclude the exchange rates of Bitcoin against other cryptocurrencies, like Ethereum or Ripple, because they are less popular, were introduced later and there is little evidence of significant arbitrage activity with respect to Bitcoin.

Tsyvinski (2018) and include the weighted average of the daily stock returns of 25 electricity companies in the USA and of 65 electricity companies in China, and the daily stock returns of Sinopec (SNP).[54]

8. To proxy for the cost of inputs relevant for manufacturing Antminers, we include the aluminum ($/25 Mt) and copper ($/25 Mt) prices –from Bloomberg– and predictors of the supply of coltan by its largest producers: the CDMNCLT index measuring the value (USD) of the mining and oil production in the Democratic Republic of Congo; and the RWEXCLVA and RWEXCLVO indices measuring the value and the volume (USD) of trade of coltan from Rwanda.[55]

After computing the log returns for the exchange rates, market indices, commodities prices, Sinopec prices and weighted averages of the electricity prices in the USA and China, a 'feature-wise normalization' or standardization –i.e features are centered around zero with unit standard deviation– is performed considering only the training dataset.[56]

4.2 ReLu DNN-CO_2 Estimation

We estimate the carbon footprint associated with Bitcoin network mining by a ReLu feedforward deep NN, proceeding in two steps. First, we obtain the optimal structure $(\widehat{L}, \{\widehat{Z}_l\}_{l=1}^{L})$ of the ReLu DNN from (12) for a given architecture size Z and set of hyperparameters, $\mathcal{H} = \{\epsilon, \lambda, \alpha, p, B, (s, \rho), (s_1, s_2, \rho_1, \rho_2)\}$. Second, we optimize/fine-tune the DNN hyperparameters $(Z, \mathcal{H})$ to solve (13) and determine $\mathbf{\Lambda}_L(\widehat{L}, \{\widehat{Z}_l\}_{l=1}^{\widehat{L}}; Z, \mathcal{H}) = [\boldsymbol{\theta}_L; (\widehat{L}, \{\widehat{Z}_l\}_{l=1}^{\widehat{L}}), Z, \mathcal{H}]$,
where $\boldsymbol{\theta}_L = \{(\mathbf{b}_1, ..., \mathbf{b}_L, b_{L+1}; \mathbf{W}_1, ..., \mathbf{W}_L, \mathbf{w}_{L+1})\}$ contains the biases and weights of all hidden units optimally allocated across layers. To validate the optimal structure obtained, we benchmark it against (cv) a DNN cross-validated architecture[57], and against (rf) a Random Forest,

[54]The Sinopec has 4.02% missing at random values at a daily frequency, which are inputted using the MissForest algorithm (Stekhoven, 2013). The maximum number of trees to be grown in each forest is set equal to 500, the maximum number of nodes for each tree is equal to 500, and the maximum number of iterations is 20. The Out-Of-Bag (OOB) estimates of the imputation error in terms of Normalized Root Mean Squared Error (NRMSE) is: 2.160×10^{-9}. Eighty percent of the observations are used to train the network, while the remaining twenty percent are exploited to evaluate the out-of-sample accuracy.

[55]Copper is largely used for the production of electrical wires due to its high conductivity, heat resistance, and low cost. Aluminum wires are used for power transmission and distributions (generally not used in households). Coltan is employed in the production of tantalum capacitors, which are essential to manufacture mining hardware and computers.

[56]Being the levels and variances of the 42 input daily series considered significantly different, 'feature-wise normalization' is done to guarantee a proper training of the ReLu DNN.

[57]We perform cross-validation only on the optimal node allocation, for given DNN depth and size. Cross-validating both depth and hidden unit allocation across layers in network architectures of different sizes, is extremely computational and time consuming, and is therefore left for future work.

both state-of-the-art in the deep learning literature. Regression trees and their extension, Random Forests, are 'tree-structured' methods commonly used for flexibly estimating regression functions where out-of-sample performance is important.[58]

The number of input variables is $P = 42$, and we allow for a maximum depth of $L = 15$.[59] Due to the high dimensionality of the combinatorial problem, it is not feasible in (cv) to cross-validate all combinations and a 4-fold cross-validation over a randomized gridsearch is implemented instead. For each case, we report the out-of-sample mean-absolute error (MAE), mean-squared error (MSE) and square root of the MSE (RMSE), with and without dropout. Different architecture sizes Z, optimization algorithms (Adam, RMSProp), weight initialization values (s, s_1, s_2), learning rates ϵ, dropout rates p, and training epochs are considered during training.[60] The default 'minibatch' size of $B = 32$ is adopted and not tuned. The Random Forest hyperspace in (rf) is defined by the following parameters: (a) the number of variables to be randomly sampled at each sample split is defined in the interval $[20, 40]$, by intervals of 2; (b) the minimum size of the terminal nodes in $[2, 20]$, by intervals of 2; and (c) the number of

[58] 'Tree-structured' methods have dictionaries of the form $\{\mathbf{1}_{\{\mathbf{X}\in R\}}\}_R$ where $\mathbf{1}_{\{.\}}$ is an indicator function, and R represents subregions of the space of all possible values of $\mathbf{X} \in \mathbb{R}^P$, $R \subseteq \mathbb{R}^P$. The most common example is $\mathbf{1}_{\{\mathbf{X}\in R\}} = \prod_{p=1}^{P} \mathbf{1}_{\{u_p \leq x_p \leq v_p\}}$ with the $2P$ coefficients $\{u_p, v_p\}_{p=1}^{P}$ representing the respective lower and upper limits of the region (hyper-rectangle) on each input x_p axis. Usually only Z disjoint regions are chosen, $\{R_z\}_{z=1}^{Z}$, so that $\mathbf{X} \in R_z \implies \widehat{f}(\mathbf{X}) = a_z$, meaning that $\mathbf{X}$ in the same region have the same 'approximation' value a_z (with an obvious abuse of notation, but with a similar interpretation). Recursive partitioning tree-structured methods are also universal approximators, in the sense defined previously, i.e. $\widehat{f}(\mathbf{X}) = \sum_{z=1}^{\infty} a_z^* \mathbf{1}_{\{\mathbf{X}\in R_z\}} = f(\mathbf{X})$. Choosing the optimal number of regions Z is a formidable combinatorial optimization problem, but recursive partitioning is an approximate solution when employing greedy optimization strategies. This effectively results in sequentially splitting the initial sample $\{y_i, \mathbf{X}_i\}_{i=1}^{N}$ starting with the single covariate x_p that minimizes the mean-squared error of the resulting subsamples (or *leaves*). Considering one different covariate at a time, the mean-squared error is therefore sequentially reduced. But too many subsamples (a *very deep tree*) would correspond to a very large Z, which risks overfitting. Therefore, in practice, a very deep tree is estimated and then *pruned* (or regularized) to a more sparse tree, using cross-validation to select the optimal depth. See Athey and Imbens (2019) for further details.

[59] As robustness check the optimization is performed considering $Z + 1$ and $Z - 1$ hidden nodes, producing the same results and thus showing convergence of the optimization algorithm.

[60] In particular, the different architecture sizes considered are $Z = \{200, 500, 800, 1674, 1800\}$,.the learning rates $\epsilon = \{0.0001, 0.001, 0.01\}$ for the Adam optimiser ($\rho_1 = 0.9, \rho_2 = 0.999$), for the Stochastic Gradient descent (SGD) with Nesterov momentum of $\alpha = 0.9$, and for the RMSProp optimiser with $\rho = 0.9$ are tuned. When the Adam optimizer is considered the He normal initializer that draws samples from a truncated normal distribution with $\mu = 0$ and $\sigma = \sqrt{2/\text{Indim}}$ where "Indim" is the number of input units in the weight tensor (Keras documentation, 2020); when instead the SGD is tuned, a truncated normal distribution with $\mu = [0.5, 0.1]$ and $\sigma = [0.02, 0.01]$ is considered. The maximum number of training epochs analyzed are: 500, 1000, 2000, 5000 and 8000, and early stopping is applied. Different dropout rates, $p = \{0.05, 0.1, 0.2, 0.3\}$ are tuned for all hidden layers, allowing also different dropout rates in different layers.

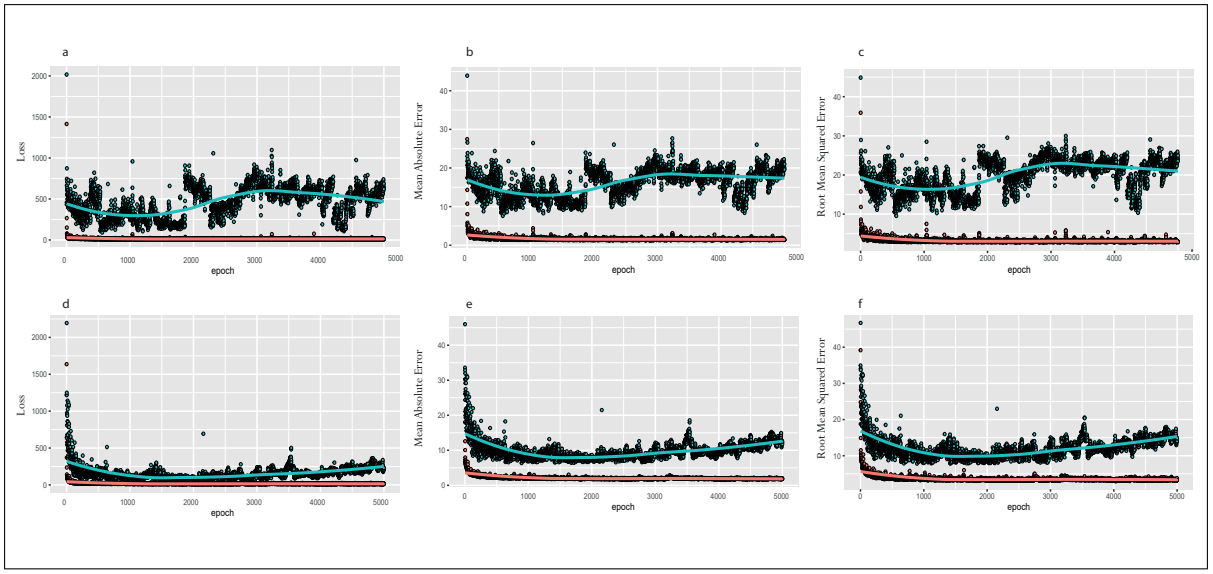

Figure 9: The Figure reports the training (in red) and validation (in green) Loss, MAE, and MSE for the neural network the structure of which is selected with the optimisation algorithm. Subfigures a, b, and c refers to a network trained without dropout; Subfigures d, e, and f to the network trained with dropout.

trees to grow in the interval $[50, 500]$, by intervals of 50.[61]

The cross-validated NN architecture size that minimizes the out-of-sample MSE is found to be $Z = 1674$, with an optimal depth of $L = 15$ and optimal allocation of hidden units $[162, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126, 126]$, for cross-validated hyperparameters: Adam optimizer, $\rho_1 = 0.9, \rho_2 = 0.999$; learning rate, $\epsilon = 0.001$, dropout rate, $p = 0.05$ for all hidden layers, and number of epochs, 5000. The out-of-sample performance of the optimal ReLu DNN measured by the MAE, MSE, and RMSE are: (i) without dropout, 7.6177, 91.2926, and 9.5527, respectively (reported in figure 9, panels a., b. and c., green curves); and (ii) with dropout, 6.3582, 61.3806, and 7.8264, respectively (reported in figure 9, panels d., e. and f., green curves). Figure 9 therefore conveys how dropout, by reducing the number of hidden units retained, increases the NN architecture sparsity, reducing overfitting and increasing out-of-sample prediction accuracy.[62]

When benchmarked against (cv), the equally-sized 4-fold crossvalidated network architecture of $[238, 48, 63, 162, 179, 90, 153, 78, 187, 154, 83, 118, 121]$ performs worse out-of-sample (with and without dropout). The values for the MAE, MSE, and RMSE are: (i) 8.0264, 103.4196, and

[61]The authors acknowledge the use of the IRIDIS High Performance Computing Facility, and associated support services at the University of Southampton, in the completion of this work.

[62]Notice that applying dropout should increase training error (red curves, in all panels a.-f. of figure 9) relative to not using dropout, since the same training sample variation is explained with less hidden units when dropout is applied. That it does not is an open research question in the ML literature.

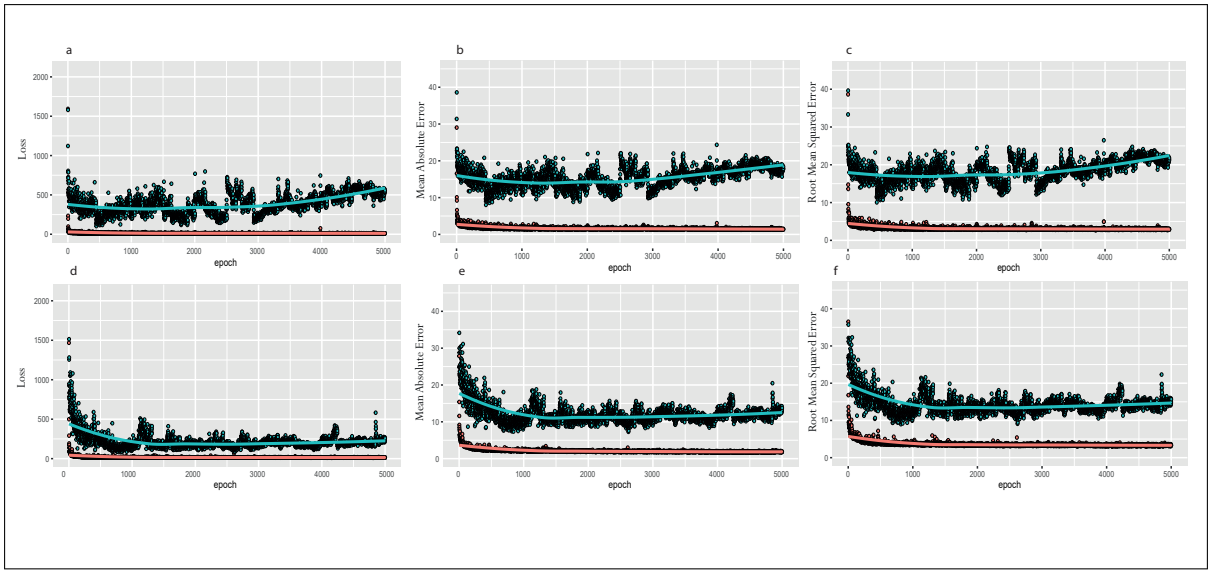

Figure 10: The Figure reports the training (in red) and validation (in green) Loss, MAE, and MSE for the neural network the structure of which is selected with randomized crossvalidation. Subfigures a, b, and c refers to a network trained without dropout; Subfigures d, e, and f to the network trained with dropout.

10.0585, respectively (without dropout: reported in figure 10, panels a., b. and c., green curves); and (ii) 7.4382, 81.5511, and 8.9936, respectively (with dropout: reported in figure 10, panels d., e. and f., green curves). Just as before, and for the same reason, figure 10 conveys that dropout leads to an increase in the out-of-sample prediction accuracy. Finally, benchmarking against (rf), a Random Forest with node size of 20, 50 trees, and 34 variables randomly sampled at each split, has associated out-of-sample MAE, MSE, and RMSE of 9.8830, 143.6868, and 11.9869, respectively. A pair-wise model comparison test statistic of the difference in out-of-sample MSE proposed in Calvo-Pardo et al. (2020a), delivers a value of 4.6411 (with associated p-value < 0.0001) of our optimal ReLu DNN against the (rf) random forest, and of 2.6976 (with associated p-value of 0.0035) against the (cv) equally-sized cross-validated ReLu DNN, with levels of statistical confidence above 1 percent.

Overall, our optimal ReLu DNN strategy to measure the carbon footprint associated with Bitcoin mining outperforms relative to the predictions obtained from (rv) a Random Forest and (cv) a cross-validated ReLu DNN architecture, both state-of-the-art ML methods. To further contribute to the literature on the carbon content of economic activity, we quantify the statistical reliability of the ML-measured CO_2 emissions associated with Bitcoin network mining.

4.2.1 Confidence Intervals

As advanced in the previous section, we construct 0.95 confidence intervals (CIs) around the first moment of the predictive distribution of the CO_2 Bitcoin network emissions (target out-

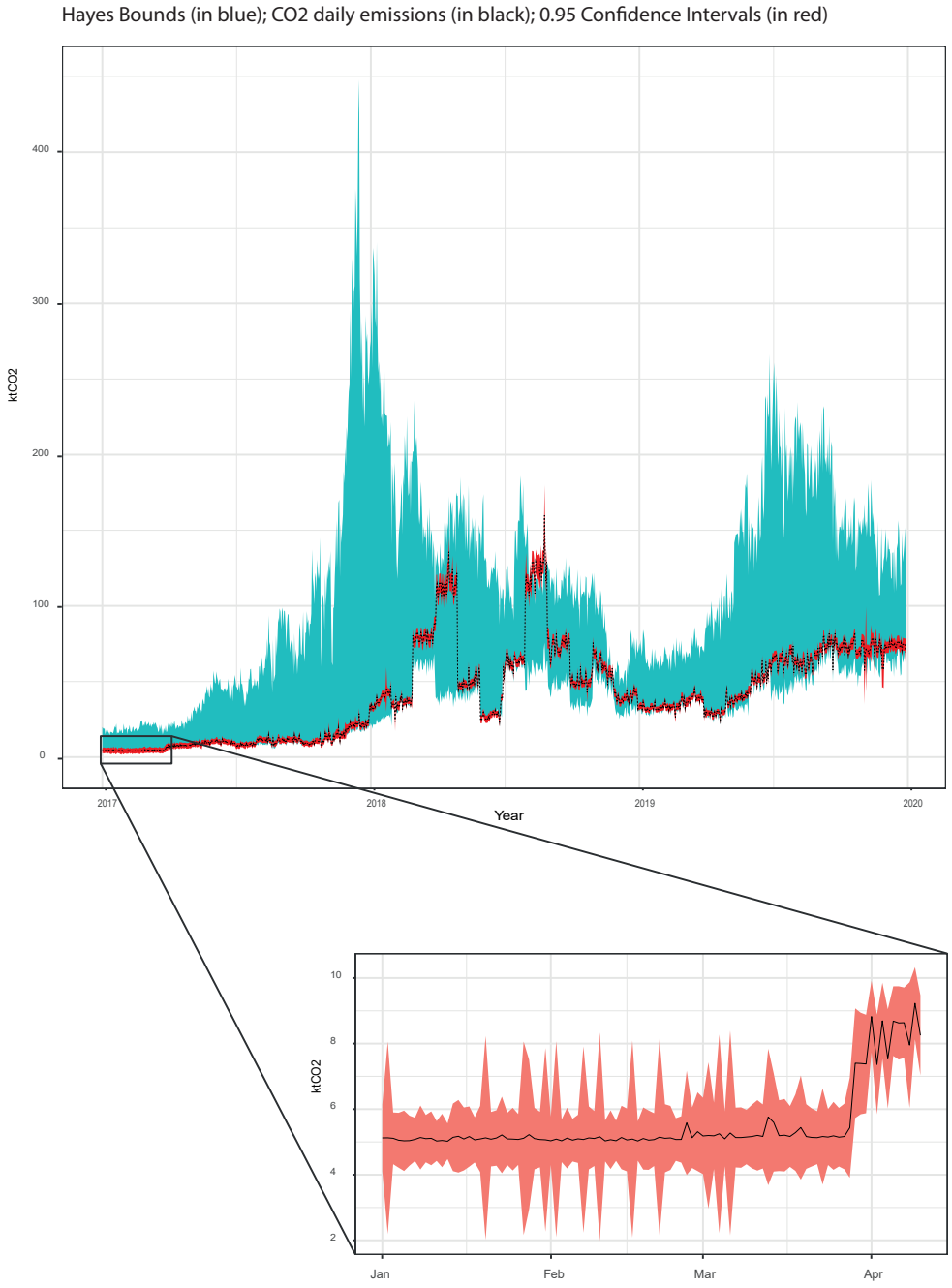

Figure 11: The Figure reports in blue the economics bounds proposed by the literature, the 0.95 confidence intervals in red, and the estimated CO_2 emissions.

put y).[63] The out-of-sample MAE, MSE, and RMSE of the predicted CO_2 emissions target output are 7.1984, 78.1076, and 8.8379, respectively, with an empirical coverage over the entire dataset of 0.22. In this case, the pair-wise model comparison test statistic in Calvo-Pardo et al. (2020a) delivers a value of 5.2406 (with associated p-value < 0.0001) of our optimal ReLu DNN against the (rf) random forest, and of 0.6438 (with associated p-value of 0.2599) against the (cv) equally-sized cross-validated ReLu DNN. Figure 11 reports the estimated CO_2 emission values and associated 0.95 CIs for the overall period. When aggregated at a yearly frequency, the corresponding CO_2 estimates [and associated 0.95 CIs] are: for the year 2017, 3.8038 $MtCO_2e$ $[3.2151, 4.3925]$ $MtCO_2e$; for 2018, 23.8313 $MtCO_2e$ $[22.1055, 25.5572]$ $MtCO_2e$; and 19.83472 $MtCO_2e$ $[18.4852, 21.1842]$ $MtCO_2e$ for the year 2019.[64]

Three main aspects stand out: first, the ML-measured daily CO_2 emissions almost always remain within the rational Bitcoin mining upper and lower bounds defined in Hayes (2015), also exploited by Stoll et al. (2019), despite of not using that information directly as inputs, $\mathbf{X}$. The exception appears to be a small window in 2018 where Bitcoin miners would appear to be operating at a loss. Second, the 0.95 confidence intervals provide a quantitative measure of the uncertainty associated with the ML-based Bitcoin mining carbon footprint, which is *substantially narrower* than the one captured by the difference between Hayes' (2015) upper and lower bounds. That difference captures the distance between the levels of emissions associated with the lowest marginal cost (or highest level of energy efficiency) and the expected marginal revenue evaluated at daily Bitcoin market prices, corresponding then to the expected daily operating margin of rational Bitcoin miners' decisions. Third, the estimates (and CIs) are in line with recent literature downward revisions of the original estimate of 69 $MtCO_2e$ provided by Mora et al. (2018) for 2017, e.g. 15.5 $MtCO_2e$ by Houy (2019), excluding unprofitable mining rigs; or 15.7 $MtCO_2e$ by Masanet et al.(2019); as well as for 2018, e.g. 43.9 $MtCO_2e$ (for Bitcoin and Ethereum) estimated by Foteinis (2018), or the lower and upper bounds of 22.0 to 22.9 $MtCO_2e$ estimated by Stoll et al. (2019) for Bitcoin mining activity.

To provide an order of magnitude, the estimates for the years 2018 and 2019 are comparable to the CO_2 yearly emissions of countries such as Bolivia, the Dominican Republic, or Croatia. Adopting the *social cost of carbon* (SCC) estimate of 62 USD per metric ton of CO_2 equivalent

[63] Concretely, *inverted dropout* on the estimated weights obtained from (13) after (12), $\widehat{\widehat{\mathbf{W}}}_l = (1/\widehat{p})\widehat{\mathbf{W}}_l, l = 1...L$, is conducted, to then run $T = 1000$ stochastic forward passes through the optimal ReLu DNN (without dropout) with weights $\widehat{\widehat{\mathbf{W}}}_l, l = 1...L$, producing a sample $\{\widehat{y}(\mathbf{X}, \widehat{\widehat{\mathbf{W}}}_1^t, ..., \widehat{\widehat{\mathbf{W}}}_L^t)\}_{t=1}^{1000}$ from sampling $T = 1000$ sets of vectors of realizations from the Bernoulli distribution $\{[\mathbf{r}_l^t]\}_{t=1}^T$ with cross-validated probability $\widehat{p}$.

[64] When instead the deterministic dropout introduced by Srivastava et al. (2014) is applied, we obtain very similar CO_2 emission levels, i.e. 4.0361 $MtCO_2$ for year 2017; 24.7312 $MtCO_2$ for 2018; and 20.9050 $MtCO_2$ for 2019, and all fall inside the 0.95 CIs provided.

(Interagency Working Group, IWG, 2016)[65], the yearly Bitcoin mining SCC is estimated to lie with 95 percent confidence in between [$199,336,200; $272,335,000] for the year 2017; in [$1,370,541,000; $1,584,546,400] for 2018; and in [$1,146,082,400; $1,313,420,400] for 2019. Recalling that the greenhouse emissions estimates reported here are a lower bound in that they exclude the electricity consumption associated with the transactions executed using Bitcoin, one could conclude that the economic social cost associated to the proof-of-work algorithm is significant. This is an important aspect that must be considered by policy makers or financial institutions that are adopting blockchain technologies for national cryptocurrency production (e.g. China), or for the emission of financial instruments (e.g. *bond-i*), because of the Paris agreement that requires to all parties to put forward policy measures intended to keep the rise in temperature below 2°C (e.g. zero net carbon emissions). Mora et al.'s (2018) projections, notwithstanding the aforementioned downward revisions, were based on the Paris agreement climate sensitivity of 2°C.[66] But the latest evidence from a global effort comprising dozens of climate-change models (in an ensemble called the Coupled Model Intercomparison Project, CMIP6)[67], feeding into the Sixth Assessment Report of the Intergovernmental Panel on Climate Change (IPCC) due next year, now indicate climate sensitivities exceeding 5°C. Williams et al. (2020) have just confirmed on the basis of the CMIP6 Met Office Unified weather-climate

[65]The National Academies of Science (2017) define the SCC as: "*The Social Cost of Carbon for a given year is an estimate in dollars, of the present discounted value of the future damage caused by a* 1 *metric ton increase in carbon dioxide (CO_2) emissions into the atmosphere in that year or, equivalently, the benefits of reducing CO_2 emissions by the same amount in that year.*" The damages include human health, flood risk, variation in the agricultural productivity, and the positive externalities associated to the ecosystem. Using Integrated Assessment Models (IAM) economists and scientists have been trying to quantify the economic impact of an increase in one tonne of CO_2 (see PAGE model by Hope et al., 1993). It is widely accepted that the aforementioned methodology suffers from an intrinsic uncertainty, and relies heavily on assumptions regarding aspects such as the population, and the gross domestic product (GDP) growth when the socioeconomic module is considered; assumptions regarding the conversion of CO_2 emissions into temperature changes, or atmospheric concentration and pressure when the climate module is considered; the IAM methodology relies also on the methodology adopted to convert the increase in temperature and other changes in environmental variables arising from an increase in greenhouse emissions into social and economic damages (damage module); and finally (discounting module) the estimate in dollars depends on the discount rate adopted to compute the present value of the monetized damages estimated through the years by the damage module. For all these reasons, we report that the discount rate used for the economic damage estimates is 2.5%, but discussing the assumptions underlying each methodology and associated estimates are beyond the scope of this paper. The IWG (2016) monetizes the SCC as 56 in 2015 and 62 in 2020 USD per $MtCO_2e$, in 2007 USD. See Hope et al. (1993) and Wang et al. (2019) for similar estimates.

[66]Climate sensitivity refers to the global warming after climate has equilibrated to a doubling of CO_2 concentration relative to pre-industrial levels, an equilibrium that might take a few hundred years to establish.

[67]See go.nature.com/3garyzc.

model, the crucial role that cloud microphysics play in the upwards revision of the climate sensitivity figures: rising temperatures affect the size and relative concentration of water and ice droplets in a cloud ('cloud-feedback problem'), leading to more supercooled water droplets and less ice droplets. If confirmed, clouds contribute less than previously thought (e.g. CMIP5) to temperature 'cooling'. Therefore, much lower levels of Bitcoin mining greenhouse gas emissions could confirm Mora et al.'s (2018) alarming projections: we cannot afford to be complacent. Cloud adjustment to climate change means that we need to redouble our efforts to cut emissions.

5 Conclusions

There is growing concern with climate change. Recent evidence from integrated weather-climate models magnifies the contribution of greenhouse emissions, making a compelling urgent call to cut on those. By focusing on the CO_2 emissions associated with Bitcoin mining, here we show that its measurement is controversial and subject to significant uncertainty. The main reason being the complexity of the underlying object of study: how much electricity is actually consumed by the global network employed in mining for Bitcoins. In a novel application of deep learning to this pressing societal issue, we were able to provide a quantitative measure of Bitcoin mining daily electricity consumption and associated CO_2 emissions, as well as of their (statistical) reliability, improving on the current methods employed in the literature. Although our estimates are in line with recent downward revisions of those provided by Mora et al.(2018), and provide substantially narrower bounds (e.g. than those provided by Stoll et al., 2019), our conclusions point towards a significant and substantial contribution towards rising world temperatures in view of the recent evidence of climate sensitivities exceeding 5°C (relative to the Paris agreement level of 2°C).

After reviewing the growing literature on deep learning, we demonstrate how ML methods can be successfully exploited to contribute to the ongoing debate. Starting from a model of rational Bitcoin mining by Hayes (2015), and based on a comprehensive set of factors reported in Kristoufek (2015), Liu and Tsivinsky (2018), McNally et al. (2018) or Jang and Lee (2018), and some novel ones, we were able to measure the carbon footprint associated with Bitcoin mining from fitting an optimized deep neural network architecture that improves upon state-of-the-art DNN methods. ML methods help in establishing how the carbon footprint of the proof-of-work algorithm is higher than those of (i) US states of Maine, New Hampshire, Rhode Island or South Dakota, of (ii) more than half the cumulative CO_2 flux from the Earth's 91 most actively degassing subaerial volcanoes (Aiuppa et al., 2019), or of (iii) economies of the size of Bolivia, the Dominican Republic, or Croatia. Our results further point towards a significant social cost of carbon (SCC) from Bitcoin mining activity, concentrated in countries with lower electricity

prices. Next to discussions about incorporating a mining tax into electricity prices, policy decision makers could also consider alternative consensus methods, such as proof-of-authority, in their efforts to curtail greenhouse emissions associated with cryptocurrencies mining and smart contracts implemented with blockchain. Finally, ML methods could be fruitfully exploited to solve the 'measurement' problem that plagues for-profit financial efforts to decarbonize the economy, enabling objective tracking of both the carbon and financial performance of investments[68].

[68] See 'Cimate change and investing: The trouble with green finance', The Economist, June 20-26th 2020.

References

[1] Aiuppa, A., T. P. Fischer, T. Plank and P. Bani (2019) "CO2 fux emissions from the Earth's most actively degassing volcanoes, 2005–2015," Nature Scientific Reports 9:5442 available at https://doi.org/10.1038/s41598-019-41901-y

[2] Allen-Zhu, Z., Li, Y. and Liang, Y. (2019) "Learning and generalization in overparameterized neural networks, going beyond two layers". In *Advances in neural information processing systems*; pp. 6158 - 6169.

[3] Arora, S., Ge, R., Neyshabur, B. and Zhang, Y. (2019) "Stronger generalization bounds for deep nets via a compression approach" *arXiv preprint arXiv:1802.05296.*

[4] Asic Miner Index (2020) *Asic Miner Index*; Available at: `https://asic-dex.com/`

[5] Athey, S. and Imbens, G.W. (2019) "Machine learning methods that economists should know about" *Annual Review of Economics, 11.*

[6] Bouoiyour, J. and Selmi, R. (2017) "The Bitcoin price formation: Beyond the fundamental sources" *arXiv preprint arXiv:1707.01284.*

[7] Brander, M., Sood, A., Wylie, C., Haughton, A. and Lovell, J. (2011) "Technical Paper| Electricity-specific emission factors for grid electricity" *Ecometrica, Emissionfactors.com.*

[8] Calvo-Pardo, H. F., Mancini, T. and Olmo, J. (2020a) "Optimal Deep Neural Networks by Maximization of the Approximation Power". Available at SSRN: `https://ssrn.com/abstract=3578850` or `http://dx.doi.org/10.2139/ssrn.3578850`

[9] Calvo-Pardo, H. F., Mancini, T. and Olmo, J. (2020b) "Extra-Neural networks: A deep ensemble estimation of the predictive distribution", MIMEO

[10] Ciaian, P., Rajcaniova, M. and Kancs, D.A (2016) "The economics of BitCoin price formation" *Applied Economics*; 48(19), pp. 1799 - 1815.

[11] Cover, M.T and Thomas, J.A. (2006) "Elements of information theory" *John Wiley & Sons.*

[12] Cybenko, G. (1989) "Approximation by superpositions of a sigmoidal function" *Mathematics of control, signals and systems*; 2(4), pp. 303 - 314.

[13] De Vries, A., 2018. Bitcoin's growing energy problem. Joule, 2(5), pp.801-805.

[14] Dittmar, L. & Praktiknjo, A. (2019) "Could Bitcoin emissions push global warming above 2 °C?" Nat. Clim. Change https://doi.org/10.1038/s41558-019-0534-5.

[15] Dyhrberg, A.H. (2016) "Bitcoin, gold and the dollar - A GARCH volatility analysis" *Finance Research Letters*; 16, pp. 85 - 92.

[16] Farrell, M.H. (2015) "Robust inference on average treatment effects with possibly more covariates than observations" *Journal of Econometrics*; 189(1), pp. 1 - 23.

[17] Farrell, M.H., Liang, T. and Misra, S. (2018) "Deep Neural Networks for Estimation and Inference", *arXiv preprint arXiv:1809.09953*.

[18] Federal Reserve Bank of St. Louis, St. Louis Fed Financial Stress Index [STLFSI], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/STLFSI, March 22, 2020.

[19] Foteinis, S. (2018) "Bitcoin's alarming carbon footprint" *Nature*; 554(7691), pp. 169 - 169.

[20] Friedman, J.H. (1994) "An overview of predictive learning and function approximation". In *From statistics to neural networks*, pp. 1 - 61: Springer, Berlin, Heidelberg.

[21] Frost and Sullivan (2018) *Bitmain and Canaan IPO filings* Available at: Bloomberg Terminal.

[22] Gal, Y. and Ghahramani, Z. (2016) "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In *international conference on machine learning*; pp. 1050 - 1059.

[23] Garcia, D., Tessone, C.J., Mavrodiev, P. and Perony, N. (2014) "The digital traces of bubbles: feedback cycles between socio-economic signals in the Bitcoin economy" *Journal of the Royal Society Interface*; 11(99), p. 20140623.

[24] Global Carbon Atlas (2020) *Global Carbon Atlas*; Available at: `http://www.globalcarbonatlas.org/en/CO2-emissions`.

[25] Goodfellow, I.J., Bengio, Y. and Courville, A. (2017) "Deep learning" *MIT press.*

[26] Goodfellow, I.J., Bulatov, Y., Ibarz, J., Arnoud, S. and Shet, V. (2013) Multi-digit number recognition from street view imagery using deep convolutional neural networks *arXiv preprint arXiv:1312.6082.*

[27] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014) "Generative adversarial nets". In *Advances in neural information processing systems*; pp. 2672 - 2680.

[28] Goodfellow, I.J., Warde-Farley, D., Mirza, M., Courville, A. and Bengio, Y. (2013) "Maxout networks" *arXiv preprint arXiv:1302.4389.*

[29] Hayes, A. (2015) "A cost of production model for bitcoin" Available at SSRN 2580904.

[30] Hope, C., Anderson, J. and Wenman, P. (1993) "Policy analysis of the greenhouse effect: an application of the PAGE model" *Energy Policy*; 21(3), pp. 327 - 338.

[31] Hornik, K. (1991) "Approximation capabilities of multilayer feedforward networks" *Neural networks*; 4(2), pp. 251 - 257.

[32] Houy, N. (2019) "Rational mining limits Bitcoin emissions" *Nature Climate Change*; 9(9), pp.655 - 655.

[33] Interagency Working Groups on the Social Cost of Greenhouse Gases (2016) *Technical Support Document: Technical Update of the Social Cost of Carbon for Regulatory Impact Analysis Under Executive Order 12866*. Washington, DC: Interagency Working Group on the Social Cost of Carbon.

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

[35] Jang, H. and Lee, J., (2017) "An empirical study on modeling and prediction of bitcoin prices with bayesian neural networks based on blockchain information" *Ieee Access*; 6, pp. 5427 - 5437.

[36] Kristoufek, L., (2015) "What are the main drivers of the Bitcoin price? Evidence from wavelet coherence analysis". *PloS one*; 10(4), pp. 1 - 15.

[37] LeCun, Y., Bengio, Y. and Hinton, G. (2015) "Deep learning", *nature*; 521(7553), pp. 436 - 444.

[38] Liu, Y. and Tsyvinski, A. (2018) *Risks and returns of cryptocurrency (No. w24877)*; National Bureau of Economic Research.

[39] Masanet, E., Shehabi, A., Lei, N., Vranken, H., Koomey, J. and Malmodin, J. (2019) "Implausible projections overestimate near-term Bitcoin CO_2 emissions". *Nature Climate Change*; 9(9), pp. 653 - 654.

[40] McNally, S., Roche, J. and Caton, S., (2018) "Predicting the price of bitcoin using machine learning". In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*; pp. 339 - 343. IEEE.

[41] Montufar, G.F., Pascanu, R., Cho, K. and Bengio, Y. (2014) "On the number of linear regions of deep neural networks". In *Advances in neural information processing systems*; pp. 2924 - 2932.

[42] Mora, C., Rollins, R.L., Taladay, K., Kantar, M.B., Chock, M.K., Shimada, M. and Franklin, E.C. (2018) "Bitcoin emissions alone could push global warming above 2 C". *Nature Climate Change*; 8(11), pp. 931- 933.

[43] Murray, M. (2018) "Blockchain explained". *Reuters Graphics*; June, 15, p.2018.

[44] National Academies of Sciences, Engineering, and Medicine (2017) "Valuing climate damages: updating estimation of the social cost of carbon dioxide" *National Academies Press.*

[45] Pascanu, R., Montufar, G. and Bengio, Y. (2013) "On the number of response regions of deep feed forward networks with piece-wise linear activations" *arXiv preprint arXiv:1312.6098*

[46] Raghu, M., Poole, B., Kleinberg, J., Ganguli, S. and Dickstein, J.S. (2017) "On the expressive power of deep neural networks". In *Proceedings of the 34th International Conference on Machine Learning*; 70, pp. 2847 - 2854.

[47] Shapiro, S.S. and Wilk, M.B. (1965) "An analysis of variance test for normality (complete samples)" *Biometrika*; 52(3/4), pp. 591 - 611.

[48] Shodan.io (2020) *IoT-search engine*; Available at: `https://www.shodan.io/`.

[49] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014) "Dropout: a simple way to prevent neural networks from overfitting" *The journal of machine learning research*; 15(1), pp. 1929 - 1958.

[50] Stekhoven, D.J. (2013) "missForest: Nonparametric Missing Value Imputation using Random Forest", *R package version 1.4.0.*

[51] Stoll, C., Klaaßen, L. and Gallersdörfer, U. (2019) "The carbon footprint of bitcoin" *Joule*; 3(7), pp. 1647 - 1661.

[52] Wang, P., Deng, X., Zhou, H. and Yu, S. (2019) "Estimates of the social cost of carbon: A review based on meta-analysis" *Journal of cleaner production*; 209, pp. 1494 - 1507.

[53] Williams, K. D., Hewitt, A. J. & Bodas-Salcedo, A. (2020). J. Adv.Model. Earth Sys. 12, e2019MS001986 also reported as "Short-term tests validate long-term estimates of climate change". *Nature* 582, 185-186 (2020) doi: 10.1038/d41586-020-01484-5.