{TinyRCE}: Multi Purpose Forward Learning for Resource Restricted Devices

On-Device Learning

TinyRCE: Multi Purpose Forward Learning for Resource Restricted Devices

Danilo Pau[1*], Andrea Pisani[1], Fabrizio M. Aymone[1], and Gianluigi Ferrari[2**]

[1] *Department of Systems Research and Application, STMicroelectronics, Agrate Brianza, MB 20041 IT*
[2] *Department of Engineering and Architecture, University of Parma, Parma, PR 43124 IT*
[*] *Fellow, IEEE and AAIA*
[**] *Senior Member, IEEE*

Abstract—The challenge of deploying learning workloads of neural networks on ultra-low power tiny devices has recently attracted several machine learning researchers in the TinyML community. A typical on-device learning session processes real-time streams of data acquired by heterogeneous sensors. In such a context, this paper proposes TinyRCE, a forward-only learning approach based on a hyperspherical classifier, which can be deployed on microcontrollers and potentially integrated into the sensor package. TinyRCE is fed with compact features extracted by a convolutional neural network, which can be trained with BP or can be an extreme learning machine with randomly initialized weights. A forget mechanism has been introduced to discard useless neurons from the its hidden layer, since they can become redundant over the time. TinyRCE has been evaluated with a new interleaved learning and testing data protocol to mimic a forward on-tiny-device workload. It has been tested over the standard MLCommons Tiny datasets used for Keyword Spotting and Image Classification, and against the respective benchmarks. 95.25% average accuracy was achieved over the former classes (vs. 91.49%) and 87.17% over the latter classes (vs. 100%, caused by overfitting). In terms of complexity, TinyRCE requires 22× less MACC than SoftMax (with 36 epochs) on the former, while it requires 5× more MACC than SoftMax (with 500 epochs) for the latter. Classifier complexity and memory footprint are negligible w.r.t. the Feature Extractor, for training and inference workloads.

Index Terms—Extreme Learning Machines, Feature Extraction, Hyper-Spherical Classifier, Keyword Spotting, On-Tiny-Device Learning, TinyML.

I. INTRODUCTION

Supervised Artificial intelligence (AI) and Tiny Machine Learning (Tiny ML) nowadays are widely used approaches. [1] benchmarked multiple Deep Neural Networks (DNNs or NNs) reporting accuracy, memory usage, complexity, and latency. As the complexity of the DNNs increased, by addressing more challenging problems, the DNNs required greater model size, bigger memory and more computational capabilities. The deployment of complex DNNs on tiny devices and their use to process sensor-generated data streams can also be affected by a decrease in inference accuracy w.r.t. the training phase [2]. This is due to numerous causes. The consequence is that continuous updates and retraining of the DNNs are required over the time. The de-facto standard training method, used by many deep learning frameworks [1], is based on backpropagation (BP) and stochastic gradient descent (SGD). Retraining with such a complex algorithm requires powerful computational and storage assets. Unfortunately, when using tiny devices, Continuous Learning (CL) is very costly on them [3]. Moreover, BP is a type of sequential learning which can be heavily affected by catastrophic forgetting (CF) [4] compromising the CL objectives. Thus, most tiny devices push sensors data to the cloud, which is capable to run highly asset-demanding AI workloads. Although cloud-based solutions [5] feature more resources than the tiny devices, they still account for many disadvantages. Privacy, security of user data, latency, and power consumption are big concerns for the TinyML community.

Therefore, there is the opportunity to reconsider supervised learning procedures and extend the state of the art beyond BP by conceiving new on-tiny-device learning algorithms. An approach to on-tiny-device CL without using BP is proposed by this paper, which is organized as follows: Section II introduces the problem statement and requirements to be fulfilled by the solution; Section III, reviews the existing related works and reports their limitations; Section IV summarizes the datasets used to shape the case studies for experimenting the learning and testing workloads; Section V explains the proposed solution, highlights differences with respect to the previous works and describes how the On-Device Learning (ODL) field was approached; the experimental results are reported in Section VI; Section VII analyzes the complexity of the proposed solution with respect to its deployment on microcontrollers (MCUs), followed by the conclusions and future works in Section VIII.

Corresponding author: D.P. Pau (e-mail: danilo.pau@st.com).
Associate Editor: .
Digital Object Identifier 10.1109/LSENS.2017.0000000

II. PROBLEM STATEMENT AND REQUIREMENTS

The research question set by this paper is: can incremental learning of multiple categories happen, totally on-line without CF, without K-fold BP and still be deployable on tiny devices? Therefore, the requirements, to solve such a problem, were set as shown in Table 1. They shall be fulfilled by the proposed or existing (if any) solution.

TABLE 1. Requirements for any Tiny ODL solution.

No.	Requirements
1.	Real-time forward learning.
2.	No BP.
3.	Deployable on tiny MCU, optionally in the sensor fitting into the embedded memory.
4.	Capable of performing multi-class classification.
5.	Capable of interleaving learning with inference workloads.
6.	Requiring the minimum temporary storage of sensor data.
7.	Featuring limited latency to run inference workloads.

III. RELATED WORKS

A. On-device learning and Extreme Learning Machines

CL DNNs aim to learn different tasks without retraining from scratch nor forgetting previous knowledge. CF [4] is the most common issue affecting the performances of CL models. A related issue affecting Tiny ODL models is concept drift (CD) [6]. [7] reported a state-of-the-art review for both ODL and ML on tiny devices and summarized the importance of the challenges faced.

Extreme Learning Machines (ELMs) are DNNs with randomly initialized weights. Only the ELMs' final dense layer is optimized through linear methods. [8] introduced the concept of ELM on edge devices. ELM was introduced by [9] for regression and multiclass classification applications. [10] proposed a survey for on-line ELM.

B. Introduction to RCE and Hyperspherical Classifiers

Restricted Coulomb Energy (RCE) classifiers [11] are a branch of hyper-spherical classifiers [12]. They are composed of three layers: input, hidden, and output. Hidden neurons are dynamically initialized hyperspheres defined by center points that are samples of the feature space and a default radius. Each hidden neuron is fully connected to the neurons of the input layer, while it is connected to only one output neuron. New hidden neurons are instantiated during the learning phase if some input feature vectors do not fall inside existing hyperspheres, according to a pairwise metric, usually the Euclidean distance. Multiple hidden neurons could ambiguously fire at once. The radius of hidden neurons that misfire and lead to incorrect predictions is reduced so that the corresponding hyperspheres cease to contain the misclassified inputs. Should a hidden neuron feature an very small radius, it can be considered redundant and therefore be removed. Unfortunately, in a naive implementation, that would not happen. [13] was a DNN to segment hands appearing into images. The skin-colored pixels were classified and segmented using the RCE. A new approach to reduce the neuron's radius has been introduced to lower the processing. However, it did not face how to remove the redundant neurons. [14] proposed a RCE inductive classifier, which was more efficient than other similar approaches. All the above approaches were not meant for MCU deployment. They adopted the traditional offline training methods and did not perform ODL on MCU.

C. MLCommons Benchmark for Keyword Spotting and Image Classification

The MLPerf Tiny Benchmark [15] is a collection of TinyML benchmark models. They were created thanks to the collaborative effort of more than 50 players, spanning from leading companies to academic research groups. It focuses on 4 use cases: Keyword Spotting (KWS), Visual Wake Words (VWW), Image Classification (IC) and Anomaly Detection (AD). In particular, the KWS model is the Depth-wise Separable Convolutional Neural Network (DS-CNN), 92.2% accurate over the Speech Commands v2 dataset [16], within 36 epochs, with BP. The IC benchmark is an implementation of ResNetv1 [17], 86.5% accurate over CIFAR-10 [18] dataset within 500 epochs.

IV. DATASETS

KWS and IC datasets shaped the case studies considered by this work. Speech Commands v2 [16] is a widely used Speech Recognition dataset that represents the former use case. It is available under the Creative Commons BY license, and is part of the TensorFlow Datasets[1] library. It provides 105,829 audio recordings of single word utterances, recorded by volunteers with their own devices and shared online in a WAV format with a sampling rate of 16 kHz. They are divided into training (85,511 samples), validation (10,102) and testing (4,890) subsets. The data is divided in 12 classes: 10 are associated with relevant keywords. The two remaining ones are associated with silent recordings and non-relevant keywords labeled as 'Unknown'. CIFAR-10 [18] is a dataset of low-res (32px×32px) RGB images collected from the Internet. It consists of 60,000 images of vehicles and animals divided in 10 classes (6,000 images per class), encoded as NumPy matrices. The dataset is divided into 5 training batches and 1 testing batch, of 10,000 images each. One task of this work was to compare the MLCommons/Tiny benchmark neural networks [15]. Therefore, the same pre-processing pipeline was used. In order to mimic the incremental introduction of new classes over time, the matrices were rearranged by label and sequentially presented to the classifier, using 20% of the data of each class as training samples and 80% as testing samples.

V. PROPOSED ODL ALGORITHM

Tiny Restricted Coulomb Energy (TinyRCE) NN is a hyperspherical classifier variant proposed by this work. The raw data is input to the CNN Feature Extractor (FE). Convolution operations into CNNs extracts abstracted features from raw data tensors. Thus, a CNN topology stripped of its final SoftMax layer is considered to be the FE. It was trained offline, using BP. A random ELM-style initialization for the FE was performed, and it impacted negatively the classifier's accuracy since it was not able to spatially separate features of the different classes. The extracted features then fed the TinyRCE classifier. For the KWS use case, the MLCommons benchmark DS-CNN was adopted as FE after removing its classification layer. For IC, the MLCommons benchmark ResNetv1 was used.

TinyRCE was designed to address all the requirements in Table 1. Only the hidden neurons of TinyRCE were optimized by the ODL procedure. Data classes were presented sequentially to mimic a CL set-up. TinyRCE adjusted its hidden and output layers by processing the features extracted by the FE. CL and the dynamic addition of new neurons into TinyRCE could quickly saturate the fixed size of the MCU embedded memory if handled in an uncontrolled way. Therefore, a culling mechanism was introduced. It worked by assigning to each hidden neuron an unsigned 8 bits integer age value and a floating point 32 bits reliability value. At time t_0, said values were set to 0 for each instantiated hidden neuron. The age value was incremented by 1 every time a neuron fired. The reliability score s_j was updated accordingly to (1). Note that $\hat{y}_t$ represents a label prediction done by the model for sample t, why y_t represents the true label of the same sample.

$$s_j^{\text{new}} = \begin{cases} s_j^{\text{old}} + \frac{R_j - dist(h_j, x_t)}{R_j} & \text{if } \hat{y}_t = y_t \\ s_j^{\text{old}} - \frac{dist(h_j, x_t)}{R_j} & \text{if } \hat{y}_t \neq y_t. \end{cases} \quad (1)$$

[1] https://www.tensorflow.org/datasets?hl=en

In (1), the increment and decrement were different: the increment's numerator favored the correct predictions that involved inputs located close to the hypersphere's center instead of those involving inputs located near its border. The decrement formula did the opposite. When the feature fell inside the influence region of a hidden neuron j that matched the ground truth annotation, s_j was increased; otherwise, decremented. This value enabled pruning of redundant neurons, which was triggered if a threshold was exceeded. The threshold dependeded on the maximum number of hidden neurons MCU could fit into its memory. When pruning the redundant neurons, both the age score and the reliability score were used. Neurons that were deemed redundant were those with the least number of firings and the lowest reliability.

It is well known that hyper-spherical classifiers produce feature spaces that overlap with each other's classes. This implies to activate simultaneously several hidden neurons associated with different classes. Furthermore, at the end of the CL workflow the hyperspheres associated with the last class dominated the feature space with respect to the other classes' hyperspheres, since they were exposed to the least corrections over time. To avoid this issues, the dynamic instantiation of the hyperspheres was modified w.r.t. the classic RCE algorithm. Also, a method to make the model more robust to noise in the data was introduced.

The dynamic instantiation of new hyperspheres was modified as follows. If an input vector was positioned in a location of the feature space not yet covered by any hypersphere, the radius of the new hypersphere to instantiate (denoted by R^*) was determined dynamically instead of being set by a user-defined standard value $\overline{R}$: since the overlapping of hyperspheres with different labels led to incorrect predictions, the model checked if the newly instantiated hypersphere overlapped with other ones that were differently labeled. If this was the case, then the radius of the new hypersphere was set to the biggest possible value that caused no such an overlap. Otherwise, the standard value was kept. This behavior is summarized in (2), which defines R^*:

$$R^* = \begin{cases} \overline{R} & \forall j, dist(h_j, x_t) > \overline{R} \\ \min_j(dist(h_j, x_t)) - R_j & \text{otherwise.} \end{cases} \quad (2)$$

Finally, to achieve better noise resilience, the implemented scoring system also tied in to the neuron resizing policy. A hypersphere was not supposed to be resized if its misfiring was caused by an data outlier. An outlier was defined as a wrong prediction fired by a hypersphere whose reliability score was in the first quartile among all existing hyperspheres. Assuming the input was recognized as data outlier, the misfiring of a hypersphere did not cause its resizing, only the decrease of its reliability score.

In the case of IC, Bayesian Optimization [19] was applied to find set of hyperparameters which maximised the testing accuracy. The variables to optimize were; the standard radius $\overline{R}$; the pruning threshold; the train-test split used to divide the data; a growth law that determined how $\overline{R}$ would change from one class to the next during CL, chosen among a constant function, a linear and an exponential growth; a weight w which was part of the target metric to optimize. This target metric is described in (3), where acc_x stands for accuracy, and N represents the number of classes.

$$\text{score} = \text{acc}_{\text{test set}} - w \min\{0, (0.9 - \frac{1}{N}\sum_{k=1}^{N} \text{acc}_{\text{class } k})\} \quad (3)$$

VI. EXPERIMENTS

For both KWS and IC, the FE produced 64-dimension features. Randomly initialized versions performed poorly. On the contrary, the off-the-shelf trained topologies from MLCommons python scripts achieved well separated feature spaces.

TinyRCE proved to outperform the SoftMax over the ODL procedure. For KWS, TinyRCE was 95.25% accurate (average on tests) w.r.t. 91.49% of the SoftMax classifier trained with 36 epochs. SoftMax performed very poorly (average accuracy 14.22%) if trained using only 1 epoch w.r.t. TinyRCE. Over the standard test set, TinyRCE performed comparably w.r.t. the benchmark network trained with BP. For IC, SoftMax performed even worse: if trained with 500 epochs, it totally overfit the training data, thus achieving 100% training accuracy while performing as a random predictor if deployed on the test set. On the contrary, TinyRCE's average accuracy over the single classes (87.17%) was closer to its accuracy on the scrambled test set (78.5%). Histograms displaying the results for both use cases are shown in fig. 1.

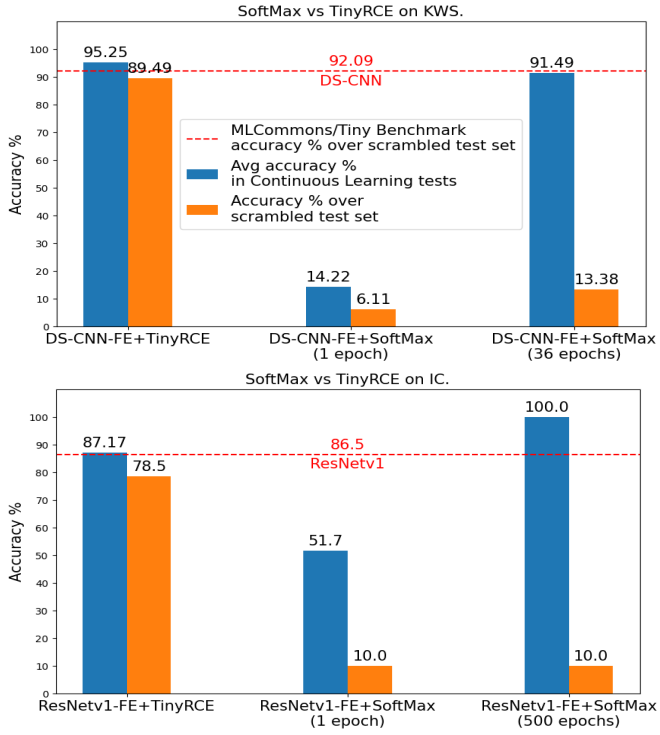

Fig. 1. Experimental results on KWS and IC.

VII. COMPLEXITY ANALYSIS FOR TINY DEVICES

Complexity was measured in Multiply and ACCumulate (MACC) operations. The equations to estimate the complexity for TinyRCE's inference and learning phases are described in (4) and (5), respectively:

$$MACC_{\text{inference}} = h \times [(n \times 5) + 10] \quad (4)$$

h is the number of hidden neurons, and n is the dimensionality of the feature vectors;

$$MACC_{\text{learning}} = \{h \times [(n \times 5) + 10]\} \times (N \times E). \quad (5)$$

N is the number of training data, and E is the number of epochs in the worst case.

TABLE 2. Complexity Profiling and Memory Footprint Analysis for the KWS use case.

Metrics		DS-CNN-FE_BP	SoftMax (1 epoch)	SoftMax (36 epochs)	TinyRCE
MACC (M)	Training	2.429E+5	**3.494**	125.8	5.643
	Inference	2.664	**9.480E-4**	**9.480E-4**	3.960E-3
RAM (KiB)	Training	866.3	**64.0**	**64.0**	**64.0**
	Inference	**64.0**	**64.0**	**64.0**	**64.0**
FLASH (KiB)	Inference	**3.120**	**3.120**	**3.120**	3.216
Inference time (ms)	NUCLEO-H743ZI2	-	**38.05**	**38.05**	38.10
	B-U585I-IOT02A	-	**162.96**	**162.96**	163.14

TABLE 3. Complexity Profiling and Memory Footprint Analysis for the IC use case.

Metrics		ResNet-FE	SoftMax (1 epoch)	SoftMax (500 epochs)	TinyRCE
MACC (M)	Training	92.75E+6	**2.100**	1,050.0	5,266.8
	Inference	12.5	**7.90E-4**	**7.90E-4**	9.24E-2
RAM (KiB)	Training	455.12	**196.60**	**196.60**	**196.60**
	Inference	**196.60**	**196.60**	**196.60**	**196.60**
FLASH (KiB)	Inference	**2.600**	**2.600**	**2.600**	2.680
Inference time (ms)	NUCLEO-H743ZI2	-	**123.61**	**123.61**	124.51
	B-U585I-IOT02A	-	**527.03**	**527.03**	530.86

The complexity analysis was performed over two MCUs: NUCLEO-H743ZI2 and B-U585I-IOT02A. The former runs at 480 MHz, with 1024 KiB of embedded RAM and 2048 KiB of embedded FLASH memory, while the latter runs at 160 MHz, with 786 KiB RAM and 2048 KiB FLASH embedded. About FEs, the analysis consisted in evaluating the MACCs needed by each layer of the NN for training and inference operations (e.g., forward pass, backward pass). About the inference, RAM stored the activation buffers, while Flash stored the trainable parameters. During the training phase the parameters shall be continuously updated, so they are stored in RAM. Hence, peak RAM utilization for all the topologies was limited by the memory bottleneck imposed by the forward pass through the frozen FEs. Total inference times were evaluated using the STM32Cube.AI Developer Cloud[2] service. For both KWS and IC, the training of the FE with BP dominated complexity and memory footprint. The difference between the two classifiers was marginal due to the FE they had in common. About KWS, peak RAM was 866.3 KiB (estimated) for training and 64 KiB for inference. The Flash memory required was 3.12 KiB for the DS-CNN and 3.216 KiB for DS-CNN-FE_BP+TinyRCE. About IC, peak RAM was 455.12 KiB for training and 196.60 KiB for inference. The Flash memory footprint was 2.68 KiB (estimated).

VIII. CONCLUSIONS AND FUTURE WORKS

This paper introduced TinyRCE, which performed forward-only incremental classification on MCU. No BP was required, except for the FE training. The FEs were off-the-shelf solutions from MLCommons Tiny working group. A specific protocol for both training and testing procedure was introduced, mimicking the streaming acquisition of raw sensor data. TinyRCE proved to be more accurate than the MLCommons benchmark models trained for multiple epochs on the ODL training-testing workload. Complexity and memory footprint were dominated by the offline-trained FEs. Future developments will be focused on devising an ELM FE, in order to eliminate the BP procedure applied to the FE. Further RAM and FLASH reduction using low bit-depth features could ease the deployability into the sensor itself. Also, new use cases regarding different data types (e.g. time of flight) can be investigated.

[2]https://stm32ai-cs.st.com

REFERENCES

[1] S. Bianco, R. Cadène, L. Celona, and P. Napoletano, "Benchmark Analysis of Representative Deep Neural Network Architectures," *IEEE Access*, vol. 6, pp. 64270–64277, Oct. 2018.

[2] D. Vela, A. Sharp, R. Zhang, T. Nguyen, A. Hoang, and O. S. Pianykh, "Temporal quality degradation in AI models," *Scientific Reports*, vol. 12, p. 11654, July 2022.

[3] M. Delange, M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, Ales Leonardis, A. Leonardis, Ales Leonardis, Greg Slabaugh, G. Slabaugh, Gregory G. Slabaugh, G. Slabaugh, G. G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks.," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, Sept. 2019.

[4] Prakhar Kaushik, Alex Gain, Adam Kortylewski, and A. Yuille, "Understanding Catastrophic Forgetting and Remembering in Continual Learning with Optimal Relevance Mapping," *ArXiv*, 2021.

[5] F. Samie, L. Bauer, and J. Henkel, "From Cloud Down to Things: An Overview of Machine Learning in Internet of Things," *IEEE Internet of Things Journal*, vol. 6, pp. 4921–4934, Jan. 2019.

[6] Yamini Kadwe and V. Suryawanshi, "A Review on Concept Drift," 2015.

[7] Danilo Pau and Prem Kumar Ambrose, "A quantitative review of automated neural search and on-device learning for tiny devices," Jan. 2022.

[8] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, vol. 2, pp. 985–990, July 2004.

[9] G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Rui Zhang, and R. Zhang, "Extreme Learning Machine for Regression and Multiclass Classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, pp. 513–529, Apr. 2012.

[10] S. Zhang, W. Tan, and Y. Li, "A survey of online sequential extreme learning machine," in *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 45–50, IEEE, 2018.

[11] M. Hudak, "RCE networks: An experimental investigation," in *IJCNN-91-Seattle International Joint Conference on Neural Networks*, vol. i, (Seattle, WA, USA), pp. 849–854, IEEE, 1991.

[12] M. H. Hassoun *et al.*, *Fundamentals of Artificial Neural Networks*. MIT press, 1995.

[13] C. Sui, N. M. Kwok, and T. Ren, "A Restricted Coulomb Energy (RCE) Neural Network System for Hand Image Segmentation," in *2011 Canadian Conference on Computer and Robot Vision*, (St Johns, Newfoundland, Canada), pp. 270–277, IEEE, May 2011.

[14] N. Fanizzi, C. d'Amato, and F. Esposito, "ReduCE: A Reduced Coulomb Energy Network Method for Approximate Classification," *Extended Semantic Web Conference*, pp. 323–337, May 2009.

[15] Colby R. Banbury, V. Reddi, P. Torelli, J. Holleman, Nat Jeffries, C. Király, Pietro Montino, David Kanter, Sebastian Ahmed, Danilo Pau, Urmish Thakker, Antonio Torrini, P. Warden, Jay Cordaro, G. D. Guglielmo, Javier Mauricio Duarte, Stephen Gibellini, Videet Parekh, Honson Tran, Nhan Tran, Niu Wenxu, and Xu Xuesong, "MLPerf Tiny Benchmark," *NeurIPS Datasets and Benchmarks*, 2021.

[16] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," 2018.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv: Computer Vision and Pattern Recognition*, Dec. 2015.

[18] "CIFAR-10 and CIFAR-100 datasets." http://www.cs.toronto.edu/~kriz/cifar.html.

[19] R. Garnett, *Bayesian Optimization*. Cambridge, United Kingdom ; New York, NY: Cambridge University Press, 2023.