StarCoder 2 and The Stack v2: The Next GenerationAnton Lozhkov1Raymond Li2Loubna Ben Allal1Federico Cassano4Joel Lamy-Poirier2Nouamane Tazi1Ao Tang3Dmytro Pykhtar3Jiawei Liu7Yuxiang Wei7Tianyang Liu25Max Tian2Denis Kocetkov2Arthur Zucker1Younes Belkada1Zijian Wang5Qian Liu12Dmitry Abulkhanov5Indraneil Paul5Zhuang Li14Wen-Ding Li26Megan Risdal24Jia Li5Jian Zhu16Terry Yue Zhuo14,15Evgenii Zheltonozhskii13Nii Osae Osae Dade28WenhaoYu20Lucas Krau5Naman Jain27Yixuan Su30Xuanli He23Manan Dey31EduardoAbati5Yekun Chai5Niklas Muennigho29Xiangru Tang5Muhtasham Oblokulov18Christopher Akiki9,10Marc Marone8Chenghao Mou5Mayank Mishra19Alex Gu17Binyuan Hui5Tri Dao21Armel Zebaze1Olivier Dehaene1Nicolas Patry1Canwen Xu25Julian McAuley25Torsten Scholak2Sebastien Paquet2Jennifer Robinson6Carolyn JaneAnderson22Nicolas Chapados2Mostofa Patwary3Nima Tajbakhsh3Yacine Jernite1Carlos Muoz Ferrandis1Lingming Zhang7Sean Hughes6Thomas Wolf1Arjun Guha4,11Leandro von Werra1,Harm de Vries2,1Hugging Face2ServiceNow Research3Nvidia4Northeastern University5Independent6ServiceNow7University of Illinois Urbana-Champaign8Johns Hopkins University9Leipzig University10ScaDS.AI11Roblox12Sea AI Lab13Technion Israel Institute of Technology14Monash University15CSIROsData6116University of British Columbia17MIT18Technical University of Munich19IBM Research20University of Notre Dame21Princeton University22Wellesley College23University College London24Kaggle25UC San Diego26Cornell University27UC Berkeley28Mazzuma29Contextual AI30Cohere31SalesforceCorresponding authors ( ) can be contacted at contact@bigcode-project.orgAbstractThe BigCode project,1an open-scientic collaboration focused on the responsible developmentof Large Language Models for Code (Code LLMs), introduces StarCoder2. In partnershipwith Software Heritage (SWH),2we build The Stack v2 on top of the digital commons of theirsource code archive. Alongside the SWH repositories spanning 619 programming languages,we carefully select other high-quality data sources, such as GitHub pull requests, Kagglenotebooks, and code documentation. This results in a training set that is 4larger than therst StarCoder dataset. We train StarCoder2 models with 3B, 7B, and 15B parameters on3.3 to 4.3 trillion tokens and thoroughly evaluate them on a comprehensive set of Code LLMbenchmarks.We nd that our small model, StarCoder2-3B, outperforms other Code LLMs of similar sizeon most benchmarks, and also outperforms StarCoderBase-15B. Our large model, StarCoder2-15B, signicantly outperforms other models of comparable size. In addition, it matches oroutperforms CodeLlama-34B, a model more than twice its size. Although DeepSeekCoder-33B is the best-performing model at code completion for high-resource languages, we ndthat StarCoder2-15B outperforms it on math and code reasoning benchmarks, as well asseveral low-resource languages. We make the model weights available under an OpenRAILlicense and ensure full transparency regarding the training data by releasing the SoftWareHeritage persistent IDentiers (SWHIDs) of the source code data.1https://www .bigcode-project .org2https://www .softwareheritage .org/1

1 IntroductionLarge Language Models for Code (Code LLMs; Chen et al. ,2021;Nijkamp et al. ,2023;Rozire et al. ,2023;Guo et al. ,2024) have rapidly emerged as powerful assistants for writing and editing code. As of January 30,2024, GitHub CoPilot has garnered over 1.3 million paying subscribers, with over 50,000 organisations optingfor the enterprise version ( MSFT Q2 Earning Call ,2024), estimated to increase developer productivity by upto 56% as well as developer satisfaction ( Peng et al. ,2023;Ziegler et al. ,2024). ServiceNow recently disclosedthat their text-to-code solution, built from ne-tuning StarCoderBase models ( Li et al. ,2023), results ina 52% increase in developer productivity ( Yahoo Finance ,2024). Despite the initial focus on generatingcode snippets from natural language instructions or other code snippets, Code LLMs exhibit the potentialto enhance all phases of the software development cycle ( Hou et al. ,2023;Fan et al. ,2023;Wang et al. ,2024;Zhuo et al. ,2023b ). This includes speeding up the implementation of new projects, improving qualityassurance for developed software, helping detect and x bugs, simplifying maintenance tasks, and easingmigration to newer software.The development process of LLMs can exhibit dierent levels of openness ( Solaiman ,2023;Ding et al. ,2022;Akiki et al. ,2022). Proprietary models like OpenAIs GPT-4 ( OpenAI et al. ,2023) and GooglesGemini ( Gemini Team et al. ,2023) provide access to the model through a paid API but do not disclosedevelopment details. On the other hand, open-weight models like Code LLaMa ( Rozire et al. ,2023),Mistral ( Jiang et al. ,2023), and DeepSeekCoder ( Guo et al. ,2024) have released the model weights. Thisenables the open-source community to run these models locally, inspect the model representations, and ne-tune them on their tasks. However, the model developers have not disclosed their training data. Consequently,content creators do not know if their data was used for training, social scientists cannot scrutinize the datasetfor bias and toxicity, and LLM developers lack information as to what extent the training set is contaminatedwith test benchmarks. More broadly, this practice hinders scientic progress as other research teams cannotreadily reuse each others training data. Other LLM development projects, like Allen AIs OLMo ( Groeneveldet al. ,2024), Eleuther AIs Pythia ( Biderman et al. ,2023), and BigSciences BLOOM ( BigScience Workshop ,2022;Scao et al. ,2022a ), have adopted a fully open development approach by releasing training data, trainingframeworks, and evaluation suites.The BigCode project was established in September 2022 as an open scientic collaboration focused on theopen and responsible development of Code LLMs. BigCode is stewarded by ServiceNow and Hugging Face inthe spirit of open governance ( BigCode collaboration et al. ,2023) and has brought together more than 1,100members from diverse academic institutes and industry labs. The community previously released The Stackv1 (Kocetkov et al. ,2023), a 6.4 TB dataset of permissively licensed source code in 384 programming languages.The Stack v1 includes a governance tool called Am I in The Stack, designed for developers to verify if theirsource code is included in the dataset. It also provides an opt-out process for those who prefer to exclude theircode from the dataset. In December 2022, the BigCode community released SantaCoder ( Ben Allal et al. ,2023), a strong-performing 1.1B parameter model trained on Java, JavaScript, and Python code from TheStack v1. Building upon this success, the community further scaled up its eort and released StarCoder onMay 4th, 2023 ( Li et al. ,2023). At its release, the 15B parameter StarCoder model was the best open-accessLLM for code.This technical report describes the development process of The Stack v2 and StarCoder2. The Stack v2 buildsupon the foundation of Software Heritages vast source code archive, which spans over 600 programminglanguages. In addition to code repositories, we curate other high-quality open data sources, including Githubissues, pull requests, Kaggle and Jupyter notebooks, code documentation, and other natural language datasetsrelated to math, coding, and reasoning. To prepare the data for training, we perform deduplication, createlters to eliminate low-quality code, redact Personally Identiable Information (PII), remove malicious code,and handle opt-outs from developers who requested to have their code removed from the dataset. With thisnew training set of 900B+ unique tokens, 4larger than the rst StarCoder dataset, we develop the nextgeneration of StarCoder models. We train Code LLMs with 3B, 7B, and 15B parameters using a two-stagetraining process ( Rozire et al. ,2023;Guo et al. ,2024). We start base model training with a 4k contextwindow and subsequently ne-tune the model with a 16k context window. We ensure that the trainingprocess does not exceed more than 5 epochs over the dataset ( Muennigho et al. ,2023). However, we push2

the number of training tokens far beyond the compute-optimal number suggested by Chinchilla (Harms law;de Vries ,2023) and train relatively small models within the range of 3.3 to 4.3 trillion tokens. We thoroughlyassess and compare the performance of these models on a suite of code LLM benchmarks ( Cassano et al. ,2023b ;Austin et al. ,2021;Chen et al. ,2021;Liu et al. ,2023a ;Lai et al. ,2023;Muennigho et al. ,2024a ;Cassano et al. ,2024;Liu et al. ,2023b ;Ding et al. ,2023;Gu et al. ,2024;Cobbe et al. ,2021;Pearce et al. ,2022;Dhamala et al. ,2021;Nozza et al. ,2021;Gehman et al. ,2020), nding that:The StarCoder2-3B model outperforms other Code LLMs of similar size (StableCode-3B andDeepSeekCoder-1.3B) on most benchmarks. Moreover, it matches or surpasses the performance ofStarCoderBase-15B.The StarCoder2-15B model signicantly outperforms other models of comparable size (CodeLlama-13B), and matches or outperforms CodeLlama-34B. DeepSeekCoder-33B is the best model atcode completion benchmarks for high-resource languages. However, StarCoder2-15B matches oroutperforms DeepSeekCoder-33B on low-resource programming languages (e.g., D, Julia, Lua,and Perl). Moreover, when we consider benchmarks that require models to reason about codeexecution ( Gu et al. ,2024) or mathematics ( Cobbe et al. ,2021), we nd that StarCoder2-15Boutperforms DeepSeekCoder-33B.The StarCoder2-7B model outperforms CodeLlama-7B but is behind DeepSeekCoder-6.7B. It is notclear to this reports authors why StarCoder2-7B does not perform as well as StarCoder2-3B andStarCoder2-15B for their size.2 Data SourcesIn this section, we elaborate on the process of obtaining training data, encompassing not just the datasourced from Software Heritage ( 2.1) but also GitHub issues ( 2.2), pull requests ( 2.3), Jupyter and Kagglenotebooks ( 2.4), documentation ( 2.5), intermediate representations ( 2.6), small math and coding datasets(2.7), and other natural language datasets ( 2.8).2.1 Source CodeSoftware Heritage We build the Stack v2 on top of the Software Heritage (SH) archive ( Abramatic et al. ,2018), maintained by the non-prot organization of the same name. The mission of Software Heritage is tocollect and preserve all knowledge taking the form of source code. We work with the SH graph dataset ( Pietriet al. ,2020), a fully deduplicated Merkle DAG ( Merkle ,1987) representation of the full archive. The SHgraph dataset links together le identiers, source code directories, and git commits, up to the entire statesof repositories, as observed during periodic crawls by Software Heritage.Extracting repositories We leverage the 2023-09-06 version of the SH graph dataset as the primarysource. We start by extracting the most recently crawled versions of all GitHub repositories and lteringthem to retain only the main branch. The branch is considered main if the repository metadata in GHArchivelists it as the default branch or if its name is main ormaster . We only extract the latest revision (commit)from the main branch and deduplicate the repositories based on the unique hashes of their contents (columndirectory\_id of the SH dataset). The repositories directory structure is reconstructed by recursivelyjoining the directory\_entry table of the dataset to itself using the directory\_id and target columns andconcatenating the directory and le names (column name) into full paths. We only traverse the directory treeup to level 64. The individual le contents are downloaded from the SH content S3 bucket if the compressedle size is less than 10MB.License detection We extract repository-level license information from GHArchive ( Github Archive ,2024)for all repositories with matching names in the SWH dataset. When the repo-level license is not available,i.e., for 96.93% of repositories, we use the ScanCode Toolkit ( ScanCode ,2024) to detect le-level licenses asfollows:3

Is the GitHublicense empty?Is the GitHub li-cense permissive?non-permissive permissiveDid ScanCodedetect licenses?no licenseAre all detected li-censes permissive?permissive non-permissiveyesnonoyesyesnoyesnoFigure 1: File-level license assignment logic.Find all les that could contain a license using a regular expression in Appendix A.3. This allows usto gather les that either explicitly contain a license (e.g., LICENSE ,MIT.txt ,Apache2.0 ) or containa reference to the license (e.g., README.md ,GUIDELINES );Apply ScanCodes license detection to the matching les and gather the SPDX3IDs of the detectedlicenses;Propagate the detected licenses to all les that have the same base path within the repository as thelicense le.Once the le-level license information is gathered, we decide whether the le is permissively licensed,non-permissively licensed, or unlicensed, following the algorithm described in Figure 1.The licenses we consider permissive are listed in Appendix A.4. This list was compiled from the licensesapproved by the Blue Oak Council ( Blue Oak Council ,2024), as well as licenses categorized as Permissiveor Public Domain by ScanCode ( ScanCode License Categories ,2024).Data licenses We consider three types of les: permissively licensed, non-permissively licensed (e.g.,copyleft), and unlicensed les. The main dierence between the Stack v2 and the Stack v1 is that we includeboth permissively licensed and unlicensed les. We exclude commercial licenses since their creators donot intend their code to be used for commercial purposes. We also exclude copyleft-licensed code due touncertainty regarding the communitys stance on using such data for LLM training and its relatively lowvolume.Language detection While the Stack v1 ( Kocetkov et al. ,2023) detects programming languages by theirle extension, we instead rely on a language classier. Specically, we use go-enry based on GitHubs librarylinguist (go-enry ,2024) to detect the programming language for each le. We detect 658 unique languagesinTheStackV2-dedup , some of which get removed at the data inspection stage (see next paragraph).3System Package Data Exchange, https://spdx .dev.4