DeepSeek-Coder-V2：打破代码智能领域闭源模型的壁垒

Qihao Zhu*, Daya Guo*, Zhihong Shao*, Dejian Yang*, Peiyi Wang, Runxin Xu, Y. Wu
Yukun Li, Huazuo Gao, Shirong Ma, Wangding Zeng, Xiao Bi, Zihui Gu, Hanwei Xu, Damai Dai
Kai Dong, Liyue Zhang, Yishi Piao, Zhibin Gou, Zhenda Xie, Zhewen Hao, Bingxuan Wang
Junxiao Song, Deli Chen, Xin Xie, Kang Guan, Yuxiang You, Aixin Liu, Qiushi Du, Wenjun Gao
Xuan Lu, Qinyu Chen, Yaohui Wang, Chengqi Deng, Jiashi Li, Chenggang Zhao
Chong Ruan, Fuli Luo, Wenfeng Liang

DeepSeek-AI

https://github.com/deepseek-ai/DeepSeek-Coder-V2

Abstract

我们提出了 DeepSeek-Coder-V2，这是一个开源的混合专家（MoE）代码语言模型，在代码专项任务上达到了与 GPT4-Turbo 相当的性能。具体而言，DeepSeek-Coder-V2 基于 DeepSeek-V2 的中间检查点，使用额外的 6 万亿 token 进行了进一步预训练。通过这种持续预训练，DeepSeek-Coder-V2 显著增强了 DeepSeek-V2 的代码生成与数学推理能力，同时在通用语言任务上保持了相当的性能。与 DeepSeek-Coder-33B 相比，DeepSeek-Coder-V2 在代码相关任务的各个方面以及推理和通用能力上均展现出显著进步。此外，DeepSeek-Coder-V2 将支持的编程语言从 86 种扩展至 338 种，并将上下文长度从 16K 扩展至 128K。在标准基准测试中，DeepSeek-Coder-V2 在代码和数学基准测试上的表现优于 GPT4-Turbo、Claude 3 Opus 和 Gemini 1.5 Pro 等闭源模型。

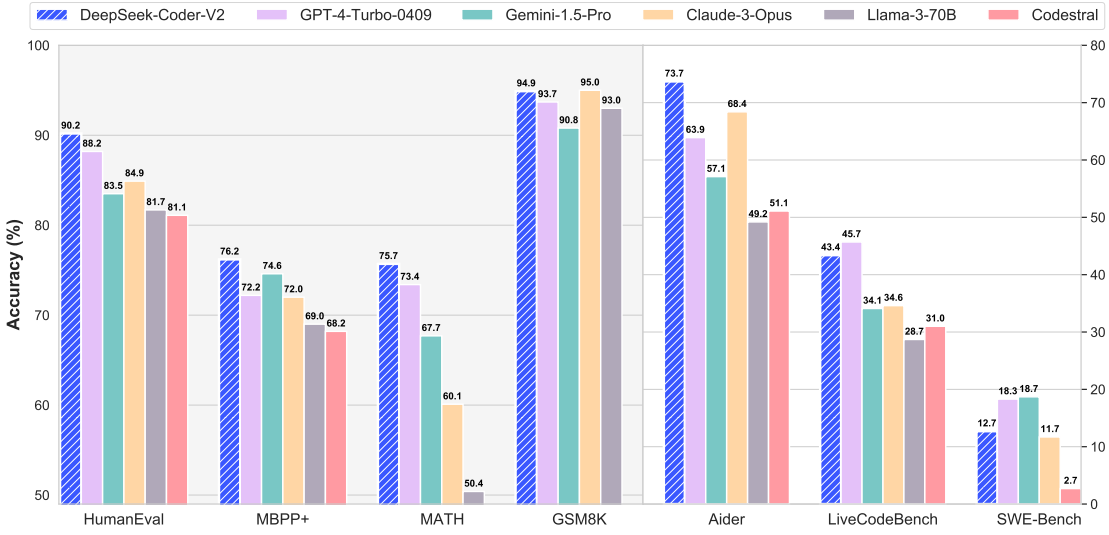

图 1 | DeepSeek-Coder-V2 在数学和代码基准测试上的性能表现。

* 核心贡献者

1. 引言

开源社区通过开发 StarCoder (Li et al., 2023b; Lozhkov et al., 2024)、CodeLlama (Roziere et al., 2023)、DeepSeek-Coder (Guo et al., 2024) 和 Codestral (MistralAI, 2024) 等开源代码模型，在推动代码智能发展方面取得了显著进展。这些模型的性能已稳步接近闭源同类模型，为代码智能的进步做出了贡献。然而，与 GPT4-Turbo (OpenAI, 2023)、Claude 3 Opus (Anthropic, 2024) 和 Gemini 1.5 Pro (Reid et al., 2024) 等最先进的闭源模型相比，仍存在明显的差距。为了弥合这一差距并进一步推动开源代码模型的发展，我们推出了 DeepSeek-Coder-V2 系列模型。这些模型以 DeepSeek-V2 (DeepSeek-AI, 2024) 为基础，并使用包含 6 万亿 token 的额外语料库进行了进一步预训练。

在预训练阶段，DeepSeek-Coder-V2 的数据集由 60% 的源代码、10% 的数学语料和 30% 的自然语言语料组成。源代码部分包含 1,1700 亿个来自 GitHub 和 CommonCrawl 的代码相关 token，采用与 DeepSeekMath (Shao et al., 2024) 相同的处理流水线。与用于训练 DeepSeek-Coder 的代码语料相比，该语料库支持的编程语言从 86 种扩展至 338 种。为了验证新代码语料的有效性，我们使用 1B 参数模型进行了消融实验，并观察到在 HumanEval（从 30.5% 提升至 37.2%）和 MBPP（从 44.6% 提升至 54.0%）基准测试上的准确率分别提升了 6.7% 和 9.4%（Austin et al., 2021a; Chen et al., 2021）。对于数学语料，我们使用相同的流水线从 CommonCrawl 收集了 2210 亿个数学相关 token，其规模约为 1200 亿 DeepSeekMath 语料库 (Shao et al., 2024) 的两倍；而对于自然语言语料，我们直接从 DeepSeek-V2 的训练语料库中采样。总体而言，DeepSeek-Coder-V2 共接受了 10.2 万亿个训练 token 的曝光，其中 4.2 万亿 token 源自 DeepSeek V2 数据集，其余 6 万亿 token 来自 DeepSeek-Coder-V2 数据集。

为了适应更长的代码输入并提升模型在各种编程场景中的适用性，我们将上下文长度从 16K 扩展至 128K token，使模型能够处理更复杂、更庞大的编码任务。在该多源语料库上对 DeepSeek-V2 进行持续预训练后，我们发现 DeepSeek-Coder-V2 在保持相当通用语言性能的同时，显著增强了模型在代码生成和数学推理方面的能力。

在对齐阶段，我们首先构建了一个指令训练数据集，其中包含了来自 DeepSeek-Coder (Guo et al., 2024) 和 DeepSeek-Math (Shao et al., 2024) 的代码与数学数据，以及来自 DeepSeek-V2 (DeepSeek-AI, 2024) 的通用指令数据。该数据集用于微调基础模型。随后，在强化学习阶段，我们采用组相对策略优化（GRPO）算法，使其行为与人类偏好保持一致。偏好数据通过编译器反馈和测试用例在代码领域收集，并开发了一个奖励模型来指导策略模型的训练。该方法确保了模型在代码任务中的响应在正确性和人类偏好方面得到优化。为了使模型在对齐后支持代码补全功能，我们在微调 16B 参数基础模型时也采用了中间填充（Fill-In-Middle）方法 (Guo et al., 2024)。

1.1. 主要贡献

综上所述，我们的主要贡献如下：

- 我们基于 DeepSeekMoE 框架推出了具有 16B 和 236B 参数的 DeepSeek-Coder-V2 模型，其激活参数仅为 2.4B 和 21B，能够高效支持多样化的计算与应用需求。此外，DeepSeek-Coder-V2 支持 338 种编程语言，最大上下文长度达 128K token。
- 我们首次尝试开发开源千亿参数代码模型，以推动代码智能领域的发展。实验结果表明，DeepSeek-Coder-V2 236B 在代码和数学任务上的表现均优于 GPT4-Turbo、Claude 3 Opus 和 Gemini 1.5 Pro 等最先进的闭源模型。
- DeepSeek-Coder-V2 模型在宽松许可协议下公开发布，允许用于研究和无限制的商业用途。

1.2. 评估与指标总结

- **代码**：在代码生成基准测试评估中，DeepSeek-Coder-V2 展现出对所有开源模型的显著优势，同时性能与 GPT4-Turbo、Claude 3 Opus 和 Gemini 1.5 Pro 等领先闭源模型持平。值得注意的是，我们在 HumanEval (Chen et al., 2021) 上取得了 **90.2**% 的得分，在 MBPP (Austin et al., 2021a) 上取得了 **76.2**% 的得分（使用 EvalPlus 评估流水线创下新的最先进结果），并在 LiveCodeBench (Jain et al., 2024)（2023 年 12 月至 2024 年 6 月的问题）上取得了 **43.4**% 的得分。此外，DeepSeek-Coder-V2 是首个在 SWEBench (Jimenez et al., 2023) 上得分超过 10% 的开源模型。
- **数学**：DeepSeek-Coder-V2 展现出强大的数学推理能力，在 GSM8K (Cobbe et al., 2021) 等基础基准测试以及 MATH (Hendrycks et al., 2021)、AIME (MAA, 2024) 和 Math Odyssey (Netmind.AI, 2024) 等高级竞赛级基准测试上，均能与 GPT-4o、Gemini 1.5 Pro 和 Claude 3 Opus 等顶级闭源模型相媲美。值得注意的是，DeepSeek-Coder-V2 在 MATH 基准测试上达到了 **75.7**% 的准确率，几乎与 GPT-4o 实现的 **76.6**% 的最先进准确率持平。此外，在 AIME 2024 竞赛中，其性能超越了这些闭源模型。
- **自然语言**：DeepSeek-Coder-V2 保持了与 DeepSeek-V2 相当的综合语言性能。例如，在使用 OpenAI simple-eval 评估流水线时，DeepSeek-Coder-V2 在 MMLU 上取得了 79.2% 的得分。在以 GPT-4 作为评判者的主观评估中，DeepSeek-Coder-V2 在 arena-hard (Li et al., 2024) 上取得了 **65.0** 分，在 MT-bench (Zheng et al., 2023) 上取得了 **8.77** 分，在 alignbench (Liu et al., 2023c) 上取得了 **7.84** 分。这些分数显著优于其他代码专用模型，甚至可与通用开源模型相媲美。

2. 数据收集

DeepSeek-Coder-V2 的预训练数据主要由 60% 的源代码、10% 的数学语料和 30% 的自然语言语料组成。由于自然语言语料直接采样自 DeepSeek-V2 的训练数据集，本节将重点介绍代码和数学数据的收集、清洗和过滤流程。同时，我们通过对比分析实验进一步验证了该数据的质量。

我们收集了 GitHub 上创建于 2023 年 11 月之前的公共仓库。我们首先应用与 DeepSeek-Coder (Guo et al., 2024) 相同的过滤规则和近似去重方法，以过滤掉质量较低和重复的源代码。为了使本文内容完整，我们简要描述这些过滤规则。首先，我们过滤掉平均行长超过 100 个字符

或最大行长超过 1000 个字符的文件。此外，我们移除字母字符比例低于 25% 的文件。除 XSLT 编程语言外，我们进一步过滤掉前 100 个字符中出现字符串 `"<?xml version="` 的文件。对于 HTML 文件，我们考虑可见文本与 HTML 代码的比例。我们保留可见文本至少占代码 20% 且不少于 100 个字符的文件。对于通常包含更多数据的 JSON 和 YAML 文件，我们仅保留字符数在 50 到 5000 之间的文件。这有效移除了大多数数据密集型的文件。通过应用这些过滤规则和近似去重，我们获得了 821B 涵盖 338 种编程语言的代码，以及 185B 代码相关文本（如 Markdown 和 Issues）。支持的编程语言列表见附录 A。我们使用与 DeepSeekV2 相同的分词器，详见 (DeepSeek-AI, 2024)。

为了从 Common Crawl 中收集与代码和数学相关的网页文本，我们遵循与 DeepSeekMath (Shao et al., 2024) 相同的流水线。具体而言，我们选择编程论坛（如 StackOverflow[1]）、库网站（如 PyTorch 文档[2]）和数学网站（如 StackExchange[3]）作为我们的初始种子语料库。使用该种子语料库，我们训练了一个 fastText 模型 (Joulin et al., 2016) 以召回更多与代码和数学相关的网页。由于中文等语言的词法分析无法通过空格完成，我们使用了来自 DeepSeek-V2 的字节对编码（BPE）分词器，这显著提高了 fastText 的召回准确率。对于每个域名，我们计算第一次迭代中收集的网页比例。收集网页比例超过 10% 的域名被归类为代码相关或数学相关。随后，我们标注这些已识别域名内与代码或数学内容相关的 URL。与这些 URL 关联的未收集网页被添加到种子语料库中。经过三轮数据收集，我们从网页中收集了 700 亿个代码相关 token 和 221B 个数学相关 token。为了进一步从 GitHub 收集高质量的源代码，我们还在 GitHub 上应用了相同的流水线，经过两轮数据收集，获得了 94B 源代码。初始种子语料库是通过手动收集包含详细描述等高质量源代码构建的。最终，新的代码语料库由来自 GitHub 和 CommonCrawl 的 1,170B 代码相关 token 组成。

为了证明新代码语料库的有效性，我们使用一个 1B 参数模型进行了消融实验（见表 1），并将其与用于训练 DeepSeek-Coder 的语料库进行了对比。使用 1T token 在新代码语料库上预训练该 1B 模型，使其在 HumanEval（从 30.5% 提升至 36.0%）和 MBPP（从 44.6% 提升至 49.0%）基准测试上的准确率分别提升了 5.5% 和 4.4%。使用 2T token 进一步训练该 1B 模型带来了额外的提升，HumanEval 和 MBPP 的得分分别上升至 37.2% 和 54.0%。因此，新的代码语料库优于用于训练 DeepSeek-Coder 的代码语料库。

模型	Token 数	Python	C++	Java	PHP	TS	C#	Bash	JS	平均	MBPP
DeepSeek-Coder-1B	1T	30.5%	28.0%	31.7%	23.0%	30.8%	31.7%	9.5%	28.6%	26.7%	44.6%
DeepSeek-Coder-V2-1B	1T	36.0%	34.8%	31.7%	27.3%	**37.7%**	34.2%	6.3%	**38.5%**	31.2%	49.0%
DeepSeek-Coder-V2-1B	2T	**37.2%**	**39.1%**	**32.3%**	**31.7%**	34.6%	**36.7%**	**12.0%**	32.9%	**32.0%**	**54.0%**

表 1 | DeepSeek-Coder 与 DeepSeek-Coder-V2 的 1B 基础模型性能对比。

[1]https://stackoverflow.com

[2]https://pytorch.org/docs

[3]https://math.stackexchange.com

3. 训练策略

3.1. 训练策略

对于 DeepSeek-Coder-v2 16B，我们使用两个训练目标：下一 Token 预测（Next-Token-Prediction）和中间填充（Fill-In-Middle, FIM）（Bavarian et al., 2022; Guo et al., 2024; Li et al., 2023b）。对于 DeepSeek-Coder-v2 236B，我们仅使用下一 Token 预测目标。此处我们简要介绍 FIM 训练策略。在开发 DeepSeek-Coder-v2-16B 时，我们采用了 FIM 训练方法，利用 PSM（前缀、后缀、中间）模式。该方法按前缀、后缀和中间的顺序构建内容重建，如下所示：

<|fim_begin|>f_{pre}<|fim_hole|>f_{suf}<|fim_end|>f_{middle}<|eos_token|>

该结构在文档级别应用，作为预打包过程的一部分。FIM 的使用比例为 0.5，与 PSM 框架保持一致，以提升训练效率和模型性能。

3.2. 模型架构

我们的架构与 DeepSeekV2 (DeepSeek-AI, 2024) 保持一致。16B 和 236B 的超参数设置分别对应于 DeepSeek-V2-Lite 和 DeepSeek-V2 所使用的设置。值得注意的是，我们在训练过程中遇到了不稳定性以及梯度值激增的问题，我们将其归因于指数归一化技术。为了解决这一问题，我们恢复使用了传统的归一化方法。

3.3. 训练超参数

与 DeepSeek V2 的方法 (DeepSeek-AI, 2024) 保持一致，我们使用 AdamW 优化器 (Loshchilov and Hutter, 2019)，配置为 $\beta_1 = 0.9, \beta_2 = 0.95$，权重衰减为 0.1。批量大小和学习率根据 DeepSeek-V2 的规范进行调整。对于学习率调度，我们采用余弦衰减策略，从 2000 步预热开始，并逐渐将学习率降低至初始值的 10%。

DeepSeek-Coder-V2 和 DeepSeek-Coder-V2-Lite 均采用相同的方法进行训练。为了保持 DeepSeek-Coder-V2 强大的自然语言理解能力，我们从 DeepSeek-V2 的一个中间检查点继续预训练过程。该中间检查点最初在 4.2T token 上进行了训练。因此，在预训练阶段，DeepSeek-Coder-V2 总共接触了 10.2T 高质量 token。

模型	DeepSeek-Coder-V2-Lite	DeepSeek-Coder-V2
总参数量 (#TP)	16B	236B
激活参数量 (#AP)	2.4B	21B
预训练 Token 数	4.2T+6T	4.2T+6T
学习率调度器	余弦衰减	余弦衰减
FIM	启用	禁用

表 2 | DeepSeek-Coder-V2 的训练设置。

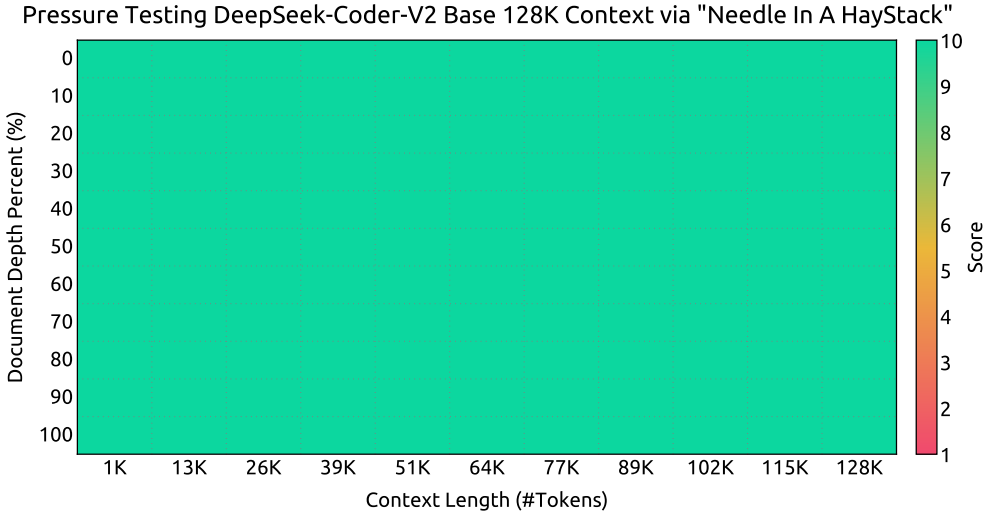

图 2 | “大海捞针”（Needle In A Haystack, NIAH）测试的评估结果。DeepSeek-Coder-V2 在高达 128K 的所有上下文窗口长度上均表现良好。

3.4. 长上下文扩展

遵循 DeepSeek-V2 的做法，我们使用 Yarn (Peng et al., 2023) 将 DeepSeek-Coder-V2 的上下文长度扩展至 128K。YARN 的超参数与 DeepSeek-V2 相同：缩放比例 s 设为 40，α 设为 1，β 设为 32。我们进一步采用两个阶段继续训练模型，以增强其处理长上下文的能力。在第一阶段，我们使用 32K 的序列长度和 1152 的批次大小训练 1000 步。在第二阶段，我们额外训练 1000 步，采用 128K 的序列长度和 288 的批次大小。需要注意的是，在长上下文扩展期间，我们提高了长上下文数据的采样比例。如图 2 所示，在“大海捞针”（Needle In A Haystack, NIAH）测试上的结果表明，DeepSeek-Coder-V2 在高达 128K 的所有上下文窗口长度上均表现良好。

3.5. 对齐

3.5.1. 监督微调

为了构建 DeepSeek-Coder-V2 Chat，我们构建了混合代码和数学数据的指令训练数据集。我们首先从 DeepSeek-Coder 和 DeepSeek-Math 中收集了 2 万条代码相关指令数据和 3 万条数学相关数据。为了保持通用能力，我们还从 DeepSeek-V2 的指令数据中采样了部分数据。最终，我们使用了包含 3 亿 token 的指令数据集。在训练过程中，我们采用余弦学习率调度策略，设置 100 步预热，初始学习率为 $5e^{-6}$。我们还使用了 100 万 token 的批次大小，总训练数据量为 10 亿 token。

3.5.2. 强化学习

我们进一步采用强化学习（RL）技术来充分激发 DeepSeek-Coder-V2 的能力，这已被证明非常有效。

提示词 我们投入了大量精力从各种来源收集与代码和数学相关的提示词，并且每个代码提示词都附带相应的测试用例。经过筛选后，提示词数据总量约为 4 万条。

奖励建模 奖励模型在强化学习训练中起着至关重要的作用。对于数学偏好数据，我们通过真实标签（ground-truth labels）获取。对于代码偏好数据，尽管代码编译器本身已经可以提供 0-1 反馈（即代码是否通过所有测试用例），但某些代码提示词的测试用例数量有限，无法提供全面覆盖，因此直接使用编译器的 0-1 反馈可能会存在噪声且并非最优。因此，我们仍然决定在编译器提供的数据上训练一个奖励模型，并在强化学习训练期间使用该奖励模型提供信号。与原始编译器信号相比，这种方法更加稳健且具有更好的泛化能力。如图 3 所示，在我们的内部测试集（Leetcode 和 Leetcode-zh）上，使用奖励模型提供强化学习训练信号的效果明显优于使用原始编译器信号。因此，在后续所有实验中，我们均采用奖励模型信号而非编译器信号。

强化学习算法 我们采用组相对策略优化（Group Relative Policy Optimization, GRPO）Shao et al. (2024) 作为我们的强化学习算法，这与 DeepSeek-V2 使用的算法相同。值得注意的是，GRPO 已被证明非常有效，且与 PPO 相比成本更低，因为它无需维护额外的评论家（critic）模型。

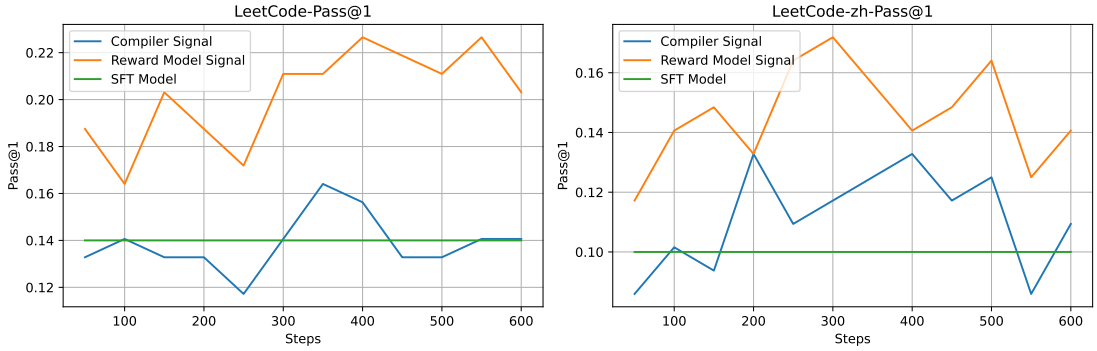

图 3 | 不同方法的性能对比

4. 实验结果

在本节中，我们在编程、数学和通用自然语言三类任务上对 DeepSeek-Coder-V2 进行评估。我们将 DeepSeek-Coder-V2 与以往的最先进大语言模型进行了对比。

- **CodeLlama** (Roziere et al., 2023) 是一系列基于 Llama2 (Touvron et al., 2023) 的代码语言模型，并在 5000 亿至 1 万亿代码 token 的数据集上继续预训练。这些模型提供四种规模：7B、13B、34B 和 70B。
- **StarCoder** (Lozhkov et al., 2024) 是一个公开可用的 150 亿参数模型。它专门在精心筛选的 Stack 数据集 (Kocetkov et al., 2022) 子集上进行训练，涵盖 86 种编程语言。

- **StarCoder2** (Lozhkov et al., 2024) 包含 3B、7B 和 15B 参数模型，在 Stack2 数据集 (Lozhkov et al., 2024) 的 3.3 万亿至 4.3 万亿 token 上进行训练，涵盖 619 种编程语言。
- **DeepSeek-Coder** (Guo et al., 2024) 包含一系列代码语言模型，参数量从 10 亿到 330 亿不等。每个模型均在 2 万亿 token 上从头训练，数据组成为 87% 的代码和 13% 的中英文自然语言。这些模型在 16K 窗口大小的项目级代码语料库上进行预训练，并引入了额外的填空任务，从而支持项目级代码补全和插入。
- **Codestral** (MistralAI, 2024) 是由 Mistral 开发的 220 亿参数模型。它在涵盖 80 多种编程语言的多样化数据集上进行训练，包括 Python、Java 和 JavaScript 等流行语言，以及 Swift 和 Fortran 等更专业的语言。
- 我们对比的通用语言模型包括 **Llama3 70B** (Meta, 2024)、**GPT-4** (OpenAI, 2023)、**Claude 3 Opus** (Anthropic, 2024) 和 **Gemini 1.5 Pro** (Reid et al., 2024)。尽管它们并非专门在大型代码语料库上训练，但在编程任务上仍达到了最先进的性能。

4.1. 代码生成

HumanEval 和 MBPP 基准测试。 HumanEval (Chen et al., 2021) [4] 和 MBPP (Austin et al., 2021b) 基准测试通常用于评估代码生成大语言模型（LLMs）的性能。HumanEval 包含 164 个 Python 任务，通过测试用例验证，用于评估代码大语言模型在零样本场景下的性能。对于 MBPP，我们使用 MBPP-Plus 版本 (Liu et al., 2023a) 来评估模型。为了测试模型的多语言能力，我们将 HumanEval 基准测试问题扩展到了另外七种语言：C++、Java、PHP、TypeScript、C#、Bash、JavaScript、Swift、R、Julia、D、Rust 和 Racket。在这两个基准测试中，我们均采用贪婪搜索策略，并使用相同的脚本和环境重新生成基线结果，以确保公平比较。

表 3 全面展示了各模型在 HumanEval 和 MBPP$^+$ 基准测试上跨多种编程语言的性能指标。DeepSeek-Coder-V2-Instruct 表现出卓越的性能，以 75.3% 的平均得分位居第二。这一表现尤为引人注目，因为它打破了闭源模型通常占据的主导地位，成为领先的开源竞争者。其得分仅略低于以 76.4% 平均得分领跑的 GPT-4o。DeepSeek-Coder-V2-Instruct 在多种语言上均展现出顶尖水平，包括在 Java 和 PHP 中取得最高分，并在 Python、C++、C#、TypeScript 和 JavaScript 中表现强劲，凸显了其在应对多样化编程挑战时的鲁棒性与通用性。

此外，DeepSeek-Coder-V2-Lite-Instruct 的表现同样令人印象深刻，超越了参数量更大的 33B 模型。凭借在平均性能上的显著优势（65.6% 对比 61.9%），它凸显了 16B 模型在规模较小的情况下仍能交付具有竞争力结果的效能。这进一步印证了该模型的高效性，以及其在模型架构和训练方法上的进步，使其能够超越更大的同类模型。

算法竞赛。 为了进一步验证模型在真实算法竞赛问题中的能力，我们利用 LiveCodeBench (Jain et al., 2024) 和 USACO 基准测试 (Shi et al., 2024) 来评估 DeepSeek-Coder-V2 的有效性。

[4] 我们使用模板 “Please complete the python function below. The final complete version of your function must be returned within a code block. Here is the unfinished function:\n ```python\n{problem_description}\n\n” 来构建指令提示词。

	#TP	#AP	Python	Java	C++	C#	TS	JS	PHP	Bash
闭源模型										
Gemini-1.5-Pro	-	-	83.5%	81.0%	78.3%	75.3%	77.4%	80.8%	74.5%	39.9%
Claude-3-Opus	-	-	84.2%	78.5%	81.4%	74.7%	76.1%	75.8%	78.3%	48.7%
GPT-4-1106	-	-	87.8%	**82.3%**	78.9%	80.4%	81.8%	80.1%	77.6%	**55.7%**
GPT-4-Turbo-0409	-	-	88.2%	81.7%	78.3%	79.1%	79.3%	80.8%	78.9%	**55.1%**
GPT-4o-0513	-	-	**91.0%**	80.4%	**87.0%**	**82.9%**	**86.2%**	**87.6%**	**79.5%**	53.8%
开源模型										
Codestral	22B	22B	78.1%	71.5%	71.4%	77.2%	72.3%	73.9%	69.6%	47.5%
DS-Coder-instruct	33B	33B	79.3%	73.4%	68.9%	74.1%	67.9%	73.9%	72.7%	43.0%
Llama3-Instruct	70B	70B	81.1%	67.7%	64.0%	69.6%	69.8%	70.2%	65.8%	36.1%
DS-Coder-V2-Lite-Instruct	16B	2.4B	81.1%	76.6%	75.8%	76.6%	80.5%	77.6%	74.5%	43.0%
DS-Coder-V2-Instruct	236B	21B	**90.2%**	**82.3%**	**84.8%**	**82.3%**	**83.0%**	**84.5%**	**79.5%**	**52.5%**

	#TP	#AP	Swift	R	Julia	D	Rust	Racket	MBPP+	平均
闭源模型										
Gemini-1.5-Pro	-	-	66.5%	53.4%	71.7%	55.8%	73.1%	48.4%	**74.6%**	68.9%
Claude-3-Opus	-	-	63.9%	55.9%	76.1%	60.3%	71.2%	**64.6%**	72.0%	70.8%
GPT-4-1106	-	-	62.7%	57.8%	69.2%	60.9%	**78.8%**	64.0%	69.3%	72.5%
GPT-4-Turbo-0409	-	-	63.9%	56.5%	69.8%	**61.5%**	**78.8%**	63.4%	72.2%	72.3%
GPT-4o-0513	-	-	**75.9%**	**65.2%**	**78.0%**	60.9%	**80.1%**	**64.6%**	73.5%	**76.4%**
开源模型										
Codestral	22B	22B	63.3%	49.7%	67.9%	32.1%	67.3%	37.3%	68.2%	63.2%
DS-Coder-instruct	33B	33B	61.4%	44.7%	53.5%	31.4%	68.6%	46.0%	70.1%	61.9%
Llama3-Instruct	70B	70B	55.1%	46.0%	62.9%	48.1%	58.3%	46.0%	68.8%	60.6%
DS-Coder-V2-Lite-Instruct	16B	2.4B	64.6%	47.8%	67.3%	45.5%	62.2%	41.6%	68.8%	65.6%
DS-Coder-V2-Instruct	236B	21B	**72.2%**	**64.0%**	**72.3%**	**64.1%**	**78.2%**	**63.4%**	**76.2%**	**75.3%**

表 3 | HumanEval 和 MBPP 基准测试上各模型的性能指标

LiveCodeBench 是一项针对大语言模型 (LLM) 代码生成能力的严谨且无数据污染的评估，它系统性地从三个著名的算法竞赛平台（LeetCode、AtCoder 和 CodeForces）随时间推移收集新颖的挑战题。由于训练数据的截止日期在 2023 年 11 月之前，我们使用了 LiveCodeBench 的子集 (1201-0601)。USACO 基准测试包含来自美国计算机奥林匹克竞赛 (USA Computing Olympiad) 的 307 道题目，并为每道题提供了高质量的单元测试、参考代码和官方解析。

表 4 展示了各语言模型在这两个基准测试上的性能表现。值得注意的是，DeepSeek-Coder-V2-Instruct 表现突出，以 43.4% 的得分与大模型中的最高分并列，与 GPT-4o 持平。这一卓越的成绩使其在总排名中位列第二，仅次于以 45.7% 综合表现领跑的 GPT-4-Turbo-0409。DeepSeek-Coder-V2-Instruct 在处理复杂编程挑战方面的出色能力，使其稳居顶尖竞争者之列，紧随领先的 GPT-4-Turbo 变体之后。

4.2. 代码补全

4.2.1. 仓库级代码补全评估

我们使用 RepoBench (Liu et al., 2023b) 来评估当前可用的参数量低于 35B 的开源代码模型在仓库级代码补全任务中的能力。该数据集由 Python 和 Java 这两种流行编程语言中多样化的真实世界、开源且采用宽松许可证的仓库构建而成。值得注意的是，RepoBench 的最新版本 (v1.1)

Model	#TP	#AP	LiveCodeBench 简单 (82)	中等 (87)	困难 (57)	总计 (226)	USACO
闭源模型							
Gemini-1.5-Pro	-	-	74.9%	16.8%	1.8%	34.1%	4.9%
Claude-3-Opus	-	-	77.2%	16.7%	0.7%	34.6%	7.8%
GPT-4-1106	-	-	78.4%	20.2%	3.5%	37.1%	11.1%
GPT-4-Turbo-0409	-	-	84.1%	**35.4%**	**6.1%**	**45.7%**	12.3%
GPT-4o-0513	-	-	**87.4%**	27.5%	4.9%	43.4%	**18.8%**
开源模型							
Codestral	22B	22B	66.5%	17.7%	0.2%	31.0%	4.6%
DS-Coder-instruct	33B	33B	51.6%	9.7%	0.4%	22.5%	4.2%
Llama3-Instruct	70B	70B	62.4%	14.4%	2.1%	28.7%	3.3%
DS-Coder-V2-Lite-Instruct	16B	2.4B	58.5%	8.0%	0.0%	24.3%	6.5%
DS-Coder-V2-Instruct	236B	21B	**84.1%**	**29.9%**	**5.3%**	**43.4%**	**12.1%**

表 4 | 在 LiveCodeBench (LCB) 和 USACO 基准测试上的性能表现。

的数据来源于 2023 年 10 月 6 日至 12 月 31 日期间创建的 GitHub 仓库，而我们的预训练数据包含 2023 年 11 月之前创建的代码。为确保该数据集未出现在我们的预训练数据中并避免数据泄露，我们仅使用 2023 年 12 月的数据。

我们的评估涵盖三种设置：跨文件首部（cross-file-first）、跨文件随机（cross-file-random）和文件内（in-file），并包含五个上下文长度级别：2k、4k、8k、12k 和 16k tokens。我们对所有待评估模型均采用贪婪搜索策略。模型被限制为每个提示最多生成 64 个新 token，并选取输出的第一个非空且非注释行作为预测结果。通过截断多余的跨文件上下文，提示的最大 token 长度被设置为 15,800。我们报告了不同上下文长度级别的平均精确匹配率（Exact Match）。如

Model	#TP	#AP	Python						Java					
			2k	4k	8k	12k	16k	平均	2k	4k	8k	12k	16k	平均
StarCoder2-Base	15B	15B	35.7%	36.7%	34.6%	27.4%	25.1%	32.1%	46.2%	45.0%	39.8%	30.5%	30.7%	38.7%
CodeLlama-Base	7B	7B	32.0%	34.4%	35.3%	33.3%	32.2%	33.5%	43.1%	42.1%	40.4%	37.0%	40.3%	40.6%
CodeLlama-Base	13B	13B	33.0%	36.5%	37.0%	34.6%	35.0%	35.2%	43.5%	44.8%	40.7%	38.6%	41.1%	41.8%
CodeLlama-Base	34B	34B	35.3%	37.5%	39.5%	34.9%	35.6%	36.6%	45.9%	45.4%	42.5%	41.0%	41.2%	43.3%
DS-Coder-Base	6.7B	6.7B	36.1%	37.5%	38.2%	34.0%	35.0%	36.2%	46.8%	46.4%	42.9%	38.8%	40.8%	43.3%
DS-Coder-Base	33B	33B	39.7%	40.1%	40.0%	36.9%	38.5%	39.1%	47.9%	47.7%	43.3%	40.9%	43.6%	44.8%
Codestral	22B	22B	**42.1%**	**44.3%**	**46.6%**	**46.6%**	**51.5%**	**46.1%**	48.3%	**47.8%**	**46.0%**	**42.2%**	**43.9%**	**45.7%**
DS-Coder-V2-Lite-Base	16B	2.4B	38.3%	38.6%	40.6%	38.3%	38.7%	38.9%	**48.8%**	45.7%	42.4%	38.1%	41.1%	43.3%

表 5 | 不同模型在 RepoBench v1.1 12 月子集上的性能表现。

表 5 所示，结果表明，尽管 DeepSeek-Coder-V2-Lite-Base 模型仅拥有 24 亿个活跃参数，但其在 Python 中的代码补全能力可与 DeepSeek-Coder-Base 33B 模型相媲美，在 Java 中则可与 DeepSeek-Coder-Base 7B 模型相媲美。与 CodeStral 相比，DeepSeek-Coder-V2-Lite-Base 模型的活跃参数仅为 CodeStral 的十分之一，因此在代码补全任务上的表现略低。然而，我们认为 DeepSeek-Coder-V2 较少的活跃参数使其在代码补全场景中具有更快的推理速度。

4.2.2. 中间填充代码补全

DeepSeek-Coder-V2-Lite 采用了一种独特的训练策略，在其预训练阶段包含了 0.5 的中间填充（Fill-In-the-Middle, FIM）比例。该方法使模型能够熟练地利用上下文（包括前序和后序代码片段）来填充空白，从而完成代码补全。这一能力对于代码补全工具尤为有利。多个开源模型，如 SantaCoder (Allal et al., 2023)、StarCoder (Li et al., 2023b) 和 CodeLlama (Roziere et al., 2023)，也利用了类似的能力，并在代码生成与补全领域树立了高标准。

为了评估 DeepSeek-Coder-V2 模型的性能，我们与主流领先模型进行了对比分析。评估基于单行填充（Single-Line Infilling）基准测试，涵盖了 Allal et al. (2023) 所述的三种不同编程语言。本次评估的主要指标为行精确匹配准确率[5]。

模型	#TP	#AP	Python	Java	JavaScript	平均
StarCoder[6]	16B	16B	71.5%	82.3%	83.0%	80.2%
CodeLlama-Base	7B	7B	58.6%	70.6%	70.7%	68.0%
CodeLlama-Base	13B	13B	60.7%	74.3%	78.5%	73.1%
DS-Coder-Base	1B	1B	74.1%	85.1%	82.9%	81.8%
DS-Coder-Base	7B	7B	79.8%	**89.6**%	86.3%	86.1%
DS-Coder-Base	33B	33B	**80.5%**	88.4%	86.6%	**86.4**%
Codestral	22B	22B	77.2%	83.2%	85.9%	83.0%
DS-Coder-V2-Lite-Base	16B	2.4B	80.0%	89.1%	**87.2%**	**86.4%**

表 6 | 不同方法在 FIM 任务上的性能表现。

该表展示了多种代码模型在三种编程语言（Python、Java 和 JavaScript）的 FIM（中间填充）任务上的性能表现，其中 Mean（平均分）指标反映了整体有效性。在对比的模型中，配置为 24 亿活跃参数的 DeepSeek-Coder-V2-Lite-Base 取得了出色的结果。它在 Python、Java 和 JavaScript 上的得分分别为 80.0%、89.1% 和 87.2%，平均分高达 86.4%，位居榜首。这证明了 DeepSeek-Coder-V2-Lite-Base 的卓越有效性，特别是在处理不同编程语言的 FIM 任务时，其性能可与评估中的其他更大规模模型相媲美。

4.3. 代码修复

为了评估模型的代码修复能力，我们使用了 Defects4J [7]、SWE-bench (Jimenez et al., 2023) 和 Aider [8] 数据集进行测试。Defects4J 是软件工程领域广泛使用的数据集，专门用于评估和测试程序修复技术。该数据集包含来自多个开源项目的真实软件缺陷，包括但不限于 Apache Commons、JFreeChart 和 Closure Compiler。数据集中的每个缺陷都附带测试套件，可用于验证程序修复工具的有效性。由于 Defects4J 中的原始缺陷可能需要修改仓库中的多个文件，从而导致上下文过长，我们从此基准中筛选出仅需修改单个方法的 238 个缺陷。

SWE-bench 是一个综合性基准测试，旨在评估大语言模型解决源自 GitHub 的真实软件问

[5]我们使用生成的第一行代码而非整个生成的代码块，因此结果与 DeepSeek-Coder 的原始报告略有不同。

[7]https://github.com/rjust/defects4j

[8]https://github.com/paul-gauthier/aider

题的能力。该基准测试会提供一个代码库及一个具体问题，要求语言模型生成能够有效解决所述问题的补丁。这一严格的评估框架确保了语言模型理解和修复真实软件问题的能力得到充分测试，为其在软件开发任务中的实用性和有效性提供了明确的衡量标准。

Aider 的代码编辑基准测试评估了大语言模型修改 Python 源文件的能力，共包含 133 个不同的编程任务。该基准测试不仅检验了大语言模型的编程技能，还评估了其根据提示词规范生成代码修改的一致性。对于 DeepSeek-Coder-V2 模型，我们采用 *whole* 格式进行评估。

模型	#TP	#AP	Defects4J	SWE-Bench	Aider
闭源模型					
Gemini-1.5-Pro	-	-	18.6%	19.3%	57.1%
Claude-3-Opus	-	-	25.5%	11.7%	68.4%
GPT-4-1106	-	-	22.8%	22.7%	65.4%
GPT-4-Turbo-0409	-	-	24.3%	18.3%	63.9%
GPT-4o-0513	-	-	**26.1**%	**26.7**%	**72.9**%
开源模型					
Codestral	22B	22B	17.8%	2.7%	51.1%
DS-Coder-Instruct	33B	33B	11.3%	0.0%	54.5%
Llama3-Instruct	70B	70B	16.2%	-	49.2%
DS-Coder-V2-Lite-Instruct	16B	2.4B	9.2%	0.0%	44.4%
DS-Coder-V2-Instruct	236B	21B	**21.0%**	**12.7**%	**73.7**%

表 7 | 不同模型在代码修复基准测试上的性能表现。由于 Llama3-Instruct 仅支持 8K 上下文长度，我们未对其在 SWE-Bench 上进行评估。

表 7 概述了不同语言模型在软件修复基准测试（包括 Defects4J、SWE-Bench 和 Aider）上的性能表现。在开源模型中，DeepSeek-Coder-Instruct 表现突出，取得了开源模型中的最佳成绩。它在 Defects4J 和 SWE-Bench 上的得分分别为 21% 和 12.7%，非常接近领先闭源模型的结果，并展现出处理较长代码序列的显著能力。值得注意的是，DeepSeek-Coder-V2-Instruct 在 Aider 上取得了 73.7% 的最高分，超越了列表中包括闭源模型在内的所有其他模型。这一卓越表现凸显了其在自动化代码修复任务中的高效性与鲁棒性，使 DeepSeek-Coder-V2-Instruct 成为该领域顶尖的开源模型，并对闭源替代方案构成了强有力的竞争。

4.4. 代码理解与推理

为了评估我们模型的代码推理能力，我们使用了 CRUXEval 基准测试。该基准测试包含 800 个 Python 函数及其对应的输入输出示例。它分为两个不同的任务：CRUXEval-I 要求大语言模型（LLM）根据给定输入预测输出，而 CRUXEval-O 则要求模型根据已知输出反推输入。这种结构旨在考验模型在正向和反向理解及推理 Python 代码的能力。表 8 展示了不同语言模型在 CruxEval 基准测试上的性能表现，该测试主要评估两个指标：CruxEval-I-COT 和 CruxEval-O-COT。在开源模型中，DeepSeek-Coder-V2-Instruct 表现尤为突出。它在 CruxEval-I-COT 和 CruxEval-O-COT 上的得分分别为 70.0% 和 75.1%，展现了其在开源领域中的卓越能力。然而，与规模更大的闭源模型相比，仍存在一定性能差距。这一差距很大程度上可归因于 DeepSeek-

模型	#TP	#AP	CruxEval-I-COT	CruxEval-O-COT
闭源模型				
Gemini-1.5-Pro	-	-	67.0%	77.5%
Claude-3-Opus	-	-	73.4%	82.0%
GPT-4-1106	-	-	75.5%	77.1%
GPT-4-Turbo-0409	-	-	75.7%	82.0%
GPT-4o-0513	-	-	**77.4%**	**88.7%**
开源模型				
Codestral	22B	22B	48.0%	60.6%
DS-Coder-Instruct	33B	33B	47.3%	50.6%
Llama3-Instruct	70B	70B	61.1%	64.3%
DS-Coder-V2-Lite-Instruct	16B	2.4B	53.0%	52.9%
DS-Coder-V2-Instruct	236B	21B	**70.0%**	**75.1%**

表 8 | 不同模型在 CruxEval 基准测试上的性能表现。

Coder-V2-Instruct 仅使用 210 亿个激活参数，远低于 GPT-4o 等更大、更先进的闭源模型。这种模型复杂度的限制可能会制约其学习与问题解决能力。

4.5. 数学推理

为了评估 DeepSeekCoder-V2 的数学推理能力，我们使用了流行的小学数学基准测试 GSM8K (Cobbe et al., 2021)，以及包括 MATH (Hendrycks et al., 2021)、2024 年美国数学邀请赛（AIME）(MAA, 2024) 和 Math Odyssey (Netmind.AI, 2024) 在内的高级竞赛级基准测试[9]。

模型	#TP	#AP	GSM8K	MATH	AIME 2024	Math Odyssey
闭源模型						
Gemini 1.5 Pro	-	-	90.8%	67.7%	2/30	45.0%
Claude-3-Opus	-	-	95.0%	60.1%	2/30	40.6%
GPT-4-1106	-	-	91.4%	64.3%	1/30	49.1%
GPT-4-Turbo-0409	-	-	93.7%	73.4%	**3/30**	46.8%
GPT-4o-0513	-	-	**95.8%**	**76.6%**	2/30	**53.2%**
开源模型						
Llama3-Instruct	70B	70B	93.0%	50.4%	1/30	27.9%
DS-Coder-V2-Lite-Instruct	16B	2.4B	86.4%	61.8%	0/30	44.4%
DS-Coder-V2-Instruct	236B	21B	**94.9%**	**75.7%**	**4/30**	**53.7%**

表 9 | 不同模型在数学推理任务上的性能表现。DeepSeek-Coder-V2-Instruct 在 AIME 2024 上使用 maj@64 策略可达到 5/30 的成绩。

表 9 中展示的结果均采用贪婪解码获得，未借助任何工具或投票技术，除非另有说明。DeepSeek-Coder-V2 在 MATH 基准测试上的准确率达到 75.7%，在 Math Odyssey 上达到 53.7%，与最先进的 GPT-4o 表现相当。此外，DeepSeek-Coder-V2 在 AIME 2024 中解决的题目数量多于其

[9]DeepSeek-Coder-V2 在这四个数学基准测试上的性能是通过零样本思维链提示获得的；每个测试问题均拼接了如下指令："\nPlease reason step by step, and put your final answer within \boxed{}."

他模型，进一步证明了其强大的数学推理能力。

4.6. 通用自然语言

由于 DeepSeek-Coder-V2 基于 DeepSeek-V2 构建，它继承了强大的自然语言处理能力，甚至在推理相关的基准测试上超越了 DeepSeek-V2。我们在标准基准测试上比较了 DeepSeek-Coder-V2 Instruct 与 DeepSeek-V2 Chat 的性能，这些基准测试涵盖中英文，包括 BigBench Hard (BBH) (Suzgun et al., 2022)、MMLU (Hendrycks et al., 2020)、ARC (Clark et al., 2018)、TriviaQA (Joshi et al., 2017)、NaturalQuestions (Kwiatkowski et al., 2019)、AGIEval (Zhong et al., 2023)、CLUEWSC (Xu et al., 2020)、C-Eval (Huang et al., 2023) 和 CMMLU (Li et al., 2023a)。此外，我们还评估了模型的开放式生成能力，包括 Arena-Hard (Li et al., 2024)、AlpacaEval2.0 (Dubois et al., 2024)、MT-Bench (Zheng et al., 2023) 和 Alignbench (Liu et al., 2023c)。评估流程和指标与 DeepSeek-V2 保持一致，其中 MMLU 使用 OpenAI 的 simple-eval 包 `https://github.com/openai/simple-evals` 进行评估。

	基准测试 (指标)	提示数	DeepSeek-V2-Lite Chat	DeepSeek-Coder-V2-Lite Instruct	DeepSeek-V2 Chat	DeepSeek-Coder-V2 Instruct
	激活参数量	-	2.4B	2.4B	21B	21B
	总参数量	-	16B	16B	236B	236B
	训练 Token 数	-	5.7T	10.2T	8.1T	10.2T
英文	BBH (EM)	3-shot	48.1	**61.2**	79.7	**83.9**
	MMLU (Acc.)	5-shot	55.7	**60.1**	78.1	**79.2**
	ARC-Easy (Acc.)	25-shot	86.1	**88.9**	**98.1**	97.4
	ARC-Challenge (Acc.)	25-shot	73.4	**77.4**	92.3	**92.8**
	TriviaQA (EM)	5-shot	**65.2**	59.5	**86.7**	82.3
	NaturalQuestions (EM)	5-shot	**35.5**	30.8	**53.4**	47.5
	AGIEval (Acc.)	0-shot	**42.8**	28.7	**61.4**	60.0
中文	CLUEWSC (EM)	5-shot	**80.0**	76.5	**89.9**	85.9
	C-Eval (Acc.)	5-shot	60.1	**61.6**	78.0	**79.4**
	CMMLU (Acc.)	5-shot	62.5	**62.7**	**81.6**	80.9
开放式生成	Arena-Hard	-	11.40	**38.10**	41.60	**65.00**
	AlpacaEval 2.0	-	16.85	**17.74**	**38.90**	36.92
	MT-Bench	-	7.37	**7.81**	**8.97**	8.77
	Alignbench	-	6.02	**6.83**	**7.91**	7.84

表 10 | DeepSeek-Coder-V2 Instruct 与 DeepSeek-V2 Chat 的性能对比。

在对比 16B 模型的性能时，可以明显看出 DeepSeek-Coder-V2-Lite-Instruct 在 BBH 和 Arena-Hard 等基准测试上优于 DeepSeek-V2-Lite-Chat。这些基准测试对模型的推理能力要求较高，而 DeepSeek-Coder-V2-Lite-Instruct 在此方面表现出色。然而，在 TriviaQA 等知识密集型基准测试中，DeepSeek-Coder-V2-Lite Instruct 的表现稍逊一筹，这主要归因于其预训练阶段使用的网络数据量相对较少。

对于 236B 模型，DeepSeek-Coder-V2 Instruct 在推理基准测试中展现出更强的实力，尤其是在 Arena-Hard 上，该基准包含大量代码、数学和推理问题。另一方面，DeepSeek-V2 Chat 在 MT-bench (Zheng et al., 2023)、AlpacaEval 2.0 (Dubois et al., 2024) 和 AlignBench (Liu et al., 2023c) 等基准测试中表现略优。这一优势可归因于 DeepSeek-V2 Chat 的通用对齐阶段。

5. 结论

在本文中，我们介绍了 DeepSeek-Coder-V2 ，旨在进一步推动代码智能领域的发展。该模型基于 DeepSeek-V2 进行持续预训练，使用了来自高质量、多源语料库的 6 万亿个 token。通过这种持续预训练，我们发现 DeepSeek-Coder-V2 在保持与 DeepSeek-V2 相当的一般语言性能的同时，显著提升了模型在代码编写和数学推理方面的能力。与 DeepSeek-Coder 相比，DeepSeek-Coder-V2 支持的编程语言数量大幅增加，从 86 种增至 338 种，并将最大上下文长度从 16K 扩展至 128K 个 token。实验结果表明，在代码和数学特定任务上，DeepSeek-Coder-V2 的性能可与 GPT-4 Turbo、Claude 3 Opus 和 Gemini 1.5 Pro 等最先进的闭源模型相媲美。

尽管 DeepSeek-Coder-V2 在标准基准测试中取得了令人瞩目的成绩，但我们发现，与 GPT-4 Turbo 等当前最先进的模型相比，其在指令遵循能力方面仍存在显著差距。这一差距导致其在 SWEbench 等复杂场景和任务中的表现不佳。因此，我们认为，代码模型不仅需要强大的编程能力，还需要卓越的指令遵循能力，以应对现实世界中复杂的编程场景。未来，我们将更加专注于提升模型的指令遵循能力，以更好地应对现实世界中的复杂编程场景，并提高开发流程的生产力。

参考文献

L. B. Allal, R. Li, D. Kocetkov, C. Mou, C. Akiki, C. M. Ferrandis, N. Muennighoff, M. Mishra, A. Gu, M. Dey, et al. Santacoder: don't reach for the stars! arXiv preprint arXiv:2301.03988, 2023.

A. Anthropic. The claude 3 model family: Opus, sonnet, haiku. Claude-3 Model Card, 2024.

J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, and C. Sutton. Program synthesis with large language models, 2021a.

J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, et al. Program synthesis with large language models. arXiv preprint arXiv:2108.07732, 2021b.

M. Bavarian, H. Jun, N. Tezak, J. Schulman, C. McLeavey, J. Tworek, and M. Chen. Efficient training of language models to fill in the middle. arXiv preprint arXiv:2207.14255, 2022.

M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374, 2021.

P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. CoRR, abs/1803.05457, 2018. URL `http://arxiv.org/abs/1803.05457`.

K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021.

DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024.

Y. Dubois, B. Galambosi, P. Liang, and T. B. Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. arXiv preprint arXiv:2404.04475, 2024.

D. Guo, Q. Zhu, D. Yang, Z. Xie, K. Dong, W. Zhang, G. Chen, X. Bi, Y. Wu, Y. Li, et al. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. arXiv preprint arXiv:2401.14196, 2024.

D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. arXiv preprint arXiv:2009.03300, 2020.

D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset. arXiv preprint arXiv:2103.03874, 2021.

Y. Huang, Y. Bai, Z. Zhu, J. Zhang, J. Zhang, T. Su, J. Liu, C. Lv, Y. Zhang, J. Lei, et al. C-Eval: A multi-level multi-discipline chinese evaluation suite for foundation models. arXiv preprint arXiv:2305.08322, 2023.

N. Jain, K. Han, A. Gu, W.-D. Li, F. Yan, T. Zhang, S. Wang, A. Solar-Lezama, K. Sen, and I. Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code, 2024.

C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan. Swe-bench: Can language models resolve real-world github issues? arXiv preprint arXiv:2310.06770, 2023.

M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In R. Barzilay and M.-Y. Kan, editors, Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147.

A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov. Fasttext. zip: Compressing text classification models. arXiv preprint arXiv:1612.03651, 2016.

D. Kocetkov, R. Li, L. Jia, C. Mou, Y. Jernite, M. Mitchell, C. M. Ferrandis, S. Hughes, T. Wolf, D. Bahdanau, et al. The stack: 3 tb of permissively licensed source code. Transactions on Machine Learning Research, 2022.

T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. P. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural questions: a benchmark for question answering research. Trans. Assoc. Comput. Linguistics, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://doi.org/10.1162/tacl_a_00276.

H. Li, Y. Zhang, F. Koto, Y. Yang, H. Zhao, Y. Gong, N. Duan, and T. Baldwin. CMMLU: Measuring massive multitask language understanding in Chinese. arXiv preprint arXiv:2306.09212, 2023a.

R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim, et al. Starcoder: may the source be with you! arXiv preprint arXiv:2305.06161, 2023b.

T. Li, W.-L. Chiang, E. Frick, L. Dunlap, B. Zhu, J. E. Gonzalez, and I. Stoica. From live data to high-quality benchmarks: The arena-hard pipeline, April 2024. URL https://lmsys.org/blog/2024-04-19-arena-hard/.

J. Liu, C. S. Xia, Y. Wang, and L. Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In Thirty-seventh Conference on Neural Information Processing Systems, 2023a. URL https://openreview.net/forum?id=1qvx610Cu7.

T. Liu, C. Xu, and J. McAuley. Repobench: Benchmarking repository-level code auto-completion systems. In The Twelfth International Conference on Learning Representations, 2023b.

X. Liu, X. Lei, S. Wang, Y. Huang, Z. Feng, B. Wen, J. Cheng, P. Ke, Y. Xu, W. L. Tam, X. Zhang, L. Sun, H. Wang, J. Zhang, M. Huang, Y. Dong, and J. Tang. Alignbench: Benchmarking chinese alignment of large language models. CoRR, abs/2311.18743, 2023c. doi: 10.48550/ARXIV.2311.18743. URL https://doi.org/10.48550/arXiv.2311.18743.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization, 2019.

A. Lozhkov, R. Li, L. B. Allal, F. Cassano, J. Lamy-Poirier, N. Tazi, A. Tang, D. Pykhtar, J. Liu, Y. Wei, et al. Starcoder 2 and the stack v2: The next generation. arXiv preprint arXiv:2402.19173, 2024.

MAA. American invitational mathematics examination - aime. American Invitational Mathematics Examination - AIME 2024, 2024. URL https://maa.org/math-competitions/american-invitational-mathematics-examination-aime.

Meta. Introducing meta llama 3: The most capable openly available llm to date. https://ai.meta.com/blog/meta-llama-3/, April 2024.

MistralAI. Codestral. https://mistral.ai/news/codestral/, 2024. Accessed: 2024-05-29.

Netmind.AI. Odyssey-math. https://github.com/protagolabs/odyssey-math/tree/main, 2024. Accessed: April 22, 2024.

OpenAI. Gpt-4 technical report, 2023.

B. Peng, J. Quesnelle, H. Fan, and E. Shippole. Yarn: Efficient context window extension of large language models. arXiv preprint arXiv:2309.00071, 2023.

M. Reid, N. Savinov, D. Teplyashin, D. Lepikhin, T. Lillicrap, J.-b. Alayrac, R. Soricut, A. Lazaridou, O. Firat, J. Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv preprint arXiv:2403.05530, 2024.

B. Roziere, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, T. Remez, J. Rapin, et al. Code llama: Open foundation models for code. arXiv preprint arXiv:2308.12950, 2023.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, M. Zhang, Y. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024.

Q. Shi, M. Tang, K. Narasimhan, and S. Yao. Can language models solve olympiad programming? arXiv preprint arXiv:2404.10952, 2024.

M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. V. Le, E. H. Chi, D. Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. arXiv preprint arXiv:2210.09261, 2022.

H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288, 2023.

L. Xu, H. Hu, X. Zhang, L. Li, C. Cao, Y. Li, Y. Xu, K. Sun, D. Yu, C. Yu, Y. Tian, Q. Dong, W. Liu, B. Shi, Y. Cui, J. Li, J. Zeng, R. Wang, W. Xie, Y. Li, Y. Patterson, Z. Tian, Y. Zhang, H. Zhou, S. Liu, Z. Zhao, Q. Zhao, C. Yue, X. Zhang, Z. Yang, K. Richardson, and Z. Lan. CLUE: A chinese language understanding evaluation benchmark. In D. Scott, N. Bel, and C. Zong, editors, Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020, pages 4762–4772. International Committee on Computational Linguistics, 2020. doi: 10.18653/V1/2020.COLING-MAIN.419. URL https://doi.org/10.18653/v1/2020.coling-main.419.

L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

W. Zhong, R. Cui, Y. Guo, Y. Liang, S. Lu, Y. Wang, A. Saied, W. Chen, and N. Duan. AGIEval: A human-centric benchmark for evaluating foundation models. CoRR, abs/2304.06364, 2023. doi: 10.48550/arXiv.2304.06364. URL https://doi.org/10.48550/arXiv.2304.06364.

A. 支持的编程语言

ABAP, ActionScript, Ada, Agda, AGS Script, Alloy, AmbientTalk, AMD GPU, AMPL, ANSYS Parametric Design Language, ANTLR, Apache Configuration, APL, AppleScript, Arc, Arduino, ASP, AspectJ, Assembly, Asymptote, Augeas, AutoHotkey, AutoIt, AWK, BC, Berry, BitBake, BlitzBasic, BlitzMax, Bluespec, BNF, Boo, Boogie, Brainfuck, BrightScript, Bro, BST, C, C#, C2HS Haskell, CADL, CapDL, Ceylon, Chapel, ChucK, Cirru, Click, Clojure, CMake, COBOL, COBOLFree, CoffeeScript, ColdFusion CFC, Common Lisp, C++, Crystal, Csound, Csound Score, CSS, CUDA, Cypher, Cython, Darcs Patch, Dart, DASM16, Debian Control File, DeviceTree, Diff, DM, Docker, Dockerfile, Dylan, EBNF, eC, Eiffel, Elixir, Elm, ELPi, Emacs Lisp, EmberScript, Erlang, Execline, F#, Factor, Fancy, Fantom, Felix, Fennel, Fish, Flux, Fortran, Fortran Fixed Form, FoxPro, FreeFem, FreeMarker, F*, Futhark, G-Code, GAP, GAS, GDScript, Genshi, Gentoo Ebuild, Gentoo Eclass, Gettext Catalog, GLSL, Glyph, Gnuplot, Go, Gosu, Grace, Gradle, Grammatical Framework, GraphQL, Graphviz DOT, Groff, Groovy, Groovy Server Pages, GSQL, Handlebars, Haskell, Haxe, HCL, HLSL, HTML, HTML Django, HTML ERB, HTML PHP, HTTP, Hy, Idris, IGOR Pro, Inform 6 Template, Inno Setup, Io, Isabelle, J, Jade, JAGS, Jasmin, Java, Java Server Pages, JavaScript, JavaScript MozPreproc, JCL, JFlex, JSON, JSONiq, JSX, Julia, Jupyter Notebook, K, Kconfig, Koka, Kotlin, KRL, Lean, Less, Lex, LFE, Lighttpd Configuration File, LilyPond, Limbo, Linker Script, Liquid, Literate Agda, Literate CoffeeScript, LLVM, Logtalk, LSL, Lua, M4, Makefile, Mako, Mason, MATLAB, Maxima, Meson, Metal, MiniScript, Mirah, Mizar, Modelica, Modula-2, Monkey, MooCode, MoonScript, Mosel, MQL, MUF, MuPAD, NASM, NCL, NetLinx, Nginx Configuration File, Nimrod, Ninja, Nit, Nix, NSIS, Nu, NuSMV, Objdump, Objective-C, Objective-C++, OCaml, Octave, Odin, OMG Interface Definition Language, ooc, Opa, OpenCL, OpenEdge ABL, OpenSCAD, Ox, Oz, Papyrus, Parrot Internal Representation, Pascal, PAWN, PEG, Perl, Perl 6, PHP, Pike, PkgConfig, POD, Pony, POV-Ray, PowerShell, Praat, Processing, Propeller Spin, Protocol Buffer, Pug, Puppet, PureBasic, PureScript, Python, Q, QML, QVTO, R, Racket, Ragel in Ruby Host, RAML, RConsole, Rd, REALbasic, ReasonML, Red, RenderScript, Ren'Py, REXX, RHTML, Ride, Robot Framework, Rouge, Ruby, Rust, S, Sage, SARL, SAS, Sass, Scala, Scheme, Scilab, SCSS, Self, Shell, ShExC, Sieve, Silver, Singularity, Slim, Smali, Smarty, Smithy, SMT, Solidity, SourcePawn, SPARQL, SQF, SQL, Squirrel, Stan, Standard ML, Stata, Stylus, SuperCollider, Swift, SWIG, SystemVerilog, Tcl, Tcsh, Tea, Terminfo, TeX, Thrift, Transact-SQL, Treetop, Turing, Twig, TypeScript, TypoScript, Unity3D Asset, Uno, UnrealScript, UrWeb, USD, Vala, VBScript, VCL, Velocity, Verilog, VHDL, VimL, Visual Basic, Vue, WebAssembly, Web IDL, Whiley, X10, XBase, XC, XML, XML Lasso, XQuery, XS, XSLT, Xtend, Xtlang, YANG, Zeek, Zephir, Zig, Zimpl