NYONIC TECHNICAL REPORT

Junfeng Tian, Rui Wang, Cong Li, Yudong Zhou, Jun Liu, Jun Wang, Lu Zhang
nyonic.ai

Abstract

This report details the development and key achievements of our latest language model designed for custom large language models. The advancements introduced include a novel Online Data Scheduler that supports flexible training data adjustments and curriculum learning. The model's architecture is fortified with state-of-the-art techniques such as Rotary Positional Embeddings, QK-LayerNorm, and a specially crafted multilingual tokenizer to enhance stability and performance. Moreover, our robust training framework incorporates advanced monitoring and rapid recovery features to ensure optimal efficiency. Our Wonton 7B model has demonstrated competitive performance on a range of multilingual and English benchmarks. Future developments will prioritize narrowing the performance gap with more extensively trained models, thereby enhancing the model's real-world efficacy and adaptability.

GitHub: `https://github.com/nyonicai/nyonic-public`

1 Introduction

This report details the development and release of our pre-trained 7B base model checkpoint and one fine-tuned model for chat. As we plan to continuously train and improve our model, we will release more base model checkpoints in the future, as well as fine-tuned models for different use cases.

Our training infrastructure is mainly built on open source PyTorch framework, incorporating various newly developed features, e.g., flash attention([Dao et al., 2022]), RoPE(Su et al. [2021]), QK-LayerNorm([Dehghani et al., 2023]), etc.

The main contributions of this work can be summarized as follows:

- We build an online data scheduler and multiplexer for flexible training data mixing.
- We train our own multilingual tokenizer to incorporate several widely-spoken languages.
- We monitor various intermediate metrics during the training and apply several normalization and regularization techniques to increase the stability.
- We consolidate our infrastructure to speed up the resuming after interruptions.
- We fine tuned the pre-trained model with open SFT datasets, as well as our private datasets.
- We enabled a few inference scenarios and provided deployment methods.

This report aims to provide a comprehensive overview of LLM training, development, SFT, inference, and deployment technology, which can benefit the community in the creation of more LLMs and the development of a wide range of real-world applications.

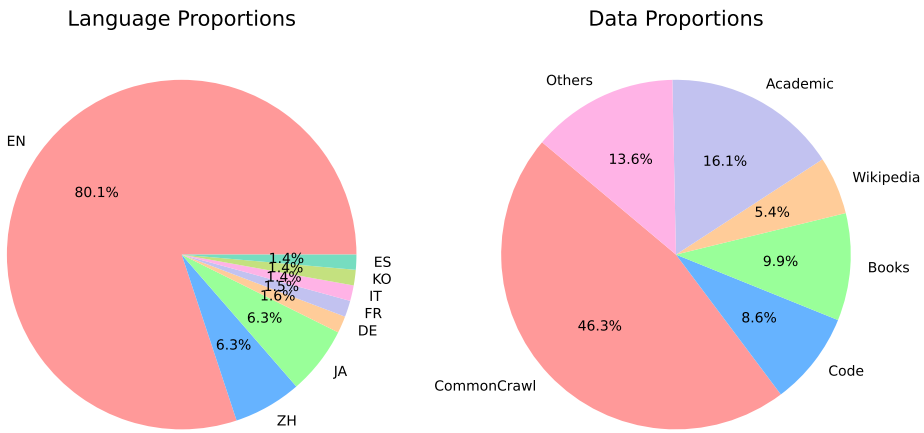

Figure 1: Language and data proportions in the training set.

2 Training

2.1 Data

The efficacy and performance of language models are significantly influenced by the quality and diversity of the data used in training. An effective pretraining dataset must encompass a broad spectrum of types, domains, and tasks to ensure comprehensive linguistic coverage.

Our curated dataset is specifically designed to meet these criteria, incorporating sources such as the Common Crawl, books, Wikipedia, code, and academic papers. Moreover, our dataset is multilingual, encompassing English, Chinese, German, French, Italian, Spanish, Japanese, and Korean, and with a significant portion of the data being in English. Figure 1 illustrates the mixture of data and the percentage they represent in the training set.

2.2 Data Engineering

In providing customized large model solutions for specific industries, the traditional approach involves offline conversion of data into token indices to create corresponding data indices. However, recent studies ([Xie et al., 2023]) have shown that compared to uniformly sampling data sources, carefully selecting training data and undergoing continual pre-training can significantly enhance model accuracy, further validating the effectiveness of data scheduling strategies.

Nevertheless, these methods still rely on offline data processing and lack mechanisms for online adjustments and real-time feedback during the training process. Restarting training not only consumes substantial computational resources but also does not necessarily yield better training outcomes.

To address this issue, we propose an innovative **Online Data Scheduler**, which integrates an online data stream with a data scheduler. The Online Data Scheduler offers several benefits:

- It enables flexible implementation of data mixing since it eliminates the need for offline conversion of data into token indices. By utilizing multiple workers in the data loader, it can concurrently process data and perform model training, without reducing training efficiency.
- The system allows for training flexibility across different stages through curriculum learning. The model is able to avoid wasting time on simple data that it has already mastered, focusing more attention on data that is challenging to learn. This

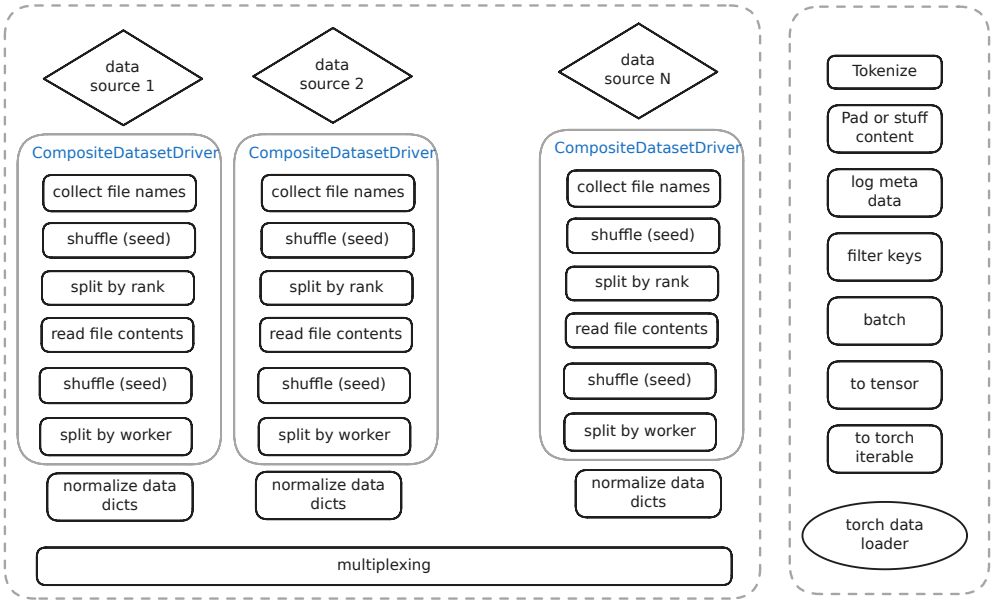

Figure 2: The data flow architecture of the Online Data Scheduler.

targeted training approach can accelerate model convergence, reduce unnecessary computations, and thereby save time and resources.

- Real-time feedback is possible. The scheduler can dynamically adjust data ratios based on the model's real-time training loss, allowing for immediate adjustments based on feedback.
- The model is capable of online learning, enabling it to adapt to new data. This allows the model to learn within a continuously changing data stream, aligning more closely with the ongoing learning processes observed in the real world.

The data flow for training large models is efficiently divided into three stages.

1. **Data Preparation** involves collecting, shuffling, and distributing file names across different ranks, with each rank responsible for reading file contents.
2. In the **Data Processing** stage, data is further shuffled, split among workers, normalized, and tokenized, ensuring consistency and readiness for model input.
3. The final stage, **Batch Preparation**, involves padding the data to uniform lengths, logging metadata, and organizing it into batches that are converted to tensors and torch iterables for training.

This streamlined process ensures that data is methodically prepared and optimized for effective model training. In particular, we present the key components of our online data scheduler, which facilitate the integration of data from various sources in a configurable manner.

- **Multiplexing**: To facilitate data mixing from various sources in a configurable manner, a Multiplexer is integrated into the real data loading pipeline. This allows for the combination or multiplexing of processed data streams from different sources into a single stream, with configurable mixture ratios.
- **Content Stuffing**: Following [Brown et al., 2020], during training, we consistently use sequences that fill the entire 2048 token context window. When individual documents are shorter than 2048 tokens, multiple documents are packed into a single sequence to enhance computational efficiency. These sequences are further augmented with additional length information, which enables them to be processed in the CUDA kernel without the need for sequence-specific masking.

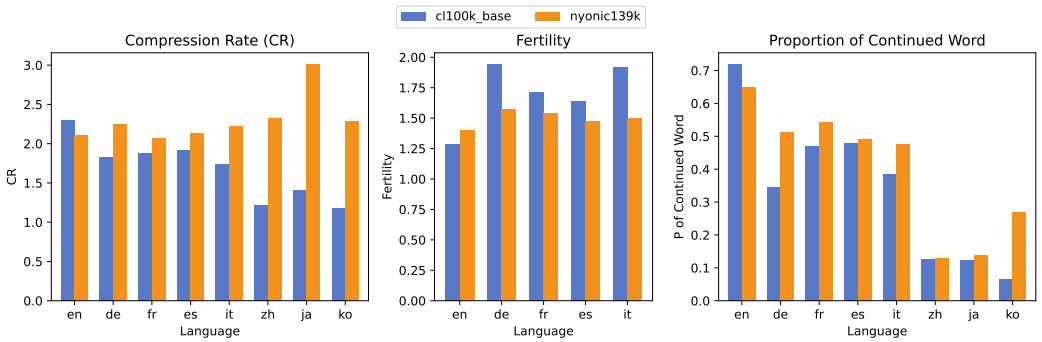

Figure 3: Comparative analysis of multilingual tokenization metrics across different languages.

- **Data Resuming**: Given that the list of files assigned to each rank remains constant for each dataset, we choose to save the processing status of files on each rank. This approach enables us to bypass files that have already been processed when resuming training. Currently, our focus is on maintaining lightweight control over the resumption process at the file level, rather than managing it with finer granularity at the record level.

2.3 Tokenization

The design of vocabulary plays a significant role in the training efficiency and downstream task performance of multilingual large language models. We employ the byte-pair encoding (BPE) algorithm, as described by [Bostrom and Durrett, 2020], using the SentencePiece implementation ([Kudo and Richardson, 2018]). To optimize performance with datasets containing abundant whitespace, such as code, we assign one special token for each sequence of whitespace ranging from 1 to 24 characters. Additionally, in line with [Touvron et al., 2023a,b], we have segmented numbers into individual digits.

Selecting an optimal vocabulary size is of paramount importance for tokenizers, as a size that is too small may impede transformer computations, while a size that is too large may hinder the model's capacity to derive meaning from sequences. We have identified the most commonly used languages by our customers, including English, Simplified Chinese, Traditional Chinese, German, French, Italian, Spanish, Japanese, and Korean. Our data sources encompass Common Crawl, Wiki, and various coding repositories. The final vocabulary size of the tokenizer, as determined by extensive experimentation, is approximately 139,000.

In Figure 3, we present a comprehensive evaluation of the multilingual tokenizer. This detailed comparison examines several critical metrics that are essential for understanding tokenizer efficiency. These metrics include the compression rate, which measures the compactness of tokenized data; fertility, which assesses the breakdown of words into subwords; and the proportion of continued words, which evaluates the segmentation consistency across various languages. The results of our analysis demonstrate that our tokenizer achieves superior compression efficiency in most languages. This indicates that it has the potential to significantly reduce operational costs by using fewer tokens to transmit equivalent or greater amounts of information.

2.4 Architecture Highlights

Our model, Wonton 7B, is built on a transformer architecture ([Vaswani et al., 2017]), with its main parameters summarized in Table 1. Compared to GPT3 ([Brown et al., 2020]), Wonton 7B incorporates a number of enhancements, which are outlined below.

Rotary Positional Embeddings (RoPE) ([Su et al., 2021]): Unlike traditional absolute positional embeddings, we implement rotary positional embeddings in each layer. RoPE has

gained widespread acceptance and has shown effectiveness in modern large language models ([Touvron et al., 2023b, Jiang et al., 2023, Bai et al., 2023]).

Pre-Norm & QK-LayerNorm : We employ pre-normalization, a prevalent strategy that enhances training stability over post-normalization ([Xiong et al., 2020]). Traditional layer normalization ([Ba et al., 2016]) is used, and it is also applied to the queries and keys before the dot-product attention process ([Dehghani et al., 2023]) to prevent excessively large values in attention logits.

Max-z Loss : Building on the auxiliary z-loss from PaLM ([Chowdhery et al., 2022]) and the max-z loss concept ([Yang et al., 2023]), we employ these techniques to regulate logits values. This ensures more stable training and robust inference across various hyperparameters. Max-z loss, in particular, offers greater effectiveness with a reduced memory demand, making it our default choice for model training.

2.5 Training Hyperparameters

The models are trained using the AdamW optimizer ([Loshchilov and Hutter, 2017]), with the hyperparameters set at $\beta_1 = 0.9$ and $\beta_2 = 0.95$. We implement a weight decay of 0.1 and apply gradient clipping at 1.0. Following the approach described in [Brown et al., 2020], we utilize a cosine decay schedule with warmup. The warmup period consists of $2,000$ steps, followed by a decay to 10% of the maximum learning rate.

Following the approach described in [Radford et al., 2019], we initialize the linear and embedding weights using a normal distribution and scale the up-projection layers at initialization by a factor of $1/\sqrt{N}$ where N represents the number of layers. This scaling is designed to optimize the variance of the outputs across different layers, facilitating more stable training.

Parameter	Value
params	6.7B
dimension	4,096
n heads	32
n layers	32
context len	2,048
vocab size	139,776
learning rate	$3.0e^{-4}$
batch size	1M

Table 1: Model architecture.

2.6 Training Infrastructures

For training the Nyonic-7B model, we utilize PyTorch ([Paszke et al., 2019]) and DeepSpeed ZeRO2 ([Rasley et al., 2020]), distributing the optimizers states across 128 NVIDIA A800 GPUs interconnected via InfiniBand. To enhance training efficiency, we incorporate FlashAttention ([Dao et al., 2022]) and xformers ([Lefaudeux et al., 2022]), achieving a training speed of approximately 3,000 tokens per GPU per second. All models are trained using BFloat16 mixed precision to ensure training stability.

We introduce additional metrics to monitor the model training process. By integrating these metrics, we aim to gain deeper insights into our model's internal dynamics and improve its performance and stability across a range of tasks.

- **Max Attention Logits**: This measures the maximum value of the attention logits within each attention block, providing insight into the attention distribution's extremities.
- **Mean Query Norm**: By averaging the norms of the query vectors, this metric assesses the average strength or magnitude of the queries before they interact with the keys, which can indicate the overall query signal strength across different layers.
- **Output Logit Mean (Pre-Softmax)**: This metric calculates the mean of the output logits just before the softmax activation, which can help in understanding the pre-activation distribution of the logits.
- **Root Mean Square of Gradient of the First Layer of MLP**: Monitoring the root mean square of the gradient for the first layer of the multi-layer perceptron

Task	Lambada (5153)	WinoGrande (1267)	HellaSwag (10042)	PIQA (1838)	RACE (1045)	BoolQ (3270)
Metric	ppl ↓	acc	acc	acc	acc	acc
Mistral 7B	3.7771	0.7364	0.6124	0.8074	0.4096	0.8376
Pythia 7B	6.9210	0.6109	0.4811	0.7514	0.3684	0.6358
Wonton 7B	6.2973	0.6456	0.4777	0.7503	0.3684	0.6685

Table 2: Performance of Wonton 7B and other open source models on the benchmarks. All models were re-evaluated on all metrics with our evaluation pipeline for accurate comparison.

(MLP) within each block can provide early indications of potential issues with gradient vanishing or exploding, which are critical for maintaining training stability.

- **Block Output RMS**: The root mean square (RMS) of each block's output offers a measure of the output signal's consistency and variability, which can be crucial for diagnosing model behavior during both training and inference phases.

Our training framework supports minute-level training resumption thanks to our implemented online data scheduler. Additionally, it accommodates training resumption from various distributing configurations to adapt to dynamic changes in the computing cluster.

2.7 Experiment Results

Basic Capabilities We select the most common evaluation sets in each category as follows:

- **Lambada** ([Paperno et al., 2016]): The LAMBADA benchmark features around 10,000 passages from the BooksCorpus, where participants are tasked with predicting a missing word in the contextually rich final sentence of each passage.
- **WinoGrande** ([Sakaguchi et al., 2019]): A large-scale dataset designed to challenge models with complex commonsense reasoning through refined pronoun disambiguation tasks.
- **HellaSwag** (Zellers et al. [2019]): This dataset pushes the limits of models' commonsense reasoning capabilities by requiring them to complete scenarios in a contextually appropriate way.
- **PIQA** ([Bisk et al., 2020]): Introduces a benchmark for physical commonsense reasoning, testing models' abilities to understand and predict physical interactions in diverse settings.
- **RACE** ([Lai et al., 2017]): A robust dataset providing a range of reading comprehension questions based on academic texts from middle and high school levels.
- **BoolQ** ([Clark et al., 2019]): Focuses on yes/no question answering, with questions naturally derived from Google searches, testing models' abilities to interpret and evaluate straightforward queries against provided texts.

Table 2 lists the performance of Wonton 7B and the other open source models on the benchmarks. Wonton 7B consistently outperforms Pythia 7B across various metrics, indicating more efficient use of its training data and better optimization strategies. However, when compared to Mistral 7B, which was trained on a significantly larger corpus, Wonton 7B still lags behind, especially in tasks that require a deep understanding of language nuances and complex reasoning. These results underscore the effectiveness of Wonton 7B against comparable models like Pythia but also highlight the challenges it faces in reaching the performance levels of more extensively trained models such as Mistral 7B. This comparison not only reflects Wonton 7B's strengths but also points towards potential areas for future improvement to bridge the gap with top-tier models.

Multilingual Understanding To assess the capabilities of Wonton 7B, we utilized various benchmark datasets that challenge models to demonstrate comprehension and reasoning across multiple languages. The datasets employed include:

Task / Language	Average	Belebele							
		English (900)	German (900)	French (900)	Italian (900)	Spanish (900)	Chinese (900)	Japanese (900)	Korean (900)
Mistral 7B	0.4073	0.4544	0.4278	0.4222	0.3900	0.4033	0.4067	0.3656	0.3889
Pythia 7B	0.3384	0.3656	0.3267	0.3644	0.3278	0.3300	0.3578	0.3156	0.3189
Wonton 7B	0.3517	0.3822	0.3500	0.3689	0.3367	0.3467	0.3567	0.3533	0.3189

Table 3: Performance comparison of Wonton 7B with other open-source models on the Belebele benchmarks.

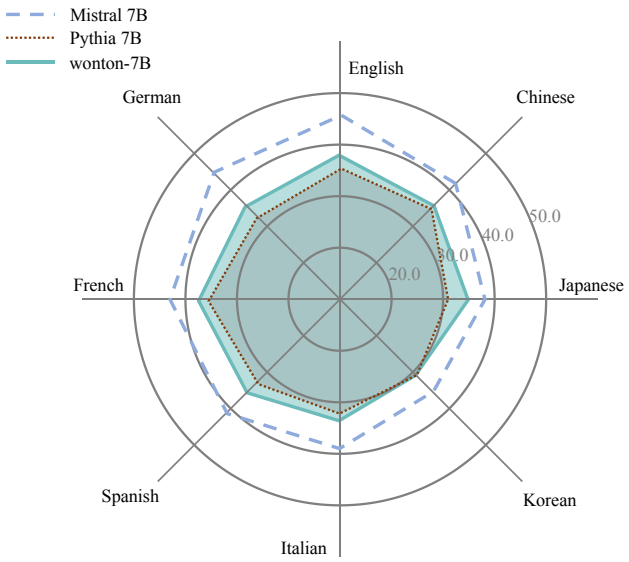

Figure 4: Performance comparison of Wonton 7B with other open-source models on the Belebele benchmarks.

- **Belebele** ([Bandarkar et al., 2023]): Tests models on tasks across eight different languages—English, German, French, Italian, Spanish, Chinese, Japanese, and Korean.

- **XNLI** ([Conneau et al., 2018]): Focuses on natural language inference in three languages, assessing how well models understand and interpret text across language barriers.

- **XStoryCloze** ([Lin et al., 2022]): Measures models' ability to logically complete stories in English, Spanish, and Chinese, testing narrative understanding and commonsense reasoning.

- **XWinograd** ([Tikhonov and Ryabinin, 2021]): A multilingual version of the Winograd schema challenge that tests models on resolving pronoun disambiguation in English, French, and Japanese.

In Table 3, Wonton 7B showed competitive results across the Belebele benchmarks. It consistently outperformed Pythia 7B, indicating more efficient use of its training data and better optimization strategies. The results in Table 4 are consistent with Belebele, reflecting improvements in its ability to handle complex linguistic constructions and contexts across different languages.

Benchmark		XNLI			XStoryCloze			XWinograd		
Language	Average	English (5010)	German (5010)	Chinese (5010)	English (1511)	Spanish (1511)	Chinese (1511)	English (2325)	French (83)	Japanese (959)
Mistral 7B	0.6581	0.5731	0.5026	0.3737	0.7882	0.6903	0.6334	0.8865	0.7470	0.7278
Pythia 7B	0.5841	0.5411	0.4605	0.3387	0.7055	0.5989	0.5381	0.8245	0.6627	0.5871
Wonton 7B	0.6153	0.5427	0.4884	0.3417	0.7075	0.6453	0.6056	0.8417	0.6627	0.7018

Table 4: Results of Wonton 7B across XNLI, XStoryCloze, and XWinograd benchmarks.

3 Inference and Deployment

3.1 Inference

In the development of our foundation models, we have implemented and tested several inference frameworks to optimize performance across different environments. Our primary focus was on leveraging the flexibility and robustness of PyTorch for native inference, which allowed us to maintain a consistent development and deployment pipeline.

To expand our model's accessibility and ease of integration, we adapted our models for use with the Hugging Face ecosystem. This adaptation enabled seamless inference capabilities and provided our team with robust, community-driven tools for model management and deployment. The Hugging Face integration also allowed us to leverage their extensive libraries and APIs, which significantly accelerated our development process. This work is yet to be published.

Additionally, we utilized NVIDIA TensorRT[1] for optimizing our models for high-performance inference on NVIDIA GPUs. This was crucial for scenarios demanding low latency and high throughput, such as real-time applications and large-scale deployments. TensorRT helped in dramatically reducing inference times while preserving the accuracy and reliability of our models.

3.2 Deployment

For deployment, we mostly chose Aliyun Elastic Algorithm Service (EAS)[2] to host and manage our models. This platform offered a scalable and secure environment that supported our operational needs, including automatic scaling, performance monitoring, and robust security measures. The use of Aliyun EAS facilitated a straightforward deployment process and enabled efficient management of model resources in a production environment.

Through these diverse technologies, we achieved a flexible, efficient, and scalable foundation model infrastructure that supports a wide range of applications and meets various business requirements.

4 Specialized Model for Chat

4.1 Data

We finetune Wonton 7B pretrained model using open source and industry datasets created by Nyonic. All The datasets are in English language. It includes:

- **Instruction Tuning with GPT-4** ([Peng et al., 2023]): GPT-4-LLM, which aims to share data generated by GPT-4 for building an instruction-following LLMs with supervised learning and reinforcement learning.
- **Flan V2** ([Wei et al., 2022]): It aims to 'Finetuned Language Models are Zero-Shot Learners'.
- **Tulu V2/Cot** ([Ivison et al., 2023]): TÜLU, open resources for instruction tuning have developed quickly, from better base models to new finetuning techniques. TÜLU

[1] `https://github.com/NVIDIA/TensorRT`

[2] `https://www.alibabacloud.com/help/en/pai/user-guide/eas-model-serving/`

2, a suite of improved TÜLU models for advancing the understanding and best practices of adapting pretrained language models to downstream tasks and user preferences.

- **Open Assitant** ([Köpf et al., 2023]): It comes in lots of conversations to democratize large language model alignment, especially multi-turn conversations.
- **Automobile Datasets**: It is created by Nyonic and aims to contain knowledge of Automobile industry.

4.2 SFT Hypterparameters

We conduct the full finetuning training based on the Nyonic pretrained model. It leverages the same training framework as the pretraining did. The prompt template is `<s> [INST] Question [/INST] Answer </s>`. We implement the SFT loss function by masking the loss question part, to only accumulate the losses of the answer part.

The optimizer is similar to the pretraining stage described at 2.5. The initial learning rate is 2e−5. We employ a 512 global batch size for smoothing training losses. We conduct 1-3 training epochs.

4.3 Experiment Results

Basic Capabilities Table 2 shows the performance of Nyonic pretrained 7B model. The fine tuned model has a comparable results on such benchmarks.

Independent Scoring We employs ChatGPT to conduct independent scoring by assigning scores ranging from 1 to10, based on the quality of model responses. The fine tuned model had achieved average 2.6 score higher than pretrained model.

5 Conclusion

We have summarized our technical development of our large language models. As we mentioned at the beginning, we will continuously train and release more model checkpoints in the future, as well as fine-tuned models for different scenarios. We hope the community can benefit from our experience and we sincerely welcome further development based on our models and collaborations on various downstream applications.

Acknowledgements

We would like to thank Johannes Otterbach, Sami Jaghouar, Manas Gaur, Zengyu Yan, Jing Su, and Sören Erichsen for their various contributions to this work.

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report, 2023.

Lucas Bandarkar, Davis Liang, Benjamin Muller, Mikel Artetxe, Satya Narayan Shukla, Donald Husa, Naman Goyal, Abhinandan Krishnan, Luke Zettlemoyer, and Madian

Khabsa. The belebele benchmark: a parallel reading comprehension dataset in 122 language variants, 2023.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *AAAI*, pages 7432–7439. AAAI Press, 2020. doi: 10.1609/aaai.v34i05.6239.

Kaj Bostrom and Greg Durrett. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online, November 2020. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. PaLM: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *NAACL-HLT*, pages 2924–2936, 2019.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. XNLI: Evaluating cross-lingual sentence representations. In *EMNLP*, pages 2475–2485, Brussels, Belgium, October-November 2018.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, 2022.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd van Steenkiste, Gamaleldin F. Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Patrick Collier, Alexey Gritsenko, Vighnesh Birodkar, Cristina Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Pavetić, Dustin Tran, Thomas Kipf, Mario Lučić, Xiaohua Zhai, Daniel Keysers, Jeremiah Harmsen, and Neil Houlsby. Scaling vision transformers to 22 billion parameters, 2023.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. Camels in a changing climate: Enhancing lm adaptation with tulu 2, 2023.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Eduardo Blanco and Wei Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. Openassistant conversations – democratizing large language model alignment, 2023.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1082.

Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. xformers: A modular and hackable transformer modelling library. `https://github.com/facebookresearch/xformers`, 2022.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona Diab, Veselin Stoyanov, and Xian Li. Few-shot learning with multilingual language models, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *ACL*, 2016.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*, 2023.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 3505–3506, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3406703.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

Alexey Tikhonov and Max Ryabinin. It's All in the Heads: Using Attention Heads as a Baseline for Cross-Lingual Transfer in Commonsense Reasoning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3534–3546, Online, August 2021.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. 2023b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining, 2023.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *ICML*, pages 10524–10533, 2020.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Haizhou Zhao, Hang Xu, Haoze Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, Juntao Dai, Kun Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Peidong Guo, Ruiyang Sun, Tao Zhang, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin Men, Xin Yu, Xuehai Pan, Yanjun Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. Baichuan 2: Open large-scale language models. Technical report, Baichuan Inc., 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *ACL*, pages 4791–4800, 2019.