Week 5: Introducing Large Language Models to a General Audience

BEH Chuen Yang

Abstract

In this report, we summarize the salient details of ”Deep Dive into LLMs like ChatGPT” by Karpathy (2025), and present them in a manner suitable for a non-technical, but otherwise well-educated general audience. It is hoped this report serves as a friendly yet insightful introduction to the topic of Language Modelling, and conveys the important intuitions and concepts that underlie the workings of Large Language Models (LLMs) such as ChatGPT.

1 Introduction

Ever since GPT-3 (Brown et al. (2020)) and ChatGPT OpenAI (2022) first demonstrated the remarkable capabilities of Large Language Models (LLMs), there has been a surge of interest in understanding how these models work and what they can do. Yet as applications of LLMs transform various aspects of our daily lives, many people still do not fully grasp the underlying principles of these models, or how to effectively interact with them.

This report, which is overwhelmingly based on Karpathy (2025), hopes to be a good starting point for those who wish to learn more about LLMs, by providing a high-level overview of the key concepts and techniques involved in their development and use.

1.1 What is an LLM?

While there is no universally accepted definition of what an LLM is (Zhao et al. (2025)), a few things remain generally agreed upon *in the modern context* (Zhao et al. (2025); Sanderson (2025); Karpathy (2025)):

- LLMs are *neural networks* trained on massive amounts of text data in order to learn the statistical patterns of language.
- LLMs typically have *billions or even trillions* of parameters (internal variables that the model tweaks during training).
- Modern LLMs typically have *diverse capabilities*, (Brown et al. (2020)) such as generating coherent text, answering questions, following instructions, and even engaging in conversations.
- Many modern LLMs are trained with *some variant of the Transformer architecture* (Vaswani et al. (2017)), which possesses a powerful attention mechanism that allows the model to determine what it should focus on in the input text. [1] [2]

[1] There are some notable exceptions such as MAMBA (State Space Model) (Gu & Dao (2024)) and RWKV (Recurrent Neural Network) (Peng et al. (2023)). Aside from RWKV which does not suffer from the same limitations, RNNs are rarely used in modern LLMs since it is difficult to parallelize them.

[2] For a great video about the Transformer architecture, see Sanderson (2025).

2 How Do I Use LLMs?

Perhaps the most common way to interact with LLMs is through a chat interface, where users can type in prompts and receive responses in natural language. (OpenAI (2022); Anthropic (2025); DeepSeek-AI et al. (2025b))

However, chatting is just a special case of a more general interaction paradigm, *prompt engineering* (Karpathy (2025)), where users can provide the LLM with *carefully crafted* inputs to elicit specific outputs.

Here are some common interaction techniques mentioned by Karpathy (2025) that fall under this umbrella: [3] [4]

Technique	Description
Zero-shot prompting	Users provide a prompt without any examples, and the model generates a response based on its pre-trained knowledge.
Few-shot prompting	Users provide a few examples of the desired input-output pairs within the prompt itself. The model then uses these examples to generate responses that follow the same pattern.
Conversational prompting	Users structure their prompts as if they are having a conversation with an AI assistant, often using special tags or markers to distinguish between user inputs and expected assistant responses.
Chain of Thought (CoT) prompting (Wei et al. (2023))	Instead of directly asking for an answer, users prompt the model to generate intermediate reasoning steps, which can significantly improve performance on tasks requiring complex logical deduction.

Table 1: Some Prompt Engineering Techniques

2.1 Where Can We Find Off-the-Shelf LLMs?

LLMs can typically be found in four main categories: Proprietary LLMs, Open-source LLMs, Inference Providers, and Local LLMs. Which one is the most suitable depends on the user's needs and resources.

Now, let us take a closer look at how language models in general are created.[5]

3 Pre-Training: Creating a Base Model

The key capability of an LLM is its ability to generate coherent and contextually relevant text, which is achieved by learning the statistical patterns of language from vast amounts of text data. This process can be achieved via *self-supervised learning* (Karpathy (2025)), where, for example, the model could learn to predict the next word in a sentence given the previous words.

A simple analogy for this process is the act of copying a textbook in order to learn its contents. While arguably not very efficient for humans, this method allows the LLM to learn the structure and patterns of language completely from scratch.

[3] As Karpathy (2025) demonstrates, not affording LLMs the opportunity to think step-by-step within the context window (as Chain-of-Thought does) can lead to poor performance on tasks that require reasoning.

[4] However, you can also use LLMs in other ways, for example, by integrating them into applications e.g. via Anthropic's MCP (Anthropic (2025)), or augmenting them with external tools like ChatGPT's search functionality (OpenAI (2022)).

[5] Note that our description considers only the seq2seq paradigm that is typically used in modern LLMs (Sutskever et al. (2014)). There are other paradigms beyond the scope of this report, such as diffusion (Nie et al. (2025)).

Type	Description
Proprietary LLMs	Among the most powerful LLMs. Developed and maintained by commercial entities, such as OpenAI's GPT series (OpenAI (2022)), Anthropic's Claude (Anthropic (2025)), and Google's Gemini (et al (2025)), typically for profit. Access is typically through pay-per-token APIs.
Open-source LLMs	Developed by various organizations and individuals, with their underlying structures and learned weights publicly available. Examples include Meta's LLaMA models (Touvron et al. (2023)) and Deepseek's models (DeepSeek-AI et al. (2025b;a)). Platforms like HuggingFace (HuggingFace (2025)) typically serve as repositories for these models.
Inference Providers	Companies that host LLMs and provide access via APIs, allowing users to run inference without needing to manage the underlying infrastructure. Examples include Hyperbolic (Hyperbolic Team (2024)) and other cloud-based services.
Local LLMs	Smaller, open-source LLMs that can be run on personal computers or local servers. Examples include Qwen3 (Yang et al. (2025)). These models can be run using tools like LM Studio (Element Labs (2025)) for local inference.

Table 2: LLMs and Where to Find Them

3.1 Data Collection & Preprocessing

The first step in creating an LLM is to *collect and tokenize* a large corpus of text data.

In the collection stage, data is typically scoured from the Internet (due to its sheer volume and ease of access), after which enough text is collected to fill tens of terabytes of disk space (Karpathy (2025); Penedo et al. (2024)).

The data is then *preprocessed* (Karpathy (2025)) to ensure that it is clean, diverse, and of high quality. A sample set of procedures is given below:

- Removing URLs with a blacklist (such as Université Toulouse 1 (2025)). Such URLs may contain adult content, malware, or other undesirable material.
- Parsing raw HTML to extract the core textual content of web pages, as the aim is not to learn HTML syntax but rather the human-readable contents of the website.
- Removing duplicate documents.
- Removing Personally Identifiable Information (PII) such as names and addresses. This is to protect the privacy of individuals whose data may have been scraped.
- (Optional) Filter documents to retain only specific languages if need be.

3.2 Tokenization

Since neural networks do not understand language symbols innately, a natural next step is to convert text data into numbers that the model can process. This process is called **tokenization** (Karpathy (2025)), where the text is broken down into smaller units called ”tokens.” (To understand why we do so, see Footnote 5.)

Typically, tokens are one-to-one mapping of numerical IDs to strings of symbols, as Table 3 shows:

For example, the GPT-3 model (Brown et al. (2020)) uses a sub-word tokenization scheme called Byte Pair Encoding (BPE) (Sennrich et al. (2016)), and it empirically appears to provide a good compromise between sequence length (i.e. how many tokens are needed to represent a piece of text) and the complexity of the model (i.e. how many ”words” the model needs to learn) (Brown et al. (2020)).

Tokenization Scheme	Symbol String
Character-level	Individual Characters (e.g. a, b, c)
Word-level	Whole Words (e.g. ”hello”, ”world”)
Sub-word	Sequences of ≥ 1 characters (e.g. ”ing”, ”pre”)

Table 3: Tokenization Schemes

3.3 Self-Supervised Learning

Once the data is tokenized, the next step is to train a Transformer (Vaswani et al. (2017)) to learn the patterns in the data. Here, the model is trained to predict the next token in a sequence, given all the tokens that have come before it.

Practically, this means that the model ingests a sequence of tokens as input, processes them through its layers, and outputs a probability distribution over the next token for each position in the sequence. See Figure 1 for an illustration of this process.

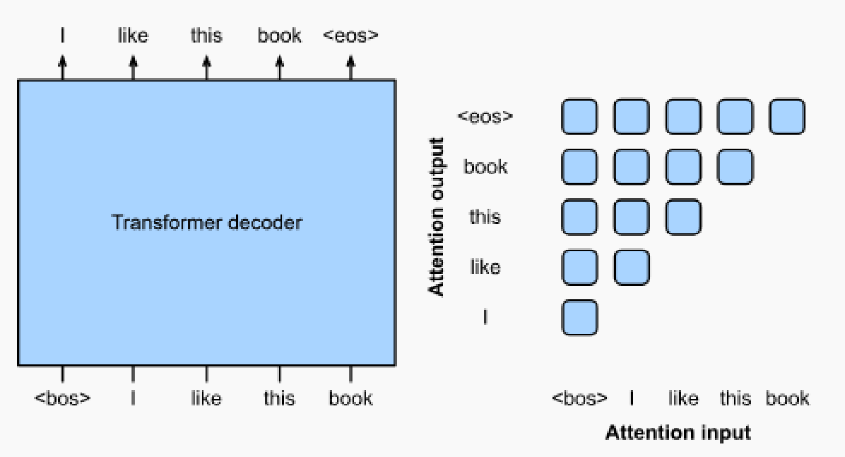

Figure 1: How an LLM like GPT-3 processes a sequence of tokens.

The model is left to learn as most other neural networks do (via backpropagation) (Karpathy (2025)), by adjusting its parameters to minimize the difference [6] between its predicted token probabilities and the actual next tokens within the sequence itself.

3.4 On The Cost of Pre-Training

As mentioned by Karpathy (2025), pre-training an LLM is a resource-intensive process. Creating solid pre-trained LLMs can take months of training on powerful hardware, often involving thousands of GPUs or TPUs working in parallel. (Yang et al. (2025); Anthropic (2025); OpenAI (2022); DeepSeek-AI et al. (2025b); et al (2025); Touvron et al. (2023))

As such, true LLM pre-training is usually the preserve of a select few immensely well-funded entities.

3.5 Aside: Some Quirks of Base Models

While pre-training equips an LLM with a broad understanding of whichever languages it was trained on and a vast repository of knowledge, the base model at this point is ”an extremely expensive autocomplete” (Karpathy (2025)). It can generate text that is coherent,

[6] This difference is usually the cross-entropy loss.

but as (Karpathy (2025)) notes, the base model, without external assistance, is not the best at generating contextually appropriate content, following instructions, answering questions, or engaging in conversations.

While such superficial flaws can be mitigated with careful prompt engineering, the base model still has some fundamental limitations that are worth noting.

- **Latent Memory**: Base models often demonstrate a surprising ability to recall information they encounter frequently during training, such as the content of a well-known Wikipedia page like the one about zebras. This is best explained by the base model having a ”latent memory” (or latent knowledge) that is burned into its parameters during pre-training. However, Karpathy (2025) suggests that the memorization is not infallible.
- **Working Memory**: Moreover, base models can utilise their context window to store information, retrieving them as necessary. We *speculate* that this is responsible for the model’s ability to perform Chain of Thought (CoT) reasoning (Wei et al. (2023)).
- **Hallucinations**: While base models are obviously capable of creating cogent text, they can inadvertently generate text that happens to be factually incorrect. Karpathy (2025) observes this to be the case with a fabricated person ”Orson Kovacs” who does not exist in reality.
- **In-Context Learning**: Base models appear to have the ability to learn from the context provided in the prompt, which allows them to adapt their responses based on examples given, even if they have never seen those examples before, or if the examples are few in number (”few-shot learning”). This is often referred to as ”in-context learning” (Brown et al. (2020)).

4 Post-Training: Taking the Base Model Further

In the last section, we learned that base models have limitations which limit their usefulness in practical applications. To address these limitations, we can take the base model and further refine it in a process known as *post-training* (Karpathy (2025)). Most deployed LLMs today are post-trained in some way or another. (Yang et al. (2025); Anthropic (2025); OpenAI (2022); DeepSeek-AI et al. (2025b); et al (2025))

Post-training has a few traits which distinguishes it from pre-training (Karpathy (2025)):

- Post-training appears to be done on a smaller, more targeted dataset, which is often much smaller than the dataset used for pre-training.
- Post-training is usually much faster than pre-training, often taking only hours instead of months.
- Compared to pre-training, post-training is more diverse in terms of the number of techniques at practitioners’ disposal to refine the LLM’s behavior.

We go through the post-training techniques mentioned by Karpathy (2025) below.

4.1 Supervised Fine-Tuning (SFT)

In SFT, the LLM continues training as it did during pre-training, but on a smaller dataset of high-quality examples encompassing a specific domain or task.

Sometimes, the examples are also curated to demonstrate the desired behavior of the LLM. For example, to create a conversational AI, the LLM would be fine-tuned on a dataset of example dialogues between a human user and an AI assistant. Such dialogues are often meticulously crafted by human contractors who follow detailed guidelines, but are increasingly generated by previous iterations of LLMs to scale up the data creation process.

To continue the analogy in pre-training, *SFT could be like providing the LLM with examples and answers from a textbook, whereupon it copies the examples and learns how to respond to similar questions.*

4.2 Reinforcement Learning (RL) Post-Training

As mentioned at the start of this section, post-training is more diverse than pre-training. Indeed, SFT is not the only technique available to refine LLM behavior. Another approach to post-training is **Reinforcement Learning (RL) post-training** (Karpathy (2025)), which refines the LLM's behavior through maximization of a reward signal.

Again continuing the analogy from pre-training, *RL post-training could be like providing the LLM with exercises that test its understanding and application of the learned material from the SFT phase, allowing it to practice and improve its skills.*

There are two main types of RL post-training (Karpathy (2025)):

- **Reinforcement Learning from Human/AI Feedback (RLHF/RLAIF)**: In this approach, the LLM tries to learn the most preferred responses based on feedback from humans or other AI systems.
- **Reinforcement Learning with Verifiable Rewards (RLVR)**: In this approach, the LLM is trained to maximize a reward signal that can be verified by an external system, such as an algorithm, a theorem prover, or a search engine, etc.

4.2.1 RLHF/RLAIF

In RLHF/RLAIF, the LLM is trained to maximize the expected reward signal from human or AI feedback. This is typically done by generating a large number of responses to various prompts, and then having humans or AI systems evaluate the quality of these responses.

Of course, this approach is very tough to scale up, so it is common instead to train a statistical reward model (Zhu et al. (2024)) which *approximates* the relative rankings of responses based on human feedback, or preferences of previously-trained LLMs. This reward model is then used to guide the LLM's learning process, allowing it to generate responses that are most preferred by humans or other LLMs

Naturally, the tradeoff is that the LLM may adjust itself to "game" the reward model, by abusing imperfections in the reward model's output to generate responses that are apparently highly ranked, but actually appear weird or nonsensical to humans or other LLMs.

4.2.2 RLVR

RLVR typically experiences fewer such issues as compared to RLHF/RLAIF, in the sense that the reward signal tends to instead be rule-based, verifiable, free of AI/human biases, and independent of statistical approximators whose inaccuracies can be exploited by the LLM.

For example, the reward signal could be whether the code written by the LLM compiles successfully, something which a human can use the same compiler to unambiguously double-check.

The exemplar of this approach is Deepseek-R1 (DeepSeek-AI et al. (2025b)), which uses Group Relative Policy Optimization (GRPO) to train the LLM on math, reasoning, and coding tasks by verifying the correctness of the LLM's responses using a theorem prover or a compiler.

However, this comes at the cost of flexibility, as a suitable reward signal may not be available for every task (how do you verify that a poem is good?).

Moreover, while RLVR is verifiable, it is not *infallible*. Should there be some kind of bug in the reward signal, the LLM may still generate responses that are inconsistent or undesirable.

4.2.3 Another Aside: Some challenges associated with RL Post-Training

While RL post-training is a powerful technique that can significantly improve the performance of LLMs, it is not without challenges.

As with most RL problems (Jones (2021)), RL post-training suffers from instability due to the multitude of moving parts within an RL post-training setup. [7]

Moreover, the LLM may still be susceptible to imperfections within the RL process, which can lead to interesting biases such as longer and longer response lengths. (Liu et al. (2025))

5 Conclusion

In this report, we have provided a beginner-friendly overview how Large Language Models (LLMs) like ChatGPT are created, from the initial pre-training phase to the subsequent post-training phase. We hope this document allows readers to gain a better understanding of the key concepts and techniques involved in LLM development, and how to effectively interact with these models.

References

Anthropic. System card: Claude opus 4 & claude sonnet 4, 2025. URL https://www-cdn.anthropic.com/4263b940cabb546aa0e3283f35b686f4f3b2ff47.pdf.

Johan Bjorck, Carla P. Gomes, and Kilian Q. Weinberger. Is high variance unavoidable in rl? a case study in continuous control, 2022. URL https://arxiv.org/abs/2110.11222.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui

[7] We speculate somewhat baselessly that the high variance inherent to most RL problems (Bjorck et al. (2022); Irpan (2018)) is mitigated somewhat by the constrained nature of the pre-trained LLM's outputs, which are already somewhat structured and coherent.

Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025a. URL https://arxiv.org/abs/2501.12948.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025b. URL https://arxiv.org/abs/2412.19437.

Inc. Element Labs. Lm studio, 2025. URL https://lmstudio.ai/.

Gemini Team et al. Gemini: A family of highly capable multimodal models, 2025. URL https://arxiv.org/abs/2312.11805.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL https://arxiv.org/abs/2312.00752.

HuggingFace. Hugging face, 2025. URL https://huggingface.co/.

Inc. Hyperbolic Team. Hyperbolic - the open access ai cloud, 2024. URL https://hyperbolic.ai/.

Alex Irpan. Deep reinforcement learning doesn't work yet. https://www.alexirpan.com/2018/02/14/rl-hard.html, 2018.

Andy L. Jones. Debugging reinforcement learning. https://andyljones.com/posts/rl-debugging.html, 2021.

Andrej Karpathy. Deep dive into llms like chatgpt, 2025. URL https://www.youtube.com/watch?v=7xTGNNLPyMI.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025. URL https://arxiv.org/abs/2503.20783.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models, 2025. URL https://arxiv.org/abs/2502.09992.

OpenAI. Introducing chatgpt, 2022. URL https://openai.com/index/chatgpt/.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale, 2024. URL https://arxiv.org/abs/2406.17557.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Jiaju Lin, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Bolun Wang, Johan S. Wind, Stanislaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. Rwkv: Reinventing rnns for the transformer era, 2023. URL https://arxiv.org/abs/2305.13048.

Grant Sanderson. Large language models explained briefly, 2025. URL https://www.youtube.com/watch?v=LPZh9BOjkQs.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units, 2016. URL https://arxiv.org/abs/1508.07909.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014. URL https://arxiv.org/abs/1409.3215.

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, 2023. URL https://arxiv.org/abs/2302.13971.

Capitole Université Toulouse 1. Blacklist ut1, 2025. URL https://dsi.ut-capitole.fr/blacklists/index_en.php.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, Illia Polosukhin, Samy Bengio, and Yiming Yang. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2025. URL https://arxiv.org/abs/2303.18223.

Banghua Zhu, Jiantao Jiao, and Michael I. Jordan. Principled reinforcement learning with human feedback from pairwise or k-wise comparisons, 2024. URL https://arxiv.org/abs/2301.11270.