AI Applications Lecture 4
Licenses and Openness

SUZUKI, Atsushi
Jing WANG

2025-09-01

Contents

1 Introduction

1.1 Review of the Previous Lecture

In the last lecture, we defined the function represented by a neural network as **generally as possible**. The key points were as follows:

- The **architecture** of a neural network is the design information, consisting of the structure of a **Directed Acyclic Graph (DAG)**, the **activation function** for each node, and the **correspondence** between edges and parameters.
- A **checkpoint** contains the **specific parameter values** obtained through training for that architecture.
- Only when both the architecture $f_{(\cdot)}$ and the checkpoint $\boldsymbol{\theta}$ are available is a **specific function** $f_{\boldsymbol{\theta}}$ uniquely determined. That is, for an input $\boldsymbol{x}$, it returns an output $\boldsymbol{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x})$.

Let's revisit this perspective using mathematical notation. For an input space $\mathcal{X}$, an output space $\mathcal{Y}$, and a parameter space $\Theta \subset \mathbb{R}^D$:

$$\text{Architecture } f_{(\cdot)} : \Theta \to \{\mathcal{X} \to \mathcal{Y}\}, \qquad \boldsymbol{\theta} \in \Theta \mapsto f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}.$$

Training is the process of determining parameters based on past data using some algorithm, and inference is the process of applying $f_{\boldsymbol{\theta}^*}$.

1.2 Learning Outcomes

By the end of this lecture, students should be able to explain and judge the following:

- Systematically explain the **various stages of releasing and licensing a neural network model** (e.g., UI/API release, architecture + checkpoint release, training code release).
- Determine whether a model provided under an **Open Source Software (OSS)** license can be **used for their own purposes**, by referencing major licenses (MIT / BSD-3 / Apache 2.0 / GPL-3).

1.3 Disclaimer

The speaker of this lecture is not a legal expert. The content presented here is **general guidance**, and for specific cases, you should consult with **lawyers in your respective country and region**. The goal is to **reasonably prevent** legal risks, and **final decisions should be made at your own responsibility** (also refer to official documents from GitHub, etc.) [5, 12, 20].

2 Degrees of Openness with Examples

2.1 From Completely Closed to Almost Open

Neural networks span a spectrum from **completely closed** (only UI/API is shared) to **almost completely open** (architecture, checkpoint, and even training code/recipes are released). Furthermore, even if a model is "open," whether others can freely use it is determined by its **license**.

Case 1: The Staged Release of GPT-2 When OpenAI's GPT-2 was announced in 2019, the weights for the largest model were subject to a **staged release**, and for a time, it was completely closed (the 1.5B weights were later fully released in November 2019) [24, 25]. It is worth noting that the means of use were limited at the time.

Case 2: LLMs as a Service As of August 2025, OpenAI provides **ChatGPT-5** via its ChatGPT interface and API. However, **inference is executed on their servers**, and the **details of the architecture and checkpoints are generally not disclosed** [26]. Users only interact with prompts and outputs.

Case 3: "Open" Models and Distribution Platforms Models like **Llama** (Meta) [17, 18], **Qwen** (Alibaba) [2], **Mistral** (Mistral AI) [19], and **DeepSeek** (DeepSeek AI) [10] are widely known as **open (Open Weights/Open Model)**. The **Hugging Face Hub** serves as a platform for their distribution and discovery (with licenses clearly stated in repositories), allowing them to be downloaded and run in conjunction with various libraries [14, 15].

3 What Does "Open" Mean?

Generally, when a model is called "open," it means that at a minimum, the **architecture (configuration files or implementation) and the trained checkpoint (weights)** are publicly available. So, what is **often not disclosed**? Typically, it is the **training data** and the **training algorithm (including the complete training recipe and scripts for reproduction)**.

Remark 3.1 (Feasibility of Inference and Training)**.** With the architecture $f_{(\cdot)}$ and checkpoint θ, **inference** is possible. However, **reproducing** the training (arriving at the exact same θ) is usually impossible unless the data, recipe, random seeds, etc., are identical. For this reason, even with "open weights," if the **insights from the training process (e.g., data curation, regularization techniques)** are not disclosed, the reproducibility and potential for adaptation are limited.

3.1 Visualizing the Relationship with a Block Diagram

Figure 1 uses the framework from the previous lecture (parametric functions) to color-code the **parts that tend to be open** (architecture and checkpoint) and the **parts that tend to remain private** (training data and training algorithm).

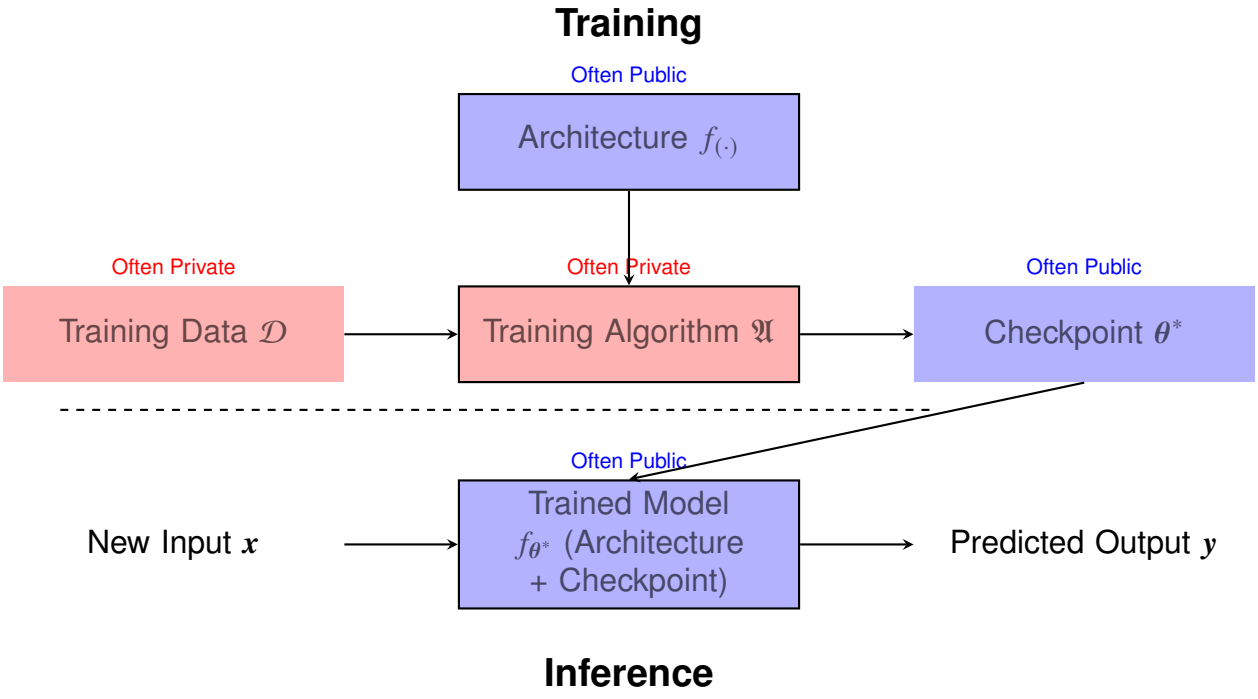

Figure 1: Color-coding of the scope of release in open models. Blue = Often Public (Architecture $f_{(\cdot)}$, Checkpoint θ^*, Trained Model f_{θ^*}), Red = Often Private (Training Data $\mathcal{D}$, Training Algorithm $\mathfrak{A}$).

4 Specific Examples of Release Stages

4.1 UI/API-only Release (Hosted)

In this model, only an **inference service** is provided via a chat UI or web API, while the **weights and detailed implementation remain private** (e.g., OpenAI's generally available models) [26].

4.2 Architecture + Checkpoint Release

Weights are distributed as **Open Weights**, allowing for local execution or use as a starting point for retraining (e.g., Mistral 7B is released under Apache 2.0 [19], DeepSeek-R1 under MIT [10]). However, the **training data and complete training procedures** are often not disclosed.

4.3 Release of Training Code and Recipes

Some initiatives also release data pipelines and training scripts to improve **training reproducibility** (e.g., AI2's OLMo project [1]).

5 License Basics (For Source Code)

Here, we summarize four widely used OSS licenses, highlighting their **practical implications** and **representative model examples**. The final determination of "can I use this?" must be based on the **specific license text and its scope**.

5.1 MIT License

This license is **very permissive**, with minimal conditions (copyright notice and disclaimer). It facilitates **commercial use, redistribution, and modification** [22]. However, it does not warrant that use of the software does not infringe on the author's patent rights, so there is a theoretical risk of infringing the author's patent rights.

- **Conditions for users**: Include the original copyright notice and permission notice in copies or derivative works.
- **Permissions for users**: "Without restriction." No fees. Commercial use is permitted.
- **Warranty and Liability**: The authors and copyright holders provide no warranty (including no warranty of non-infringement of third-party rights) and assume no liability.
- **Model examples**: **DeepSeek-v3**, **DeepSeek-R1** (DeepSeek AI) [10].

5.2 BSD 3-Clause License

Similar to MIT, this license is **highly permissive** but includes a clause prohibiting the use of developers' or contributors' names for endorsement without permission [21].

- **Conditions for users**: Include the original copyright notice and permission notice in copies or derivative works. Do not use the names of copyright holders or contributors for promotional purposes without permission.
- **Permissions for users**: Redistribution and "use" (effectively unrestricted). No fees. Commercial use is permitted.
- **Warranty and Liability**: The authors and copyright holders provide no warranty and assume no liability.
- **Model/Library example**: The classic machine learning library **scikit-learn** is under BSD-3 [27]. (Whether it includes pre-trained models depends on the specific distribution.)

5.3 Apache License 2.0

This license grants users freedoms similar to the MIT License, with the added benefit for users of an **explicit grant of patent licenses**. This means that even if the author holds patents, you can use them without a fee (however, this does not protect the user if a third party holds a patent) [4].

- **Conditions for users**: Include the original copyright notice and license text in copies or derivative works.
- **Permissions for users**: Free of charge; specific actions are enumerated but are nearly unrestricted. Commercial use is permitted. In addition to a copyright license, a royalty-free patent license is also granted (subject to termination conditions related to patent litigation).
- **Warranty and Liability**: The authors and copyright holders provide no warranty (including no warranty of non-infringement of third-party rights) and assume no liability.
- **Model examples**: **Mistral 7B** was released under Apache 2.0 [19]. The Hugging Face **Transformers** library (which implements many architectures) is under Apache 2.0 [13].

5.4 GPL

The GPL (GNU General Public License) is a representative license that embodies the philosophy of **Copyleft**. As it is less commonly used in the context of AI models and its pros and cons are complex, we will only touch on it briefly. Copyleft is the idea that while allowing the free use, modification, and redistribution of software, it **obligates derivative works to be under the same license (freedom)**. This aims to ensure that the freedom of the software is perpetually inherited.

GPLv3 has a **strong copyleft** provision, requiring that if you distribute software created using GPL code, the source code of the entire software must also be released under GPLv3 [11]. To address the fact that using the software on a server-side does not trigger the source code disclosure obligation, the **AGPL-3** was created, which extends the definition of ”distribution” to include providing services over a network.

- **Model Example**: **YOLOv5** uses AGPL-3.0, which means that if you provide a web service that incorporates this model, you are obligated to release the source code of the entire service [28].
- **Practical Implications**: It imposes strong restrictions on integration into services and code mixing (which can lead to unintended requirements to disclose source code). Careful consideration is required, especially for commercial use.

5.5 The Danger of No License

If source code or a model has no license specified, it is understood by **default that all rights are reserved**, and **no permission is granted for use, modification, or redistribution** [5, 12]. Such materials should be avoided.

6 Model-Specific Licenses (Open-Weight Type)

Recent **open-weight** distributions sometimes include **custom terms that are not standard OSS licenses**, such as use restrictions or monthly active users (MAU) thresholds.

- **Llama series (Meta)**: The **Community License** for the Llama 3 series allows **commercial use with restrictions**, subject to conditions on scale and application (the **700 million MAU** threshold clause is well-known for Llama 2 and 3) [17, 18].
- **Qwen 2.5 (Alibaba)**: The **Qianwen License** for large models in the Qwen 2.5 series (e.g., Qwen/Qwen2.5-72B-Instruct) includes custom conditions, such as a **100 million MAU** threshold [2]. However, other models in the same Qwen 2.5 series, like Qwen/Qwen2.5-7B-Instruct, are licensed under the Apache License 2.0, allowing for relatively free use. This shows that licenses can differ even for models from the same company and series, so caution is needed. As far as the lecturer has investigated, Qwen 3 and later versions are licensed under the Apache License 2.0, permitting relatively free use.

6.1 CreativeML Open RAIL-M and its Derivatives

The **CreativeML Open RAIL-M** license is used by image generation AIs like **Stable Diffusion** [6]. RAIL stands for **R**esponsible **AI L**icense, and as its name suggests, it aims to promote ”responsible AI use.”

This license permits the free use, modification, and distribution of the model and code, but it is characterized by imposing **explicit Use Restrictions**. Specifically, it includes clauses that prohibit use for purposes such as:

- Illegal purposes
- Generating content intended to harm or exploit others
- Spreading misinformation or disinformation
- Uses that violate individual privacy

More importantly, it requires that when distributing **derivative works** created under this license, the **same use restrictions must be imposed on downstream users**. This obligation of ”inheritance of restrictions” is analogous to the copyleft philosophy of GPL (inheritance

of license). However, since it does not mandate that the derivative work must have the exact same license, it is not typically called a copyleft license.

7 Creative Commons

While **CC licenses** are widely used for documents, images, and data, they are generally **unsuitable for the distribution of mutable and executable software/models**. Key variations include **CC BY 4.0** (Attribution) [7], **CC BY-SA 4.0** (ShareAlike) [9], **CC BY-NC 4.0** (NonCommercial) [8], and others.

8 Case Study: Separation of Rights Holders

Even if an architecture implementation (e.g., Hugging Face **Transformers**) is under Apache 2.0, the **license for each checkpoint is separate**. Since **rights and conditions differ for each set of weights even within the same framework**, it is **essential to check the model card and LICENSE file** [13, 15].

- Example: **DeepSeek-R1** weights are MIT [10], the **Mistral** family is Apache 2.0 [19], and **Llama** has its own custom license [2, 17].

9 Model Versions and License Differences Across Versions

Companies that focus on developing large-scale models may release multiple models (checkpoint + architecture) under the same series name. While it can be inferred that the training processes to obtain these checkpoints are similar, they are mathematically distinct entities. It is important to note that there is no guarantee that the licenses for all these models will be the same.

Example 9.1 (Qwen 2.5 (Alibaba))**.** The **Qianwen License** for large language models in the Qwen 2.5 series by Alibaba (e.g., Qwen/Qwen2.5-72B-Instruct) includes custom conditions such as a **100 million MAU** threshold [2]. In the same Qwen 2.5 series, however, models like Qwen/Qwen2.5-7B-Instruct are licensed under the Apache License 2.0, allowing for relatively free use. Qwen 3 is licensed under Apache License 2.0, also allowing for relatively free use.

As the example above shows, licenses can differ for models from the same company and series, so caution is needed. It is necessary to check the model card for the specific model you wish to use.

By the way, we mentioned model names like Qwen/Qwen2.5-72B-Instruct and Qwen/Qwen2.5-7B-Instruct, both of which are part of the Qwen2.5 series. Some students might be curious about the differences between them. While this course aims to avoid delving into the details

of individual models, I will provide a brief explanation for interested students. When there are multiple models within the same series, they generally differ in two main ways: either the architecture is different, resulting in a different number of real numbers in the checkpoint (also called the number of parameters), or the architecture is the same, but the checkpoint is different. I will explain briefly below, but I must re-emphasize that the license should be checked for each case.

9.1 Number of Parameters (Billion Parameters)

The scale of a model is expressed by the **total number of real numbers in its checkpoint (parameters)**. For example, "7B" means the checkpoint contains approximately 7 billion (7×10^9) real numbers (B is for Billion). This is sometimes stated as the model consisting of 7 billion parameters, where each real number is called a parameter. In mathematical terms, the parameters (checkpoint) are a real vector represented as:

$$\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_D) \in \mathbb{R}^D.$$

A "7B model" refers to the dimension D of this vector being approximately 7 billion. Generally, a higher parameter count increases the model's expressive power, but also increases computational costs and memory requirements.

9.2 Numerical Representation: Floating-Point Numbers and Quantization

Each parameter θ_i is stored on a computer in a specific numerical format. This format directly affects the model's file size and inference speed.

- **fp32 (single-precision floating-point)**: A standard format that uses 32 bits to represent a number. It has a wide dynamic range and high precision but consumes a large amount of memory. A 7B model would require approximately $7 \times 10^9 \times 32$ bit ≈ 28 GB of storage.
- **fp16 (half-precision floating-point)**: Uses 16 bits for representation. It halves memory usage and computation compared to fp32, but its smaller range of representable numbers (dynamic range) can make training unstable.
- **bf16 (Bfloat16)**: A 16-bit format developed by Google Brain. Like fp16, it uses 16 bits, but it allocates 8 bits to the exponent, the same as fp32, thereby maintaining a wide dynamic range and improving training stability in deep learning [16].
- **Quantization**: A technique for approximating parameters with a smaller number of bits (e.g., **int8**, **int4**). It can significantly reduce model size and accelerate inference, but generally at the cost of a slight decrease in model accuracy.

9.2.1 Precise Definition of Floating-Point Numbers (IEEE 754)

The standard for handling real numbers in computers is **IEEE 754**. This standard represents a floating-point number in binary using three parts: a **Sign**, an **Exponent**, and a **Mantissa/Fraction**.

A value V is defined using a sign bit $S \in \{0, 1\}$, an integer E represented by the exponent bit string, and a fraction M represented by the mantissa bit string, as follows:

$$V = (-1)^S \times (1.M)_2 \times 2^{E-\text{bias}}$$

Here, $(1.M)_2$ is the binary representation using an "implicit bit," where an implicit '1' is assumed before the mantissa bit string M. The exponent E is not used directly; a fixed **bias** is subtracted to get the actual exponent. This allows for efficient representation of both positive and negative exponents.

Each format (fp32, fp16, bf16) differs in the number of bits allocated to these three parts.

Table 1: Bit Allocation for Floating-Point Formats

Format	Total Bits	Sign	Exponent	Mantissa	Bias
fp32	32	1	8	23	127
fp16	16	1	5	10	15
bf16	16	1	8	7	127

The differences in bit allocation create a trade-off between the **representable range of numbers (dynamic range)** and **precision (the granularity between adjacent numbers)** for each format.

9.2.2 Comparison of fp16 and bf16: Dynamic Range vs. Precision Trade-off

Although both fp16 and bf16 are 16-bit formats, their characteristics differ significantly.

- **fp16** has a longer **mantissa of 10 bits**, giving it **high precision**. However, its short **exponent of 5 bits** results in a **narrow dynamic range**. When dealing with very large or very small numbers, it is prone to **overflow** (exceeding the maximum representable value) or **underflow** (becoming closer to zero than the minimum representable value).
- **bf16** has an **exponent of 8 bits**, the same as fp32, giving it a **very wide dynamic range**. However, its short **mantissa of 7 bits** means its **precision is lower than fp16**.

Let's look at this difference with a concrete example.

Example 9.2 (A number representable in bf16 but not in fp16)**. Large number**: Consider $V = 100,000$.

- **fp16**: The maximum representable normalized number has an exponent of $E_{\text{max}} = 15$ and a maximum mantissa, which is $(2 - 2^{-10}) \times 2^{15} \approx 1.999 \times 32768 = 65504$. Therefore, $100,000$ exceeds the range of fp16 and will **overflow** to infinity (∞).
- **bf16**: Since it has the same exponent as fp32, it can represent numbers up to approximately 3.4×10^{38}. $100,000$ is $1.52587890625 \times 2^{16}$, and the exponent 16 fits within the exponent field as $16 + 127 = 143$, so it can be **represented without issue**.

Small number: Consider $V = 1.0 \times 10^{-5}$.

- **fp16**: The smallest positive normalized number is $2^{-14} \approx 6.1 \times 10^{-5}$. Since 1.0×10^{-5} is smaller than this, it will **underflow** to zero or become a denormalized number with significantly reduced precision.
- **bf16**: The smallest positive normalized number is 2^{-126}, which is extremely small. 1.0×10^{-5} can be **represented** with ease.

Example 9.3 (A number representable in fp16 but not in bf16)**. A number requiring precision**: Consider $V = 1 + 2^{-8} = 1.00390625$. In binary, this is $(1.00000001)_2 \times 2^0$.

- **fp16**: The mantissa has 10 bits. It can **accurately represent** the '1' in the 8th decimal place as $(1.\underline{0000000100})_2$.
- **bf16**: The mantissa has only 7 bits. When trying to represent $(1.\underline{00000001})_2$, the '1' in the 8th bit position does not fit and is rounded to the nearest value (round-to-nearest-even). In this case, it gets rounded to $(1.0000000)_2 = 1$, creating an error. Thus, bf16 **cannot distinguish** between 1 and $1 + 2^{-8}$.

Thus, in deep learning, the wide dynamic range of bf16 can contribute to stability when dealing with potentially huge gradients or parameter values, while fp16 may be advantageous for tasks where precision is critical.

9.3 Instruction Finetuning and "Instruct" Models

Suffixes like "Instruct" or "Chat" in a model's name suggest that it has undergone additional training to adapt to specific tasks, known as **Instruction Finetuning**.

A base large language model (foundation model) is trained to predict the next word from a vast corpus of text. While it possesses general knowledge, its ability to follow a user's specific instructions (e.g., "Summarize...", "Translate...") is limited.

Therefore, additional fine-tuning is performed using a dataset of "instruction (prompt) and desired response" pairs. This allows the model to acquire the ability to engage in conversational interactions and follow diverse instructions [29]. The details of this tuning method may be covered in a later lecture of this course.

10 Practical Guidelines and Caveats

- **”Open” does not mean free to use**: The **license text** governs everything. Always check the **LICENSE** and **Terms of Service**/**AUP** on the model page.
- **Apache 2.0 is a safe choice**: If you want your model to be widely used but have little commercial intent, Apache 2.0 is easy to handle due to the clarity of its **patent clauses** [4].
- **The legal nature of checkpoints is an immature area**: It is reasonable to consider them as having aspects of both **source code** and **data**, and that **no consensus has been established** [3, 23]. A policy of **”when in doubt, do not use”** is the safest approach.

11 Summary

- Many ”open models” release their **architecture and checkpoints**, but the **training data and learning algorithms** are often kept private.
- Even if a model is ”public,” whether **third parties can freely use it** is determined by its **license**. In practice, one must understand the nature of licenses like **MIT** / **BSD-3** / **Apache 2.0** / **GPL-3** and always check the **LICENSE and model card for each model**.

12 Next Time

Next time, we will **download and run a specific open large language model locally** to confirm hands-on the division of roles between the **neural network part (architecture + checkpoint)** and the **surrounding components (tokenizer, pre/post-processing, inference runtime)**.

References

[1] AI2. Olmo: An open language model. https://allenai.org/olmo, 2024–2025.

[2] Alibaba Cloud. Qwen license agreement (e.g., qwen2.5-72b-instruct). https://huggingface.co/Qwen/Qwen2.5-72B-Instruct/blob/main/LICENSE, 2024–2025.

[3] All Things Open. The open source ai definition: Why we need it. https://allthingsopen.org/articles/the-open-source-ai-definition-why-we-need-it, 2024.

[4] Apache Software Foundation. Apache license, version 2.0. http://www.apache.org/licenses/LICENSE-2.0.

[5] Choose a License. No license. https://choosealicense.com/no-permission/.

[6] CompVis and Stability AI. Creativeml open rail-m. https://huggingface.co/spaces/CompVis/stable-diffusion-license/raw/main/license.txt, 2022.

[7] Creative Commons. Attribution 4.0 international (cc by 4.0). https://creativecommons.org/licenses/by/4.0/.

[8] Creative Commons. Attribution-noncommercial 4.0 international (cc by-nc 4.0). https://creativecommons.org/licenses/by-nc/4.0/.

[9] Creative Commons. Attribution-sharealike 4.0 international (cc by-sa 4.0). https://creativecommons.org/licenses/by-sa/4.0/.

[10] DeepSeek-AI. deepseek-ai/deepseek-r1. https://huggingface.co/deepseek-ai/DeepSeek-R1/blob/main/LICENSE, 2025. LICENSE: MIT.

[11] Free Software Foundation. Gnu general public license, version 3. https://www.gnu.org/licenses/gpl-3.0.en.html.

[12] GitHub Docs. Licensing a repository. https://docs.github.com/articles/licensing-a-repository.

[13] Hugging Face. Transformers (license: Apache-2.0). https://github.com/huggingface/transformers.

[14] Hugging Face Docs. Downloading models. https://huggingface.co/docs/hub/en/models-downloading, 2025.

[15] Hugging Face Docs. Licenses (hub). https://huggingface.co/docs/hub/en/repositories-licenses, 2025.

[16] D. Kalamkar et al. Bfloat16: The secret to high performance on cloud tpus. https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus, 2019.

[17] Meta. Llama community license. https://ai.meta.com/llama/license/, 2023–2024.

[18] Meta AI. Introducing meta llama 3. https://ai.meta.com/blog/meta-llama-3/, 2024.

[19] Mistral AI. Announcing mistral 7b. https://mistral.ai/news/announcing-mistral-7b/, 2023.

[20] Open Source Guides. The legal side of open source. https://opensource.guide/legal/.

[21] Open Source Initiative. The bsd 3-clause license. https://opensource.org/license/bsd-3-clause/.

[22] Open Source Initiative. The mit license. https://opensource.org/license/mit/.

[23] Open Source Initiative. Drafts of the open source ai definition. https://opensource.org/ai/drafts, 2024–2025.

[24] OpenAI. Better language models and their implications. https://openai.com/index/better-language-models/, 2019.

[25] OpenAI. Releasing gpt-2: 1.5b release. https://openai.com/index/gpt-2-1-5b-release/, 2019.

[26] OpenAI. Models. https://platform.openai.com/docs/models, 2025.

[27] scikit-learn. License. https://github.com/scikit-learn/scikit-learn/blob/main/COPYING.

[28] Ultralytics. yolov5 (license: Agpl-3.0). https://github.com/ultralytics/yolov5/blob/master/LICENSE.

[29] Jason Wei et al. Finetuned language models are zero-shot learners. arXiv preprint arXiv:2109.01652, 2022. ICLR 2022.