AI Applications Lecture 4

Licenses and Openness

SUZUKI, Atsushi
Jing WANG

September 1, 2025

Outline

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**.

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:

- The **architecture** is the design information: a **DAG structure**, **activation functions**, and the **edge-parameter mapping**.

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:

- The **architecture** is the design information: a **DAG structure**, **activation functions**, and the **edge-parameter mapping**.
- A **checkpoint** contains the **specific parameter values** obtained through training.

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:

- The **architecture** is the design information: a **DAG structure**, **activation functions**, and the **edge-parameter mapping**.
- A **checkpoint** contains the **specific parameter values** obtained through training.
- Only with both the architecture $f_{(\cdot)}$ and the checkpoint θ is a **specific function** f_θ uniquely determined.

1.1 Review of the Previous Lecture

Let’s revisit this perspective using mathematical notation.

1.1 Review of the Previous Lecture

Let's revisit this perspective using mathematical notation.

For an input space $\mathcal{X}$, an output space $\mathcal{Y}$, and a parameter space $\Theta \subset \mathbb{R}^D$:

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

1.1 Review of the Previous Lecture

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 determining parameters $\boldsymbol{\theta}^*$ based on data.
- **Inference** is applying the final function $f_{\boldsymbol{\theta}^*}$.

1.2 Learning Outcomes

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

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**.

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**.
- Determine whether a model provided under an **Open Source Software (OSS)** license can be **used for their own purposes**, by referencing major licenses.

1.3 Disclaimer

Important Disclaimer

The speaker of this lecture is not a legal expert. The content presented here is **general guidance**.

1.3 Disclaimer

Important Disclaimer

The speaker of this lecture is not a legal expert. The content presented here is **general guidance**.

- For specific cases, you **must consult with lawyers** in your respective country and region.

1.3 Disclaimer

Important Disclaimer

The speaker of this lecture is not a legal expert. The content presented here is **general guidance**.

- For specific cases, you **must consult with lawyers** in your respective country and region.
- The goal is to **reasonably prevent** legal risks.

1.3 Disclaimer

Important Disclaimer

The speaker of this lecture is not a legal expert. The content presented here is **general guidance**.

- For specific cases, you **must consult with lawyers** in your respective country and region.
- The goal is to **reasonably prevent** legal risks.
- **Final decisions should be made at your own responsibility. Refer to official documentation [1, 2, 3].**

Degrees of Openness with Examples

2. Degrees of Openness with Examples

Neural networks span a spectrum from **completely closed** to **almost completely open**.

2. Degrees of Openness with Examples

Neural networks span a spectrum from **completely closed** to **almost completely open**.

- **Completely Closed:** Only a User Interface (UI) or Application Programming Interface (API) is shared. You send inputs and get outputs, but see nothing else.

2. Degrees of Openness with Examples

Neural networks span a spectrum from **completely closed** to **almost completely open**.

- **Completely Closed:** Only a User Interface (UI) or Application Programming Interface (API) is shared. You send inputs and get outputs, but see nothing else.
- **Almost Open:** Architecture, checkpoints, and even the training code or "recipes" are released.

2. Degrees of Openness with Examples

Neural networks span a spectrum from **completely closed** to **almost completely open**.

- **Completely Closed:** Only a User Interface (UI) or Application Programming Interface (API) is shared. You send inputs and get outputs, but see nothing else.
- **Almost Open:** Architecture, checkpoints, and even the training code or "recipes" are released.
- A model's **license** determines if others can freely use it, regardless of how "open" it is.

Example (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** due to safety concerns [4].

Example (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** due to safety concerns [4].

- Initially, the most powerful 1.5B parameter model was completely closed.

Example (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** due to safety concerns [4].

- Initially, the most powerful 1.5B parameter model was completely closed.
- The full model was eventually released in November 2019 after a period of study and debate [5].

Example (LLMs as a Service)

As of August 2025, OpenAI provides **ChatGPT-5** via its ChatGPT interface and API.

Example (LLMs as a Service)

As of August 2025, OpenAI provides **ChatGPT-5** via its ChatGPT interface and API.

- **Inference is executed on their servers**.

Example (LLMs as a Service)

As of August 2025, OpenAI provides **ChatGPT-5** via its ChatGPT interface and API.

- **Inference is executed on their servers**.
- The **details of the architecture and checkpoints are generally not disclosed** [6].
- Users only interact with prompts and outputs.

Example ("Open" Models and Distribution Platforms)

Models like **Llama** (Meta), **Qwen** (Alibaba), **Mistral** (Mistral AI), and **DeepSeek** (DeepSeek AI) are known as **open (Open Weights/Open Model)**.

Example ("Open" Models and Distribution Platforms)

Models like **Llama** (Meta), **Qwen** (Alibaba), **Mistral** (Mistral AI), and **DeepSeek** (DeepSeek AI) are known as **open (Open Weights/Open Model)**.

- The **Hugging Face Hub** is a major platform for their distribution [7].

Example ("Open" Models and Distribution Platforms)

Models like **Llama** (Meta), **Qwen** (Alibaba), **Mistral** (Mistral AI), and **DeepSeek** (DeepSeek AI) are known as **open (Open Weights/Open Model)**.

- The **Hugging Face Hub** is a major platform for their distribution [7].
- You can download the architecture and checkpoints to run locally.

Example ("Open" Models and Distribution Platforms)

Models like **Llama** (Meta), **Qwen** (Alibaba), **Mistral** (Mistral AI), and **DeepSeek** (DeepSeek AI) are known as **open (Open Weights/Open Model)**.

- The **Hugging Face Hub** is a major platform for their distribution [7].
- You can download the architecture and checkpoints to run locally.
- Licenses are (usually) clearly stated in the model repositories. [8, 9, 10, 11, 12]

What Does "Open" Mean?

3. What Does "Open" Mean?

Generally, when a model is called "open," it means that at a minimum, the following are publicly available:

3. What Does "Open" Mean?

Generally, when a model is called "open," it means that at a minimum, the following are publicly available:

- The **architecture** (as configuration files or code).

3. What Does "Open" Mean?

Generally, when a model is called "open," it means that at a minimum, the following are publicly available:

- The **architecture** (as configuration files or code).
- The trained **checkpoint** (the weights).

3. What Does "Open" Mean?

So, what is **often not disclosed** even for "open" models?

3. What Does "Open" Mean?

So, what is **often not disclosed** even for "open" models?

Typically, the private components are:

- The **training data**.

3. What Does "Open" Mean?

So, what is **often not disclosed** even for "open" models?

Typically, the private components are:

- The **training data**.
- The **training algorithm** (including the complete training recipe and scripts for reproduction).

3. What Does "Open" Mean?

Remark (Feasibility of Inference and Training)

With the architecture $f_{(\cdot)}$ and checkpoint θ, **inference** is possible.

Remark (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.

3. What Does "Open" Mean?

Remark (Feasibility of Inference and Training)

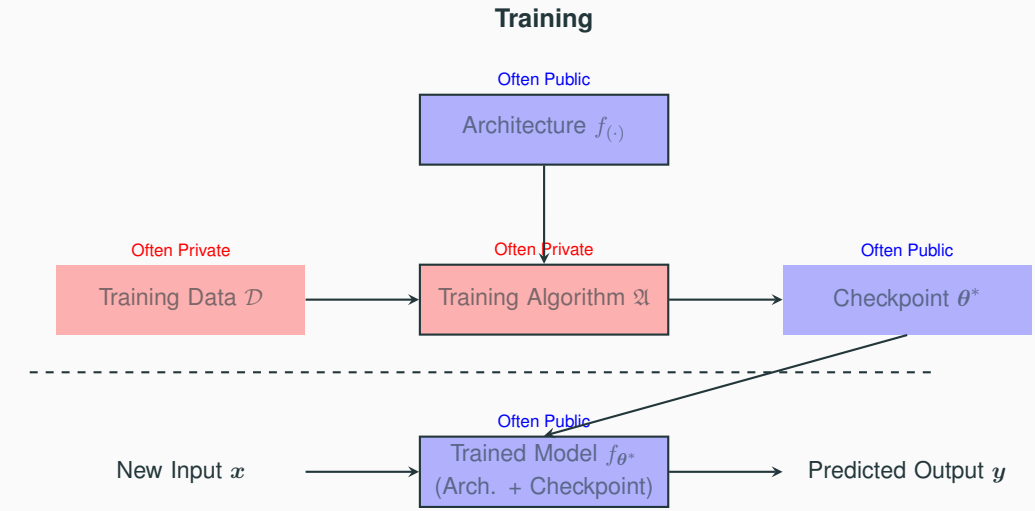
With the architecture $f_{(\cdot)}$ and checkpoint θ, **inference** is possible.

However, **reproducing** the training (arriving at the exact same θ) is usually impossible unless the data, recipe, random seeds, etc., are identical.

Even with "open weights," if the **insights from the training process** are not disclosed, the reproducibility and potential for adaptation are limited.

3.1 Visualizing the Relationship

This diagram uses our framework to show what's typically open vs. private.



Specific Examples of Release Stages

UI/API-only Release (Hosted)

UI/API-only Release (Hosted)

- Only an **inference service** is provided.

UI/API-only Release (Hosted)

- Only an **inference service** is provided.
- The **weights and detailed implementation remain private**.

UI/API-only Release (Hosted)

- Only an **inference service** is provided.
- The **weights and detailed implementation remain private**.
- Example: OpenAI's generally available models like GPT-4 and beyond [6].

Architecture + Checkpoint Release

Architecture + Checkpoint Release

- Weights are distributed as **Open Weights**.

Architecture + Checkpoint Release

- Weights are distributed as **Open Weights**.
- This allows for local execution or use as a starting point for retraining.

Architecture + Checkpoint Release

- Weights are distributed as **Open Weights**.
- This allows for local execution or use as a starting point for retraining.
- Examples: Mistral 7B (Apache 2.0) [10], DeepSeek-R1 (MIT) [11].

Architecture + Checkpoint Release

- Weights are distributed as **Open Weights**.
- This allows for local execution or use as a starting point for retraining.
- Examples: Mistral 7B (Apache 2.0) [10], DeepSeek-R1 (MIT) [11].
- The **training data and complete training procedures** are often not disclosed.

Release of Training Code and Recipes

Release of Training Code and Recipes

- Some initiatives also release data pipelines and training scripts.

Release of Training Code and Recipes

- Some initiatives also release data pipelines and training scripts.
- This is done to improve **training reproducibility**.

Release of Training Code and Recipes

- Some initiatives also release data pipelines and training scripts.
- This is done to improve **training reproducibility**.
- Example: AI2's OLMo project [13].

License Basics (For Source Code)

5. License Basics (For Source Code)

Here, we summarize four widely used Open Source Software (OSS) licenses.

5. License Basics (For Source Code)

Here, we summarize four widely used Open Source Software (OSS) licenses.

The focus is on their **practical implications**.

5. License Basics (For Source Code)

Here, we summarize four widely used Open Source Software (OSS) licenses.

The focus is on their **practical implications**.

F

inal 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.

5.1 MIT License

This license is **very permissive**, with minimal conditions.

- **Conditions for users**: Include the original copyright notice and permission notice in copies or derivative works.

5.1 MIT License

This license is **very permissive**, with minimal conditions.

- **Conditions for users**: Include the original copyright notice and permission notice in copies or derivative works.
- **Permissions for users**: "Without restriction." Commercial use is permitted.

5.1 MIT License

This license is **very permissive**, with minimal conditions.

- **Conditions for users**: Include the original copyright notice and permission notice in copies or derivative works.
- **Permissions for users**: "Without restriction." Commercial use is permitted.
- **Warranty and Liability**: No warranty and no liability for the authors.

5.1 MIT License

This license is **very permissive**, with minimal conditions.

- **Conditions for users**: Include the original copyright notice and permission notice in copies or derivative works.
- **Permissions for users**: "Without restriction." Commercial use is permitted.
- **Warranty and Liability**: No warranty and no liability for the authors.
- **Model examples**: **DeepSeek-v3**, **DeepSeek-R1** [11].

5.2 BSD 3-Clause License

Similar to MIT, this license is **highly permissive**.

5.2 BSD 3-Clause License

Similar to MIT, this license is **highly permissive**.

It includes a clause prohibiting the use of developers' or contributors' names for endorsement without permission [14].

5.2 BSD 3-Clause License

Similar to MIT, this license is **highly permissive**.

It includes a clause prohibiting the use of developers' or contributors' names for endorsement without permission [14].

- **Conditions for users**: Include notices; do not use names for promotion.

5.2 BSD 3-Clause License

Similar to MIT, this license is **highly permissive**.

It includes a clause prohibiting the use of developers' or contributors' names for endorsement without permission [14].

- **Conditions for users**: Include notices; do not use names for promotion.
- **Permissions for users**: Essentially unrestricted, including commercial use.

5.2 BSD 3-Clause License

Similar to MIT, this license is **highly permissive**.

It includes a clause prohibiting the use of developers' or contributors' names for endorsement without permission [14].

- **Conditions for users**: Include notices; do not use names for promotion.
- **Permissions for users**: Essentially unrestricted, including commercial use.
- **Warranty and Liability**: No warranty and no liability.

5.2 BSD 3-Clause License

Similar to MIT, this license is **highly permissive**.

It includes a clause prohibiting the use of developers' or contributors' names for endorsement without permission [14].

- **Conditions for users**: Include notices; do not use names for promotion.
- **Permissions for users**: Essentially unrestricted, including commercial use.
- **Warranty and Liability**: No warranty and no liability.
- **Model/Library example**: The classic machine learning library **scikit-learn** [15].

5.3 Apache License 2.0

This license grants users freedoms similar to the MIT License.

5.3 Apache License 2.0

This license grants users freedoms similar to the MIT License.

It has the added benefit for users of an **explicit grant of patent licenses** [16].

5.3 Apache License 2.0

This license grants users freedoms similar to the MIT License.

It has the added benefit for users of an **explicit grant of patent licenses** [16].

- **Conditions for users**: Include notices. If you modify files, you must state that.

5.3 Apache License 2.0

This license grants users freedoms similar to the MIT License.

It has the added benefit for users of an **explicit grant of patent licenses** [16].

- **Conditions for users**: Include notices. If you modify files, you must state that.
- **Permissions for users**: Broad permissions, including commercial use. Explicit patent grant.

5.3 Apache License 2.0

This license grants users freedoms similar to the MIT License.

It has the added benefit for users of an **explicit grant of patent licenses** [16].

- **Conditions for users**: Include notices. If you modify files, you must state that.
- **Permissions for users**: Broad permissions, including commercial use. Explicit patent grant.
- **Model examples**: **Mistral 7B** [10], Hugging Face **Transformers** library [17].

5.4 GPL

The GPL (GNU General Public License) embodies the philosophy of **Copyleft**.

5.4 GPL

The GPL (GNU General Public License) embodies the philosophy of **Copyleft**.

Copyleft: While allowing free use, modification, and redistribution, it **obligates derivative works to be under the same license**.

5.4 GPL

The GPL (GNU General Public License) embodies the philosophy of **Copyleft**.

Copyleft: While allowing free use, modification, and redistribution, it **obligates derivative works to be under the same license**.

This aims to ensure that the freedom of the software is perpetually inherited.

5.4 GPL

- **GPLv3**: Has a **strong copyleft** provision. If you distribute software containing GPLv3 code, the source code of the entire software must be released under GPLv3 [18].

5.4 GPL

- **GPLv3**: Has a **strong copyleft** provision. If you distribute software containing GPLv3 code, the source code of the entire software must be released under GPLv3 [18].
- **AGPL-3**: A variant that closes the "server-side loophole." The source code disclosure obligation is triggered even if the software is just used to provide a service over a network, without being distributed.

5.4 GPL

- **GPLv3**: Has a **strong copyleft** provision. If you distribute software containing GPLv3 code, the source code of the entire software must be released under GPLv3 [18].
- **AGPL-3**: A variant that closes the "server-side loophole." The source code disclosure obligation is triggered even if the software is just used to provide a service over a network, without being distributed.
- **Model Example**: **YOLOv5** uses AGPL-3.0 [19].

Practical Implications of Copyleft Licenses:

Practical Implications of Copyleft Licenses:

- They impose strong restrictions on integration into services and mixing with other code.

Practical Implications of Copyleft Licenses:

- They impose strong restrictions on integration into services and mixing with other code.
- This can lead to unintended requirements to disclose your own source code.

Practical Implications of Copyleft Licenses:

- They impose strong restrictions on integration into services and mixing with other code.
- This can lead to unintended requirements to disclose your own source code.
- Careful consideration is required, especially for commercial use.

5.5 The Danger of No License

What if source code or a model has **no license** specified?

5.5 The Danger of No License

What if source code or a model has **no license** specified?

I

f no license is specified, it is understood by **default that all rights are reserved**.

5.5 The Danger of No License

What if source code or a model has **no license** specified?

I

f no license is specified, it is understood by **default that all rights are reserved**.

- You have **no permission** for use, modification, or redistribution [1, 2].

5.5 The Danger of No License

What if source code or a model has **no license** specified?

I

f no license is specified, it is understood by **default that all rights are reserved**.

- You have **no permission** for use, modification, or redistribution [1, 2].
- Such materials should be avoided.

Model-Specific Licenses (Open-Weight Type)

6. Model-Specific Licenses (Open-Weight Type)

Recent **open-weight** models sometimes use **custom terms that are not standard OSS licenses**.

6. Model-Specific Licenses (Open-Weight Type)

Recent **open-weight** models sometimes use **custom terms that are not standard OSS licenses**.

These often include use restrictions or thresholds based on Monthly Active Users (MAU).

Example (Llama series (Meta))

The **Community License** for the Llama 3 series allows **commercial use with restrictions** [8, 20].

6. Model-Specific Licenses

Example (Llama series (Meta))

The **Community License** for the Llama 3 series allows **commercial use with restrictions** [8, 20].

- The terms are subject to conditions on scale and application.

6. Model-Specific Licenses

Example (Llama series (Meta))

The **Community License** for the Llama 3 series allows **commercial use with restrictions** [8, 20].

- The terms are subject to conditions on scale and application.
- The **700 million MAU** threshold clause is well-known for Llama 2 and 3.

Example (Qwen 2.5 (Alibaba))

The license for the Qwen series can vary even within the same version family.

Example (Qwen 2.5 (Alibaba))

The license for the Qwen series can vary even within the same version family.

- The largest models (e.g., Qwen2.5-72B-Instruct) used a custom **Qianwen License** with a **100 million MAU** threshold [9].

Example (Qwen 2.5 (Alibaba))

The license for the Qwen series can vary even within the same version family.

- The largest models (e.g., Qwen2.5-72B-Instruct) used a custom **Qianwen License** with a **100 million MAU** threshold [9].
- However, smaller models in the same series (e.g., Qwen2.5-7B-Instruct) are licensed under the standard **Apache License 2.0**.

Example (Qwen 2.5 (Alibaba))

The license for the Qwen series can vary even within the same version family.

- The largest models (e.g., Qwen2.5-72B-Instruct) used a custom **Qianwen License** with a **100 million MAU** threshold [9].
- However, smaller models in the same series (e.g., Qwen2.5-7B-Instruct) are licensed under the standard **Apache License 2.0**.
- Later versions, like Qwen 3, seem to have standardized on Apache 2.0.

6.1 CreativeML Open RAIL-M

The **CreativeML Open RAIL-M** license is used by image generation AIs like **Stable Diffusion** [21].

6.1 CreativeML Open RAIL-M

The **CreativeML Open RAIL-M** license is used by image generation AIs like **Stable Diffusion** [21].

RAIL stands for **R**esponsible **AI** **L**icense.

6.1 CreativeML Open RAIL-M

The **CreativeML Open RAIL-M** license is used by image generation AIs like **Stable Diffusion** [21].

RAIL stands for **R**esponsible **AI L**icense.

It permits free use, modification, and distribution, but imposes **explicit Use Restrictions**.

6.1 CreativeML Open RAIL-M

The license 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

6.1 CreativeML Open RAIL-M

The license 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

I

t requires that when distributing **derivative works**, the **same use restrictions must be imposed on downstream users**.

Creative Commons

7. Creative Commons

While **CC licenses** are widely used for documents, images, and data, they are generally **unsuitable for software/models**.

7. Creative Commons

While **CC licenses** are widely used for documents, images, and data, they are generally **unsuitable for software/models**.

- They were not designed with source code and executable programs in mind.

7. Creative Commons

While **CC licenses** are widely used for documents, images, and data, they are generally **unsuitable for software/models**.

- They were not designed with source code and executable programs in mind.
- Key variations include:
  - **CC BY 4.0** (Attribution) [22]
  - **CC BY-SA 4.0** (ShareAlike) [23]
  - **CC BY-NC 4.0** (NonCommercial) [24]

Case Study: Separation of Rights Holders

8. Case Study: Separation of Rights Holders

It is crucial to understand that the license for the **code** and the license for the **weights** can be different.

8. Case Study: Separation of Rights Holders

It is crucial to understand that the license for the **code** and the license for the **weights** can be different.

Example

The Hugging Face **Transformers** library is under the permissive Apache 2.0 license [17].

8. Case Study: Separation of Rights Holders

It is crucial to understand that the license for the **code** and the license for the **weights** can be different.

Example

The Hugging Face **Transformers** library is under the permissive Apache 2.0 license [17].

However, the **license for each checkpoint** you download to use with that library is **separate**.

8. Case Study: Separation of Rights Holders

Rights and conditions differ for each set of weights.

8. Case Study: Separation of Rights Holders

Rights and conditions differ for each set of weights.

You must **check the model card and LICENSE file for each specific model** [7].

8. Case Study: Separation of Rights Holders

Rights and conditions differ for each set of weights.

You must **check the model card and LICENSE file for each specific model** [7].

For example, within the same Transformers framework:

- **DeepSeek-R1** weights are MIT [11].
- The **Mistral** family is Apache 2.0 [10].
- The **Llama** family has its own custom license [8, 9].

Model Versions and License Differences Across Versions

9. Model Versions and License Differences

Companies often release multiple models under the same series name.

9. Model Versions and License Differences

Companies often release multiple models under the same series name.

However, there is **no guarantee** that the licenses for all models in a series will be the same.

9. Model Versions and License Differences

Companies often release multiple models under the same series name.

However, there is **no guarantee** that the licenses for all models in a series will be the same.

Example (Qwen 2.5 (Alibaba))

As we saw, different models in the Qwen 2.5 series had different licenses (custom vs. Apache 2.0) [9]. You must check the license for the specific model you wish to use.

9. Model Versions and License Differences

You might be curious about names like "Qwen2.5-72B-Instruct".

9. Model Versions and License Differences

You might be curious about names like "Qwen2.5-72B-Instruct".

While this course avoids deep dives into specific models, a brief explanation is useful. Models in a series typically differ in:

- **Scale**: The number of parameters (e.g., 7B vs 72B).

9. Model Versions and License Differences

You might be curious about names like "Qwen2.5-72B-Instruct".

While this course avoids deep dives into specific models, a brief explanation is useful. Models in a series typically differ in:

- **Scale**: The number of parameters (e.g., 7B vs 72B).
- **Fine-tuning**: Whether it's a base model or tuned for a specific task (e.g., "Instruct").

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)**.

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)**.

"7B" means the checkpoint contains approximately 7 billion (7×10^9) parameters.

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)**.

"7B" means the checkpoint contains approximately 7 billion (7×10^9) parameters.

In mathematical terms, the parameters (checkpoint) are a real vector:

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

A "7B model" means the dimension D of this vector is approximately 7 billion.

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)**.

"7B" means the checkpoint contains approximately 7 billion (7×10^9) parameters.

In mathematical terms, the parameters (checkpoint) are a real vector:

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

A "7B model" means the dimension D of this vector is approximately 7 billion.

Generally, a higher parameter count increases expressive power but also computational cost.

9.2 Numerical Representation

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

9.2 Numerical Representation

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

Common formats include:

- **fp32 (single-precision)**: 32 bits. High precision, large memory. A 7B fp32 model needs about $7 \times 4 = 28$ GB.

9.2 Numerical Representation

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

Common formats include:

- **fp32 (single-precision)**: 32 bits. High precision, large memory. A 7B fp32 model needs about $7 \times 4 = 28$ GB.
- **fp16 (half-precision)**: 16 bits. Halves memory usage, but has a smaller dynamic range.

9.2 Numerical Representation

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

Common formats include:

- **fp32 (single-precision)**: 32 bits. High precision, large memory. A 7B fp32 model needs about $7 \times 4 = 28$ GB.
- **fp16 (half-precision)**: 16 bits. Halves memory usage, but has a smaller dynamic range.
- **bf16 (Bfloat16)**: 16 bits. Wide dynamic range like fp32, but lower precision than fp16. Good for training stability [25].

9.2 Numerical Representation

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

Common formats include:

- **fp32 (single-precision)**: 32 bits. High precision, large memory. A 7B fp32 model needs about $7 \times 4 = 28$ GB.
- **fp16 (half-precision)**: 16 bits. Halves memory usage, but has a smaller dynamic range.
- **bf16 (Bfloat16)**: 16 bits. Wide dynamic range like fp32, but lower precision than fp16. Good for training stability [25].
- **Quantization**: Using even fewer bits (e.g., **int8**, **int4**). Drastically reduces size and speeds up inference, often with a small accuracy drop.

9.2.1 Precise Definition of Floating-Point Numbers

The standard for floating-point numbers is **IEEE 754**. A number is represented by three parts: a **Sign**, an **Exponent**, and a **Mantissa/Fraction**.

9.2.1 Precise Definition of Floating-Point Numbers

The standard for floating-point numbers is **IEEE 754**. A number is represented by three parts: a **Sign**, an **Exponent**, and a **Mantissa/Fraction**.

A value V is defined as:

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

9.2.1 Precise Definition of Floating-Point Numbers

The standard for floating-point numbers is **IEEE 754**. A number is represented by three parts: a **Sign**, an **Exponent**, and a **Mantissa/Fraction**.

A value V is defined as:

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

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

9.2.2 Comparison of fp16 and bf16

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

9.2.2 Comparison of fp16 and bf16

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

- **fp16**: Longer **mantissa (10 bits)** $\implies$ **high precision**. Shorter **exponent (5 bits)** $\implies$ **narrow dynamic range**. Prone to **overflow** or **underflow**.

9.2.2 Comparison of fp16 and bf16

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

- **fp16**: Longer **mantissa (10 bits)** $\implies$ **high precision**. Shorter **exponent (5 bits)** $\implies$ **narrow dynamic range**. Prone to **overflow** or **underflow**.
- **bf16**: Longer **exponent (8 bits)** $\implies$ **wide dynamic range** (same as fp32). Shorter **mantissa (7 bits)** $\implies$ **lower precision**.

9.2.2 Comparison of fp16 and bf16

Example (A number representable in bf16 but not in fp16)

Large number: Consider $V = 100,000$.

9.2.2 Comparison of fp16 and bf16

Example (A number representable in bf16 but not in fp16)

Large number: Consider $V = 100,000$.

- **fp16**: The maximum representable value is 65,504. Therefore, $100,000$ **overflows** to infinity (∞).

9.2.2 Comparison of fp16 and bf16

Example (A number representable in bf16 but not in fp16)

Large number: Consider $V = 100,000$.

- **fp16**: The maximum representable value is 65,504. Therefore, $100,000$ **overflows** to infinity (∞).
- **bf16**: Has the same dynamic range as fp32. It can be **represented without issue**.

9.2.2 Comparison of fp16 and bf16

Example (A number representable in fp16 but not in bf16)

A number requiring precision: Consider $V = 1 + 2^{-8} = 1.00390625$. In binary, this is $(1.00000001)_2$.

9.2.2 Comparison of fp16 and bf16

Example (A number representable in fp16 but not in bf16)

A number requiring precision: Consider $V = 1 + 2^{-8} = 1.00390625$. In binary, this is $(1.00000001)_2$.

- **fp16**: The mantissa has 10 bits. It can **accurately represent** the ‘1‘ in the 8th decimal place.

9.2.2 Comparison of fp16 and bf16

Example (A number representable in fp16 but not in bf16)

A number requiring precision: Consider $V = 1 + 2^{-8} = 1.00390625$. In binary, this is $(1.00000001)_2$.

- **fp16**: The mantissa has 10 bits. It can **accurately represent** the ‘1‘ in the 8th decimal place.
- **bf16**: The mantissa has only 7 bits. The 8th bit does not fit and is rounded. It **cannot distinguish** between 1 and $1 + 2^{-8}$.

9.3 Instruction Finetuning

Suffixes like “Instruct” or “Chat” suggest the model has undergone **Instruction Finetuning**.

9.3 Instruction Finetuning

Suffixes like “Instruct” or “Chat” suggest the model has undergone **Instruction Finetuning**.

- A base large language model (foundation model) is trained to predict the next word.

9.3 Instruction Finetuning

Suffixes like “Instruct” or “Chat” suggest the model has undergone **Instruction Finetuning**.

- A base large language model (foundation model) is trained to predict the next word.
- Additional fine-tuning is performed using a dataset of "instruction (prompt) and desired response" pairs [26].

9.3 Instruction Finetuning

Suffixes like “Instruct” or “Chat” suggest the model has undergone **Instruction Finetuning**.

- A base large language model (foundation model) is trained to predict the next word.
- Additional fine-tuning is performed using a dataset of "instruction (prompt) and desired response" pairs [26].
- This teaches the model to be a helpful assistant, follow diverse instructions, and engage in conversation.

Practical Guidelines and Caveats

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.

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** [16].

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** [16].
- **The legal nature of checkpoints is an immature area**: It is reasonable to consider them as having aspects of both **source code** and **data**. **No legal consensus has been established** [27, 28].

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** [16].
- **The legal nature of checkpoints is an immature area**: It is reasonable to consider them as having aspects of both **source code** and **data**. **No legal consensus has been established** [27, 28].
- A policy of **"when in doubt, do not use"** is the safest approach.

Summary

11. Summary

- Many "open models" release their **architecture and checkpoints**, but the **training data and learning algorithms** are often kept private.

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**.

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**.

Next Time

12. Next Time

Next time, we will **download and run a specific open large language model locally**.

12. Next Time

Next time, we will **download and run a specific open large language model locally**.

This hands-on exercise will confirm the division of roles between:

- The **neural network part** (architecture + checkpoint).

12. Next Time

Next time, we will **download and run a specific open large language model locally**.

This hands-on exercise will confirm the division of roles between:

- The **neural network part** (architecture + checkpoint).
- The **surrounding components** (tokenizer, pre/post-processing, inference runtime).

Reference i

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

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

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

Reference ii

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

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

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

Reference iii

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

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

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

Reference iv

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[25] 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.

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

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

[28] 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.