**2. Open Source Investigation**

**2.1 Detailed findings from the group's research**

Open-source licenses are key in the open-source community because they make collaboration possible, ensure creativity, and set limits for both users and contributors. Research revealed that licenses help shape how members of the community participate and manage their work [1]. As a result of the MIT License’s permissiveness, many people can work together and modify the software easily, promoting a team spirit [3]. In the open-source world, copyleft licenses, such as the GPL v3, share and protect the software by requiring openness from revisions and may prevent some people from contributing, for example, commercial applications [1]. Because of these licenses, contributors are able to share and modify code with fewer restrictions, but this is still essential for getting involved in open source.

**2.2 Explanation of open-source licenses**

An open-source license ensures that open-source software complies with the rules from the Open-Source Initiative (OSI), and that it can be used, changed, shared, and distributed. Such licenses must include the permission to use, modify, and share software for free and protect developers’ rights [1]. Normally, these licenses contain a rights grant, explain the rules of use, exclude warranties, and state the conditions of liability [2].

**2.2.1 Common Open-Source Licenses**:

* **MIT License**: Created at MIT, this permissive license encourages unrestricted usage, modification, and distribution, requiring that the copyright notice and permission notice be kept intact [3]. Because it is both simple and adaptable, it is widely used in the creation of repositories of curated resources.
* **Apache 2.0 License**: Permissive, it includes patent grants and mandates documentation of changes, suitable for enterprise-backed projects [4].
* **GNU General Public License (GPL) v3**: A copyleft license, it requires derivative works to remain open-source, emphasizing software freedom [1].
* **BSD Licenses**: Permissive, available in 2-clause and 3-clause variants, they are flexible for both open-source and proprietary use [4].

**2.3 Overview of different ways of making contributions：**

Contributing means understanding the various ways to participate, each of which helps with a project’s creation and management [5]. To take part in open-source projects effectively, you must understand the primary ways to contribute [5]. Here, we share the primary methods that people use to contribute, for example, by coding, documenting, fixing bugs, and managing the community.

* **Code Contributions:** Writing, changing, or reviewing code can be done to add features, repair bugs, or make a project more efficient. The usual process for contributors is to work on a branch, make changes, and submit their work for a review after forking the project [6].
* **Documentation:** peoples can contribute to the project documentation by creating or updating README files, tutorials, or API references. If documentation is clear, it improves the experience of users and makes it simpler for new people to contribute [5].
* **Bug Reports:** It is always beneficial to identify and report bugs, especially if you are beginning to work on a project. When reporting a bug, include a detailed explanation, the steps to reproduce it, and what is expected instead of the bug, which you should add to the issue tracker of the project [6].
* **Community Management:** Forum discussions and event planning are a few of the responsibilities, along with ensuring that everyone interacts positively and that communication is clear. Community managers encourage a warm setting and ensure that members communicate clearly [5].

In order to efficiently achieve the open-source practice goals of this course, our team conducted a systematic investigation and screening on "how to find suitable projects, how to participate in the open-source community, and how to actually carry out contributions". Finally, we selected [Awesome Deep Vision] from a large number of projects as our contribution target. Below is a comprehensive analysis of our process from project search to final determination.

2.4 Suitable projects finding process

2.4.1 Project Screening Logic and Adaptability Analysis

At the initial stage of the project selection, we did not rely directly on GitHub's popular recommendation list or existing lists. Instead, we started from reality and established the "three criteria": contribution feasibility, completion feasibility, and improvement feasibility, as the core indicators for evaluating the adaptability of open-source projects.

The team members have diverse backgrounds, with some having experience in implementing deep learning algorithms and being familiar with visual model structures and common datasets; other members are proficient in web development, script toolchains, and document organization. This means that we can undertake both low-level code modifications and various contribution forms such as literature maintenance, document optimization, and automated script construction. Therefore, we consciously avoided the following types of projects:

* Projects with high technical barriers and limited contribution windows: such as kernel development, system low-level modules, and hardware adaptation projects. These projects require a long time for reading and debugging, and it is extremely difficult to complete effective PRs within the course cycle. Example: <https://github.com/torvalds/linux>
* Projects that have stopped maintenance or have closed contribution channels: for example, certain firmware or hardware platform repositories, although publicly available, have long been without responses in PRs, making contributions difficult to be accepted. Example: <https://github.com/raspberrypi/firmware>
* Industrial-level frameworks or projects led by large organizations: such as some compiler tools supported by large companies or frameworks strictly following CI/CD processes. They have extremely high contribution norms and are not suitable for rapid involvement by learning-oriented teams. Example: <https://github.com/llvm/llvm-project>

Ultimately, we focused our target on project types with clear structure, high technical compatibility, clear documentation orientation, and relatively open communities, especially preferring projects maintained by individuals or small teams that allow "lightweight substantive contributions". These projects not only have room for content updates and structure optimization but also form a community feedback loop, facilitating our mastery of the open-source collaboration process.

2.4.2 The Process of Selecting Suitable Projects

During the project search process, we adopted the following strategies:

Keyword combination search: Use keywords such as "awesome deep learning vision", "open contribution", and "computer vision curated list" on GitHub for filtering, and conduct preliminary selection based on the number of stars and the level of active maintenance;

* Tool-assisted evaluation: Utilize tools like GitHub Insights, Commit Activity, and Issues Tracker to quickly assess the health and response mechanism of the projects.
* Comparison of literature and courses: Refer to the deep visual tasks covered in current courses (such as classification, detection, 3D reconstruction, etc.) to ensure that the project topic has a high degree of overlap with the learning content of the courses.
* Team meeting scoring mechanism: Develop a scoring table with three items: "technical relevance", "contribution space", and "community response", and compare each item of the initial selected projects one by one. Then make the final decision through team voting.

2.4.3 Reasons for Selecting the Final Project

The [Awesome Deep Vision] project ultimately stood out for the following reasons:

* Relevant theme: The project focuses on the classification and resource integration of deep learning methods in machine vision, covering most of the task scenarios we have learned and studied;
* Broad content but lagging maintenance: This project was widely cited by the community, but it has not been systematically updated in the past two years, with issues such as broken links, missing literature, outdated directory structure, etc., and there is a clearly visible space for optimization;
* Diverse contribution forms: The project supports non-code contributions, such as resource updates, classification supplements, automatic scripts, document standardization, etc., which align with the diverse skill sets of our team members;
* Friendly license and open structure: The project adopts the MIT license, has low collaboration barriers, is convenient for derivation and redistribution, and the maintainers have shown a good willingness to respond in historical PRs;

In summary, this project not only has a clear problem orientation and feasible contribution paths, but also can exercise our ability to interact with real open-source communities, and is a highly compatible point between the course goals and personal growth paths.

**2.5 process for submitting contributions**

The typical process for getting your changes into an open-source project, often assisted by GitHub [6], consists of the following: creating a copy of the project repository, cloning it locally, branching your work, coding or editing the project, and committing your adjustments. The first stage is forking the project repository and cloning it locally, next by creating a branch to avoid anything changing in the main code. After that, contributors work on changes by programming code, updating the documentation, reporting issues, or helping with community management, all while committing the modifications with specific messages. After making the changes, contributors push them to their forked repository and submit a pull request with details of the contribution and any relevant problems. Once improvements are added and the pull request is sent, the maintainers can provide feedback, so it is suitable for merging. With the help of community guides, contributors are able to gradually develop the project while meeting its rules [5].

Reference:

[1] Open Source Initiative, “Licenses & Standards,” [Online]. Available: https://opensource.org/licenses  
[2] D. Riehle, “Free and Open Source Software Licenses Explained,” *IEEE Computer*, vol. 52, no. 6, pp. 82-86, June 2019.  
[3] R. G. Kula, C. De Roover, and D. M. German, “A Large-Scale Empirical Study of Open Source License Usage,” in *Proc. 46th Int. Conf. Softw. Eng. (ICSE)*, Lisbon, Portugal, 2024, pp. 1-12.  
[4] M. Kapitsaki and G. Charalambous, “Open Source Licenses and the Creative Commons Framework: License Selection and Usage,” *J. Softw. Eng. Res. Develop.*, vol. 7, pp. 1-15, 2019.

[5] D. Riehle, “The Economic Motivation of Open Source Software: Stakeholder Perspectives,” IEEE Computer, vol. 40, no. 4, pp. 25-32, Apr. 2007.

[6] GitHub Docs, “Contributing to Open Source,” [Online]. Available: https://docs.github.com/en/get-started/exploring-projects-on-github/contributing-to-a-project

Bibtex格式：

[1]

[2]:

@article{ballhausen2019free,

title={Free and open source software licenses explained},

author={Ballhausen, Miriam},

journal={Computer},

volume={52},

number={6},

pages={82--86},

year={2019},

publisher={IEEE}

}

[3]:

@inproceedings{wu2024large,

title={A large-scale empirical study of open source license usage: Practices and challenges},

author={Wu, Jiaqi and Bao, Lingfeng and Yang, Xiaohu and Xia, Xin and Hu, Xing},

booktitle={Proceedings of the 21st International Conference on Mining Software Repositories},

pages={595--606},

year={2024}

}

[4]:

@article{lin2006open,

title={Open source licenses and the creative commons framework: License selection and comparison},

author={Lin, Yi-Hsuan and Ko, Tung-Mei and Chuang, Tyng-Ruey and Lin, Kwei-Jay and others},

journal={Journal of information science and engineering},

volume={22},

number={1},

pages={1--17},

year={2006},

publisher={Citeseer}

}

[5]:

@article{riehle2007economic,

title={The economic motivation of open source software: Stakeholder perspectives},

author={Riehle, Dirk},

journal={Computer},

volume={40},

number={4},

pages={25--32},

year={2007},

publisher={IEEE}

}

[6]: