Exploration of Solution for AI Reproducibility

Zhenxing Wang

Report for the Master of Science in Pathway
from the
University of Surrey

Department of Electronic Engineering
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, Surrey, GU2 7XH, UK

May 2021
Supervisors: Prof. Mark Plumbley, Andres Fernandez

©Zhenxing Wang 2021

DECLARATION OF ORIGINALITY

I confirm that the project dissertation I am submitting is entirely my own work and that any material used from other sources has been clearly identified and properly acknowledged and referenced. In submitting this final version of my report to the JISC anti-plagiarism software resource, I confirm that my work does not contravene the university regulations on plagiarism as described in the Student Handbook. In so doing I also acknowledge that I may be held to account for any particular instances of uncited work detected by the JISC anti-plagiarism software, or as may be found by the project examiner or project organiser. I also understand that if an allegation of plagiarism is upheld via an Academic Misconduct Hearing, then I may forfeit any credit for this module or a more severe penalty may be agreed.

MSc Dissertation Title: Exploration of Software Solutions for AI Reproducibility

Author Name: Zhenxing Wang

Author Signature： Date: 31/08/21

Supervisors: Prof. Mark Plumbley, Andres Fernandez

WORD COUNT

Number of Pages: 38
Number of Words: 11998

ABSTRACT

Artificial intelligence research faces a reproducibility crisis, mainly due to the increased complexity of the experimental setups involved. This slows down progress and poses some risks. As a result, it has gained more attention in the last few years.

In this paper, we discuss the causes of the reproducibility problem and the technical and non-technical solutions proposed to solve it and we also discuss the effects and benefits of each solution. Subsequently, we have reviewed the four challenges at DCASE (Detection and Classification of Acoustic Scenes and Events), taken a closer look at the issue of reproducibility, gain insight into the reproducibility problem and summarised our recommendations for improving reproducibility.

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

*The report quotes from the interim report, so there is some repetition.

As the fields of artificial intelligence (AI) and machine learning (ML) have grown in recent years, so have the calls from many academics that AI/ML is currently in a reproducibility crisis. The reproducibility of scientific experimental results is a cornerstone in science, therefore, the scientific community has encouraged researchers to publish their contributions is verifiable and understandable. In computer science, reproducibility typically requires that researchers expose code and data so that the data can be analysed like the original work described in the publication. With the development of Artificial Intelligence, a large number of research papers on AI were published and academics wanted to use these published papers for further research, but the inability to reproduce the original papers became the first challenge to scientific progress, therefore improve AI reproducibility will be the very important thing. Reproduction is essential to prove that the information produced by the experiment can be used consistently in the real world and that non-random results are obtained. An algorithm tested only by its creator is unreliable and irresponsible, and others should get the same results when run, or significantly closer results when run on another computer or when different data is entered.[1] Compared to traditional computer science and technology, AI, it is a new technical science that studies and develops theories, methods, techniques and applied systems for modelling, extending and expanding human intelligence. A computer or software program (often referred to as a program), on the other hand, is a set of instructions that instruct a computer or other information-processing capable device at each step of its operation, usually written in some programming language and running on some target architecture. Therefore, artificial intelligence is more complex and involves more mathematical knowledge and data than traditional programs.

We need to know the definition of reproducibility if we want to solve it. Many definitions of reproducibility have been summarized in the findings of many previous studies.

We use the definition of 'reproducible' given in the National Academies of Science, Engineering and Medicine 2019 report 'Reproducibility and Replicability in Science' as the guidance [32]:

> *"We define reproducibility to mean computational reproducibility – obtaining consistent computational results using the same input data, computational steps, methods, code, and conditions of analysis; and replicability to mean obtaining consistent results across studies aimed at answering the same scientific question, each of which has got its own data. In short, reproducibility involves the original data and code; replicability involves new data collection and similar methods used by previous studies."*

During the experiment, what prevents AI replication from happening as intended is the lack of access to three things: code, data and hardware.

This paper presents the background of the field of AI reproducibility, which includes a summary of problems and solutions of code, data and hardware. We summarize various problems encountered by previous scholars in the research process, and group them into two categories. Meanwhile, the solutions are divided into technical and non-technical parts, where the non-technical part is divided into policies, laws, awards, etc., and the technical part is divided into tools and cloud platforms. Also, a community study was conducted based on the original program.

More detailed summaries of solutions. In non-technical part, we introduce Pineau's checklist, checklist contains what information a paper should contain. This information is beneficial for reproducing the particle. The checklist is also used as one of the review standards of the paper review team. At the same time, there is also the platform provided by DCASE (Detection and Classification of Acoustic Scenes and Events). On the DCASE platform, there are many challenges related to paper reproduction, and these challenges also greatly encourage researchers to reproduce papers. Secondly, Raff also proposed quantitative approaches, which are also very helpful to restoring existing ones. Furthermore, Stodden also proposed a legal framework and sharing policy, constraint in laws, license and policies.

Also, there are many tools that help reproducible, such as Sumatra's library and DCASE's fancylogger, which can be logged with these two tools or anything the author used to, the core purpose of any form of logging is to record as completely as possible. There is also introduce Model Zoo and PyTorch Hub, which can be used for automated replication. There is also introduced Conda for software environment management, and docker for control, etc. The most important thing is we also introduce the cloud platform solution in great detail.

In the meantime, it is important to introduce the DCASE community and DCASE utilities. Building a community for AI is something that can help with reproducibility, and in a survey by Stodden, many people pointed out that they share in order to build a community, a survey by Stodden showed that the sharing behavior of scientists[3]. Among the motivations for being willing to share data. In summary, scientists are active contributors to the community and are willing to build it by sharing code. [3] DCASE is an audio research community that has experienced widespread use and success with AI methods since the advent of deep learning. [2] With the reproducibility crisis facing AI, these researchers may also face the problem of reproducibility in AI. Moreover, the relevance and size of this community are growing rapidly, DCASE is now well known to scholars worldwide, and the competitions offered by DCASE attract a vast number of talented scholars. So we reproduce it based on the challenges of DCASE.

Also, as an acoustics-focused community, DCASE utilities provide the tools needed to conduct acoustics-related research, such as containers, dataset processing, and methods related to features, significantly improving the efficiency of research. In order to experience the actual reproduction

process with the DCASE utilities, we chose DCASE2020 Task1b, DCASE2016 Task1&Task3 as the reproduction target. We will use previous literature and our replicated experience to summarize a hypothesis, conduct research in the DCASE community based on the summarized and obtain conclusions, and then provide relevant solutions based on the findings.

Ultimately, based on a review of previous solutions and a replication of the DCASE community challenge, a summary of ideas for improving reproducibility was used to enhance AI reproducibility.

1.1 Background and Context

Compared with other fields such as biology, physics, or sociology, which conduct experiments in the natural or social world, the reliability and reproducibility of experiments in computer science should be more robust. Because to the common mind, a computer is a fixed thing, similar to if we input one plus one, then the computer will give us back a result that equals two. Unlike, for example, biology, which has just been mentioned, because there is so much variation and uncertainty in those disciplines. So, it was thought that the computer should be something more certain, more precise.

In the field of computer science, although many early discoveries came from mathematics and theoretical analysis, in recent years, new knowledge has increasingly come from practical experiments [4]. Most of the experimental equipment is designed and manufactured by humans. However, it is essential for the discovery to be effective and reliable that the experimental process is reproducible and draws consistent results and conclusions. However, this highly controlled environment will proceed step by step according to people's program settings and then output the result. However, unexpectedly, as the complexity of the setups increases, it becomes more difficult for researchers to reproduce the work of others [5][6]. This became particularly serious with the advent of Deep Learning and the latest advances in AI, where the complexity of the involved setups exploded.

Collberg and Proebsting tried to execute the code published in 601 papers, which shows this without communicating with the author; their efforts achieved a 32.1% experimental success. When communicating with the author, their efforts achieved a 48.3% success rate[7]. A 2016 survey in the journal “Nature” showed that more than 70% of researchers failed to try to reproduce another researcher's experiment, and more than 50% of researchers failed to reproduce their experiments [8].

At the report from the NeurIPS (Neural Information Processing Systems) 2019 Reproducibility Program, Pineau introduced the current "reproducibility crisis" in the field of scientific research in the paper "Improving Reproducibility in Machine Learning Research." The paper shows some survey results, which can be seen very intuitively to reproduce the impact in scientific research. In a survey for 1,500 scientists conducted by "Nature" magazine [8] 90% of the interviewees believed that the "reproducibility" problem was a crisis in the field of scientific research. Among them, 52% of the respondents thought this problem was serious. Similarly, Edward Raff, in "A Step Toward

Quantifying Independently Reproducible Machine Learning Research", [14] a total of 255 papers published from 1984 to 2017 were reproduced, 162 (63.5%) were successfully reproduced, and 93 were not successfully reproduced.

Concern has grown in the scientific community related to the reproducibility of scientific results and the concern is not unjustified. In 2015, DeepMind's Volodymyr Mnih, Koray Kavukcuoglu et al. published a paper "Human-level control through deep learning in the journal Nature" [9], which proposed a model named DRL, since then, DRL (Deep Reinforcement Learning), which combines DL (deep learning) and RL (reinforcement learning), has quickly become the focus of the artificial intelligence community.

However, researchers have begun to reproduce on DRL but did not achieve good results because the published literature does not provide details of essential parameter settings and engineering solutions. Many algorithms are difficult to reproduce. In September 2017, the research group led by well-known Reinforcement Learning experts Henderson et al. published the paper "Deep Reinforcement Learning that Matters" [10]. They pointed out the problem directly and said that they studied reproducibility in deep RL, and they found that both the internal and external sources of uncertainty may cause difficulties in reproducing the baseline algorithm. This article has aroused enthusiastic responses in academia and industry. Many people agree with this and have strong doubts about the actual capabilities of DRL.

This is not the first time that the research team challenge DRL. In July 2017, the research group used sufficient experiments to study multiple factors that make the DRL algorithm challenging to reproduce. And then, they made a report titled "Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control" which showed in detail through examples in the process of reproducing multiple policy gradient-based algorithms. Reproduce difficulties caused by uncertain factors. [11]

In 2020, it was reported that the algorithm used by McKinney et al. for breast cancer screening using AI is non-reproducible. The article believes that the nuances of the computer code may have a significant impact on the training and evaluation of the results and may lead to unexpected consequences. Therefore, the transparency of the actual computer code form used to study the model and arrive at its final parameter set is critical to the reproducibility of the study. [12]

Reproducible problems are not limited to a small number of small-scale machine learning research teams. Even large companies that spend a lot of money are often unable to verify the results of their papers. Last year, the Yann LeCun publicly questioned that Google Brain's papers could not be reproduced [13]. Many top researchers in the NLP field were unable to reproduce the results obtained by Google Brain's language model Transformer-XL, which once caused heated discussions in the community. [13]

To explore what went wrong in the reproduction process, we need to start with the entire reproduction process. First of all, researchers would be interested in some papers that reach the new State-Of-The-Art (SOTA). And then decide to start reproducing. The next step is to search for the project code and run the code on the data set used by the author. At this time, the researchers need to check that the project has the following elements [15]

1) The research paper: including all the source files from which the manuscript was built;
2) The experiment: Code and instructions used in the experiment include all source code; Documentation using any code, including Pseudocode; clear parameter list, settings and dependencies used to achieve the results described in the paper; and the description of the experiment;
3) The data: In addition to the complete original data, a document about the data is also needed. The document needs to describe various contents about the data, such as the source of the data, the composition of the data, how the data is used in the research, etc.;
4) The result of the experiment: About the various results of this experiment, and the interpretation of these results, etc.;

Understanding these can successfully reproduce the SOTA results in the paper. But the process of reproducing was not satisfactory. There are some common problems when reproducing a machine learning paper [16]:

1) Incomplete or missing README;
2) Undeclared dependencies, buggy code, and missing pre-trained model;
3) Undisclosed parameters;
4) Private Dataset or missing pre-processing steps;
5) Unrealistic requirement for GPU resources.

At the same time, the "Deep Reinforcement Learning that Matters" explained in detail the problems that occurred in the reproduction of machine learning papers.[10] For example:

1) There is no way to access the training data, or the way the data is split different;
2) Insufficient or wrong code required to run the test;
3) Inadequate index specifications for report results;
4) Select only part of the report, or exaggerate the conclusion.

Furthermore, Edward Raff summarized the reasons for the failure to reproduce successfully, for example [14],

1) Unclear symbols or language only explained part of the algorithm and did not say it in a way that is easy for readers to understand, but said it very obscure;
2) Lack of algorithm steps or details;
3) Many papers will formulate the need to use gradients as functions or other companies but do not specify these data;
4) Lack of hyperparameters or similar nuance details.

The lack of very detailed descriptions is the first challenge in the AI replication process. Because AI is studied with a large number of formulas and parameters, as well as a description of the software environment, the lack of description of this information is the first obstacle that prevents AI from being reproduced.

At the same time, all the problems found in the reproduction process listed above can be clearly seen as the result of incomplete (wrong) explanatory articles or codes. In addition to the subjective factors of the authors, there are some objective factors, such as the law.

There is usually no way to validate scientific statements for artificial intelligence to produce published findings without access to code and data. As a unit of scholarly communication, the explanatory narrative, code, and data used to produce the results are included. The reason for including code and data is to facilitate the generation of genuinely reproducible studies. Initiating computer replication usually means providing data, software, and scripts for generating results, and using these materials is combined with intellectual property law. [5] Software exists in two main ways, and the first is source code, the second is complied, complied for scientific communication is entirely inadequate, scientific communication is more reflected in the exchange of source code. Hence, the source code is essential for scientific development. [5]

Datasets are a critical part of AI. There are bound to be barriers to their reuse and sharing that do not come from intellectual property law, such as confidentiality and privacy concerns or ownership rights from industry or other external collaborators. [5] At the same time, for technologies that can be used in clinics and associated with human life, the standard of transparency should be increased.[17]

1.2 **Objectives**

By examining the background, we believe that the nature of the many phenomena that cannot be reproduced is due to the failure to record the experimental process in sufficient detail. So, the goal is to improve reproducibility by asking scientists to keep as detailed a record as possible. Therefore, the objectives of this thesis are divided as follows:

First, study the previous solutions and analyse the previous solutions. Secondly, by studying the

previous solution and through the experience of replication, a better solution is proposed. According to the final goal, it is broken down into the following tasks:

1) Review the pervious solutions to get a landscape view of reproducibility and summarize previous problems that can occur in paper reproduction
2) Gain a deeper understanding through the reproduction of 4 tasks and gain a real reproduction experience
3) Suggestions for improving reproducibility based on the 2) results

2 BACKGROUND THEORY AND LITERATURE REVIEW

2.1 Background Theory

To summarize, we can classify the reproducibility problems raised above. I think it can be divided into two categories roughly.

1) The first category is that the paper itself has fraudulent behaviour or have a mistake. This type of paper will never be able to reproduce the results in the paper, no matter how perfect we require it and how advanced technical means we use.

2) The second category can be summarized as a lack of sufficient information. This information can include experimental reports, including data, including wrong expressions, etc. Such papers can be reproduced, but more effort is required. What we are currently discussing more is the second situation.

Based on the summary of Raff, Pineau, Stodden, and others, and also based on the 2) above, we can broken down 2) even more to that the first reason for the failure to reproduce is that there is no very detailed record code, and the second part is that it cannot be disclosed due to special reasons.

In more detail, in developing a tool for tracking simulation experiments, something like an electronic lab notebook for computational science, there are some challenges:

1) different researchers have very different ways of working and different workflows: command line, GUI, batch-jobs (e.g., in supercomputer environments), or any combination of these for various components (simulation, analysis, graphing, etc.) and phases of a project.

2) Some projects are essentially solo endeavour, others collaborative projects, possibly distributed geographically.

3) As much as possible should be recorded automatically. If it is left to the researcher to record necessary details, some details will be missed or left out, particularly under the pressure of deadlines.

So, based on the challenges as mentioned above that scientists encounter while working, the following solutions are provided.

2.2 Non-Software Solutions

The first proposed non-software components include policies, rules, law, awards, and communities. These solutions emerged mainly from the subjective consciousness of the scientist who valued reproducibility as a problem. Reproducibility is improved through paper submission policies, intellectual property protection, rewards and challenges for reproduction, and contributions to the community

Policy: The Machine Learning Reproducibility Checklist was published by Pineau on NeurIPS. In this checklist, when a paper is to be submitted, the paper needs to include information to help reproducible work. NeurIPS introduced a reproducibility plan at the 2019 Machine Learning Research Conference, which aims to improve the entire community's behavior, communication, and standards. This program consists of three parts:

1) Code submission policy;
2) Reproducibility challenges for the entire community;
3) Machine learning reproducibility checklist as part of the paper submission process. [4]

The program of reproducibility of NeurIPS is mainly aimed at the policies set up to improve reproducibility. The reproducibility checklist is a solution to solve the reproducibility problem, and most scientists have also recognized it. At the same time, this checklist has been Integrated into arXiv.org. The detailed checklist is in Appendix1[4].

Because the checklist policy is set, therefore, the author will be investigated when they were submitting the paper. According to statistical results, 97% of the answers stated that the mathematics, algorithm, or model in the paper had been clearly described, and 3% of the answers may reflect the author's question because they want there is margin for interpretation. 89% of submitters were asked, "For all figures and tables that present empirical results, indicate if you include: A description of how experiments were run." [4]

Because of the emergence of the checklist, when scientists submit the paper, whether the content written in the paper has played a very important role in facilitating the reproduction of other researchers, this checklist not only allows scientists to pay attention to what they were concerned about the crucial experimental part that was ignored before, and because of this checklist, when the paper is being reviewed, or when it is read by other researchers, the reproducibility of the paper can also be evaluated, such as Researchers can use the checklist to determine whether the paper is difficult to reproduce or not. Reviewers can also use the checklist to make review judgments.

During the operation of this checklist, some statistical data produced encouraged those who made contributions in the field of reproducibility and proved that their efforts were not in vain. First of all, when there is no checklist, we think that the checklist will make the research field more complicated through common sense, but in fact, it did not reduce the number of people submitting papers. In 2019, the number of papers submitted to NeurIPS was increased by 40%, proving that the checklist's introduction did not lead to a significant decline in interest. Secondly, the number of authors willing to submit code has also increased significantly, from less than 50% before to 75%. At the same time, the number of participants in the reproducible challenge is rising. Finally, one-third of the reviewers believe that the answers to the checklist are helpful. Meanwhile, the reviewers will give higher scores

to the paper based on this list.[4]

The role of the checklist is prominent, Pineau's efforts have made the problem of reproducibility one step further, but the performance of the checklist is only the content review at the submission stage of the paper. In other words, it only focuses on whether the process and guidelines are compliant, and the paper is sufficient scientists described the content of the paper in more detail, which increased the probability of reproducing of the paper, the checklist is a great method. And with the Pineau policy, we can actually confirm that the problems we mentioned earlier, the lack of detailed documentation is the most basic problem that leads to non-reproducibility.

Community&Award: Because the problem of reproducibility made some researchers very disappointed, so there has been a community about reproducing the paper. One of them created a machine learning research feedback platform called "Papers Without Code" [25] to specifically publish papers that no one can reproduce. This resource website hosts more than 40,000 research implementation codes. If you cannot reproduce the paper, this question may be answered on "Papers Without Code". Researchers will submit papers that cannot be reproduced on this platform on this platform, and then the platform will contact the original author.

From this platform to the present, they have had about 43 submissions, 21 of which were approved and 10 of which were resolved due to the author submitting their code. Although this result may not seem so good, the more significant effect of this community is that it exposes more of the papers that are mostly noticed and not reproducible to the general public, greatly contributing to the importance scientists place on reproducibility.

A community called DCASE releases new tasks each year and offers prizes based on players' contributions. DCASE 2020 Challenge will provide awards for open-source and innovative methods. These awards are meant to encourage open science and reproducibility, and therefore the Reproducible System Award is directly based on these criteria. Reproducible system award will be offered for the highest-scoring method that is open-source and fully reproducible. In addition, through their Judges' Award, they want to encourage novel and innovative approaches. [30] Each task will offer two awards. These awards aim to encourage participants to openly publish their code and use novel and problem-specific approaches and are therefore not directly based on the evaluation set performance ranking.

Through Stodden's previous introduction to the scientist's love of community [3], reproducing content can contribute to the community, or present papers that cannot be reproduced, in ways that are positive for scientific progress.

Law: Stodden raises the legal issues related to the use of data, code, etc. For article publication, scientists can sell the copyright of their articles to third-party structures or publish them in open access journals with a Creative Commons license for their article at the time of publication. The most

basic Create Commons license (CC-BY), essentially, it allows unrestricted downstream use so long as attribution is given to the original author. Here we would prefer to discuss the legal perspective on data and code. Code has the most complex interactions with Intellectual Property Law since it is subject to copyright and patent. [5]

In response to the sharing of source code, there are several open code licenses that impose few restrictions on reuse beyond attribution, creating an intellectual property framework similar to traditional scientific norms. [5] Berkeley Software Distribution (BSD), for example, allows downstream use, copying, and distribution of unmodified or modified source code, but the requirement is that the license needs to follow the source code.

For data sharing, the openness of the Dataset should be evaluated before the project starts, and the data needs to be released to the maximum extent possible. The Dataset should be available in a recognized repository in the field, such as OpenML and DCASE, which provide a large number of datasets on acoustics. In the case of private data, both DataVerse and Dryad can host datasets from any domain. [5]

Reproducibility is also aided at the legal level. These rules and sharing methods allow scientists to share with confidence without fear of their legal rights being infringed.

We can feel that the reproducible problems have begun to move in a positive direction through the above solutions, but this was not enough, because it is all about help and supervision from the community, society and other people. Most importantly, we also need technology, method, specific operations to meet the requirement of reproducibility.

We can use some tools to monitor our experimental process and results, or we can use some tools to help scientists produce paper automatically so that the entire production process can be recorded and measured.

2.3 Technical Solution

In order to solve the increasing demand for the reproducibility of papers, Facebook launched PyTorch Hub, a model-sharing library similar to TensorFlow Hub. Users can submit and browse models, which greatly improves the reproducibility of papers. PyTorch Hub, a simple API and workflow, provides basic building blocks to improve the reproducibility of machine learning research. PyTorch Hub contains a series of pre-trained model libraries related to image classification, image segmentation, generation, and conversions. [19] PyTorch Hub tries to improve research in the most fool proof way reproducibility of work. How simple is it? Yann LeCun tweeted that it only takes one line of code to call all models in the warehouse [17] and publish your model through a pull request. At the same time, PyTorch Hub integrates Google Colab and integrates the paper code combined with the website Papers With Code we mention about above.[19][25]

For software version control, Conda is now more commonly used for version control. Conda is a general package management system designed to build and manage any language and any type of software. For example, package management is similar to pip, and environment management allows users to easily install different versions of python and switch quickly.

Also, the use of containers has very great benefits for reproducibility. Experimental learning and research are not easy for an individual. Specific steps to fit and compile the source code, set specific runtime options, import datasets, etc., are necessary operations. These operations can be added to the container, allowing the researcher to reproduce the content in fewer steps. Not only is this convenient for researchers, but it also reduces unnecessary errors during the reproduction process. Docker is popular in the scientific community for a number of reasons, including [24]:

1) Simplifying application packaging;
2) Increasing transparency in how applications are built;
3) Improving collaboration between researchers;
4) Supporting a consistent operating environment by ensuring repeatability.

Containers provide more consistent operating environments and scientific portability to other research applications by providing a more consistent operating environment. Scientists may want to validate results with another test, repeat a test later, or they may want to repeat their analysis with new data, and containers can also help them.

There is no complete record is the fundamental reason for the lack of repeatability in computational research. [20] At the same time, it is shown in a survey that the complete record is not done because if the automatic or semi-automatic recording is to be done, then this part of the work will increase by a certain amount. The workload requires time to learn, so scientists are not particularly willing to do this kind of thing. [20] Therefore, Sumatra is a software tool that supports reproducible computing research and helps scientists complete experimental records by automatically capturing the process details of all experiments. Sumatra automates the necessary but tedious and error-prone process, which means finding out which code version was used to generate which output. At the same time, you can compare the two results to indicate whether the code has changed. Sumatra's tools are integrated into python. It is simple and easy to use. This kind of dependency that records the experiment is not only Sumatra, but there is also a similar - *dcase_util.ui.FancyLogger()* on DCASE, which is very simple and easy to use.[21] The tools of Sumatra and DCASE are just two of many experimental recording tools. Still, both of these tools provide an excellent start to help scientists perform experimental recordings, assisting scientists in completing some complex and troublesome tasks. Let scientists focus more on their research.

Using a virtual machine to encapsulate the entire operating system and software environment is

another way.[22] We know that we will use the computer and the operating system as long as we do research and development. Then we can use virtualization software, such as VMware, to separate a part of the PC. It is a virtual machine, which can keep all the research and development logs to a great extent. At the same time, because the virtual machine can be customized for configuration, such as setting memory, setting storage space, etc., the hardware conditions will no longer be the reason for non-reproducible. After the development, the virtual machine can be exported to OVF format [26], and then other scientists can reconstruct the same computing environment based on this file. Moreover, because of the emergence of cloud computing services, such as Azure, AWS, and other cloud computing providers, the virtual machines provided by them are cheap and can be customized to meet the requirements of various GPU, CPU, storage, and operating systems.

The idea of reproducing based on the cloud platform Bill Howe has also done in-depth research in "Reproducibility, Virtual Appliances, and Cloud Computing". The application of virtualization to support repeatable research does not require any significant changes in the researcher's workflow. The same experiment can be performed in a virtual environment and a physical environment, using the same code, data, and environment. After the experiment is completed, the experimenter will save a snapshot of the virtual machine, make it publicly available, and reference it in all appropriate places. Paper readers who want to reproduce the results can start their own instance author's virtual device without incurring additional costs to the author (and only cause minimal costs to the copier) and re-execute the experiment. In addition, the experiments, modifications, and extensions implemented by the new replicator can be saved in a new virtual machine image and re-shared as needed [23].

Under this solution, the most significant advantage is that it can capture as many variables as possible. The virtual machine allows researchers to share their entire Research and Development environment, including data, code, logs, usage history, intermediate results, numbers, notes, failed experiments, operating system configuration details, and more. This is a complete laboratory environment, more accurate.

In addition to the traditional use of virtual machines as a solution, copy the virtual machine image, or call the interface can solve the problem of reproducibility. Many machine learning platforms are beginning to be accepted by everyone nowadays, and closed-source machine learning platforms can help with reproducibility. Because on these machine learning platforms there exists an AI ecosystem that includes not only hardware support but also a lot of software to support the development of AI. For example, CodaLab, Kaggle, etc., and solutions providers such as Google, Amazon, Microsoft, etc., also use machine learning services as their cloud products.

"Out-of-the-Box Reproducibility: A Survey of Machine Learning Platforms" which is mention in the features, is Notebook, Source code management, Software dependencies, Workflows, Data repositories, Hardware specifications, Experiment citation.[27] These are the criteria for measuring

whether a paper can be simply reproduced. It is proposed a quantitative method to alleviate this problem. It starts from the three aspects of Experiment, Method, and Data, decomposes these three aspects into variables, and then evaluates and scores each variable. Its quantitative method evaluates and compares the current platform of machine learning and has concluded that BEAT and Floyd hub provide the best support for reproducibility. The most commonly used machine learning platforms offered by large companies, Microsoft, Amazon, and Google, have poor support for reproducibility.

But actually, considering the market share, AWS and Azure should be used more as the target platform. According to Synergy Research Group, 'Amazon and Microsoft Maintain their Grip on the Market but Others are also Growing Rapidly' [28], Amazon and Microsoft, as the two cloud platform providers with the largest market share in the world, have guaranteed technical support, technology ecology, and software maturity, and because of their commercialization, all types of services are available at discounted prices. When computing demands exceed affordable local workstations, cloud-hosted environments offer the well-known benefit of being able to perform high-performance computing without a physical cluster. Less obvious, especially to the uninitiated, is the transformative potential of cloud-hosted environments in bridging the digital divide between cash-poor and resource-rich labs, and in enhancing modern research groups with remote personnel and trainees.[29]

The work we reproduce later is based on Azure Machine Learning Platform, where the GPU model used is NVIDIA TESLA K100, the retail price is £5,400, but the price for using its services on Azure is $2.58/hour. Meanwhile, the VM also provides 56G RAM and 360GB Disk are included in the prize. Another benefit of using cloud computing is that currently, in the COVID-19 environment, many companies and schools are choosing to work from home, which leads to collaboration issues in the research process. If you use VPN to connect with the local server room, there will be network instability and speed. If the experimental environment is deployed to the cloud platform, this problem can be solved, and the collaboration between members of the laboratory will be more convenient.

2.4 Summary

Above, we did two things. First, we studied the problem of reproducibility and obtained the main causes of non-reproducibility from the original papers, blogs, and other sources and grouped them into two categories. Second, we studied the previous solutions, divided into non- software solutions, which can be done by policies to require scholars' papers, by laws to protect scholars' intellectual property, by communities to promote active sharing behaviors, and by awards to motivate scholars to conduct research. Software solutions are also discussed. The use of automated replication tools and documentation tools for complete documentation of experimental content, the use of Conda and Docker for software environment configuration, and cloud-based solutions are also discussed.

3 TECHNICAL CHAPTER

To better understand the issue of reproducibility, I reproduced the DCASE2020 Task1b baseline, DCASE 2016 Task1 baseline and DCASE 2016 Task3 baseline from the DCASE challenge.

Acoustic Scene Classification (ASC) is concerned with correctly identifying real-world sounds to a given set of environments, such as subway stations, street traffic, or public squares. The sound of an acoustic scene contains a lot of information and rich content, which makes accurate scene prediction difficult, thus making it a fascinating research problem. As a result, ASC has been an attractive research area. For decades, the DCASE Challenge [33] has provided benchmark data and a competitive platform to facilitate the study and analysis of acoustic scenes.

In DCASE 2020 Task1, there are two different subtasks in Task1. Task 1a focuses on the robustness of ASC systems. The goal is to facilitate research to address device mismatch, which is a common situation in ASC applications. The main objective is to design a device-invariant system that can well classify ten scenes of audio recorded by different devices. Good classification of ten scene audios recorded by other devices without using any device information during the evaluation phase. Task 1b focused on the model of the ASC system Scale of the ASC system. [34]

In DCASE 2016 Task1 and Task3, Task1 provides 15 scenes and classifying the sounds of the 15 scenes is the main objective of this task. With regard to task3, TUT provides data for a wide range of scenes, divided into indoor and outdoor, with indoor data such as banging, birds singing and outdoor data such as Glass jingling and people walking. The main objective is to classify them. [39]

We will use DCASE library and container for replication. DCASE Models, an open-source Python library for rapid prototyping of environmental sound analysis systems, focus on deep learning models. In addition to a collection of functions for dataset processing, data preparation, feature extraction, and evaluation, it includes a model interface to standardize the interaction of machine learning methods with other system components. This also provides an abstraction layer that can reduce the use of different machine learning backends.[31]

The main goal is to facilitate all aspects of the typical research pipeline for DCASE-related problems, emphasizing deep learning models. The library has been carefully designed to be easily extended and integrated with other software tools. It provides an abstraction of everyday tasks such as data preparation, data augmentation, feature extraction, model training, and evaluation which is shown in figure 1. This allows rapid prototyping of new methods and simplifies the work required to release (and maintain) the code for a new model. In addition, the library aims to provide several baseline reference implementations, including already trained models, to facilitate the comparison of methods. In this way, we strive to provide a low barrier to entry for students and researchers new to the field. [31]

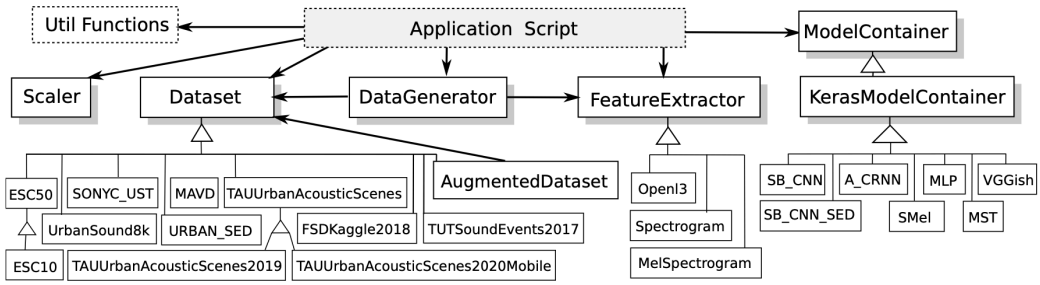

Figure 3-1 Class diagram of DCASE-models [31]

3.1 Experimental Design and Methods

3.1.1 Experimental Process Design

The process set up for the reproduction work is as follows:

1） Review and evaluate the target results. The implemented results are first analysed to see if there are any problems with the results. There is an EVALUATION dataset on DCASE, which shows the accuracy of the participants' models. Overfitting can occur in the deep learning process, and this is shown on the development dataset, where the model is exceptionally accurate. Still, the performance on the evaluation dataset differs significantly from the performance on the development dataset.

2） Review and collect relevant code and data. As mentioned in 2.1 above, before reproducing the article, we need to check whether all the files required for reproducing the article exist, including but not limited to the README, dependencies and dataset. These files will help us to understand how the author thinks and how the program works.

3） Test run the original code and correct the written code to run the computational experiments. Running the original code is to first ensure that the original code runs error-free, which helps us understand the logic of the author's code writing and experiment with the author's code to gain a deeper understanding of the author's intent for each line of code.

4） Analyse the code and re-code it. In these reproduction work, our work chain is divided into dataset processing in Data Loading - Setting Up Models - Training Models - Model Optimization - Saving Models. So, ideally, if the authors provide the source code, we can first find the location of these modules on the source code.

5） They are automating computational experiments and visualization. After we find the position of each module, we then determine the parameters that the authors set on the model. These parameters are important for our training. The code is then rewritten to run the computational experiments and visualizations to generate figures and tables that match exactly or as close to them as possible exact matches, or as close as possible, to the figures and tables in the

original article.

3.1.2 Workspace Introduction

I put the experimental platform on the Microsoft Azure Machine Learning Platform for the experiment. Originally, we were going to put the development platform on a virtual machine, configure the virtual machine from scratch, build the environment, and provide a virtual machine image for user access at the end of the replication according to our solution above, but we were more willing to try new technologies than the current popular machine learning platforms, so we made the task of this replication in Azure Machine Learning Platform.

Azure Machine Learning can be used for any kind of machine learning, from classical ML to deep learning, supervised and unsupervised learning. Whether you prefer to write Python or R code with the SDK or work with no-code/low-code options in the studio, you can build, train, and track machine learning and deep-learning models in Azure Machine Learning Workspace.[35]

Why choose Azure's machine learning platform as a reproduction and research platform?

1) Fully functional. Azure ML has integrated Jupyter Notebook, Docker, pipeline, Dataset, environment control, and other functions, which can manage our development process very efficiently.

2) Computing power. GPU-based virtual machines are available, which is very friendly to the average student because not all students can afford the high cost of GPUs. And Azure provides virtual machines at lower prices and with student discounts for general training, which is also the trend for enterprises or academic institutions to use public cloud services.

3) Data management. By creating a dataset, a reference to the location of the data source is then created, along with a copy of its metadata. Since the data remains in its existing location, no additional storage costs are incurred, and there is no risk to the integrity of your data sources. Also, Azure provides some public data and data annotation services.

4) Snapshots and logs. When we submit a run, Azure Machine Learning compresses the directory containing the script into a zip file and sends it to the compute target. The zip file is then decompressed, and the script is run there. Azure Machine Learning also stores the zip file as a snapshot as part of the run log. Anyone with access to the workspace can browse the run log and download the snapshot. The logs can be used to monitor the status of the run in real-time or to view the results when finished, or you can use any other logging method you prefer.

5) Share. As mentioned above, after training, we can package the project and create images that provide open interfaces that other scholars can access. [36]

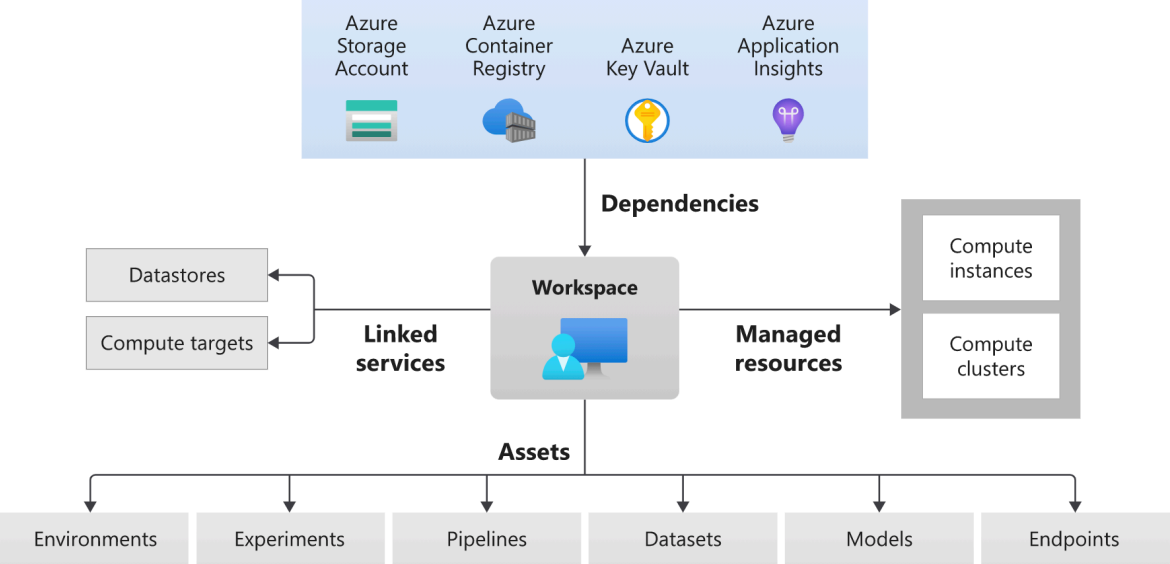

Figure 3.1.2-1 The Architecture of Azure Machine Learning [36]

3.2 Experimental Details

3.2.1 Review and Evaluate the Target Results

Check the information contained in the author's README file before starting the experiment. The author provides the following file in this experiment, as shown in figure 4.2.1-1, which contains the source code file, configuration file, README file, and requirement. [40] The README contains the dependencies that are needed to run this experiment.

In this task, the authors provide the structure of the file, which includes the source code with the README file. At the same time, the role of each file is also clearly explained. The Dataset for this task is also provided, divided into the development dataset and the evaluation dataset. Development is used as the development dataset for the challenger, and the evaluation dataset is used to measure the model provided by the challenger. Details of the software environment are also offered and also uses the Conda we mentioned above.

In the same approach as for task1b, we analysed task1 and task3 of DCASE 2016 as well, using README files or technical reports to find elements that would help us to reproduce them, such as file structure, package version information, operation environment, relevant results, etc.

```
.
├── task1a.py                    # Baseline system for subtask A
├── task1a.yaml                  # Configuration file for task1a.py
├── task1b.py                    # Baseline system for subtask B
├── task1b.yaml                  # Configuration file for task1b.py
|
├── extra.yaml                   # Example file to show how to extend the system through configuration
├── extra.yaml                   # Example file to show how to run multiple configurations
|
├── model_size_calculation.py    # Utility function for calculating model size for subtask B
├── utils.py                     # Common functions shared between tasks
|
├── README.md                    # This file
└── requirements.txt             # External module dependencies
```

Figure 3.2.1-1 The File Structure

```
conda create --name tf-dcase
conda activate tf-dcase
conda install ipython
conda install numpy
conda install tensorflow-gpu==1.14.0
conda install -c anaconda keras-gpu=2.2.4
conda install -c anaconda cudatoolkit
conda install -c anaconda cudnn
conda install librosa
pip install absl-py==0.9.0
pip install dcase_util==0.2.12
pip install sed_eval
```

Figure 3.2.1-2 The Environment Information

Scene class	Accuracy	Log loss
Indoor	82.0 %	0.680
Outdoor	88.5 %	0.365
Transportation	91.5 %	0.282
Average	**87.3 %** (+/- 0.7)	0.437 (+/- 0.045)

Figure 3.2.1-3 The Results for The Development Dataset

3.2.2 Review and Collect Relevant Code and Data

In this section, we will look at and analyse the source code and the associated Dataset. The file structure shows that the files provided by the authors include the python source code of task1b and the docker file used to control task1b. Since task1b duplicates the functional modules of task1a, the authors also imported the same parts from task1a to task1b, so we need to Analyse them in conjunction with task1a when reproducing the work. What we need to understand when we analyse the code is the author's code logic. Only if we understand the author's logic can we fix the code better if it runs with errors.

Figure 3.2.2-1 Workflow

When analysing the code, we mainly analyse it according to the module, and the analysis process is shown in the following figure 3.2.2-1. We use same approach for DCASE 2016 task1 and task3.

1) The first step is data storage, which is usually achieved by organizing data on a disk according to file naming conventions and directory structures. The data for this task were collected by Detection and Classification of Acoustic Scenes and Events and Tampere University, Audio Research Group, data hosted at Zenodo, DCASE provides the interface on this Dataset.
2) The second and third steps are feature extraction and feature normalization. The purpose of this step is to extract the features of the audio file using the method provided by DCASE library. Also, DCASE has a function for normalization, so we can easily normalize the data.
3) The fourth step is learning, in which the architecture of the model is confirmed. In this task, CNN is used, so we need to know how many layers of CNN are and the parameter settings of each layer, such as convolution kernel, activation function, etc.
4) Steps 5 and 6 are for testing and evaluation.

3.3 Experimental Result and Insight

3.3.1 Experimental Result

The results of task1b were successfully reproduced by the description of 3.2.2. The result given by the authors is 87.3% (+/-0.7), and this reproduction task gets 86.6%, which is within the range of the results given by the authors.

Scene	Accuracy	Log loss
Indoor	76.7%	0.750
Outdoor	89.6%	0.305
Transportation	93.5%	0.215
Average	86.6%	0.415

Figure 3.3.1-1 The Result of Reproduction for Task1b Baseline

3.3.2 Experimental Analyse

The replication of task1b gave the same result. However, there are some minor problems in the process of replication.

1) The exact python version is not provided when installing TensorFlow-GPU because the version of TensorFlow is 1.14.0, which only supports 2.7.*/3.6.* /3.7.* python version, but during the reproduction, the latest version of python is 3.9.*, which causes the installation of TensorFlow to fail. So, we tried two python versions. The first time we tried python 3.7, but when we ran it, we got the error *"ValueError: unsupported pickle protocol: 5"*. The reason for this error is the python version, so we tried python 3.6 for the second time, and it worked.

2) The original version of the code is not compatible. In the README, only some necessary dependency versions may be listed. Still, these dependencies may not be enough, and some dependencies do not have versions, leading to the reproduction process. Some dependencies have updated versions, and the description in the file requirement.txt is not precise enough. For example, there is a piece of information shown tensorflow>=1.14.0. The version of tensorflow2.0.* is more different from the version of tensorflow1.14.*. There are code changes required. Although it is not much, it is also necessary to fill in the version number as much as possible when illustrating and directly show the exact version during development.

3) Different package versions may lead to different results. After replicating the author's environment, we update the versions to the latest ones with the following details as figure 3.3.3-1. On the basis that we do not change the parameters of the author's model, our outgoing results want to compare with the original results. For this experiment, we obtained the same

results as before.

```
conda create --name tf-dcase
conda activate tf-dcase
conda install ipython=7.22.0
conda install numpy=1.20.3
conda install tensorflow-gpu==2.4.1
conda install -c anaconda keras-gpu=2.4.3
conda install -c anaconda cudatoolkit=10.1.243
conda install -c anaconda cudnn=7.6.5
conda install librosa=0.8.1
pip install absl-py==0.13.0
pip install dcase_util==0.2.18
pip install sed_eval=0.2.1
```

Figure 3.3.2-1 The version of Package for Task1b Baseline

Scene	Accuracy	Log loss
Indoor	76.7%	0.750
Outdoor	89.6%	0.305
Transportation	93.5%	0.215
Average	86.6%	0.415

Figure 3.3.2-2 The Result of DCASE 2020 Task1b Reproduction

```
File-wise evaluation, over 4 folds
   Scene label           | Nref : Nsys | Accuracy |  | Fold1    | Fold2    | Fold3    | Fold4    |
   ======================+=============+==========+  +==========+==========+==========+==========+
   beach                 |   78 :   60 |  69.3 %  |  |  84.2 %  |  66.7 %  |  78.9 %  |  47.4 %  |
   bus                   |   78 :   76 |  78.3 %  |  |  68.4 %  |  65.0 %  |  94.7 %  |  85.0 %  |
   cafe/restaurant       |   78 :   93 |  83.2 %  |  |  66.7 %  |  94.7 %  |  71.4 %  | 100.0 %  |
   car                   |   78 :   73 |  87.2 %  |  |  70.0 %  |  89.5 %  |  89.5 %  | 100.0 %  |
   city_center           |   78 :   88 |  85.5 %  |  |  83.3 %  |  73.7 %  |  89.5 %  |  95.5 %  |
   forest_path           |   78 :   66 |  81.0 %  |  |  57.1 %  | 100.0 %  |  66.7 %  | 100.0 %  |
   grocery_store         |   78 :   61 |  65.0 %  |  |  52.6 %  |  81.0 %  |  89.5 %  |  36.8 %  |
   home                  |   78 :   92 |  82.1 %  |  | 100.0 %  |  55.6 %  |  95.0 %  |  77.8 %  |
   library               |   78 :   66 |  50.4 %  |  |  47.6 %  |  38.9 %  |  15.0 %  | 100.0 %  |
   metro_station         |   78 :   97 |  94.7 %  |  |  84.2 %  |  94.4 %  | 100.0 %  | 100.0 %  |
   office                |   78 :   80 |  98.6 %  |  | 100.0 %  | 100.0 %  |  94.4 %  | 100.0 %  |
   park                  |   78 :   21 |  13.9 %  |  |  10.0 %  |   5.6 %  |   0.0 %  |  40.0 %  |
   residential_area      |   78 :  145 |  77.7 %  |  |  78.9 %  |  47.6 %  | 100.0 %  |  84.2 %  |
   train                 |   78 :   45 |  33.9 %  |  |  16.7 %  |  31.6 %  |  26.1 %  |  61.1 %  |
   tram                  |   78 :  107 |  85.4 %  |  |  88.9 %  |  88.9 %  |  63.6 %  | 100.0 %  |
   ======================+=============+==========+  +==========+==========+==========+==========+
   Overall accuracy      | 1170 : 1170 |  72.4 %  |  |  67.2 %  |  68.9 %  |  71.6 %  |  81.9 %  |
[Done]
```

Figure 3.3.2-3 The Result of DCASE 2016 Task1 Reproduction

From the experiment dcase2016 task1 we can draw some conclusions:

1) because the project was developed in 2016, the version of python used is 2.7, which is detailed in the README by the author. Python3, which is used by most developers today, would have been an unnecessary hassle in the environment configuration session if the author had not detailed it here. There are many differences between python2 and python3, and different versions of python can lead to completely different results, so it is vital to be clear about the

version.

2) Since we are using python 2.7, some of the dependencies have been updated considerably. However, the author gives detailed version information in the requirements, and when the author prompts for >= a version, we default to the smallest version. One of the scipy installations requires python2 pip to be installed, and requires some experience to know how to run it.

3) Using Conda for software environment management is crucial. In my experiments, I use Conda for environment management because Conda can provide a completely independent environment, so when I use each package, it will not conflict with other versions of packages, which guarantees the stability of the software environment

4) The 2016 challenge is now 5 years away, but it can still be reproduced now, two of which are very crucial, the first is the robustness of the code, because the code structure is complete and the author uses container several times, so the code logic structure is clear, we found no errors in the code in this experiment, but the higher readability of the code can make it very easy to modify the code as well. Also, there is a lot of explanatory text in the source code to explain what the code does and how it is executed. The second reason is that the author gives a very detailed description of the environment, and the various versions are very detailed.

From the experiment dcase2016 task3 we can draw some conclusions:

Unfortunately, I failed in the task3 reproduction process. I am sure that if I had spent more time on task 3, there would have been a way to reproduce it. However, after various attempts I was unable to reproduce it, and the time spent on it was very higher compared to the previous two tasks, so I gave up on task3.

The main reason is the version conflict, from the structure of the code there is no problem, but in the reproduction process there is an incompatibility of the software version. In the requirement file, the required version of NumPy is "numpy>=1.9.2", but in this version, there will be an error occurred "numpy.dtype has the wrong size, try recompiling". The reason for this error is that the version of NumPy is too low. Upgrading to 1.11 gives the error "numpy.ufunc has the wrong size, try recompiling" and the NumPy version does not match the scikit-learn version. Upgrading the NumPy version to 1.12/1.13 again, there is another error occurred which detail is "numpy boolean negative, the `-` operator, is not supported, use the `~` operator or the logical_not function instead". This is because the version of NumPy is too high.

Therefore, the task3 implementation failed because we were unable to find a better way to replicate it due to updates to the software versioning environment, updates to the dependencies associated with each package, and the complexity of the dependencies between packages.

As can be seen, with the very rapid pace of technology updates at the moment, TensorFlow was introduced in 2017, when there were a lot of people using TensorFlow for development, yet in October 2019, version 2.0 of TensorFlow was released, after the release of which there were mixed reviews, we do not comment on its good or bad, but it has to be said that 2.0 differs more from 1.0 in that many of the methods have been changed, and the differences between 2.0's dependency and 1.0's dependency have also changed, which has led to some programs becoming extremely complex in the software environment due to problems with the newer iterations of the version, and if we fail to keep detailed records, it can lead to reproducing older articles when errors occur.

To make the replication task more diverse, we replicated an article from DCASE2020 Task1b [41]. In this article, which belongs to the same task as task1b mentioned earlier, the source code is not available on the DCASE community, so the key information to reproduce the task can only be obtained from its technical report.

For the analysis process as shown in figure3.2.2-1, we can obtain from its technical report, firstly, the result data about this task, but also about the core CNN architecture and the method of Disout in this task. Disout is based on Dropout, this method enhances the generalization ability of deep neural net-works by studying feature plot disturbances.[41]. When performing reproduction work, for feature_extraction, where some of the parameters are unclear, in order to ensure the accuracy of the replication, we had to contact the authors for help, and they generously provided the source code to facilitate the replication work. The final result we obtained was as follows:

Scene	DD-CNN	
	Accuracy	Loss
Indoor	89.10%	0.402
Outdoor	90.20%	0.296
Transportation	97.30%	0.075
Average	92.00%	0.257

Figure 3.3.2-4 The Result of DCASE 2020 Task1b Challenger Reproduction

3.3.3 Experimental Insight

From the above, some lessons can be drawn from the above information, the first being that Authors should **use version control for their work**. In this task, only one version was released. But actually, this task is continued all the time, not quite the same every year. If you can do version update iterations, such as using TensorFlow 2.0, it is more beneficial for scholars to understand the author's intention. When they do so, they should specify the information that uniquely identifies the exact code they are using to produce their results. Without version control, identification can be made by specifying the base software version used along with any patches that have been applied.

So, we suggest author:

1) use some form of version control, for example, a tool like Git or Mercurial

2) if version control is not possible, explicitly save different versions of their code with version numbers. Explicitly mention the results of the version publication used to produce the results.

3) If a simulation can no longer be done after fixing a bug, for example, because resources have been exhausted, provide multiple commit hashes or unique identifiers that have been used. [37]

Secondly, **Clarify the dependencies of the software and their versions**. Software frameworks are often convenient or necessary for performing scientific calculations. Many frameworks depend on other software packages for their functionality, but these dependencies may evolve at different rates over time. For example, in this task, the authors specify that the version of dcase_util is 0.2.12, but do not specify what the version of librosa is. The version of librosa at the time of the authors' research may have been 0.7.*, because the authors released the code in early 2020, but in this task, the version of librosa has been updated to 0.8.1. Although there are not many updates, this is only for librosa, and the lack of a clear version for other packages with more significant updates will cause the replication to fail. Authors should provide detailed versioning information for each of these dependencies; this information will make it possible to reconstruct the original software stack used to produce the results. So, we suggest author:

1） Using Docker images or other container solutions, it is possible to "save" all the necessary dependencies for computational experiments. Use Dockerfile instead of preparing Docker images manually, which captures all the dependency information needed to use computational experiments.

2） Use a package management system, such as Conda. These systems need to build all the necessary dependencies, and they also keep track of the versioning information for these dependencies.

In this task, because of the author's specificity (it was a challenge on DCASE), there were no problems other than the two aspects mentioned above, and the author did an outstanding job. Firstly, the authors provide a very detailed README file, which was easy and quick to understand what the task was doing, what was being provided, and what results were needed when I knew nothing about the task. Secondly, the authors describe their neural network in great detail as well, so you can see the architecture of the neural network very intuitively. Also, the authors show their results, and this result is 0.7% up or down on the accuracy metric, which is a very responsible result and shows that the authors did not choose one of the best results to present.

Third, **as much as possible, the parameters are disclosed**. In the field of artificial intelligence,

there are very many parameters involved when we train a model, for example, in feature extraction, for example, when designing a convolutional network, we need parameters, so they are very important for us to be able to reproduce the paper better. For the pervious tasks, the most praise worthy thing is that the authors provide the source code, and the source code uses docker, puts all the parameter files in YAML files, and also provides the parameter files for baseline expansion.

Krafczyk MS et al. also did the same thing in 'Learning from reproducing computational results: introducing three principles and the Reproduction Package'. [37] The author presented more comprehensive guidelines of benefits for reproduction. Some of them were encountered in this task, and some were not. The ones that were not encountered were related to AI, such as:

1) To make available all artifacts that support published results without exceeding legal and ethical barriers, for example, putting the source code on GitHub, which in our current task the authors have done. Also, offered to host the data on Zenodo.org with a license to use it. In this task, the data is hosted on Zenodo.org and has MIT licenses.

2) Avoid using absolute or hard-coded file paths in your code. Absolute file paths, such as C:\Project\Dataset or /home/user/Dataset will almost certainly not work on a different system than the one on which the experiment was originally run. on a different system than the one on which the experiment was originally run. The author should use a file path defined relative to the current directory, e.g.. /Dataset or . /data/Dataset. new

3) Provide an explicit mechanism for setting and reporting random seed values. When using algorithms that rely on random number generators, authors should implement their scripts to allow the user to set the initial seed value. Setting the seed value can produce the same computational result in subsequent runs of the program. In addition, authors should report the random seeds used for the results they present in their publications. Recommendation:

    a. Ensure that seed values can be set and report the seed values used in the article experiments.

    b. Avoid setting the seed value to a variable amount like time or allow the user to disable this setting.

    c. Allow the user to set the seed value, e.g., by providing an option that is available via the command line.

4) Report expected errors and tolerances for any published results, including any uncertainties in the software or computing environment. Not all results can be bitwise reproducible (exactly the same as every bit published), thus leading to discrepancies (errors) between what the authors expect and what the code produces. [37] In this case, the author reports this error range and/or implements tests incorporating this error range. If they do not, users produce results

that differ from those published and do not state a tolerance level. They may believe they are producing incorrect results when they are not.

5) Disclosure of resource requirements for computational experiments. Cutting-edge research often requires the use of large amounts of computational resources, whether in the form of vast amounts of memory or many computational nodes. [37] As an essential step, authors should report the hardware on which they run their computational experiments. If possible, including a small test case that a user with regular hardware can run is an excellent step to not only test that the code runs well but also to allow others to verify that they can get and run the code correctly before moving to a large computational resource. In this task, one challenger successfully completed the challenge, but when reproducing his task, it was found that under the hardware conditions: NVIDIA TESLA K100*2, 56GRAM, 380GB Disk, 6 Cores, it ran very slowly, taking about 90 minutes per epoch to run, probably because the author used an innovative method, but such a slow speed would have lost most users Patience. So, it is also indispensable to disclose detailed hardware information.

    a. Report the hardware used to run the experiment to obtain the results in the article.

    b. When running the code, use the time command to measure the time required for the experiment and report this time as a rough estimate of the expected running time of the experiment.[37]

3.4 Experimental Conclusion

From the above experiments, we conclude. In fact, when reproducing a paper, the effect of reproduction varies depending on the difficulty of the paper. However, in general, papers without source code and data are the most difficult to reproduce. We discuss what we need to do to make the reproducibility go further, so we come up with the following conclusion:

1) We would like the authors to disclose the source code and dataset as much as possible and to have a detailed technical report, which would be extremely useful for replication. If it is not possible to disclose the source code, it would be useful to provide an interface to the code so that it can be called by other scholars, or, if not, to include the code ideas in the technical documentation. If it is not possible to disclose the datasets, it is also possible to use public datasets for development or to host them on third party platforms.

2) Where both source code and datasets are available, we expect authors to describe the technical reports and associated supporting documentation as clearly as possible.

3) We strongly recommend that the research environment is in the cloud

4 CONCLUSION

We reviewed the reproducibility issues and analysed the original solutions. Based on the original solution, we reproduced four tasks at DCASE to demonstrate that the problems were real, and we made more detailed requirements and recommendations to improve reproducibility.

5 ACKNOWLEDGMENT

Many thanks to Professor Mark Plumbley and Andres Fernandez for their support of my research. Many thanks to my parents, Baohua Wang and Huanlian Wang, for their support.

APPENDIX

[1] The Machine Learning Reproducibility Checklist

The Machine Learning Reproducibility Checklist (v2.0, Apr.7 2020)

For all **models** and **algorithms** presented, check if you include:

- ❑ A clear description of the mathematical setting, algorithm, and/or model.
- ❑ A clear explanation of any assumptions.
- ❑ An analysis of the complexity (time, space, sample size) of any algorithm.

For any **theoretical claim**, check if you include:

- ❑ A clear statement of the claim.
- ❑ A complete proof of the claim.

For all **datasets** used, check if you include:

- ❑ The relevant statistics, such as number of examples.
- ❑ The details of train / validation / test splits.
- ❑ An explanation of any data that were excluded, and all pre-processing step.
- ❑ A link to a downloadable version of the dataset or simulation environment.
- ❑ For new data collected, a complete description of the data collection process, such as instructions to annotators and methods for quality control.

For all shared **code** related to this work, check if you include:

- ❑ Specification of dependencies.
- ❑ Training code.
- ❑ Evaluation code.
- ❑ Pre-trained model(s).
- ❑ README file includes table of results accompanied by precise command to run to produce those results.

For all reported **experimental results**, check if you include:

- ❑ The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results.
- ❑ The exact number of training and evaluation runs.
- ❑ A clear definition of the specific measure or statistics used to report results.
- ❑ A description of results with central tendency (e.g. mean) & variation (e.g. error bars).
- ❑ The average runtime for each result, or estimated energy cost.
- ❑ A description of the computing infrastructure used.

Reproduced from: *www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist-v2.0.pdf*

REFERENCES

[1] Plesser HE. Reproducibility vs. Replicability: A Brief History of a Confused Terminology. Front Neuroinform. 2018;11:76. Published 2018 Jan 18. doi:10.3389/fninf.2017.00076

[2] DCASE Community, http://dcase.community/community_info, DCASE Community

[3] Stodden, Victoria, The Scientific Method in Practice: Reproducibility in the Computational Sciences (February 9, 2010). MIT Sloan Research Paper No. 4773-10, Available at SSRN: https://ssrn.com/abstract=1550193 or http://dx.doi.org/10.2139/ssrn.1550193

[4] Pineau, Joelle et al. "Improving Reproducibility in Machine Learning Research (A Report from the NeurIPS 2019 Reproducibility Program)." *ArXiv* abs/2003.12206 (2020): n. pag.

[5] Stodden, V. (2014). What computational scientists need to know about intellectual property law: A primer. In Implementing Reproducible Research (pp. 325-340). CRC Press. https://doi.org/10.1201/b16868

[6] E. Bjornson, "Reproducible Research: Best Practices and Potential Misuse [Perspectives]," in IEEE Signal Processing Magazine, vol. 36, no. 3, pp. 106-123, May 2019, doi: 10.1109/MSP.2019.2898421.

[7] Collberg, C., Proebsting, T., and Warren, A.M. Repeatability and Benefaction in Computer Systems Research: A Study and a Modest Proposal. Technical Report TR 14-04. Department of Computer Science, University of Arizona, Tucson, AZ, Dec. 2014; http:// repeatability.cs.arizona.edu/v2/RepeatabilityTR.pdf

[8] Baker, M. 1,500 scientists lift the lid on reproducibility. Nature 533, 452–454 (2016). https://doi.org/10.1038/533452a

[9] Silver, D., Huang, A., Maddison, C. et al. Mastering the game of Go with deep neural networks and tree search. Nature 529, 484–489 (2016). https://doi.org/10.1038/nature16961

[10] Henderson, Peter, Islam, Riashat, Bachman, Philip, Pineau, Joelle, Precup, Doina and Meger, David Deep Reinforcement Learning that Matters. (2017). , cite arxiv:1709.06560Comment: Accepted to the Thirthy-Second AAAI Conference On Artificial Intelligence (AAAI), 2018 .

[11] Islam, R., Henderson, P., Gomrokchi, M., & Precup, D. (2017). Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control. *ArXiv, abs/1708.04133*.

[12] Haibe-Kains, B., Adam, G.A., Hosny, A. et al. Transparency and reproducibility in artificial intelligence. Nature 586, E14–E16 (2020). https://doi.org/10.1038/s41586-020-2766-y

[13] https://twitter.com/ylecun/status/1246056786274156545, Yann LeCun

[14] Raff, Edward. "A Step Toward Quantifying Independently Reproducible Machine Learning Research." NeurIPS (2019).

[15] V. Stodden, "The Legal Framework for Reproducible Scientific Research: Licensing and

Copyright," in Computing in Science & Engineering, vol. 11, no. 1, pp. 35-40, Jan.-Feb. 2009, doi: 10.1109/MCSE.2009.19.

[16] Common Problems When Reproducing A Machine Learning Paper. Derek chia, https://medium.com/@derekchia/common-problems-when-reproducing-a-machine-learning-paper-17178515d6c6 July, 2020

[17] Yann LeCun, https://twitter.com/ylecun/status/1138167583155183617, Yann LeCun

[18] Edward Raff, "Quantifying Independently Reproducible Machine Learning", The Gradient, 2020.

[19] Towards Reproducible Research with PyTorch Hub, https://pytorch.org/blog/towards-reproducible-research-with-pytorch-hub/, Team PyTorch, June 10, 2019

[20] Davison, Andrew & Mattioni, Michele & Samarkanov, D & Tele'nczuk, B. (2014). Sumatra: A Toolkit for Reproducible Research.

[21] DCASE Utilities, https://dcase-repo.github.io/dcase_util/ui.html#fancylogger, DCASE

[22] Piccolo, S.R., Frampton, M.B. Tools and techniques for computational reproducibility. GigaSci 5, 30 (2016). https://doi.org/10.1186/s13742-016-0135-4

[23] Howe, B.. "Reproducibility, Virtual Appliances, and Cloud Computing." (2018).

[24] S. F. J. Apostal, D. Apostal and R. Marsh, "Containers and Reproducibility in Scientific Research," 2018 IEEE International Conference on Electro/Information Technology (EIT), 2018, pp. 0525-0530, doi: 10.1109/EIT.2018.8500088.

[25] One researcher's mission to encourage reproducibility in machine learning, Ben Dickson, https://bdtechtalks.com/2021/03/01/papers-without-code-machine-learning-reproducibility/, March 2021

[26] VMware Workstation Pro Documentation, https://docs.vmware.com/cn/VMware-Workstation-Pro/16.0/com.vmware.ws.using.doc/GUID-D1FEBF81-D0AA-469B-87C3-D8E8C45E4ED9.html, VMware Document

[27] R. Isdahl and O. E. Gundersen, "Out-of-the-Box Reproducibility: A Survey of Machine Learning Platforms," 2019 15th International Conference on eScience (eScience), 2019, pp. 86-95, doi: 10.1109/eScience.2019.00017.

[28] Amazon and Microsoft Maintain their Grip on the Market but Others are also Growing Rapidly , https://www.srgresearch.com/articles/amazon-and-microsoft-maintain-their-grip-market-others-are-also-growing-rapidly, RENO, NV, April 29, 2021

[29] Cloudy with a Chance of Peptides: Accessibility, Scalability, and Reproducibility with Cloud-Hosted Environments Benjamin A. NeelyJournal of Proteome Research 2021 20 (4), 2076-2082DOI: 10.1021/acs.jproteome.0c00920

[30] Reproducible system award, http://dcase.community/challenge2020/awards, DCASE awards

[31] Zinemanas, Pablo & Hounie, Ignacio & Cancela, Pablo & Font, Frederic & Rocamora, Martín & Serra, Xavier. (2020). DCASE-models: A Python library for computational environmental sound analysis using deep-learning models.

[32] National Academies of Sciences, Engineering, and Medicine, Reproducibility and Replicability in Science. Washington, DC: The National Academies Press, 2019.

[33] A. Mesaros et al., "Detection and Classification of Acoustic Scenes and Events: Outcome of the DCASE 2016 Challenge," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 26, no. 2, pp. 379-393, Feb. 2018, doi: 10.1109/TASLP.2017.2778423.

[34] Hu, H., Yang, C.H., Xia, X., Bai, X., Tang, X., Wang, Y., Niu, S., Chai, L., Li, J., Zhu, H., Bao, F., Zhao, Y., Siniscalchi, S.M., Wang, Y., Du, J., & Lee, C. (2020). Device-Robust Acoustic Scene Classification Based on Two-Stage Categorization and Data Augmentation. *ArXiv, abs/2007.08389*.

[35] What is azure machine learning, https://docs.microsoft.com/en-us/azure/machine-learning/overview-what-is-azure-ml, Azure Team

[36] How Azure Machine Learning works: Architecture and concepts, https://docs.microsoft.com/en-us/azure/machine-learning/concept-azure-machine-learning-architecture

[37] Krafczyk MS, Shi A, Bhaskar A, Marinov D, Stodden V. 2021 Learning from reproducing computational results: introducing three principles and the Reproduction Package. Phil. Trans. R. Soc. A 379: 20200069. https://doi.org/10.1098/rsta.2020.0069

[38] Plesser HE. Reproducibility vs. Replicability: A Brief History of a Confused Terminology. Front Neuroinform. 2018;11:76. Published 2018 Jan 18. doi:10.3389/fninf.2017.00076

[39] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. *TUT database for acoustic scene classification and sound event detection.* In 24th European Signal Processing Conference 2016 (EUSIPCO 2016). Budapest, Hungary, 2016.

[40] DCASE2020 - Task 1 - Baseline systems, https://github.com/toni-heittola/dcase2020_task1_baseline.git, Toni Heittola, Tampere University

[41] DCASE2020Task1b, http://dcase.community/documents/challenge2020/technical_reports/DCASE2020_Zhao_40.pdf, Jingqiao Zhao et. al.