SC2TOOLS: STARCRAFT II TOOLSET AND DATASET API

A PREPRINT

Andrzej Białecki[*,1], **Piotr Białecki**[2], **Piotr Sowiński**[1,4], **Mateusz Budziak**[2], **Jan Gajewski**[3]

[1]Warsaw University of Technology
[2]Independent Researcher
[3]Józef Piłsudski University of Physical Education in Warsaw
[4]NeverBlink, Poland

May 5, 2025

ABSTRACT

Computer games, as fully controlled simulated environments, have been utilized in significant scientific studies demonstrating the application of Reinforcement Learning (RL). Gaming and esports are key areas influenced by the application of Artificial Intelligence (AI) and Machine Learning (ML) solutions at scale. Tooling simplifies scientific workloads and is essential for developing the gaming and esports research area.

In this work, we present "SC2Tools", a toolset containing multiple submodules responsible for working with, and producing larger datasets. We provide a modular structure of the implemented tooling, leaving room for future extensions where needed. Additionally, some of the tools are not StarCraft 2 exclusive and can be used with other types of data for dataset creation.

The tools we present were leveraged in creating one of the largest StarCraft 2 tournament datasets to date with a separate PyTorch and PyTorch Lightning application programming interface (API) for easy access to the data.

We conclude that alleviating the burden of data collection, preprocessing, and custom code development is essential for less technically proficient researchers to engage in the growing gaming and esports research area. Finally, our solution provides some foundational work toward normalizing experiment workflow in StarCraft 2

Keywords gaming · esport · data processing · toolset · dataset preparation

1 Introduction and Background

Computer games as fully controlled simulated environments were used in major scientific works that showcased the application of Reinforcement Learning (RL). As such, computer games can be viewed as one of the many components of major breakthroughs and advancements in RL applications [1–7].

Despite heightened interest in research on gaming and esports, there are limited high-level libraries and tools made for rapid experimentation in some game titles. Researchers from various research disciplines have shown their interest in exploring gaming and esports, including: (1) psychology [8], (2) computer science [9, 10], (3) education [11, 12], (4) medical sciences [13], and others [14, 15]. The ability to tie these topics with the in-game data cannot be overstated.

When such software is available, it is often hard to use for less technically proficient researchers. Data parsing libraries are prevalent in computer games, such as Counter-Strike [16, 17], Rocket League [18], Dota 2 [17, 19], and finally in StarCraft 2 [20–22].

[*] Corresponding author: `andrzej.bialecki94@gmail.com`
[*] Institutional contact: `andrzej.bialecki.dokt@pw.edu.pl`

Esports can be treated as a subset of gaming with additional requirements for players, such as tournament presence, organized play, training, and professionalization [23]. The study of esports is multidisciplinary in nature [24, 25]. Due to the growing academic interest in the area of gaming and esports [26–29], it is key to provide tools for researchers capable of simplifying the process of acquiring large datasets efficiently, not only for authors interested in the area of computer science [30, 31].

In case of our implementation, we focus on solving problems within the StarCraft 2 (SC2) infrastructure ecosystem. StarCraft 2 is a real-time strategy game developed by Blizzard Entertainment. The game is known as one of the most prominent real-time strategy (RTS) esports titles [32, 33]. It is also characterized by its fast-paced gameplay and a high skill ceiling [34]. These attributes make for a great environment for testing various AI agents [6, 35–37]. Moreover, research in StarCraft 2 is not limited to AI agents – there are efforts to analyze the game from various perspectives and provide insights that can assist players in their gameplay [38, 39].

Our software collection is an open-source implementation of data extraction, and data interfacing tools for StarCraft 2. We solve the problem of ease of access to the data encoded in files with ".SC2Replay" extension by using an open-source file extractor for proprietary MoPAQ (MPQ) file format. From this point on, we will refer to the MPQ files with the ".SC2Replay" extension as SC2Replay files.

So far, our software was leveraged in preparation of major datasets: "SC2ReSet" [40] and "SC2EGSet" [41] with an accompanying peer-reviewed and published Data Descriptor article [42]. The output of our software was used in varying contexts indirectly. authors cited our work, some of them following the general flow of our exploration [43]. Others put emphasis on statistical calculation within esports landscape [44]. Finally authors describe our work in surveys of related work when working in another games [45].

Our solution uses the official StarCraft 2 replay file format specification provided via Blizzard Entertainment GitHub repository [46]. Specifically, in "SC2InfoExtractorGo", we extend the community-built Golang implementation of the parser [21, 47]. The output of our software pipeline is a fully prepared dataset, ready for use with our extension of PyTorch [48] and PyTorch Lightning interfaces [49]. Our goal was to lower the technical knowledge required to obtain data from in-game replays.

2 Software Description

Our software publication consists of multiple modules that the user can match to their specific needs. To easily extend our toolset, the main repository of "SC2Tools" contains multiple git submodules. Each submodule is a separate repository with the logic required to perform a specific tasks on the SC2Replay files. The motivation for this structure is twofold. Firstly, it makes evolving the toolset easier, as modules can be easily replaced, or new ones added. Secondly, users have the option of using only a portion of the pipeline. The full list of current submodules is as follows: (1) "SC2InfoExtractorGo" [50], (2) "DatasetPreparator" [51], (3) "SC2AnonServerPy" [52]. (4) "SC2_Datasets" [53].

In case of developing future tools, deprecating or improving the existing implementations, updates will be made to the common open-source repository. In their current state, all of the tools can be either used independently or combined in a data processing pipeline. The pipeline can interoperate with other software tools, as long as they use standard SC2Replay files for inputs to the "SC2InfoExtractorGo". Finally, loading the data for experiments is supported as long as the output is saved in Java Script Object Notation (JSON) files with ".json" extension, and conforms to a pre-defined schema defined by the "SC2_Datasets" parser. From now on, we will refer to such files as JSON files. In the current version extending the software with additional tools is possible, and we encourage the community to contribute to the project for future releases.

2.1 Software Architecture

Tools and scripts in our repository have singular responsibilities. Each of our submodules fulfills a specific part of the data processing needs within the pipeline. The full pipeline in a simplified pictorial form is showcased in Figure 1. Note the distinctive steps of data pre-processing are explained in DatasetPreparator introducing the "DatasetPreparator" Python utility scripts. Further, data processing Golang tool implementatino is explained in SC2InfoExtractorGo introducing the "SC2InfoExtractorGo". Finally, data modeling or post-processing tasks are introduced in SC2_Datasets diving deeper on "SC2_Datasets" as a Python API implementations for PyTorch [48] and PyTorch Lightning [49].

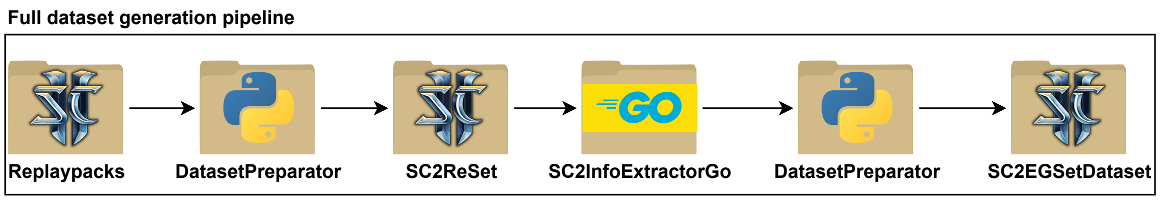


Figure 1: Simplified full pipeline using SC2Tools to create two datasets, "SC2ReSet" [40] and "SC2EGSet" Dataset [41]. Initially introduced in [42]. Figure 1

2.1.1 DatasetPreparator

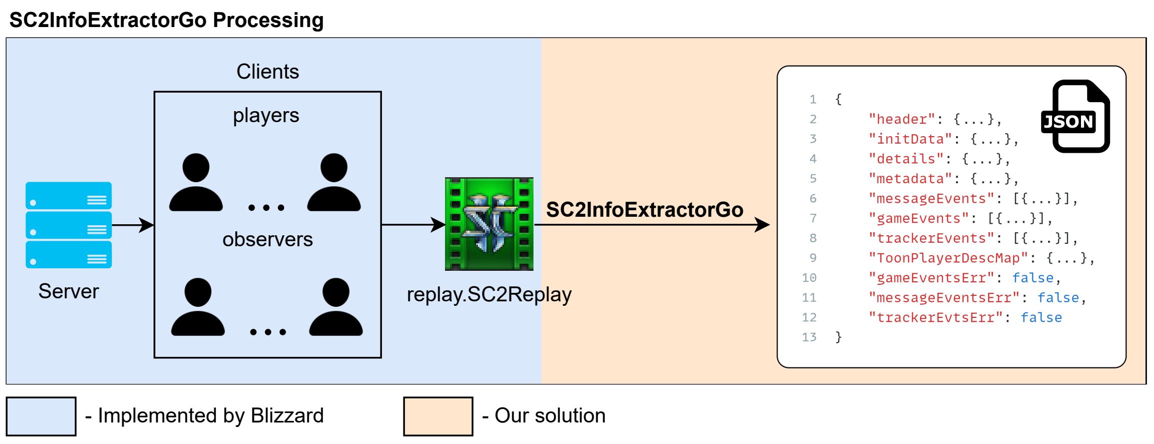
The "DatasetPreparator" [51] submodule is a set of scripts that ease the process of working with major collections of raw data (replaypacks/datasets). A full list of scripts is as follows:

1. "directory_flattener.py"; flattens the nested directory structure of the replaypacks,
2. "directory_packager.py"; packages all of the directories in the specified input directory,
3. "file_renamer.py"; renames the files in the directory to follow a specific naming convention (e.g., to match the dataset schema),
4. "json_merger.py"; merges two JSON files into one,
5. "processed_mapping_copier.py"; copies the auxiliary files generated by "directory_flattener.py" to matching output directories. Built specifically to prepare the "SC2EGSet" prior to packaging,
6. "sc2_map_downloader.py"; wraps "SC2InfoExtractorGo" to run the map downloading step,
7. "sc2egset_replaypack_processor.py"; wraps "SC2InfoExtractorGo" to run the replaypack processing step on multiple directories at once,
8. "sc2egset_pipeline.py"; wraps the entire processing pipeline used to obtain the "SC2ReSet" and "SC2EGSet" datasets,
9. "sc2reset_replaypack_downloader.py"; downloads the raw (flattened) replaypacks of "SC2ReSet" [40] for users that wish to use their own tools for data processing.

In the context of our work, this submodule is responsible for preparing directory structure, execution of "SC2InfoExtractorGo" on the data, and packaging the dataset for hosting. Finally, the current capabilities include downloading the raw replaypacks of "SC2ReSet" [40] for ease of "SC2EGSet" Dataset reproduction [41].

2.1.2 SC2InfoExtractorGo

The SC2InfoExtractorGo as a submodule is a tool responsible for extracting the data from SC2Replay files, it depends on previously published open-source lower-level libraries [21, 47]. The tool is written in Golang and is shipped as a binary file (release), and as a Docker image via DockerHub. A simplified depiction of the data extraction is available on Figure 2.



```
{
    "header": {...},
    "initData": {...},
    "details": {...},
    "metadata": {...},
    "messageEvents": [{...}],
    "gameEvents": [{...}],
    "trackerEvents": [{...}],
    "ToonPlayerDescMap": {...},
    "gameEventsErr": false,
    "messageEventsErr": false,
    "trackerEvtsErr": false
}
```

Figure 2: Pictorial representation of the "SC2InfoExtractorGo" functionality [50]. Replays contain the events which happened during gameplay (blue background), our implementations extracts this data and outputs it for further analysis by the user (orange background).

2.1.3 SC2_Datasets

One of our solutions, "SC2_Datasets" [53] interfaces with the JSON files produced by the "SC2InfoExtractorGo" [50]. This includes all of the classes and methods required to load a single JSON, a collection of JSON files (representing a replaypack), and finally a way of loading an entire dataset (a collection of replaypacks). The pictorial representation of the "SC2_Datasets" functionality is presented on Figure 3.

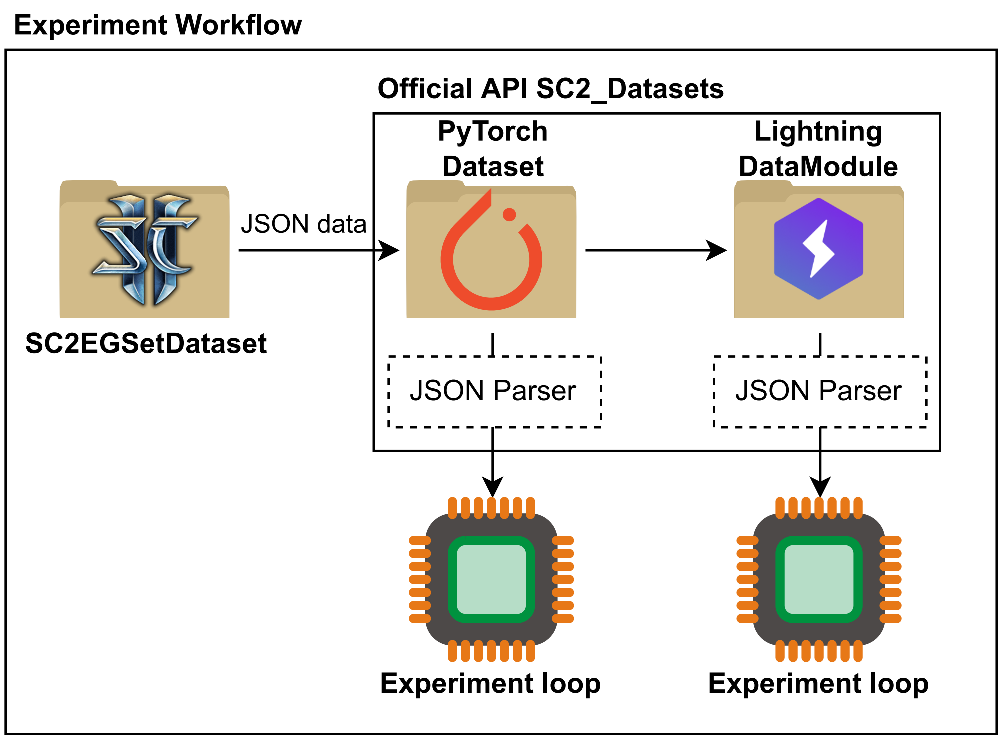


Figure 3: Loading the output of the SC2InfoExtractorGo for machine learning and artificial intelligence use with "SC2_Datasets".

Users have the ability to extend our solution and apply it to their data via the PyTorch [48] and PyTorch Lightning [49] interfaces.

2.2 SC2AnonServerPy

In the process of extracting the information from the StarCraft 2 replays, the users have the ability to choose if nicknames of the players should be anonymized with a separate tool "SC2AnonServerPy" [52], this functionality may be key for laboratories that wish to share their datasets with a wider community.

2.3 Software Functionalities

Main functionality of this software collection introduce a repeatable way of working with StarCraft 2 data for research and data analysis. Users need to verify if their specific use case is permitted by the Blizzard End User License Agreement (EULA). Our software package includes file-wrangling tools such as: flattening nested directory structure, data-parallel replay file parsing (extraction), data cleanup, exporting replay data to JSON, and finally data loading into PyTorch [48] and PyTorch Lightning [49]. We have developed a modular system of tools solving specific issues of data processing with expandability in mind.

Main contribution of the work that we present is the “SC2InfoExtractorGo” [50], as introduced above. The most important procedure of the data extraction pipeline is showcased in Figure 4.

Within “DatasetPreparator” [51] there are multiple scripts that solve specific problems that may be present when researching StarCraft 2, including a wrapper for the "SC2InfoExtractorGo". For example to reproduce the “SC2ReSet” and “SC2EGSet”, scripts from “DatasetPreparator” would be executed consecutively as follows:

1. “directory_flattener.py” to flatten the nested directory structure of replaypacks that often have a complex structure with meaningful directory naming conventions,
2. “directory_packager.py” to obtain “SC2ReSet” by creating archives of the previously flattened directories,
3. “sc2egset_replaypack_processor.py” (requires “SC2InfoExtractorGo”) to process the replaypacks and obtain the initial version of “SC2EGSet”,
4. “processed_mapping_copier.py” to copy the auxiliary files generated by “directory_flattener.py” to matching output directories.
5. “file_renamer.py” to rename the files in directories to follow a specific naming convention (e.g., to match the dataset schema),
6. “directory_packager.py” to obtain the final version of “SC2EGSet” by packaging all of the directories in the specified input directory.

2.4 Code Snippets

Due to the complex nature of our software, and number of operations that are run for every replay, we have created multiple functions that define the steps of the data extraction process with “SC2InfoExtractorGo”. Function that is ran for every replay is showcased in Figure 4.

```go
func FileProcessingPipeline(
    replay string,
    anonymizer *GRPCAnonymizer,
    englishToForeignMapping map[string]string,
    flags CLIFlags,
) (bool, CleanedReplay, ReplaySummary, string) {

    replayData, err := rep.NewFromFile(replay)
    if err != nil {
        return false, CleanedReplay{}, ReplaySummary{}, "rep.NewFromFile() failed"
    }
    defer replayData.Close()

    if flags.PerformIntegrityCheck {
        integrityOk, failureReason := checkIntegrity(replayData)
        if !integrityOk {
            reason := fmt.Sprintf("checkIntegrity() failed: %s", failureReason)
            return false, CleanedReplay{}, ReplaySummary{}, reason
        }
    }
    if flags.PerformValidityCheck {
        if flags.FilterGameMode&Ranked1v1 != 0 && gameIs1v1Ranked(replayData) {
            if !validate1v1Replay(replayData) {
                return false, CleanedReplay{}, ReplaySummary{}, "validateReplay() failed"
            }
        }
    }
    if flags.PerformFiltering {
        if !filterGameModes(replayData, flags.FilterGameMode) {
            return false, CleanedReplay{}, ReplaySummary{}, "filterGameModes() failed"
        }
    }

    cleanOk, cleanReplayStructure := extractReplayData(replayData, englishToForeignMapping, flags.PerformCleanup)
    if !cleanOk {
        return false, CleanedReplay{}, ReplaySummary{}, "cleanReplay() failed"
    }

    summarizeOk, summarizedReplay := summarizeReplay(&cleanReplayStructure)
    if !summarizeOk {
        return false, CleanedReplay{}, ReplaySummary{}, "summarizeReplay() failed"
    }
    if grpcAnonymizer != nil {
        if !anonymizeReplay(&cleanReplayStructure, anonymizer, flags.PerformChatAnonymization) {
            return false, CleanedReplay{}, ReplaySummary{}, "anonymizeReplay() failed"
        }
    }
    return true, cleanReplayStructure, summarizedReplay, ""
}
```

Figure 4: Golang-inspired pseudocode algorithm for processing a single replay file using SC2InfoExtractorGo [50].

After the initial processing with a pre-defined pipeline, the output JSON files can be loaded with any programming language capable of reading this format for further processing. In our case this is showcased in "SC2_Datasets" repository, building on top of the JSON files an API for rapid experimentation with ML and AI methods using PyTorch [48] and PyTorch Lightning [49].

3 Usage Information

3.1 DatasetPreparator

3.1.1 Usage of Directory Flattener

It is common for StarCraft 2 tournament replaypacks to be sorted in multiple subdirectories. There is some information to be inferred from the names of the directories. Using this script flattens the diectory structure and prepares it for a simplified further processing.

3.1.2 Usage of Directory Packager

Finally, after all of the replaypack, or dataset processing is done, we have prepared a utility script that creates a ".zip" archive out of all top-level directories.

3.1.3 Usage of Processed Mapping Copier and File Renamer

As described above, after using the "directory_flattener.py" script, one of its side effects is the creation of "processed_mapping.json" file. When preparing a dataset, these files can be treated as additional metadata. Our software in its pipeline includes a script called "processed_mapping_copier.py", it iterates over each of the input directories and matches it against the directory in the output directory and copies the "processed_mapping.json" file.

To facilitate dataset creation, in most cases replaypack directories are often named after the tournament at which they were collected. File renaming script makes sure that the resulting ".zip" archive is renamed to match the tournament name. Some additional auxilliary files are created. This includes: (1) package summaries; containing some basic information about the number of processed replays and other in-game information. (2) previously copied "processed_mapping.json" file considering the directory structure of a replaypack might have been flattened. (3) "processed_failed.log" file containing the information about which files failed to process, and which files were processed successfully. (4) "main_log.log" file, containing all of the logs for debugging. In case of all of these files the "file_renamer.py" unifies the file names to become prepended with the tournament name e.g., "main_log.log" file becomes "TournamentName2024_main_log.log".

3.1.4 Usage of SC2EGSet Dataset Processing Pipeline

One of the scripts ("sc2egset_pipeline.py") simplifies all of the steps required to produce a StarCraft 2 dataset. We use this code to easily reproduce "SC2ReSet", and "SC2EGSet".

3.1.5 Usage of SC2EGSet Replaypack Processor

For a user that wishes to reproduce "SC2EGSet" from "SC2ReSet", a separate script is available that runs multiple instances of SC2InfoExtractorGo, The script iterates over each of the directory in the input path, and runs the "SC2InfoExtractorGo" on each of them with hardcoded parameters. Our solution implements multiprocessing Besides this functionality, the script does not offer much more utility than the original "SC2InfoExtractorGo" executable.

3.1.6 Usage of Other Scripts

Using the "sc2reset_replaypack_downloader.py" via its command line arguments makes it is possible to download all available replaypacks published as "SC2ReSet" hosted in an Zenodo repository [40]. When using "SC2ReSet" to run the rest of the processing pipeline, there is no need to execute the "directory_flattener.py", each replaypack was pre-processed before upload. After downloading "SC2ReSet" [40], it should be available under the directory as specified by the user.

3.2 Direct Use of SC2InfoExtractorGo

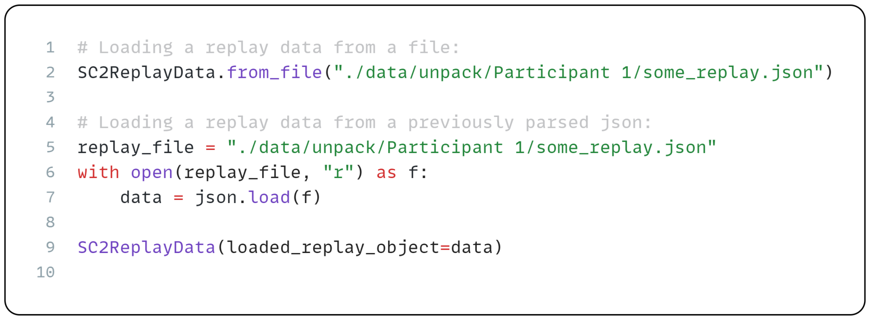
Next step pertains to the data extraction with "SC2InfoExtractorGo". The input directory for the command line usage of "SC2InfoExtractorGo" should reflect the directory where the user stored the replays which they would like to process. To ensure smooth reproduction of SC2EGSet the "SC2InfoExtractorGo" should be ran against each of the replaypack directories separately to produce output corresponding to data from a single tournament.

3.3 Usage of SC2AnonServerPy

In some cases, the user might want to anonymize the data due to the privacy, ethical, or legislative concerns. We provide additional service named "SC2AnonServerPy". As a separate gRPC service it is capable of receiving a string type containing a player nickname, and return a unique identifier as a string type for the requested player. The anonymization server is not constrained to StarCraft 2 data and can be used as long as the user provides the expected input type.

3.4 Running Experiments With SC2_Datasets

After extracting the data from SC2Replay files, any further processing, and experiments are possible with the "SC2_Datasets" Python package [53]. Loading a single JSON file following the structure defined in the "SC2_Datasets"



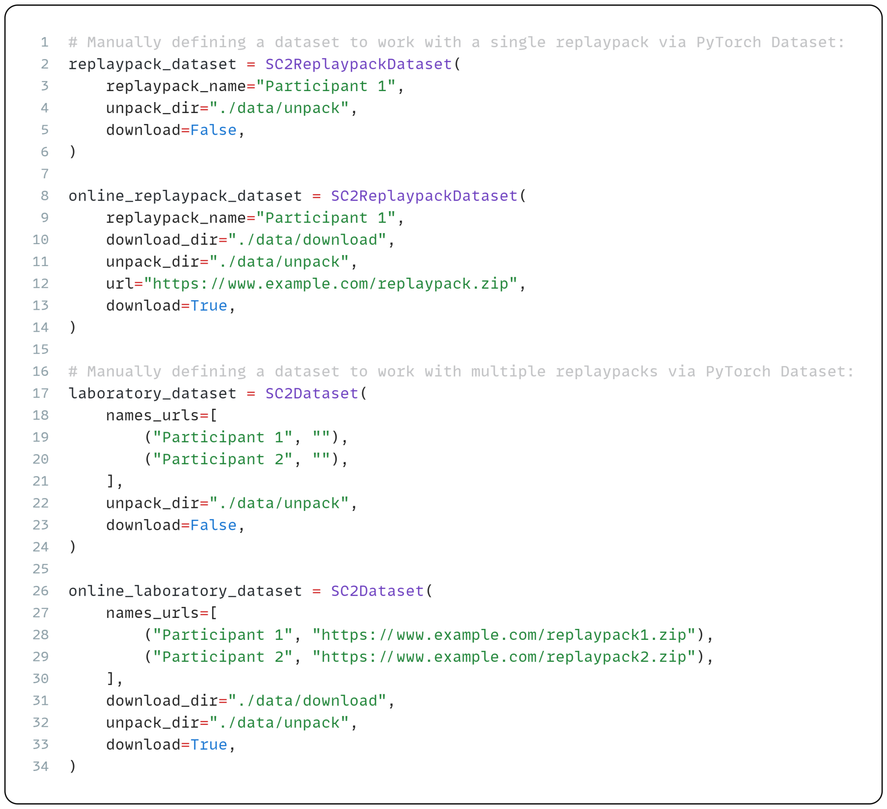
```python
# Loading a replay data from a file:
SC2ReplayData.from_file("./data/unpack/Participant 1/some_replay.json")

# Loading a replay data from a previously parsed json:
replay_file = "./data/unpack/Participant 1/some_replay.json"
with open(replay_file, "r") as f:
    data = json.load(f)

SC2ReplayData(loaded_replay_object=data)

```

Figure 5: Pictorial representation of code used to load a single replay, as defined in [53].



```python
# Manually defining a dataset to work with a single replaypack via PyTorch Dataset:
replaypack_dataset = SC2ReplaypackDataset(
    replaypack_name="Participant 1",
    unpack_dir="./data/unpack",
    download=False,
)

online_replaypack_dataset = SC2ReplaypackDataset(
    replaypack_name="Participant 1",
    download_dir="./data/download",
    unpack_dir="./data/unpack",
    url="https://www.example.com/replaypack.zip",
    download=True,
)

# Manually defining a dataset to work with multiple replaypacks via PyTorch Dataset:
laboratory_dataset = SC2Dataset(
    names_urls=[
        ("Participant 1", ""),
        ("Participant 2", ""),
    ],
    unpack_dir="./data/unpack",
    download=False,
)

online_laboratory_dataset = SC2Dataset(
    names_urls=[
        ("Participant 1", "https://www.example.com/replaypack1.zip"),
        ("Participant 2", "https://www.example.com/replaypack2.zip"),
    ],
    download_dir="./data/download",
    unpack_dir="./data/unpack",
    download=True,
)
```

Figure 6: Example usage of the PyTorch [48] dataset interface as defined in [53].

parser can be seen on Figure 5. To load the output of a processed dataset that exists either on the drive or online, the user should initialize a class as visualized on Figure 6. Additionally, PyTorch Lightning [49] datamodule interfaces can be used as they are included in the API Note that the users have full control and customizability of the code. In case of our implementations for "SC2EGSet" we provide an interface to use the data. Similar approach can be used with data from other sources, as long as the data is formatted in a way that is compatible with the "SC2_Datasets" parser.

4 Potential Impact

There exist many implementations built for the purpose of parsing replay files [54]. These tools and libraries require expert programming skills to extract and interact with the resulting data. Many research approaches involve scientists that may not posses such expert knowledge in programming, but nonetheless interested in investigating esports (e.g., in psychology, biomechanics, social sciences and humanities – SSH, and others) [55–57]. Lowering the technical overhead needed to interact with in-game data can open gaming and esports to researchers with various non-technical backgrounds. Furthermore, integrating SSH scientists in the research process is not only a requirement in some funding programs, but also a practical necessity, if one aims to conduct socially responsible studies [58, 59].

Before introducing our software, users were bound to write their own tools extracting the data from StarCraft 2 replay files. Our solution outputs easy-to-use JSON files adhering to a specific, well-documented schema definition https://sc2-datasets.readthedocs.io/en/latest/autoapi/index.html. Additionally, the data extraction toolset efficiently leverages modern multi-core processors (using Golang goroutines), making the process of data extraction faster. This has real implications on day-to-day research, as it allows for faster experimentation and iteration on one's methods.

Within the intended user group, the software was created to assist with the process of StarCraft 2 data processing. Mainly, the software fulfilled the research needs of our team and other collaborating research teams, which led to processing and creating a dataset [42]. Additionally, an API interface was created to load and work with the data in PyTorch [48] and PyTorch Lightning [49].

Due to the End User License Agreement (EULA) provisions specified by the game publisher (Blizzard), the commercial use of the extracted game data directly is limited. Nonetheless, one can extract valuable insights from the data and transfer them to the industry in a manner compliant with the EULA. In the past, research conducted on StarCraft 2 data has yielded fruitful ventures in online tooling [39, 60–62]; and research [6, 30, 35, 36].

5 Conclusions

We conclude that despite there being some software packages available, they often require additional programming skills and knowledge. Our solution provides a simple to use executable file and a set of scripts to work with StarCraft 2 data. Additionally, we conclude that our software solves a very specific infrastructure problem that is prevalent in the gaming and esports research on StarCraft 2.

In its current version our toolset "SC2Tools" is capable of simplifying the work associated with handling files used to create a StarCraft 2 dataset. We are planning to keep updating the software to include more tools, features, and functionalities. Additionally, due to the capability of our software to output JSON files, We claim full interoperability with other replay parsing solutions as long as they keep the same output format.

Finally, based on our previous experience in successfully creating a published dataset that was leveraged in other published material [30], we conclude that our efforts were not in vain and such infrastructure development may be useful to others.

Conflict of Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgements

We would like to acknowledge various contributions by the members of the technical and research community, with special thanks to: Timo Ewalds (DeepMind, Google), Anthony Martin (Sc2ReplayStats), and András Belicza for assisting with our technical questions.

Authors' Contributions

- Conceptualization: Andrzej Białecki;
- Methodology: Andrzej Białecki, Piotr Białecki;
- Software: Andrzej Białecki;
- Technical Oversight: Andrzej Białecki, Piotr Białecki;
- Code Review: Andrzej Białecki, Piotr Białecki, Piotr Sowiński;
- Writing - Original Draft: Andrzej Białecki;
- Writing - Review & Editing: Andrzej Białecki, Piotr Sowiński, Piotr Białecki, Jan Gajewski;
- Licensing and Legal Analysis: Andrzej Białecki, Mateusz Budziak;
- Supervision: Andrzej Białecki, Jan Gajewski;

References

[1] I. Szita, *Reinforcement Learning in Games*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 539–577. [Online]. Available: https://doi.org/10.1007/978-3-642-27645-3_17 (Cited on page: 1).

[2] M. A. Samsuden, N. M. Diah, and N. A. Rahman, "A Review Paper on Implementing Reinforcement Learning Technique in Optimising Games Performance," in *2019 IEEE 9th International Conference on System Engineering and Technology (ICSET)*, 2019, pp. 258–263. [Online]. Available: https://doi.org/10.1109/ICSEngT.2019.8906400 No citations.

[3] M. Lanctot, E. Lockhart, J.-B. Lespiau, V. Zambaldi, S. Upadhyay, J. Pérolat, S. Srinivasan, F. Timbers, K. Tuyls, S. Omidshafiei, D. Hennes, D. Morrill, P. Muller, T. Ewalds, R. Faulkner, J. Kramár, B. D. Vylder, B. Saeta, J. Bradbury, D. Ding, S. Borgeaud, M. Lai, J. Schrittwieser, T. Anthony, E. Hughes, I. Danihelka, and J. Ryan-Davis, "OpenSpiel: A framework for reinforcement learning in games," *CoRR*, vol. abs/1908.09453, 2019. [Online]. Available: http://arxiv.org/abs/1908.09453 No citations.

[4] K. Shao, Z. Tang, Y. Zhu, N. Li, and D. Zhao, "A Survey of Deep Reinforcement Learning in Video Games," 2019. [Online]. Available: https://arxiv.org/abs/1912.10944 No citations.

[5] C. S. Jayaramireddy, S. V. V. S. S. Naraharisetti, M. Nassar, and M. Mekni, "A Survey of Reinforcement Learning Toolkits for Gaming: Applications, Challenges and Trends," in *Proceedings of the Future Technologies Conference (FTC) 2022, Volume 1*, K. Arai, Ed. Cham: Springer International Publishing, 2023, pp. 165–184. [Online]. Available: https://doi.org/10.1007/978-3-031-18461-1_11 No citations.

[6] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov 2019. [Online]. Available: https://doi.org/10.1038/s41586-019-1724-z (Cited on pages: 2, 9).

[7] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, L. Gilpin, P. Khandelwal, V. Kompella, H. Lin, P. MacAlpine, D. Oller, T. Seno, C. Sherstan, M. D. Thomure, H. Aghabozorgi, L. Barrett, R. Douglas, D. Whitehead, P. Dürr, P. Stone, M. Spranger, and H. Kitano, "Outracing champion Gran Turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, Feb 2022. [Online]. Available: https://doi.org/10.1038/s41586-021-04357-7 (Cited on page: 1).

[8] M. J. Campbell, A. J. Toth, A. P. Moran, M. Kowal, and C. Exton, "Chapter 10 - eSports: A new window on neurocognitive expertise?" in *Sport and the Brain: The Science of Preparing, Enduring and Winning, Part C*, ser. Progress in Brain Research, S. Marcora and M. Sarkar, Eds. Elsevier, 2018, vol. 240, pp. 161–174. [Online]. Available: https://doi.org/10.1016/bs.pbr.2018.09.006 (Cited on page: 1).

[9] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *J. Mach. Learn. Res.*, vol. 21, no. 1, 01 2020. (Cited on page: 1).

[10] Y. Pu, S. Wang, R. Yang, X. Yao, and B. Li, "Decomposed Soft Actor-Critic Method for Cooperative Multi-Agent Reinforcement Learning," 2021. [Online]. Available: https://arxiv.org/abs/2104.06655 (Cited on page: 1).

[11] E. O. Jensen, T. Hanghøj, and P. Bukovica Gundersen, "Repositioning Vulnerable Youth Through Educational Esports Programmes," *European Conference on Games Based Learning*, vol. 18, no. 1, pp. 447–454, 10 2024. [Online]. Available: http://dx.doi.org/10.34190/ecgbl.18.1.2896 (Cited on page: 1).

[12] S. Jenny, J. Gawrysiak, and N. Besombes, "Esports.edu: An inventory and analysis of global higher education esports academic programming and curricula," *International Journal of Esports*, vol. 1, no. 1, 09 2021. [Online]. Available: https://www.ijesports.org/article/59/html (Cited on page: 1).

[13] K. Krarup and H. Krarup, "The physiological and biochemical effects of gaming: A review," *Environmental Research*, vol. 184, p. 109344, 2020. [Online]. Available: https://doi.org/10.1016/j.envres.2020.109344 (Cited on page: 1).

[14] J. T. Holden, A. Kaburakis, and R. Rodenberg, "The Future Is Now: Esports Policy Considerations and Potential Litigation," *Journal of Legal Aspects of Sport*, vol. 27, no. 1, pp. 46–78, Feb. 2017. [Online]. Available: http://dx.doi.org/10.1123/jlas.2016-0018 (Cited on page: 1).

[15] E. Nagorsky and J. Wiemeyer, "The structure of performance and training in esports," *PLOS ONE*, vol. 15, no. 8, pp. 1–39, 08 2020. [Online]. Available: https://doi.org/10.1371/journal.pone.0237584 (Cited on page: 1).

[16] P. Xenopoulos, “awpy,” https://github.com/pnxenopoulos/awpy, 2020, acessed: 2024.09.20. (Cited on page: 1).

[17] skadistats, “clarity,” https://github.com/skadistats/clarity, 2013, acessed: 2025.01.01. (Cited on page: 1, 1).

[18] N. Babcock, “boxcars,” https://github.com/pnxenopoulos/awpy, 2016, acessed: 2024.09.20. (Cited on page: 1).

[19] odota, “core,” https://github.com/odota/core, 2014, acessed: 2025.01.01. (Cited on page: 1).

[20] Blizzard, “s2client-proto,” 2017, acessed: 2024.09.20. [Online]. Available: https://github.com/Blizzard/s2client-proto (Cited on page: 1).

[21] A. Belicza, “s2prot,” https://github.com/icza/s2prot, 2016, acessed: 2024.10.12. (Cited on pages: 2, 3).

[22] G. Kim, D. Joerg, K. Leung, A. Hanhikoski, C. Clauss, R. Précenth, D. Neise, H. Wainwright, C. Zemek, Andrene, srounet, I. Kelly, J. Chung, B. Deng, Talv, Chazzz, J. Gravel, rejuxst, A. Nickelsen, Gusgus01, D. Fulton, DasFranck, R. Sanbhadti, S. Krum, T. Gates, B. Peschier, C. Lundquist, D. Kuhta, eqy, and K. Li, “ggtracker/sc2reader: v1.8.0 various fixes and improvements,” may 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6519543 (Cited on page: 1).

[23] J. Formosa, N. O’Donnell, E. M. Horton, S. Türkay, R. L. Mandryk, M. Hawks, and D. Johnson, “Definitions of Esports: A Systematic Review and Thematic Analysis,” *Proc. ACM Hum.-Comput. Interact.*, vol. 6, no. CHI PLAY, Oct. 2022. [Online]. Available: https://doi.org/10.1145/3549490 (Cited on page: 2).

[24] T. Brock, “Ontology and interdisciplinary research in esports,” *Sport, Ethics and Philosophy*, vol. 0, no. 0, pp. 1–17, 2023. [Online]. Available: https://doi.org/10.1080/17511321.2023.2260567 (Cited on page: 2).

[25] A. D. Pizzo, Y. Su, T. Scholz, B. J. Baker, J. Hamari, and L. Ndanga, “Esports Scholarship Review: Synthesis, Contributions, and Future Research,” *Journal of Sport Management*, vol. 36, no. 3, pp. 228 – 239, 2022. [Online]. Available: https://doi.org/10.1123/jsm.2021-0228 (Cited on page: 2).

[26] G. Yamanaka, M. Campos, O. Roble, and L. Mazzei, “eSport: a state-of-the-art review based on bibliometric analysis,” *Journal of Physical Education and Sport*, vol. 21, pp. 3547–3555, 12 2021. (Cited on page: 2).

[27] J. G. Reitman, M. J. Anderson-Coto, M. Wu, J. S. Lee, and C. Steinkuehler, “Esports Research: A Literature Review,” *Games and Culture*, vol. 15, no. 1, pp. 32–50, 2020. [Online]. Available: https://doi.org/10.1177/1555412019840892 No citations.

[28] D. Tang, R. K.-w. Sum, M. Li, R. Ma, P. Chung, and R. W.-k. Ho, “What is esports? A systematic scoping review and concept analysis of esports,” *Heliyon*, vol. 9, no. 12, Dec 2023. [Online]. Available: https://doi.org/10.1016/j.heliyon.2023.e23248 No citations.

[29] A. Białecki, B. Michalak, and J. Gajewski, “Esports training, periodization, and software—a scoping review,” *Applied Sciences*, vol. 14, no. 22, 2024. [Online]. Available: https://doi.org/10.3390/app142210354 (Cited on page: 2).

[30] B. Ferenczi, R. Newbury, M. Burke, and T. Drummond, “Carefully Structured Compression: Efficiently Managing StarCraft II Data,” 2024. [Online]. Available: https://arxiv.org/abs/2410.08659 (Cited on pages: 2, 9, 9).

[31] A. Smerdov, B. Zhou, P. Lukowicz, and A. Somov, “Collection and Validation of Psychophysiological Data from Professional and Amateur Players: a Multimodal eSports Dataset,” 2020. [Online]. Available: https://arxiv.org/abs/2011.00958 (Cited on page: 2).

[32] T. Y. Qian, J. J. Zhang, J. J. Wang, and J. Hulland, “Beyond the Game: Dimensions of Esports Online Spectator Demand,” *Communication & Sport*, vol. 8, no. 6, pp. 825–851, 2020. [Online]. Available: https://doi.org/10.1177/2167479519839436 (Cited on page: 2).

[33] D. Y. Jin, “Historiography of Korean Esports: Perspectives on Spectatorship,” *International Journal of Communication*, vol. 14, pp. 3727–3745, 2020. [Online]. Available: https://ijoc.org/index.php/ijoc/article/view/13795 (Cited on page: 2).

[34] L. Migliore, *What Is Esports? The Past, Present, and Future of Competitive Gaming*. Cham: Springer International Publishing, 2021, pp. 1–16. [Online]. Available: https://doi.org/10.1007/978-3-030-73610-1_1 (Cited on page: 2).

[35] W. Ma, Q. Mi, Y. Zeng, X. Yan, Y. Wu, R. Lin, H. Zhang, and J. Wang, “Large Language Models Play StarCraft II: Benchmarks and A Chain of Summarization Approach,” 2024. [Online]. Available: https://arxiv.org/abs/2312.11865 (Cited on pages: 2, 9).

[36] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson, “The StarCraft Multi-Agent Challenge,” 2019. [Online]. Available: https://arxiv.org/abs/1902.04043 (Cited on page: 9).

[37] T. Pearce and J. Zhu, "Counter-Strike Deathmatch with Large-Scale Behavioural Cloning," in *2022 IEEE Conference on Games (CoG)*, 2022, pp. 104–111. [Online]. Available: https://doi.org/10.1109/CoG51982.2022.9893617 (Cited on page: 2).

[38] M. Seeger, "sc2-ai-coach," https://github.com/manuelseeger/sc2-ai-coach, 2022, acessed: 2024.10.12. (Cited on page: 2).

[39] A. Martin, "sc2replaystats," https://sc2replaystats.com/, acessed: 2025.01.01. (Cited on pages: 2, 9).

[40] A. Białecki, "SC2ReSet: StarCraft II Esport Replaypack Set," 06 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5575796 (Cited on pages: 2, 3, 3, 3, 7, 7).

[41] A. Białecki, N. Jakubowska, P. Dobrowolski, A. Szczap, R. Białecki, and J. Gajewski, "SC2EGSet: StarCraft II Esport Game State Dataset," 06 2023. [Online]. Available: https://doi.org/10.5281/zenodo.5503997 (Cited on pages: 2, 3, 3).

[42] A. Białecki, N. Jakubowska, P. Dobrowolski, P. Białecki, L. Krupiński, A. Szczap, R. Białecki, and J. Gajewski, "SC2EGSet: StarCraft II Esport Replay and Game-state Dataset," *Scientific Data*, vol. 10, no. 1, p. 600, Sep 2023. [Online]. Available: https://doi.org/10.1038/s41597-023-02510-7 (Cited on pages: 2, 3, 9).

[43] M.-J. Kim, D. Lee, J. S. Kim, and C. W. Ahn, "Surrogate-assisted Monte Carlo Tree Search for real-time video games," *Engineering Applications of Artificial Intelligence*, vol. 133, p. 108152, 2024. [Online]. Available: https://doi.org/10.1016/j.engappai.2024.108152 (Cited on page: 2).

[44] A. J. H. Antoine Dupuy, Mark J. Campbell and A. J. Toth, "On the necessity for biomechanics research in esports," *Sports Biomechanics*, vol. 0, no. 0, pp. 1–13, 2024, pMID: 38747541. [Online]. Available: https://doi.org/10.1080/14763141.2024.2354440 (Cited on page: 2).

[45] S. Johar, "Modelling Player Skills in Rocket League through a Behavioural Pattern Mining Approach," *PatternIQ Mining*, 02 2024. [Online]. Available: http://dx.doi.org/10.70023/piqm243 (Cited on page: 2).

[46] Blizzard, "s2protocol," https://github.com/Blizzard/s2protocol, 2013, acessed: 2024.09.12. (Cited on page: 2).

[47] A. Belicza, "mpq," https://github.com/icza/mpq, 2016, acessed: 2024.10.12. (Cited on pages: 2, 3).

[48] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, vol. 32. [Online]. Available: https://dl.acm.org/doi/10.5555/3454287.3455008 (Cited on pages: 2, 2, 5, 5, 6, 8, 9).

[49] W. Falcon and The PyTorch Lightning team, "PyTorch Lightning," 3 2019. [Online]. Available: https://github.com/PyTorchLightning/pytorch-lightning (Cited on pages: 2, 2, 5, 5, 6, 8, 9).

[50] A. Białecki, L. Krupiński, and P. Białecki, "Kaszanas/SC2InfoExtractorGo: 2.1.1 SC2InfoExtractorGo Release," jun 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5296788 (Cited on pages: 2, 4, 4, 5, 6).

[51] A. Białecki, P. Białecki, and L. Krupiński, "Kaszanas/DatasetPreparator: 2.0.0 SC2DatasetPreparator Release," jun 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5296664 (Cited on pages: 2, 3, 5).

[52] A. Białecki and P. Białecki, "Kaszanas/SC2AnonServerPy: 1.0.0 SC2AnonServerPy Release," jul 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5138313 (Cited on pages: 2, 5).

[53] A. Białecki, P. Białecki, and A. Szczap, "Kaszanas/SC2_Datasets: 1.0.2 SC2_Datasets Release," 08 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7028797 (Cited on pages: 2, 4, 7, 8, 8).

[54] G. Kim, D. Joerg, K. Leung, A. Hanhikoski, C. Clauss, R. Précenth, D. Neise, H. Wainwright, C. Zemek, Andrene, srounet, I. Kelly, J. Chung, B. Deng, Talv, Chazzz, J. Gravel, rejuxst, A. Nickelsen, Gusgus01, D. Fulton, DasFranck, R. Sanbhadti, S. Krum, T. Gates, B. Peschier, C. Lundquist, D. Kuhta, eqy, and K. Li, "ggtracker/sc2reader: v1.8.0 Various fixes and improvements," 05 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6519543 (Cited on page: 8).

[55] J. Kegelaers, O. Mairesse, M. Van Heel, P. Wylleman, J. Verschueren, L. Van Ruysevelt, S. Bessi, J. Rainaud, M. Watson, M. Borg, I. Pedraza-Ramirez, J. Ylänne, J. Ragnarsson, D. Milijkovic, I. Bonilla, A. Chamorro, A. Díaz-Moreno, P. Davis, and M. Trotter, "European report: Mental health outcomes in esports players," 03 2025. [Online]. Available: https://www.researchgate.net/publication/389859780_European_report_Mental_health_outcomes_in_esports_players (Cited on page: 8).

[56] A. Dupuy, M. Campbell, and A. Toth, "Differentiating right upper limb movements of esports players who play different game genres," *Scientific Reports*, vol. 15, 02 2025. No citations.

[57] D. Y. Wohn and G. Freeman, “Live streaming, playing, and money spending behaviors in esports,” *Games and Culture*, vol. 15, no. 1, pp. 73–88, 2020. [Online]. Available: https://doi.org/10.1177/1555412019859184 (Cited on page: 8).

[58] J. Graf, “Bringing concepts together: Interdisciplinarity, transdisciplinarity, and ssh integration,” *Fteval Journal for Research and Technology Policy Evaluation*, no. 48, pp. 33–36, 2019. [Online]. Available: https://doi.org/10.22163/FTEVAL.2019.364 (Cited on page: 9).

[59] G. Sonetti, O. Arrobbio, P. Lombardi, I. M. Lami, and S. Monaci, “”only social scientists laughed”: Reflections on social sciences and humanities integration in european energy projects,” *Energy Research & Social Science*, vol. 61, p. 101342, 2020. (Cited on page: 9).

[60] S. Tool, “Spawning tool,” https://lotv.spawningtool.com/, 05 2013, accessed: 2025.03.17. (Cited on page: 9).

[61] D. Chan, “Sc2 revealed,” https://sc2revealed.com/, 06 2020, accessed: 2025.03.17. No citations.

[62] E. Fonn, “Aligulac,” http://aligulac.com/, 04 2011, accessed: 2025.03.17. (Cited on page: 9).