The Journal of Open Source Software

SC2Tools: StarCraft II Toolset and Dataset API

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

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

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

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 (Jayaramireddy et al., 2023; Lanctot et al., 2019; Samsuden et al., 2019; Shao et al., 2019; Szita, 2012; Vinyals et al., 2019; Wurman et al., 2022).

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 (Campbell et al., 2018), (2) computer science (Pu et al., 2021; Rashid et al., 2020), (3) education (Jenny et al., 2021; Jensen et al., 2024), (4) medical sciences (Krarup & Krarup, 2020), and others [Holden et al. (2017);Nagorsky2020]. 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 [Xenopoulos (2020);ClarityGitHub], Rocket League (Babcock, 2016), Dota 2 (odota, 2014; skadistats, 2013), and finally in StarCraft 2 (Belicza, 2016b; Blizzard, 2017; G. Kim et al., 2022a).

Esports can be treated as a subset of gaming with additional requirements for players, such as tournament presence, organized play, training, and professionalization (Formosa et al., 2022). The study of esports is multidisciplinary in nature (Brock, 2023; Pizzo et al., 2022). Due to the growing academic interest in the area of gaming and esports (Białecki et al., 2024; Reitman et al., 2020; Tang et al., 2023; Yamanaka et al., 2021), 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 (Ferenczi et al., 2024; Smerdov et al., 2020).

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 [Qian et al. (2020);Dal2020]. It is also characterized by its fast-paced gameplay and a high skill ceiling (Migliore, 2021). These attributes make for a great environment for testing various AI agents Vinyals et al. (2019). 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 (Martin, n.d.; Seeger, 2022).

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" (Białecki, 2022) and "SC2EGSet" (Białecki, Jakubowska, Dobrowolski, Szczap, et al., 2023) with an accompanying peer-reviewed and published Data Descriptor article (Białecki, Jakubowska, Dobrowolski, Białecki, et al., 2023). 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 (M.-J. Kim et al., 2024). Others put emphasis on statistical calculation within esports landscape (Antoine Dupuy & Toth, 2024). Finally authors describe our work in surveys of related work when working in another games (Johar, 2024).

Our solution uses the official StarCraft 2 replay file format specification provided via Blizzard Entertainment GitHub repository (Blizzard, 2013). Specifically, in "SC2InfoExtractorGo", we extend the community-built Golang implementation of the parser (Belicza, 2016b, 2016a). The output of our software pipeline is a fully prepared dataset, ready for use with our extension of PyTorch (Paszke et al., 2019) and PyTorch Lightning interfaces (Falcon & The PyTorch Lightning team, 2019). Our goal was to lower the technical knowledge required to obtain data from in-game replays.

Software Description

Our software 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" (Białecki, Krupiński, et al., 2022), (2) "DatasetPreparator" (Białecki, Białecki, & Krupiński, 2022), (3) "SC2AnonServerPy" (Białecki & Białecki, 2021). (4) "SC2_Datasets" (Białecki, Białecki, & Szczap, 2022).

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.

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 introduced in# "DatasetPreparator" for the Python utility scripts. Further, data processing Golang tool implementatino is explained in# introducing the "SC2InfoExtractorGo". Finally, data modeling or post-processing tasks are introduced in# diving deeper on "SC2_Datasets" as a Python API implementations for PyTorch (Paszke et al., 2019) and PyTorch Lightning (Falcon & The PyTorch Lightning team, 2019).

The Journal of Open Source Software

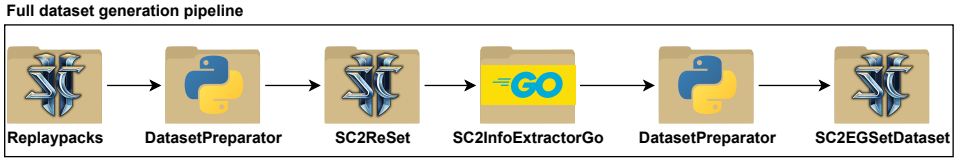

Figure 1: Simplified full pipeline using SC2Tools to create two datasets, "SC2ReSet" (Białecki, 2022) and "SC2EGSet" Dataset (Białecki, Jakubowska, Dobrowolski, Szczap, et al., 2023). Initially introduced in (Białecki, Jakubowska, Dobrowolski, Białecki, et al., 2023).

DatasetPreparator

The "DatasetPreparator" (Białecki, Białecki, & Krupiński, 2022) 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" (Białecki, 2022) 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" (Białecki, 2022) for ease of "SC2EGSet" Dataset reproduction (Białecki, Jakubowska, Dobrowolski, Szczap, et al., 2023).

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 (Belicza, 2016b, 2016a). 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.

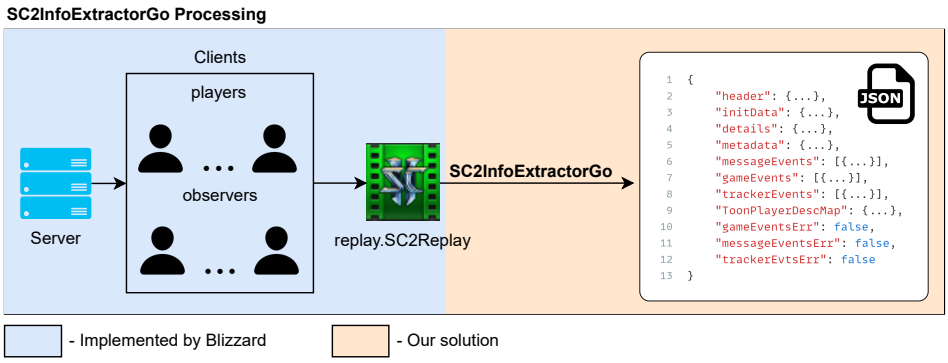

Figure 2: Pictorial representation of the "SC2InfoExtractorGo" functionality (Białecki, Krupiński, et al., 2022). Replays contain the events which happened during gameplay (blue background), our implementations extracts this data and outputs it for further analysis by the user (orange background).

SC2_Datasets

One of our solutions, SC2_Datasets (Białecki, Białecki, & Szczap, 2022) interfaces with the JSON files produced by the SC2InfoExtractorGo (Białecki, Krupiński, et al., 2022). 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.

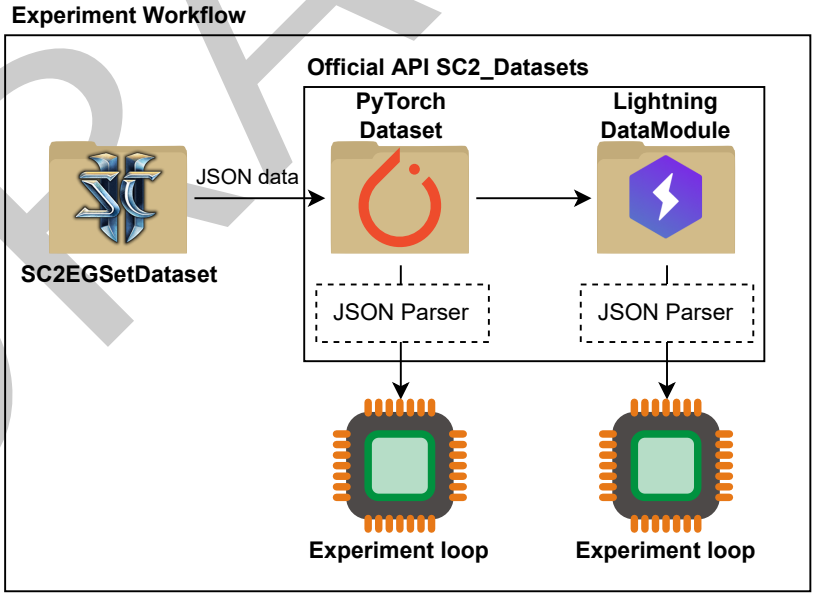

Figure 3: Loading the output of the SC2InfoExtractorGo for machine learning and artificial intelligence use with "SC2_Datasets".

Users have the ability to extend our solution and apply it to their data via the PyTorch (Paszke et al., 2019) and PyTorch Lightning (Falcon & The PyTorch Lightning team, 2019) interfaces.

{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" (Białecki & Białecki, 2021), this functionality may be key for laboratories that wish to share their datasets with a wider community.

{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 (Paszke et al., 2019) and PyTorch Lightning (Falcon & The PyTorch Lightning team, 2019). 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" (Białecki, Krupiński, et al., 2022), as introduced above. The most important procedure of the data extraction pipeline is showcased in Figure 4.

Within "DatasetPreparator" (Białecki, Białecki, & Krupiński, 2022) 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.

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.

```
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 (Białecki, Krupiński, et al., 2022).

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 (Paszke et al., 2019) and PyTorch Lightning (Falcon & The PyTorch Lightning team, 2019).

Usage Information

DatasetPreparator

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.

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.

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".

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".

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.

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 (Białecki, 2022). 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" (Białecki, 2022), it should be available under the directory as specified by the user.

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.

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.

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 (Białecki, Białecki, & Szczap, 2022). Loading a single JSON file following the structure defined in the "SC2_Datasets" 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 (Falcon & The PyTorch Lightning team, 2019) 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.

```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 (Białecki, Białecki, & Szczap, 2022).

```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 (Paszke et al., 2019) dataset interface as defined in (Białecki, Białecki, & Szczap, 2022).

Potential Impact

There exist many implementations built for the purpose of parsing replay files (G. Kim et al., 2022b). 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) (Dupuy et al., 2025; Kegelaers et al., 2025; Wohn & Freeman, 2020). 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 (Graf, 2019; Sonetti et al., 2020).

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 (Białecki, Jakubowska, Dobrowolski, Białecki, et al., 2023). Additionally, an API interface was created to

load and work with the data in PyTorch (Paszke et al., 2019) and PyTorch Lightning (Falcon & The PyTorch Lightning team, 2019).

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 (Chan, 2020; Fonn, 2011; Martin, n.d.; Tool, 2013); and research (Ferenczi et al., 2024; Ma et al., 2024; Samvelyan et al., 2019; Vinyals et al., 2019).

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 (Ferenczi et al., 2024), 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.

Antoine Dupuy, A. J. H., Mark J. Campbell, & Toth, A. J. (2024). On the necessity for biomechanics research in esports. *Sports Biomechanics*, *0*(0), 1–13. https://doi.org/10.1080/14763141.2024.2354440

Babcock, N. (2016). *Boxcars*. https://github.com/pnxenopoulos/awpy.

Belicza, A. (2016a). *Mpq*. https://github.com/icza/mpq.

Belicza, A. (2016b). *s2prot*. https://github.com/icza/s2prot.

Białecki, A. (2022). *SC2ReSet: StarCraft II Esport Replaypack Set* (Version 2.0.0) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.5575796

Białecki, A., & Białecki, P. (2021). *Kaszanas/SC2AnonServerPy: 1.0.0 SC2AnonServerPy Release* (Version 1.0.0). Zenodo. https://doi.org/10.5281/zenodo.5138313

Białecki, A., Białecki, P., & Krupiński, L. (2022). *Kaszanas/DatasetPreparator: 2.0.0 SC2DatasetPreparator Release* (Version 2.0.0). Zenodo. https://doi.org/10.5281/zenodo.

5296664

Białecki, A., Białecki, P., & Szczap, A. (2022). *Kaszanas/SC2_Datasets: 1.0.2 SC2_Datasets Release* (Version 1.0.2). Zenodo. https://doi.org/10.5281/zenodo.7028797

Białecki, A., Jakubowska, N., Dobrowolski, P., Białecki, P., Krupiński, L., Szczap, A., Białecki, R., & Gajewski, J. (2023). SC2EGSet: StarCraft II Esport Replay and Game-state Dataset. *Scientific Data*, *10*(1), 600. https://doi.org/10.1038/s41597-023-02510-7

Białecki, A., Jakubowska, N., Dobrowolski, P., Szczap, A., Białecki, R., & Gajewski, J. (2023). *SC2EGSet: StarCraft II Esport Game State Dataset* (Version 2.0.0) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.5503997

Białecki, A., Krupiński, L., & Białecki, P. (2022). *Kaszanas/SC2InfoExtractorGo: 2.1.1 SC2InfoExtractorGo Release* (Version 2.1.1). Zenodo. https://doi.org/10.5281/zenodo.5296788

Białecki, A., Michalak, B., & Gajewski, J. (2024). Esports training, periodization, and software—a scoping review. *Applied Sciences*, *14*(22). https://doi.org/10.3390/app142210354

Blizzard. (2013). *s2protocol*. https://github.com/Blizzard/s2protocol.

Blizzard. (2017). s2client-proto. In *GitHub*. https://github.com/Blizzard/s2client-proto

Brock, T. (2023). Ontology and interdisciplinary research in esports. *Sport, Ethics and Philosophy*, *0*(0), 1–17. https://doi.org/10.1080/17511321.2023.2260567

Campbell, M. J., Toth, A. J., Moran, A. P., Kowal, M., & Exton, C. (2018). Chapter 10 - eSports: A new window on neurocognitive expertise? In S. Marcora & M. Sarkar (Eds.), *Sport and the brain: The science of preparing, enduring and winning, part c* (Vol. 240, pp. 161–174). Elsevier. https://doi.org/10.1016/bs.pbr.2018.09.006

Chan, D. (2020). *SC2 revealed*. https://sc2revealed.com/.

Dupuy, A., Campbell, M., & Toth, A. (2025). Differentiating right upper limb movements of esports players who play different game genres. *Scientific Reports*, *15*. https://doi.org/10.1038/s41598-025-90949-6

Falcon, W., & The PyTorch Lightning team. (2019). *PyTorch Lightning* (Version 1.4). https://doi.org/10.5281/zenodo.3828935

Ferenczi, B., Newbury, R., Burke, M., & Drummond, T. (2024). *Carefully Structured Compression: Efficiently Managing StarCraft II Data*. https://arxiv.org/abs/2410.08659

Fonn, E. (2011). *Aligulac*. http://aligulac.com/.

Formosa, J., O'Donnell, N., Horton, E. M., Türkay, S., Mandryk, R. L., Hawks, M., & Johnson, D. (2022). Definitions of Esports: A Systematic Review and Thematic Analysis. *Proc. ACM Hum.-Comput. Interact.*, *6*(CHI PLAY). https://doi.org/10.1145/3549490

Graf, J. (2019). Bringing concepts together: Interdisciplinarity, transdisciplinarity, and SSH integration. *Fteval Journal for Research and Technology Policy Evaluation*, *48*, 33–36.

Holden, J. T., Kaburakis, A., & Rodenberg, R. (2017). The Future Is Now: Esports Policy Considerations and Potential Litigation. *Journal of Legal Aspects of Sport*, *27*(1), 46–78. https://doi.org/10.1123/jlas.2016-0018

Jayaramireddy, C. S., Naraharisetti, S. V. V. S. S., Nassar, M., & Mekni, M. (2023). A Survey of Reinforcement Learning Toolkits for Gaming: Applications, Challenges and Trends. In K. Arai (Ed.), *Proceedings of the future technologies conference (FTC) 2022, volume 1* (pp. 165–184). Springer International Publishing. https://doi.org/10.1007/978-3-031-18461-1_11

Jenny, S., Gawrysiak, J., & Besombes, N. (2021). Esports.edu: An inventory and analysis of

global higher education esports academic programming and curricula. *International Journal of Esports*, *1*(1). https://www.ijesports.org/article/59/html

Jensen, E. O., Hanghøj, T., & Bukovica Gundersen, P. (2024). Repositioning Vulnerable Youth Through Educational Esports Programmes. *European Conference on Games Based Learning*, *18*(1), 447–454. https://doi.org/10.34190/ecgbl.18.1.2896

Johar, S. (2024). Modelling Player Skills in Rocket League through a Behavioural Pattern Mining Approach. *PatternIQ Mining*. https://doi.org/10.70023/piqm243

Kegelaers, J., Mairesse, O., Van Heel, M., Wylleman, P., Verschueren, J., Van Ruysevelt, L., Bessi, S., Rainaud, J., Watson, M., Borg, M., Pedraza-Ramirez, I., Ylänne, J., Ragnarsson, J., Milijkovic, D., Bonilla, I., Chamorro, A., Díaz-Moreno, A., Davis, P., & Trotter, M. (2025). *European report: Mental health outcomes in esports players*. https://doi.org/10.13140/RG.2.2.10296.25605

Kim, G., Joerg, D., Leung, K., Hanhikoski, A., Clauss, C., Précenth, R., Neise, D., Wainwright, H., Zemek, C., Andrene, srounet, Kelly, I., Chung, J., Deng, B., Talv, Chazzz, Gravel, J., rejuxst, Nickelsen, A., … Li, K. (2022b). *ggtracker/sc2reader: v1.8.0 Various fixes and improvements* (Version v1.8.0). Zenodo. https://doi.org/10.5281/zenodo.6519543

Kim, G., Joerg, D., Leung, K., Hanhikoski, A., Clauss, C., Précenth, R., Neise, D., Wainwright, H., Zemek, C., Andrene, srounet, Kelly, I., Chung, J., Deng, B., Talv, Chazzz, Gravel, J., rejuxst, Nickelsen, A., … Li, K. (2022a). *Ggtracker/sc2reader: v1.8.0 various fixes and improvements* (Version v1.8.0). Zenodo. https://doi.org/10.5281/zenodo.6519543

Kim, M.-J., Lee, D., Kim, J. S., & Ahn, C. W. (2024). Surrogate-assisted Monte Carlo Tree Search for real-time video games. *Engineering Applications of Artificial Intelligence*, *133*, 108152. https://doi.org/10.1016/j.engappai.2024.108152

Krarup, K. B., & Krarup, H. B. (2020). The physiological and biochemical effects of gaming: A review. *Environmental Research*, *184*, 109344. https://doi.org/10.1016/j.envres.2020.109344

Lanctot, M., Lockhart, E., Lespiau, J.-B., Zambaldi, V., Upadhyay, S., Pérolat, J., Srinivasan, S., Timbers, F., Tuyls, K., Omidshafiei, S., Hennes, D., Morrill, D., Muller, P., Ewalds, T., Faulkner, R., Kramár, J., Vylder, B. D., Saeta, B., Bradbury, J., … Ryan-Davis, J. (2019). OpenSpiel: A framework for reinforcement learning in games. *CoRR*, *abs/1908.09453*. http://arxiv.org/abs/1908.09453

Ma, W., Mi, Q., Zeng, Y., Yan, X., Wu, Y., Lin, R., Zhang, H., & Wang, J. (2024). *Large Language Models Play StarCraft II: Benchmarks and A Chain of Summarization Approach*. https://arxiv.org/abs/2312.11865

Martin, A. (n.d.). *sc2replaystats*. https://sc2replaystats.com/.

Migliore, L. (2021). What Is Esports? The Past, Present, and Future of Competitive Gaming. In L. Migliore, C. McGee, & M. N. Moore (Eds.), *Handbook of esports medicine: Clinical aspects of competitive video gaming* (pp. 1–16). Springer International Publishing. https://doi.org/10.1007/978-3-030-73610-1_1

odota. (2014). *Core*. https://github.com/odota/core.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 32). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

Pearce, T., & Zhu, J. (2022). Counter-Strike Deathmatch with Large-Scale Behavioural

Cloning. *2022 IEEE Conference on Games (CoG)*, 104–111. https://doi.org/10.1109/CoG51982.2022.9893617

Pizzo, A. D., Su, Y., Scholz, T., Baker, B. J., Hamari, J., & Ndanga, L. (2022). Esports Scholarship Review: Synthesis, Contributions, and Future Research. *Journal of Sport Management*, *36*(3), 228–239. https://doi.org/10.1123/jsm.2021-0228

Pu, Y., Wang, S., Yang, R., Yao, X., & Li, B. (2021). *Decomposed Soft Actor-Critic Method for Cooperative Multi-Agent Reinforcement Learning*. https://arxiv.org/abs/2104.06655

Qian, T. Y., Zhang, J. J., Wang, J. J., & Hulland, J. (2020). Beyond the Game: Dimensions of Esports Online Spectator Demand. *Communication & Sport*, *8*(6), 825–851. https://doi.org/10.1177/2167479519839436

Rashid, T., Samvelyan, M., De Witt, C. S., Farquhar, G., Foerster, J., & Whiteson, S. (2020). Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.*, *21*(1).

Reitman, J. G., Anderson-Coto, M. J., Wu, M., Lee, J. S., & Steinkuehler, C. (2020). Esports Research: A Literature Review. *Games and Culture*, *15*(1), 32–50. https://doi.org/10.1177/1555412019840892

Samsuden, M. A., Diah, N. M., & Rahman, N. A. (2019). A Review Paper on Implementing Reinforcement Learning Technique in Optimising Games Performance. *2019 IEEE 9th International Conference on System Engineering and Technology (ICSET)*, 258–263. https://doi.org/10.1109/ICSEngT.2019.8906400

Samvelyan, M., Rashid, T., Witt, C. S. de, Farquhar, G., Nardelli, N., Rudner, T. G. J., Hung, C.-M., Torr, P. H. S., Foerster, J., & Whiteson, S. (2019). *The StarCraft Multi-Agent Challenge*. https://arxiv.org/abs/1902.04043

Seeger, M. (2022). *sc2-ai-coach*. https://github.com/manuelseeger/sc2-ai-coach.

Shao, K., Tang, Z., Zhu, Y., Li, N., & Zhao, D. (2019). *A Survey of Deep Reinforcement Learning in Video Games*. https://arxiv.org/abs/1912.10944

skadistats. (2013). *Clarity*. https://github.com/skadistats/clarity.

Smerdov, A., Zhou, B., Lukowicz, P., & Somov, A. (2020). *Collection and Validation of Psychophysiological Data from Professional and Amateur Players: a Multimodal eSports Dataset*. arXiv. https://doi.org/10.48550/ARXIV.2011.00958

Sonetti, G., Arrobbio, O., Lombardi, P., Lami, I. M., & Monaci, S. (2020). "Only social scientists laughed": Reflections on social sciences and humanities integration in european energy projects. *Energy Research & Social Science*, *61*, 101342.

Szita, I. (2012). Reinforcement Learning in Games. In M. Wiering & M. van Otterlo (Eds.), *Reinforcement Learning: State-of-the-Art* (pp. 539–577). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-27645-3_17

Tang, D., Sum, R. K., Li, M., Ma, R., Chung, P., & Ho, R. W. (2023). What is esports? A systematic scoping review and concept analysis of esports. *Heliyon*, *9*(12). https://doi.org/10.1016/j.heliyon.2023.e23248

Tool, S. (2013). *Spawning tool*. https://lotv.spawningtool.com/.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., … Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, *575*(7782), 350–354. https://doi.org/10.1038/s41586-019-1724-z

Wohn, D. Y., & Freeman, G. (2020). Live streaming, playing, and money spending behaviors in

eSports. *Games and Culture*, *15*(1), 73–88. https://doi.org/10.1177/1555412019859184

Wurman, P. R., Barrett, S., Kawamoto, K., MacGlashan, J., Subramanian, K., Walsh, T. J., Capobianco, R., Devlic, A., Eckert, F., Fuchs, F., Gilpin, L., Khandelwal, P., Kompella, V., Lin, H., MacAlpine, P., Oller, D., Seno, T., Sherstan, C., Thomure, M. D., … Kitano, H. (2022). Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, *602*(7896), 223–228. https://doi.org/10.1038/s41586-021-04357-7

Xenopoulos, P. (2020). *Awpy*. https://github.com/pnxenopoulos/awpy.

Yamanaka, G., Campos, M., Roble, O., & Mazzei, L. (2021). eSport: a state-of-the-art review based on bibliometric analysis. *Journal of Physical Education and Sport*, *21*, 3547–3555. https://doi.org/10.7752/jpes.2021.06480