
DEPARTMENT OF TELECOMMUNICATIONS

*Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.*

*We hereby express our thanks to our supervisor – Piotr Chołda, PhD, who was guiding the process of creation of this thesis. The support in granting the needed resources for the study and feedback provided by Mr. Chołda are invaluable.*
*Special thanks to Krzysztof Rusek, PhD – for consulting the approach for proper machine learning model execution.*
*This research was supported in part by PLGrid Infrastructure.*

# Contents

# 1. Implementation

## 1.1. Persistence and security

The Data Collection module consists of several submodules, but the core of it is the backend API[1] which was written in the 8th version of Java[2]. The responsibilities of that API are mainly the persistence of captured data and user authentication.

The whole architecture of the API is based on the Spring[3] framework which is widely used for such purposes and provides many capabilities in fields of securing and persisting data. Spring is divided into many projects that create compatible ecosystem and make it possible to develop scalable web applications. In order to make development in Spring easier and faster the authors of the framework created Spring Boot[4] which sets up the environment for the developer and provides many easily imported libraries (so-called starters) to use.

To perform databases operations using the idea of object–relational mapping (ORM) the implementation of Java Persistence API[5] (JPA) provided by Spring Boot was used. The choice of the database was in the scope of relational databases because the captured data forms the time series which are easily mapped to the structure of the table. The authors decided to use PostgreSQL[6] because of the popularity and wide range of support. The data was stored in three different tables. The first consists of the user's login, password and granted authority which was required to distinguish administrators from ordinary users. The second one includes sequences of user's captured actions with timestamps, screen resolutions and types of the events. The last one stores all possible action types with the description.

The collected data is sent to the API via Representational State Transfer (REST), which is nowadays one of the most popular interfaces between web applications, because of its simplicity and native support in many different programming languages and technologies. The schema of the interface was established at the beginning of the work on this module, so it was possible to create the API and the fronted application simultaneously.

[1] Application Programming Interface
[2] *https://www.java.com/*
[3] *https://spring.io/*
[4] *https://spring.io/projects/spring-boot*
[5] *https://www.oracle.com/java/technologies/persistence-jsp.html*
[6] *https://www.postgresql.org/*

To protect the data from invalid properties, the input was also validated before saving to the database in such a way that all of persisted data contained all required elements. Securing such an application from malware users is also important issue, so the access was restricted only for signed users that existed in the database. The security was provided by using OAuth2[7] and JSON Web Token[8] (JWT). The API was working as a OAuth2 provider which issued JWT tokens and authorized them basing on the user's credentials stored in the database. Such a mechanism allowed to control access to the application and enforce registration before the use of the system.

The API was prepared with two different switchable profiles: admin oriented and user oriented. They were created to provide flexibility in accounts creation — respectively — only by admin and anyone. In further work only user oriented profile was used.

## 1.2. Presentation and reverse proxy

The second core submodule created for the purpose of collecting the data from the users is the frontend application exposed to the internet. This can be split further into two different functional modules — client and server-side.

### 1.2.1. Client side module

The client-side module is an end-user presentation layer that is built with the HTML5[9] and CSS3[10] — technologies that are the core and standard for modern web applications nowadays. HTML is a widely used markup language used to create hypertext documents and CSS is used as a style-sheet for modifying the design of web documents. The content of the website is a free, prebuilt template taken from the Colorlib[11] collection which web templates are licensed under the CC BY 3.0 License [12]. The mechanism that collects users' mouse actions is designed as a "plugin" script, meaning that the template can be changed easily, so different styled environments that serve various purposes can be used to collect the data.

To use the system and participate in the research, an interested user is required to first accept the consents of usage of the website as well as accept the usage of the cookies and then registration is allowed and required, and afterward, the user is redirected to the login page. To persists consistency of the registration and login, the custom static web pages were created to be easily transferable between the templates. On each document, the FAQ panel with more information regarding the project is exposed as a drop-down list. To authenticate, users are required to provide the credentials to log in to the system. These credentials are then sent through https — which means that they are secured and encrypted — to

[7] *https://oauth.net/2/*
[8] *https://jwt.io/*
[9] *https://html5.org/*
[10] *https://www.w3.org/Style/CSS/Overview.en.html*
[11] *https://colorlib.com/*
[12] Creative Commons Attribution 3.0 License — *https://creativecommons.org/licenses/by/3.0/*

the reverse proxy server which then performs some action to authenticate a user, sets the cookie with granted JWT[13] token and redirects the user to the homepage. More on the user authentication and the token obtaining sequence is described in The whole sequence is described and shown in subsection 1.2.2).

After a successful login, the token allows for site usage and ensures the identity of the user. From now on, the actions performed by the user are recorded into the batches and sent to the API every two seconds. The event listeners are awaiting four different action types: mouse move, mouse-down, mouse-up, mouse wheel action. Collected actions are packed into JSON object and sent to the server side API (described in 1.2.2) — the fields included are shown in the Listing 1.1.

**Listing 1.1.** JSON schema for mouse event batches

### 1.2.2. Server side module

The server side module is build with Node.js[14] runtime with the Express[15] framework on the top of it. Node.js is an asynchronous JavaScript runtime, which is widely used to build high-end, scalable commercial applications. Express is light-weight and simple to use, yet powerful web framework for Node.js, which allows for a fast and convenient HTTP server set-up server for serving the static web documents over the internet. This module is serving the purpose of reverse proxy between clients and the backend API.

The main responsibilities of the reverse proxy are signing up the user, token granting, validation and caching, serving static web documents storing and passing the data to for persistence.

When a new user tries to sign-up, secured with TLS credentials from the sign-up form are sent to one of the proxy endpoints, where they are then passed to the backend API. Because the backend is not exposed anywhere over the public network, there is no need to secure the credentials with TLS.
The user is granted JWT Access and Refresh tokens after logging in using the login form. The user credentials are secured with TLS and received by proxy API, whereas the proxy server is additionally appending the Client id and Client secret for OAuth2 server to the request "Authorization" header, as the proxy server authenticates to OAuth2 server with HTTP Basic authentication scheme. Credentials of the user that wishes to log in are included in the body of the request. When the user is properly authenticated, the OAuth2 server responds with a valid JWT token which is then set as a cookie with HttpOnly, Secure, and SameSite strict options. When the token is successfully granted, it is also saved to the Redis, which is a simple to use key-value NoSQL database suitable for caching. The retrieval of such a token is very fast, so this is serving the purpose of the caching system, which results in a great efficiency improvement and lower response time of the server. For each interaction and request for resources, the user has to hold a valid JWT Token. First, the token is searched in the cache database. If it does not figure there, the proxy server is attempting to check the token with the backend OAuth2 server. If the server responds

[13] *https://jwt.io/*
[14] *https://nodejs.org/*
[15] *https://expressjs.com/*

with Bad Request status or the access token is not set on the user request, the refresh token is being used for access token renewal. The complete sequence visualization can be seen in Fig. 1.1. The proxy server is storing the user mouse data received from the client side and periodically passes it to the backend API for persistence.

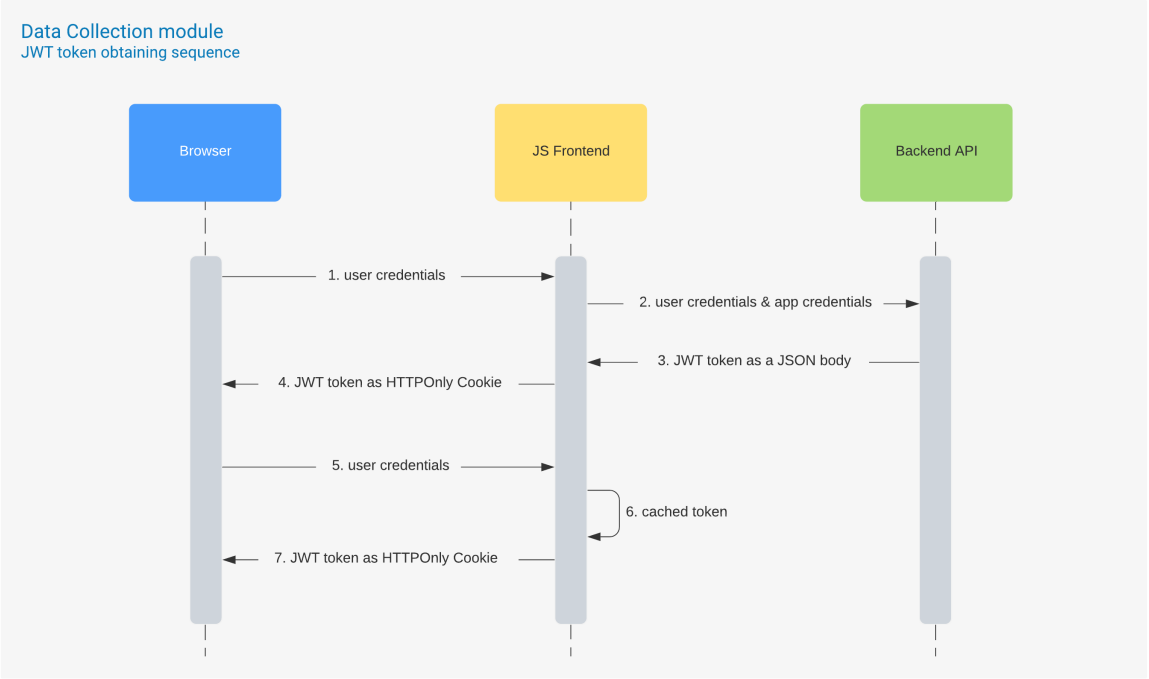


**Fig. 1.1.** JWT token obtaining sequence

## 1.3. Performance tests

Solutions presented in previous chapters were deployed to the Heroku Cloud Application Platform[16] to deliver the possibility of using the system among a wide range of users. Heroku as a cloud provider has a free pricing plan for non-commercial apps, which is suitable for scientific usage such as this work. The deployment process is performed using the idea of containerization based on Docker[17] which is a pretty convenient way of application delivery to the web server. To enable such a possibility, the Docker-files with container configuration description were prepared for both frontend and backend services and also docker-compose configuration files were created to allow easy testing and development in the local environment.

[16] *https://www.heroku.com/*
[17] *https://www.docker.com/*

In order to deliver reliable and efficient product to the users, the performance tests were prepared using JavaScript[18] programming language in Node.js[19] environment. Those tests cover mainly the token exchange and refreshment and also allow to test the behavior of application during increased load produced by using the system by many different users. The results of the tests showed that the Heroku free pricing plan had some limitations regarding database storage volume and response time from the application. It turned out that the database in such a plan permit to store only 20,000 rows overall which is unacceptable for the data stream with the time resolution measured in milliseconds because a single user may cause exceeding the limit in a relatively short time. Moreover, the described cloud system suspends the working of the application when it is idle and the waking up time is very long which is manifested in a usage lag after a short period. Described issues with the Heroku Cloud Application Platform forced the authors to find another suitable cloud provider.

## 1.4. Deployment and orchestration

In order to overcome the issues described in the previous chapter, the cloud providers' research was conducted, and based on the results it was decided to adopt the Google Cloud Platform[20] (GCP) as a deployment environment.

Unlike Heroku, GCP had no free pricing plan, but it does not restrict the usage of resources and therefore it met the project assumptions, and moreover, it has some extra advantages like native support for containers orchestration. The latter is provided by using Kubernetes[21], the open-source solution introducing flexible deployments, scaling and container management. Those features drastically simplify working with application development in a cloud but require additional setup and configuration files.

Kubernetes natively supports the usage of Docker containers so the ones prepared before can be reused, but the deployment process requires a so-called Deployment configuration files that enable tuning the resources assigned to the single instance of an application and also the number of the replicas of the application. The basic unit managed by Kubernetes is Pod which may consist of many single Docker containers, but in this work the Pod is associated with a single container of application. The described scaling method is known as Horizontal Pod Scaling and can be seen in the Fig. 1.2 as a several same Pods in single Deployment. It improves reliability and allows to increase the limit of the maximal load accepted by a single deployment of the application because the load is split into the mirror instances of the same application that work in parallel.

To deliver the load to all of the mirror applications, Kubernetes uses Services as an entry point to a group of Pods that are managed by one Deployment (in the Fig. 1.2 they are drawn on the edge of the Deployment rectangles to mark their gateway nature). The Service exposes the Deployment under the

[18] *https://developer.mozilla.org/en-US/docs/Web/JavaScript*
[19] *https://nodejs.org/*
[20] *https://cloud.google.com/*
[21] *https://kubernetes.io/*

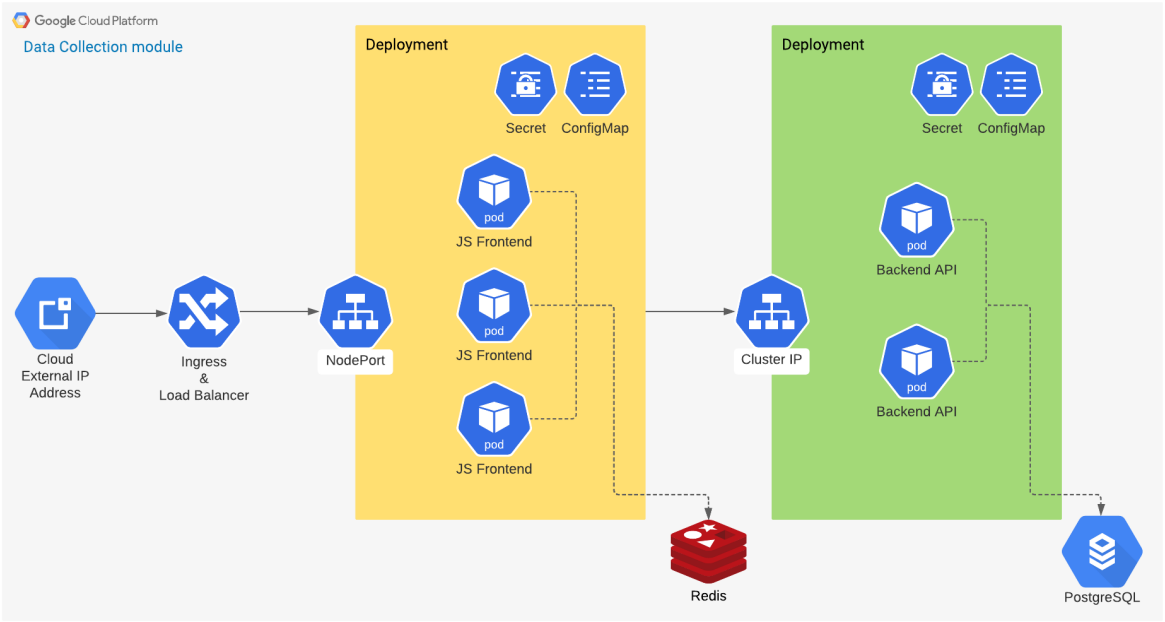


**Fig. 1.2.** Cloud architecture

single DNS[22] name and updates the underlying IP address in cases of its change, but it also works as a load balancer which distributes the load among the managed Pods. Kubernetes has several different Service types but in this work two of them are used: Cluster IP that exposes the Deployments in the scope of Kubernetes but hides them from external access and NodePort that permits the access using port-forwarding. The latter one is dictated by the usage of Google Cloud HTTP(S) load balancer and it also requires additional configuration by using an Ingress and obtaining the static external IP address.

To provide encryption of connection, Transport Layer Security protocol (TLS) should be used, which requires the SSL Server Certificate, which can be issued only for existing valid domain names, thus the authors were forced to buy such a domain name, while the certificate is issued by GCP.

The configuration also includes some sensitive data, such as database credentials, OAuth2 secret, and SSH-RSA keys, thus the Secrets are used with encoded content. The insensitive data are included in ConfigMaps that are similar to the Secrets but their content is plain text. These configurations are matched to the appropriate component by labels and therefore they are drawn inside the Deployment rectangles in Fig. 1.2, in practice they can be mounted as volumes basing on the component labels and can also expose their values as environmental variables.

The whole architecture of the collector system is presented in Fig. 1.2, but the figure has several simplifications such as a way of database deployment. In contrast to the Redis cache database, the main database instance is not deployed in Kubernetes itself, but the Cloud SQL service with PostgreSQL is used to achieve straightforward management and reliability.

[22] Domain Name System

## 1.5. Custom bot

In order to prepare a dataset that consists of two classes — the valid user and human impersonating bot, the custom bot was prepared.

The main idea behind this module is to first create scenarios that next could be executed by the bot software. To achieve the most convenient creation of scenarios, the authors prepared a utility to record the coordinates and actions of the mouse. Using this approach, the bot could impersonate a user in a more natural way. Recorded mouse events are stored as a YAML[23] schema, which is a human-readable standard for data serialization. The structure of a YAML scenario schema is shown in Listing 1.2.

**Listing 1.2.** YAML schema for bot event scenarios

After the scenarios are recorded, the mechanism to execute them is required. The Custom Bot executor script reads the YAML file and loads it to the memory as a list of events, that is then traversed. The executor script performs an action stored as an "event" filed at the coordinates $(x, y)$. If the coordinates are different than the current mouse position, the cursor is moved, but to make it more natural, movement is based on the Bézier curve concept. The Bézier curves are used to effectively represent a smooth curve on a computer screen, as stated by B. T. Bertka [**bezier-curves**]. Knowing this, this concept can be used to simulate smooth human-like movements.

To be able to run a set of scenarios, the module introduces a so-called "batch runner" for predefined cases. This automation saves a lot of time and effort because a set of scenarios can be executed at once to generate data for the bot class.

## 1.6. Data serialization

The persisted data from the database is to serve as the input for a machine learning model, however, due to inconvenient usage of SQL queries for such purposes, the serialization tool was developed in order to translate the state of the database records to binary files. In the presented solution, such files are treated as immutable, so the operation of serialization can be performed only once, which results in the improvement of the required time to read the data by the machine learning model. Moreover, binary files allow straightforward sending and storing data in external infrastructures, such as PLGrid[24], because it does not require maintenance of database engine in that environment. Such an approach also makes it possible to process data and prepare them in a way that is required by the machine learning model.

To fulfill these requirements, the serializer and deserializer tools were developed for saving sequences in binary files and further reading those data in Python script. The serialization is performed using a tool written in Go[25] and provides additional tuning of resulted binary files and configuration of connection

[23] *https://yaml.org/*
[24] *http://www.plgrid.pl/*
[25] *https://golang.org/*

to the database. Among the others, the tool allows choosing the type of event generated by the user such as mouse click or move, minimum sequence length that should be considered as valid data, minimum screen resolution in order to filter the actions from mobile devices or the time gap between two actions that should be considered as the boundary between two sequences. The results of such filtration are saved in the chosen directory dividing the output into the user's directories and saving each separate sequence in a single file, so the output consists of many user's directories each containing many single sequence files. It is also possible to use a so-called one-user mode that enables generating output data only for a single arbitrary chosen user for debugging purposes.

In order to make serialization uncomplicated and transferable between different programming languages, the serialization framework was used. At the beginning the chosen one was Apache Avro[26] which allows to defining the schema in simple JSON file. The serialization is performed with help of a library that allows reading schema and saving programming language native objects to binary files. In the presented solution serialization should be performed using the library for Go and the deserialization with support of the library for Python, however, they proved to be incompatible which resulted in errors in deserialized data.

To avoid invalid data and to do not spend too much time on finding the bug in those libraries, another approach was taken by applying the Protocol Buffers[27] technology. Protocol Buffers or simply Protobuf is a method of serializing data to the binary form, but the real advantage of it is official multilingual support by generating a serializing code in required programming language. Protobuf also requires the definition of a schema like Apache Avro, but unlike Avro, the config file format is developed especially for Protobuf, at the same time it is also readable and effortless to write. Basing on the created schema the code was generated both for the serializer and the deserializer, but in the case of deserializer, the data is directly read to the Pandas[28] Dataframe objects, which provides a simple interface to manipulate huge amount of data. The deserializing tool was designed to work directly in the front of the neural network with additional preprocessing step, therefore it was written in Python to provide flexibility in adapting this feature in further work. It is also able to read data from the directory tree created by the serializer which enables seamless integration between these two parts.

## 1.7. Prometheus computing cluster

Data processing and machine learning model training and evaluation require a lot of computing resources. Performing such computations locally, on a single processing core is not the best approach, at least not the most effective method.

To enhance the process of manipulating the collected data, the computing cluster was used as a part of the educational grant issued by the PLGrid. To make the most out of the computing cluster,

[26] *https://avro.apache.org/*

[27] *https://developers.google.com/protocol-buffers*

[28] *https://pandas.pydata.org/*

plenty of helpers and steering scripts were prepared to automate the process of managing the Prometheus supercomputer.

Prometheus cluster is sharing the resources among multiple users. Therefore, the computation is scheduled as a job by the SLURM Workload Manager[29]. To submit a batch job to the SLURM manager, the input shell script is needed and in this case, the *sbatch_job_config* script which contains the job description was prepared.

On the top of the *sbatch* script, a helper shell script exists — *run-plgrid-job*, which is obtaining useful values that steer the SLURM scheduler as well as triggers the notification job described in previous subsection 1.9.4.

For automation, the Git version control system was used to implement a very simple continuous delivery system. The combination of the git versioning system with its aliases allowed to create a *deploy* alias, which was simply performing the pushing to the remote system — in this case, the Prometheus cluster — but also connecting to it and submitting the job. After successfully submitting, the user receives a notification on Slack. When the job starts, crashes, or finishes properly, the notification is sent again to the Slack. This gives a very convenient insight into the process and progress of the job without constantly checking for the status manually. The whole setup can be done easily via the *make* program with prepared *Makefile* file.

## 1.8. Data preprocessing

To fulfill the requirements of the adapted transfer learning model the preprocessing tool was developed alongside a whole bunch of settings. The collected data sequences are in form of coordinates' series, so it is necessary to represent them as 3-dimensional pictures which are the input to the convolutional neural network. The described tool allows transferring those sequences into multidimensional arrays of integers and also scale them to the required input size of a used transfer learning model.

The data consists of isolated points that are result of discrete sampling, so to improve performance efficiency these points on the "painting" are interpolated using a linear interpolation mechanism. It was verified on the collected dataset that interpolated data results in better accuracy of a model in comparison to the isolated ones. The presented tool works also as a splitter between training and testing sets using the provided ratio between them and as a label assigner which allows adding the corresponding labels to the single sequence basing on the provided identifier of the bot user.

Using this tool it is also possible to increase the number of bot samples by using several repetitions of the original bot set in cases of an imbalanced dataset. The output of the preprocessor is a tuple of training and testing dataset along with labels, which can be directly the input of a neural network model.

[29] SLURM homepage — *https://slurm.schedmd.com/documentation.html*

## 1.9. Bot Detection module

To validate the consistency and quality of the gathered data the Bot Detection module is introduced. This module includes various tools and utilities helpful for manipulating the data, model observations and results analysis. The selected submodules are briefly described in the following subsections.

### 1.9.1. CSV writer

To persist machine learning model execution results and statistics for later analysis, the outcome data should be stored in a consistent way. The CSV[30] file is a convenient file format to persist this kind of data because the values from different execution sessions can be easily appended using established schema. The CSVWriter class introduces methods for appending the data to the file. The other helper methods allow for checking the correctness of the file and its content.

### 1.9.2. Uploading charts on web

To see whether the machine learning model is actually getting better during the training process, accuracy and loss charts can be used as a really helpful method to determine the learning curve. To receive a "live" preview of the model execution results with charts as a notification in Slack communicator, the charts needed to be stored somewhere on the internet. The easiest way to achieve this requirement was to use the API of the Imgur[31] platform. Thanks to the official module that can be used with Python, the platform offers a simple way to upload the images to the service and returns the URL to the stored image when successfully uploaded.

### 1.9.3. Results terminating

When the model was executing the calculations multiple times, the results for each iteration were collected. This is giving a really interesting insight into the model and its stability. The tool called *ResultTerminator* was serving this purpose. The helper methods were used to determine whether the file is existing already and if not creating it. To avoid concurrent saving problems, the file is achieving the lock and releasing it when the content is successfully saved to CSV[32] file.

### 1.9.4. Slack notifier

Working with PLGrid infrastructure and running a job on the Prometheus computing cluster was very helpful in terms of delegating the great amount of computing load onto external resources. However, this solution has its downsides. One of the major issues is the job queueing mechanism, which does not determine when the job will be executed and therefore it requires constant manual checking for results.

[30] Comma-Separated Values
[31] *https://imgur.com/*
[32] Comma-Separated Values

To avoid that somehow, the authors prepared a custom mechanism, that can monitor the status of the job and notify the Prometheus user through the Slack[33] channel. Three following message templates were prepared for the purpose of notification:

1. Simple message — Generic message type build as a JSON template with help of Builder design pattern to have flexibility in customizing the message.
2. Pending job message — able to notify about: job title, reporter, commit hash, job registration date and time, header for app preview, info message taken from a commit. Build as a special case of Simple type message.
3. Results message — The most complex message with all the statistic results produced as a result of the model computation. Prepared as a JSON template with the help of the Builder design pattern.

To avoid that somehow, the authors prepared a custom mechanism, that can monitor the status of the job and notify the Prometheus user through the Slack[34] channel using observability-like architecture.

### 1.9.5. Statistics calculation

The prepared model was intended to be running several times in order to measure the consistency and stability of the results. To properly measure and visualize the model scores, a dedicated module for processing the results was prepared. Two main core parts can be distinguished — the plotting utilities and the statistics metrics calculation submodule.

The submodule for calculating a metrics allows for producing the following model statistics:

1. Mean accuracy
2. Mean loss
3. Mean FAR[35]
4. Mean FRR[36]
5. Mean true negatives
6. Mean false negatives
7. Mean true positives
8. Mean false positives

The submodule for plotting the charts allows for the following:

[33] *https://slack.com/* – a messaging app for teams
[34] *https://slack.com/* – a messaging app for teams
[35] False Acceptance Rate
[36] False Rejection Rate

1. Generic linear plot creation used to display accuracy and loss charts

2. Creating the accuracy percentile histogram

The above metrics and functionalities are designed to produce the averaged model results for multiple executions. The developed model is stable and the error is irrelevant between each execution.

## 1.10. Machine learning model

The machine learning model prepared and trained on the collected dataset was the part of the data evaluation final part.

The dataset collected during the thesis preparation is not sufficient enough due to the limitation for proper and full machine learning process. Google cloud services are able to handle the deployment of the whole system efficiently. However, this form of hosting and maintaining the infrastructure of the application comes with its price. The whole infrastructure was running for about two months and during this time generated the cost of a total $300. This amount of cash is the highest financial outlay that could have been incurred by the authors of the work since no other scholarship than PLGrid[37] computation cluster was granted for the purpose of preparing the thesis. Longer exposure on the web would cost extra money, that could not be afforded. The other matter is that the engineering thesis defending has its term specified, thus the project has been carefully thought-out in the manner of time since the beginning of the implementation. In order to meet the adopted milestones and goals, the data collection period had to follow strict deadlines. The duration of the period when the data was collected could not be extended to broader terms.

The remedy for a small amount of data collected was to use the transfer learning technique. As stated in the "A survey of transfer learning" [**transfer-learning-def**] by Weiss et al. article — "In certain scenarios, obtaining training data that matches the feature space and predicted data distribution characteristics of the test data can be difficult and expensive. Therefore, there is a need to create a high-performance learner for a target domain trained from a related source domain. This is the motivation for transfer learning. Transfer learning is used to improve a learner from one domain by transferring information from a related domain". The base feature vector for transferring to the bot recognition domain was taken from the TensorFlow Hub[38] website. The model was pre-trained on ImageNet dataset[39] using the architecture of Inception V3, which is showing great potential in the terms of the computer vision with improved performance over the previous versions, with a relatively modest computation cost [**inception-v3**]. The model was built with the following dimensions of 299×299×3 for the data input. The topmost layers were added for the transferring and tuning to the bot detection domain. For the optimization of the model, the optimizer called Adam with the learning rate fixed at the $0{,}5 \times 10^{-6}$ point was used. This and other

[37] *http://www.plgrid.pl/*

[38] *https://www.tensorflow.org/hub*

[39] *http://www.image-net.org/* — "An image database organized according to the WordNet hierarchy"

setup was experimentally tested, and the presented above was giving the best, yet still not very exciting results.

### 1.10.1. Model execution strategy

As stated before, the model is running several times. To save time and optimize resources usage, the first approach for running the model was to execute it in a multi-threaded way. For this purpose, the module which was facilitating that was created. Soon this approach was abandoned due to the encountered problems. The main problem was that the SLURM manager is not able to split the resources in a way to the multithreaded approach to be sensible. The resources were assigned to the whole computing node on which only 2 GPU cards were able to be allocated. This caused the problem of many machine learning models trying to run on a single card.

When run on the Prometheus cluster, the better approach was to split the computation between two GPUs and run the model sequentially. For this purpose, the Mirrored Strategy distribution technique from the TensorFlow library was used whereas training is split across multiple replicas. Thanks to the split and the computing power of GPU, single execution is fast, thus can be multiplied several times.

# 2. Final results and summary discussion

## 2.1. Results

Presented model was trained and tested using various parameters. The changes were affected by the input data as well as the neural network parameters. The final dataset properties used to train the presented model are described in Tab. 2.1. The input picture size was forced by the used transfer learning model. This limitation had an effect on resizing original pictures of sequences to the required one by the used neural network.

**Tab. 2.1.** Dataset properties

| | |
|---|---|
| Users | 46 |
| Human users | 45 |
| Bot users | 1 |
| Sequences | 639 |
| Minimal sequence length | 50 |
| Model input picture size | 299 x 299 [px] |

**Tab. 2.2.** Confusion matrix values

| | |
|---|---|
| False negatives | 19 |
| False positives | 0 |
| True negatives | 327 |
| True positives | 0 |
| False rejection rate | 100% |
| False acceptance rate | 0% |

The total amount of sequences depends on the minimal sequence length, because sequences that were shorter than 50 were simply rejected. The sequence split point was established by the time interval between two consecutive actions and in presented work was fixed to 1 second. The minimal sequence length was fixed based on several attempts. Shorter sequences than the determined ones resulted in apparently lower accuracy, when longer ones did not improve performance at all. The length of 50 seemed to be the golden mean between accuracy and amount of sequences.

The charts presented in Fig. 2.1 and Fig. 2.2 show that the model accuracy reaches the level of 94% with 69.5% value of the loss function. It can be seen that the model accuracy during the testing part is in most epochs higher than the one in the training phase. It is caused by an imbalanced dataset which was the input for the presented model, where the amount of user data for testing was greater than the one generated by a bot. It is the direct effect of having more user data in the whole dataset. The impact is also seen on the loss function chart (Fig. 2.2), where values are lower in the testing part.

Moreover, due to the low amount of data, the model had some problems with prediction. At the beginning of the learning procedure, user and bot data were treated with equal importance. With the

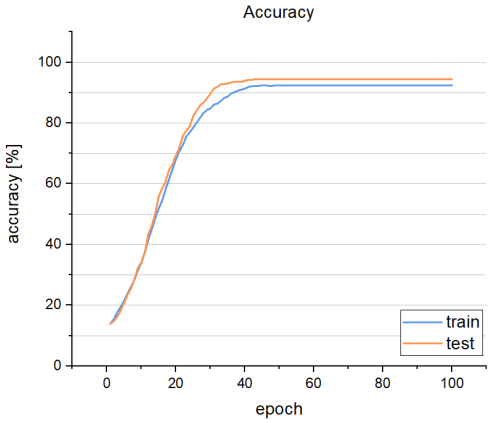


**Fig. 2.1.** Accuracy of presented model

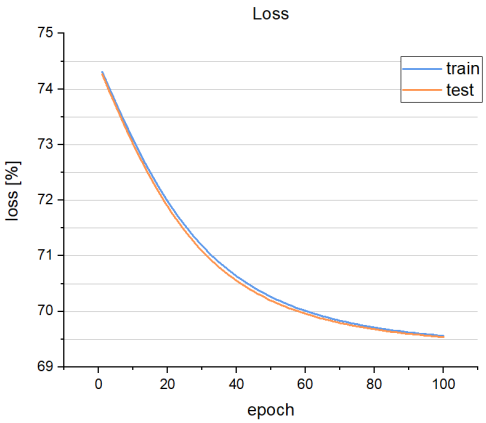


**Fig. 2.2.** Value of loss function used in model

successive epochs, user data outshone bot dataset. It can be seen (Tab. 2.2) that after the learning process, there were no true positives, which suggests that the model learned only one type of data. The model always predicted that the input data was the user's one. It is the result of a large amount of user data in comparison to bot sequences, whereas it is not the result of having only one bot user and many human users, because each user's sequence was joined into a global one.

Chong et al. [**Main**] present similar approach to related problem, but the results achieved in this study show that the presented solution behaves unilaterally. In many cases from cited work, authors achieved very low values of FRR[1] and FAR[2]. In this work, using CNN's[3] as a core of a solution, totally different results were achieved. The authors of this study consider the imbalanced dataset as the main foundation of that unexpected behavior. Despite the numerous efforts taken, the results are unsatisfactory. The obtained accuracy chart (Fig. 2.1) is somehow expected due to the dominance of one kind of data in the binary selection problem. However, the loss function (Fig. 2.2) as well as confusion matrix (Tab. 2.2) have undesirable values.

## 2.2. Limitations, conditions and problems

Although the model was considered using different sets of parameters and various approaches, it did not perform well. Due to certain limitations, some aspects of the model performance could not have been improved. Some problems that have a great impact on the overall results are described in the following sections.

[1]False Rejection Rate
[2]False Acceptance Rate
[3]Convolutional Neural Network

### 2.2.1. Dataset limitation

The main reason for poor performance has to be an insufficient dataset used in the study. The search for proper and publicly available data resulted in failure. In other related works, like Chong et al. [**Main**] or Antal et al. [**balabit1**] points at the Balabit Mouse Challenge Dataset[4] as the most popular and comprehensive, yet still quite small data source for mouse sequences and behaviors. However, this dataset is not appropriate for the problem considered in this study.

The Balabit dataset consists of user sessions recorded on the remote machine. The data includes the timing and position of the mouse cursor of ten different users. The split dataset represents training sessions and test sessions.

In the training sessions, one can find 65 legitimate sessions of various lengths, wherein total gives 2,253,871 rows of registered user mouse actions. The test sessions consist of 1,611 shorter sequences, that in total have 2,357,714 recorded actions, where during the session execution of the legal user, the illegitimate action happens — the session is taken over by another user. Illegal sessions are the mix of two legitimate users, and when the model considers the example it should assume one user, that is then interrupted and replaced by another user somewhere between the session.

This data does not relate to the problem given in the scope of this work — the model that can distinguish legitimate human user and non-human bot behavior is taken into consideration. This particular case is derived from the general problem of distinguishing two or more users and represents a more specific case of using mouse behavioral biometrics.

### 2.2.2. Custom dataset research participation limitation

Since no public dataset is available, the goal of this work was to collect exclusive and dedicated data for purpose of the study. The custom environment[5] created as a playground for research participants was designed to collect and record the mouse data, but it did not serve the responsiveness of a real commercial website, and therefore it may be causing some confusion among the subject users. By that means, data collecting was in some kind suggestive and task-oriented. Given factors could have a negative impact on the quality of the collected data.

The other thing is that participation in the study was completely voluntary and community-based. The advertisement for the ongoing study was posted on a couple of Facebook[6] groups, which was the most available and large user community base. However, such an approach resulted in non-supervised data gathering, and therefore some user actions could not be assessed as properly executed and caused disturbances and noises in the set.

The research gathered 63 unique users. Overall user mouse actions collected are equal to 334,184 rows. The number of bot actions registered and collected is equal to only 24,791 rows.
Such an uneven ratio of the data made the dataset imbalanced, which resulted in the model making

[4] *https://github.com/balabit/Mouse-Dynamics-Challenge*
[5] *https://github.com/Mouse-BB-Team/Data-Collection*
[6] https://facebook.com/

assumptions about every sequence biased towards the class of human users. Training set contained about 269,231 bot and user sequences, whereas the test set contained 89,744 sequences — even using transfer learning technique, the dataset was too small for efficient model training.

### 2.2.3. Finance limitation

Another problem encountered when preparing the thesis was the limit of the finance intended. Because the research was planned to reach many different users it had to be deployed and hosted on trusted and reliable resources.

Cloud services are really convenient way to handle such a project — the hosting, computation, and storage resources can be acquired on-demand, with no time and commitment. In the variety of different cloud solutions, in this case, the Google Cloud Platform[7] was selected and used. However, the cost of this kind of resources is high enough to be a limitation for the work.

### 2.2.4. Time limitation

Due to the time limitation, the duration of the period when the data was collected could not be extended to broader terms, thus the research and the voluntary participation in data collection were canceled during the further implementation of the project — the bot detection part[8], where the machine learning model was in build.

## 2.3. Conclusions and further study

### 2.3.1. Summary

During the work on the presented solution, the steps to limit the impact of the imbalanced dataset were taken. As an example, linear interpolation was used by connecting the points in recorded sequences. Each used sequence originally consisted of many single discrete points without any additional pieces of information. Interpolation provided an order between discrete coordinates and allowed feeding the neural network with additional information.

Another considered approach was a manipulation of the input data size. The user's sequences were limited to the number of total bot sequences. This solution was aimed to balance the dataset at the cost of fewer data. The results of this approach turned out insufficient. Because of the total amount of bot sequences, the total size of the dataset drastically shrank, which resulted in a performance deterioration. On the other hand, the duplication of the bot sequences was used. The idea was similar to the one before, but instead of reducing, the number of bot samples was increased by using a single sample several times. It resulted in an artificially balanced dataset. However, this approach did not increase performance at all.

[7] *https://cloud.google.com/*
[8] *https://github.com/Mouse-BB-Team/Bot-Detection*

Manipulation of the distribution of labels between training and testing dataset was also considered. It was done by performing either an equalization of the total number of both types in the testing dataset and the distribution of samples between both datasets. The first solution did not affect model performance, but the second one slightly improved overall performance if the ratio was close to 50:50. When the number of training samples was significantly greater than testing ones, the accuracy decreased due to a very small number of bot samples in testing dataset.

Yet another attempt to reduce the impact of the inappropriate dataset was changing the dataset itself. The developed serializing tool made it possible to create a few datasets from recorded data with different minimal sequence length limits. Using longer sequences meant that the overall number of them would be smaller. The authors tested several ones and found out that the best performance was for a length equal to 50, as it was mentioned before.

All of the presented approaches tended to minimize the dataset problem. Some of them slightly improved performance and those were considered in the final solution. Despite the efforts and attempts for improving the model, the described problem significantly worsened the performance of the model.

### 2.3.2. Further study

According to the described issues, the authors find further study mainly in the improvement of the dataset. Some efforts may be taken to extend the size of the recorded data. Firstly, recording sequences of a bigger group of users may be proposed as a solution, especially enlarge the bot users to prevent imbalance. Such a solution should improve the overall performance of the model or at least suggest other problems related to the quality of the dataset.
If the quality of the recorded samples would be inappropriate, the collection module[9] should be reviewed and improved. The main object of interest should be the method of gathering the samples. Delays and synchronization that can disturb the reliability of the data may be considered as a major area of study.

The different approach that may be taken into consideration is another machine learning model. In the presented solution, the focus was on the two-dimensional convolutional neural network, taking an example from related works like Chong et al. [**Main**] and Wei et al. [**Inspiration**]. The work [**Main**] also shows other solutions, especially recursive neural networks. Those kinds of neural networks are very popular in problems where the order and time intervals between samples have natural interpretations. The problem which is described in this work also has similar properties.

These two described areas of study are found by the authors as the major to improve performance and reliability. To deliver a safe and reliable solution to the commercial market further study is necessary.

[9] *https://github.com/Mouse-BB-Team/Data-Collection*

# List of data schemas