Computing and Informatics, Vol. 40, 2021, 930–956, doi: 10.31577/cai_2021_4_930

CURRENT TRENDS IN SOFTWARE ENGINEERING BACHELOR THESES

Jacek Dajda, Michał Idzik, Jakub Sroka, Mikołaj Sikora
Wiktor Pawłowski, Maciej Smołka, Przemysław Jabłecki
Filip Ślazyk, Maciej Malawski, Emilia Majerz
Aleksandra Pasternak, Witold Dzwinel

Institute of Computer Science
AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Kraków
e-mail: {dajda, miidzik, smolka, malawski, dzwinel}@agh.edu.pl, {jakubsr, mikolajsi, pawlowski, jablecki, slazyk, majerz, pasternak}@student.agh.edu.pl

Wojciech Kania, Bogumiła Hnatkowska, Wojciech Thomas

The Department of Applied Informatics
Wroclaw University of Science and Technology
27 Wybrzeże Wyspiańskiego st., 50-370 Wrocław
e-mail: wojciech.kania@zoho.com, {bogumila.hnatkowska, wojciech.thomas}@pwr.edu.pl

Joanna Świebocka-Więk

The Department of Applied Informatics, Jagiellonian University
ul. Gołębia 24, 31-007 Kraków
e-mail: joanna.swiebocka-wiek@uj.edu.pl

Andrzej Paszkiewicz

Rzeszów University of Technology
Al. Powstańców Warszawy 12, 35-959 Rzeszów
e-mail: andrzejp@prz.edu.pl

Abstract. This article presents short analysis and observations on current trends and directions in conducting engineering theses in the field of computer science. This report is based on collected bachelor theses in AGH Computer Science Department for academic year 2020/2021 as well as the conducted competition for the best engineering theses held during XXII KKIO 2021 Software Engineering Conference. The awarded works are briefly presented as an illustration to the drawn conclusions.

Keywords: Software engineering, bachelor thesis, tools and solutions, analysis

1 INTRODUCTION

Every year all around the globe thousands of students graduate from their universities and join the software development community. What is more, the majority of the graduates already work during their studies gaining professional experience and acquiring academic knowledge at the same time. This mixture of youth, ideas and some professional experience results in new inspirations and trends in the community which turn into new start-ups, popular frameworks or developer tools. Having said that, it seems valuable to spend some time and inspect the topics and standards of the bachelor theses in the area of Computer Science.

Bachelor theses are meant to prove students skills and readiness for the professional job market. Usually, they have a form of one-year software projects realized by single or more graduates (maximum 4) who form small teams. This report is based on bachelor theses in the Computer Science Department in the University of Science and Technology for the academic year 2020/2021 and the conducted competition for the best engineering theses held during XXII KKIO 2021 Software Engineering Conference. In total 66 final projects were analyzed in terms of teams sizes, types of topics and best diploma theses.

This paper is structured as follows. First, the general data and analysis results are presented. Afterwards, there come short summaries of the best diploma theses. Finally, the conclusion tries to summarize the current trends, explain them and forecast the near future in terms of student academic projects.

1.1 Team Size

Figure 1 shows the variations in team sizes in numbers. Most projects were realized by teams consisting of 2 or 3 graduates. This can be interpreted as a good balance between the formal graduation procedures and the concept of simulating professional team environment.

Obviously, the team size is influenced by many factors, but it seems that small teams are supposed to address more ambitious subjects and deliver more mature results.

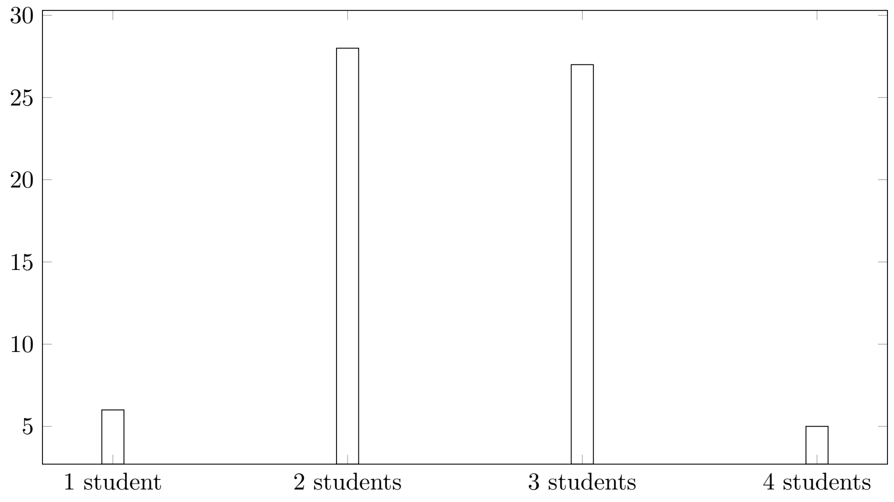


Figure 1. Size of bachelor project teams

1.2 Thesis Characteristics

There are different types and themes of bachelor projects in computer science. Some projects aim at creating functional and usable software products which are not innovative in terms of technology or architecture. Other types of projects focus on solving difficult problems and require a considerable amount of research and an innovative approach. Another interesting aspect is whether a project requires design and implementation of some sophisticated or non-standard algorithms. The utilization of neural networks and machine learning algorithms is another important factor, especially in the context of bachelor projects.

Every project is different and has its unique goals and characteristics. As a result, projects can be often assigned to multiple factors which are described above. For example, a project can result in a usable software product and utilize neural networks in its core functionality. There are also projects that require research and prototyping to solve the posed problem and deliver the final product.

Figure 2 shows characteristics of 66 final bachelor projects having considered the following aspects:

Research. Project requires a considerable amount of research before specification and implementation could be started.

Product. The main goal and final result of the project is well defined, usable and delivered software product.

Algorithms. The project requires design, realization and validation of dedicated sophisticated algorithms.

Machine Learning. The project requires utilization (which does not mean their design) of machine learning algorithms which can be achieved by applying proper software libraries delivered by other parties.

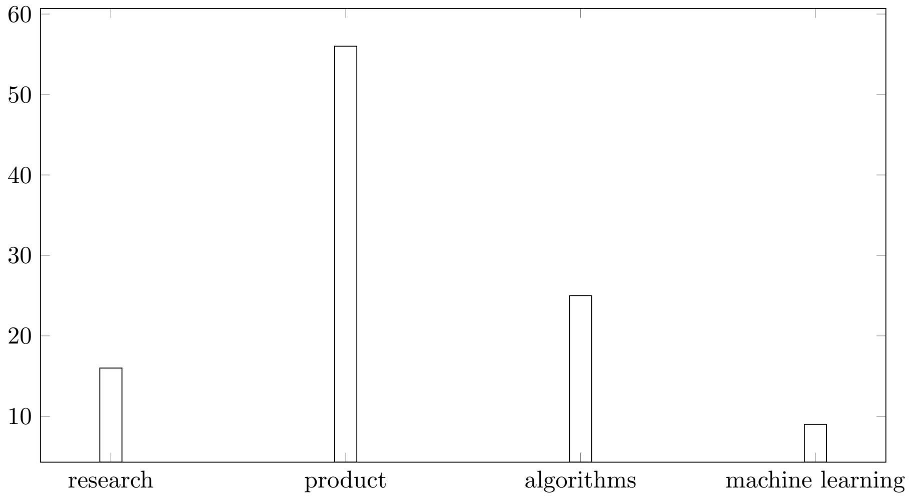


Figure 2. Bachelor projects characteristics

The first observation is that most of the bachelor theses resulted in delivering software products. However, it can be noticed that around 25 % of projects required a considerable amount of research. In some cases, the development of dedicated algorithms was the main goal of the thesis, in opposition to the standard product-based approach. It can be observed that almost 40 % of projects required solving problems by designing proper algorithms.

The following conclusions can be made:

- Majority of the projects have strictly engineering theme focusing on building well designed technological software products.
- Considerable number of projects focus on solving given problems leaving technological aspects as secondary.
- One-quarter of projects manifest scientific nature usually including aspects of machine learning and neural networks methods applied to solving specific problems.

1.3 Best Works

The organized competition for the best engineering theses held during XXII KKIO 2021 Software Engineering Conference awarded the following 4 diploma theses:

- Wojciech Kania: "Intuitive sound processing application", Supervisor: Bogumiła Hnatkowska;
- Mikołaj Sikora, Wiktor Pawłowski and Jakub Sroka: "Rule-based system for solving functional harmony exercises", Supervisor: Maciej Smołka;
- Filip Ślazyk and Przemysław Jabłecki: "A tool for comparison and integration of feature selection algorithms for modeling of response to targeted therapy for patients with hairy cell leukemia", Supervisor: Maciej Malawski;
- Emilia Majerz and Aleksandra Pasternak: "System for analyzing damage to the surface of aircraft structures using convolutional neural networks", Supervisor: Witold Dzwinel.

Submitted diploma theses were evaluated against the following criteria:

- **Topic originality and method of its implementation.** This criterion evaluated work innovative approach in the context of existing solutions, possibilities of practical application, the undertaken approach to the identified problem and its implementation.
- **Throughout quality.** This criterion evaluated the quality of the work perceived in multiple dimensions such as maturity of final results, comprehensibility and clarity of the thesis, completeness in terms of information contained, editorial level.
- **Closeness with Software Engineering research discipline.** This criterion evaluated references to Software Engineering discipline such as software quality assessment tools, tools supporting automatic testing, communication or source code implementation and others of a similar kind.

All awarded works delivered usable tools which solve quite specific needs: from sound processing on mobile devices to identification of corrosion on the surface of aircraft elements. None of the works referred to a trivial problem nor application Majority of works required a considerable amount of research and specific domain knowledge to be obtained before designing and implementation of the final solution.

In the following sections, the four highlighted works are shortly presented. Each work is described in the same scheme and includes the following key aspects as a problem description, solution concept and scope, method and implementation, project results, conclusions and future work.

2 INTUITIVE SOUND PROCESSING APPLICATION

Raising popularity of mobile devices has created the need for software capable of processing creative content. The showcased engineering thesis, titled "Intuitive sound processing application", is focused on the design and implementation of a user-friendly tool for editing audio on smartphones.

2.1 Problem Description

Sound processing applications for a very long time were firmly rooted in the desktop market, as only computers could operate on quantities of data required in these processes. That is no longer the case – implementing easy to use, reliable and efficient solution for audio edition in a mobile environment is an interesting challenge, considering different hardware possibilities of smartphones, and the specifics of their operating systems. This thesis focused on the Android operating system. The problems faced during the design and implementation phases can be generally divided into *performance* and *usability* related.

While talking about performance, it is difficult not to mention several design decisions that were taken by the operating system creator. Android differentiates the main thread, which operates the GUI and is essential for the application performance, and worker threads, as described in the platform's documentation [1]. Complicated tasks reserving access to slow resources or performing lengthy operations should be given to the worker threads. The second highlight is the operating memory restriction policy, which prevents applications being run on the device from hogging too much resources. This restriction was the most most troubling in this context. To put it into perspective, raw numerical data read from a 10 minutes long audio file with standard quality (Audio CD standard, IEC 60 908 [3]) can take up about 100 MB of operating memory. The device used for development purposes had 6 GB of RAM installed, which corresponded to imposed limit of roughly 512 MB – after crossing this threshold, the application was automatically closed. Problems such as these were uniformly well addressed by the existing solutions – multiple optimizations, such as partial loading and samples grouping, were introduced by their authors to elevate the UI responsiveness – similar conclusions can be drawn when the speed of applying built-in audio effects and transformations is being considered.

Second category of problems originates among others from naively transforming the solutions known from the desktop environment into mobile devices, with little thought put into assessing their differences and adjusting the design accordingly. In some of the existing solutions, one can often observe complex menu hierarchies, important views and controls rendered too small due to screen size differences, as well as unnecessary complex sequences of user actions leading to frequently used functionalities. Such design flaws can be intimidating to the user, who after the first bad impression can uninstall the software. Finally, lack of physical keyboard can be seen as an important disadvantage in the context of media edition, as it removes the capabilities to create key bindings for core operations. There is also another aspect of usability – the included range of functionalities, closely related with the audio transformations that made it to the final release.

2.2 Solution Concept and Scope

The main goal was to achieve the possibility of manipulating a single- or dual-channel audio track, with support for cutting, moving and inserting audio fragments, as well as sound edition via included audio effects implementations. The user should be able to load and save external data in a few popular audio formats, both lossy and lossless, as well as record, save and edit signal without exiting the application. User actions should be performed by simple interaction with intuitive user interface controls, or usage of supported touch gestures. By the end of development, the application was ready to be used as a personal voice recordings creator and a simple universal audio content editor.

The general concepts applied to minimize the impact of performance restraints were simple in nature. Used mechanisms were already provided by the programming language used. One of the dangers to look out for, with the background of memory efficiency, is extensive garbage collection. Multiple garbage collector invocations can slow down the application execution, as the operating system uses the main thread to operate it. The key in resolving this problem is *avoiding dynamic memory reallocation*, which includes pre-initialization of the collections used for numeric data storage to an extent supported by the user's device. Additionally, *avoiding built-in primitive types boxing* as well as *using arrays as means of data transport* were crucial to the software creation and usability. As stated by Marcin Moskala in his book, "Effective Kotlin" [2], the boxed numeric types used in collection can increase the usage of memory up to 5 times, and slow down an exemplary operation of mean calculation by up to 25 %. Faster copying of data from or to arrays is supported natively in most cases, arrays of primitive types also consume much less space than dynamically expandable collections, such as lists or sets.

As stated previously, one can often see user interfaces unaccustomed to the mobile environment. The author tried to avoid such mistakes, simplifying the control flow, views and controls in the key areas. The most important control – signal waveform – was highlighted and enlarged, occupying most of the screen space, additionally quick access was provided to the key functionalities, such as the undo or audio fragment insertion buttons. As for the means of control, touch gestures support with appropriate introduction should be considered crucial in the author's opinion, for in many contexts they can perform the same roles as key bindings known from the desktop software. In created software, the touch gestures are able not only to scale the UI fragments, as also supported by other existing solutions, but also to move and delete audio fragments across the rendered audio track and even edit parameters of basic effects, such as amplitude control. In terms of implemented transformations, the final release included *Echo*, *Chorus*, *Distortion*, *Reverb*, periodic *Panning* and basic gain control.

2.3 Method and Implementation

Architecture of presented solution was divided into three layers, the *Data Access Layer* responsible for the IO operations, the *Services Layer* which contained the mechanisms responsible for main features of the application, and the *Presentation Layer*, itself structured according to MVP pattern, which implemented the model and generated the view of data as seen by the user.

The application was designed with modularity in mind – the goal was to enable easy addition of new audio transformations. A programmer could easily add *Audio Effects*, as well as *Audio Generators* to the application sources, as they exposed a common public interface. The previous was used to edit the already existing audio, the latter to insert new audio fragments generated in-app on demand. All the supported operations influencing the source data were described by the *Audio Transaction* interface. They were managed by the *Transaction Manager* mechanism, responsible for maintaining the log of operations, reversing them on demand, as well as keeping a reference to the *Cache Manager*, which could create temporary files with snapshots of audio fragments as needed. Finally, default interfaces of communication between the mentioned services were implemented – they operated using buffering, partially loading the requested data into arrays of fixed sizes. These interfaces included *Audio Sources* as well as *Audio Destinations*, being respectively on the supplying and consuming ends of the communication, as well as a helper interface *Audio Middleman*, implementations of which used for relaying the status of audio operations to the outer world.

Most of the effects were implemented using a combination of comb filters – basic audio filters capable of creating different kinds of delays. Their principle of operation involves summing the input signal with itself, fed forward or backward, delayed and amplified [4]. The implementation used *circular buffers* of fixed length for storing the state of the effect, allowing the transformation to still be buffer-based and memory efficient. Some of the transformations included were simply parameterized functions, which were ready to be applied to the signal out-of-the-box, without the need for additional effect state variables.

2.4 Project Results

As parts of the thesis, project documentation and a mobile application called *Sound-Mash* were created, allowing for import and export of audio in six formats, recording voice memos, signal manipulation, including cutting and moving fragments in the audio track and also application of chosen audio filters, mentioned in Section 2.2. For conveying information about the signal, an interactive, generated oscillogram was used, the solution also included the functionality of playing selected audio fragments.

Figure 3 presents the main view of the application, with dual-channel audio track waveform and a fragment of it selected. Upon selection, the menu button can be seen in the upper right corner, containing the implemented effects. Figure 4 presents the parameters screen overlay of the Echo effect, with a short description and supported range provided. The parameters can be also manipulated through touch – the values being changed are then showcased in the upper part of the screen without the parameters overlay shown, being directly steered by user input. For example, sliding the finger downwards and upwards can decrease and increase the gain value appropriately in the gain control transformation. More precise values can be entered by unfolding the overlay parameters screen with a dedicated button.

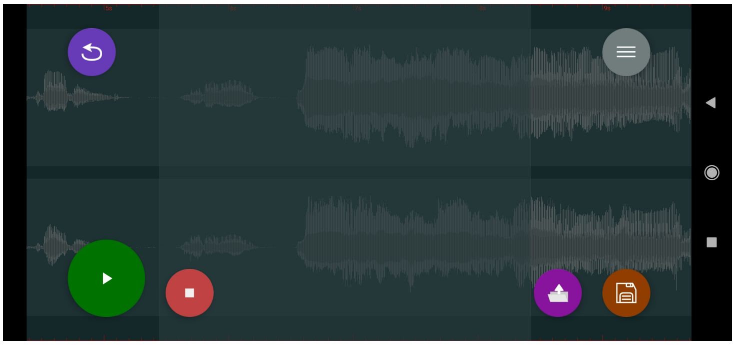


Figure 3. Main screen of the designed application

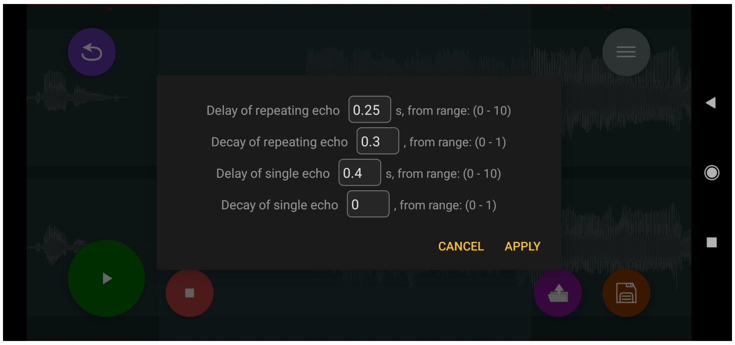

Delay of repeating echo 0.25 s, from range: (0 - 10)
Decay of repeating echo 0.3 , from range: (0 - 1)
Delay of single echo 0.4 s, from range: (0 - 10)
Decay of single echo 0 , from range: (0 - 1)
CANCEL APPLY

Figure 4. Exemplary parameters screen

2.5 Conclusions and Future Work

In retrospect, much improvement in the area of application design is still needed. Insights from external testers early on in the project could be very beneficial for its development, in terms of the UI, as well as in the requirements scope. In this regard it is often preferred to use existing solutions, gradually implementing improvements. The architectural cohesion should be planned beforehand, but there is a very high possibility that the new knowledge acquired will prove earlier designs flawed. Finding the right way of communication between the application modules is the key to well designed solutions. A flawed decision is often better than a lack of decision, once the expected deadlines of the project approaches try to deliver solution as full as possible, even if some compromises have to be made.

Future work includes among others improving the efficiency of the solution in terms of the audio formats supported, length of signal available for processing and rendering the oscillogram. For this, redesigning the way of storing data for operating memory usage optimization will be crucial. Including another batch of audio effects, together with options of changing the audio characteristics, such as bit depth and sampling frequency, would expand the functionality. In the end, wide-scale functional and non-functional testing procedures would need to be executed, playing a key role in enhancing the solution's reliability.

3 RULE-BASED SYSTEM FOR SOLVING FUNCTIONAL HARMONY EXERCISES

In this section we describe a system called *HarmonySolver* devoted to solving problems from the area of functional harmony, which is one of main fields of music theory. The presented system has the form of a plugin for the *Musescore* popular open-source musical score processor [9]. The system encodes musical rules developed during the Baroque period and prevalent also in later music, that govern assembling notes into melodies and chord sequences. Using these rules we can treat all basic types of problems arising in functional harmony: four-part harmonization based on a given sequence of harmonic functions, figured bass realization and soprano harmonization. Namely, we turn the problems into constrained discrete optimization tasks and solve them by means of classical graph algorithms.

3.1 Problem Description

Harmony is a branch of music theory that rules the assembling sounds into chords. Functional harmony is a type of harmony that was developed in the seventeenth century and remained a key element of the mainstream music until the beginning of the twentieth century. It is still in use in many genres of modern music and it is the foundation of other types of harmony. Due to its importance and great influence on modern harmony it is taught in detail in music schools. It is based on creating four-part scores with four separate voices: the highest soprano, alto, tenor and the lowest bass. Each voice has its own melody and obeys specific constraints, for example own scale. Moreover, each note in one voice forms a specific combination with notes in the other three voices: a chord. A main feature of functional harmony is to base on a specific *major* or *minor* musical scale related to the key of composition. Each note in the scale can be used as a basis for a chord consisting of three or more different sounds. Importantly, each chord has its own *function* that determines its role and usage in a musical piece. The most important harmonic functions are: *tonic* (denoted T) based on the first note of the scale, *subdominant* (S) based on the fourth note and *dominant* (D) based on the fifth note. Functional harmony provides many rules in two contexts: vertical and horizontal. The vertical context corresponds to constructing chords: its rules refer to relations among different voices in the same chord. The horizontal context rules are thought to avoid inappropriate connections between neighboring chords. There are three fundamental types of problems arising in functional harmony:

- *four-part harmonization based on given harmonic functions* – as an input we have a sequence of harmonic function symbols grouped in measures with given key and time signature;
- *figured bass realization* – as an input we have a complete bass-voice melody decorated with special symbols (mainly digits) which determine chords (and their functions) that can be build on top of the bass notes;
- *soprano harmonization* – as an input we have only a soprano-voice melody and appropriate chords have to be selected to match the given melody.

These problems are main types of exercises used in music schools because they touch some key aspects of the composition of musical piece. In each problem the solution is a sequence of chords grouped into measures (see Figure 5).

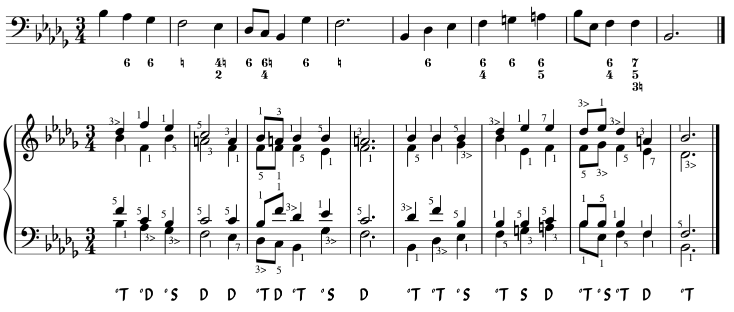


Figure 5. Figured bass realization (second type) problem formulation with a solution generated by HarmonySolver

For a thorough course in functional harmony we refer the reader to textbooks like [5].

3.2 Solution Concept and Scope

The above-mentioned fundamental problems are also the three main use cases for *HarmonySolver*. To solve problems of the first and second type we transform them to optimization tasks with constraints. The objective is a numeric measure of solution quality, whereas the constraints are obtained from the functional harmony rules divided into *hard rules* and *soft rules*. A candidate solution cannot break any hard rule, hence such a rule becomes a domain constraint. For each broken soft rule the solution quality measure is appropriately made worse, which disfavors a solution but does not necessarily reject it. In the first type of problems the constraints have a form of a sequence of harmonic function symbols. A figured bass realization task, i.e. a problem of the second type, is transformed to the first type by mapping pairs $\{bass\ note, bass\ symbol\}$ to harmonic functions with an additional constraint imposed by the fixed bass-voice melody. In the soprano harmonization problem we solve two sub-problems sequentially. First we find a harmonic function sequence that matches a given soprano melody. The result is a problem of the first type with an additional constraint defined by the fixed soprano-voice melody. Hence it can also be transformed into an optimization problem in a way analogous to the previous types.

The crucial part of the system is the domain model (see Figure 6). It allows us to properly encode functional harmony rules.

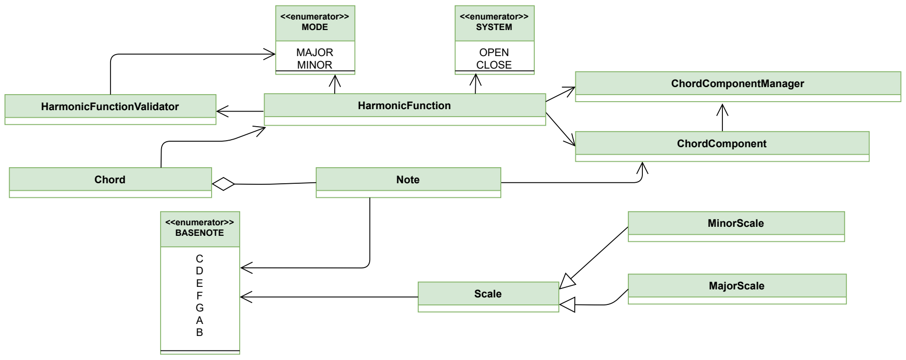


Figure 6. Domain model

The heart of the model is the *HarmonicFunction* class containing the following properties: *functionName* (T, S or D), *degree* (base note of a chord), *position* (chord component in soprano voice), *inversion* (chord component in bass voice), *delay* (list of pairs of chord components, which should be delayed from first to second one in pair), *extra* (additional chord components), *omit* (omitted chord components), *down* (flag indicating if the base note of chord is lowered), *system* (open, close or undefined), *mode* (major or minor), *key* (indicates if the chord comes from key other than piece base key, can be undefined) and *isRelatedBackwards* (used for backward deflections).

For evaluating both vertical and horizontal contexts we introduce two very important components in our system: *generator* and *evaluator*. *Generator* finds all objects matching a given single input specification and satisfying all rules. The type of generated objects depends on the specification, i.e., for a given harmonic function chords are generated and for a given soprano note the result is a set of admissible harmonic functions. *Evaluator* evaluates connections between neighboring chords and computes penalties.

3.3 Method and Implementation

HarmonySolver has been implemented as a plugin for the *Musescore* popular score editor, which exposes QML/JavaScript developer API. Plugins are thought to be short, consisting of one QML file containing both business logic and a GUI, but this assumption can be relaxed through importing external JavaScript files. As realizing the system as a Musescore plugin was a requirement, we decided to implement it in JavaScript and use a single QML file is the interface to Musescore.

To solve optimization problems described in the previous subsection we construct a directed graph representation of all admissible solutions to a given task. A simpler representation called *SingleLevelGraph* (see Figure 7) has been designed for tasks with a given sequence of harmonic functions. Its vertices are aggregated in layers, one for each harmonic function in a task and each vertex stands for only one of possible chord forms for that harmonic function. Edges represent legal connections between chords of neighboring layers and their weights are penalties obtained from soft rules evaluation. There are also two extra vertices: *begin* and *end* that are connected to each node from first and last layer, respectively. Generally *SingleLevelGraph* is a DAG without unreachable and useless nodes, with only one source (*begin*) and only one sink (*end*). Moreover, all paths from *begin* to *end* have the same length and represent complete admissible solutions. Finally, solving first-type problems consists in finding the shortest path in already built graph (cf. [6]).

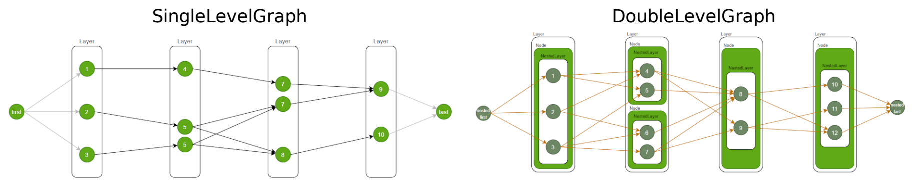


Figure 7. Graph structures used in HarmonySolver

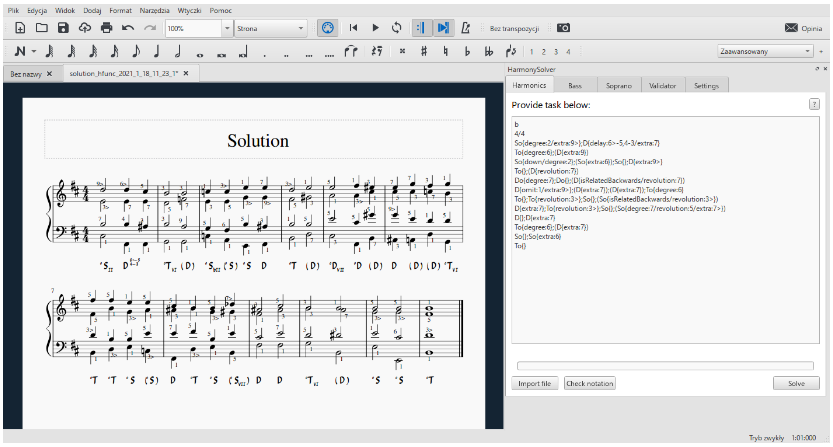
Some of the soft rules require considering triples of subsequent chords. To make it possible the graphs must be transformed in such a way that all predecessors of a single node have the same content, i.e., a harmonic function. We meet this requirement by performing so-called *Single Previous Content (SPC) transform*, that is our original idea and is based on the duplication of appropriate vertices and edges.

The process of building a *SingleLevelGraph* is crucial in our solution method. First, layers are filled with vertices, then edges between neighboring layer nodes are added based on the result of evaluation of hard rules. Afterwards, unreachable and useless nodes are removed. Then the SPC transform is performed and finally edge weights are set according to the evaluation of soft rules.

The *SingleLevelGraph* structure is not sufficient for the hardest type of considered problems, i.e. the soprano harmonization. In this case we develop the structure into *DoubleLevelGraph*. It can be described as two nested *SingleLevelGraphs*, where the first level represents harmonic functions and the second level contains admissible forms of respective chords. Using such a structure makes the problem of finding optimal harmonization more tractable.

3.4 Project Results

Final product that we created is a fully functional plugin for Musescore program [8] that can solve all three basic types of functional harmony exercises.



Provide task below:

```
b
4/4
So{degree:2/extra:9>};D{delay:6>-5,4-3/extra:7}
To{degree:6};(D{extra:9})
So{down/degree:2};(So{extra:6});So{};D{extra:9>}
To{};(D{revolution:7})
Do{degree:7};Do{};(D{isRelatedBackwards/revolution:7})
D{omit:1/extra:9>};(D{extra:7});(D{extra:7});To{degree:6}
To{};To{revolution:3>};So{};(So{isRelatedBackwards/revolution:3>})
D{extra:7};To{revolution:3>};So{};(So{degree:7/revolution:5/extra:7>})
D{};D{extra:7}
To{degree:6};(D{extra:7})
So{};So{extra:6}
To{}
```

Figure 8. Plugin view with a solution of problem of the first type

Figure 8 shows a Musescore window with HarmonySolver views presenting a formulation (right) and solution (left) of a problem of the first type.

One thing to note is that the plugin execution time is not very short: solving complicated exercises can take about 20 seconds. Our tests showed that the reason is not an inefficient implementation but performance issues with Musescore JavaScript runtime. Executing the same code in other environments lasted 100 times shorter. Nevertheless, since the manual way of solving such exercises takes about several dozen minutes, our result is satisfactory even with the slow Musescore runtime.

To test the correctness of generated solutions, we created many tests on the basis of a Polish classical harmony coursebook [7]. For all exercises our system provided correct answers. We also performed a UX test on group of musicians, organists and music theorists. They had a chance to evaluate our system, check if it generates correct solutions and give scores to those solutions. The average score for solution quality was 7.8/10 which is a great achievement. The usability of the product was considered very good as well.

3.5 Conclusions and Future Work

Looking at the results of final projects and the feedback received during tests it is fair to say that our system achieved a success. Our pure algorithmic and mathematical way of solving musical exercises led to solutions with high musical quality, which has been verified through tests.

One way to improve the existing project is to boost the performance by changing the solution architecture and avoid the use of the MuseScore runtime. A possible solution is to create an external HTTP service and expose an API endpoints that will handle requests for solving given exercise. Also, the integration with other systems would be straightforward in this way. We have already started work on that field and have a solution written with Scala and OpenAPI that already works and reduces the solution time down to the order of one second.

Another idea worth exploring is to use machine learning (e.g., neural networks) in solving the soprano harmonization exercises. There are already some solutions that use artificial intelligence and machine learning in this context. In our opinion combining the pure algorithmic and strict way based on harmonic rules and a generative model like neural network could lead to some very interesting results.

4 A TOOL FOR COMPARISON AND INTEGRATION OF FEATURE SELECTION ALGORITHMS FOR MODELING OF RESPONSE TO TARGETED THERAPY FOR PATIENTS WITH HAIRY CELL LEUKEMIA

This paper presents the development process and obtained results related to the tool for analysis of cytometry data of hairy cell leukemia (HCL) patients.

4.1 Problem Description

The development of diagnostic techniques has made it possible to more accurately measure parameters of the condition of the human body. It has led to the emergence of deep phenotyping, an approach in medicine which is characterized by precise detection of the phenotypic aberrations in individuals, supporting more precise detection of diseases in patients and providing targeted healthcare solutions [14]. Moreover, it is also increasingly easier to determine the accurate composition of blood, utilizing methods such as flow cytometry. This particular technique allows medical researchers to ascertain the exact count of different cell populations in a blood sample. It is particularly useful in the detection of HCL and modelling of response to targeted therapy.

Analysis of the flow cytometry data is a complex task, given the fact that hundreds of cell types are detected during this process, thus it requires statistical and computational approaches to prove viable in medical application, considering the complexity of the immune system and interdependencies among even small cell populations. Feature selection is a particularly non-trivial operation in this context. Hence, there exists a need to develop algorithms and computational tools for supporting that task, which is a crucial step in the generation of classification models for data obtained from advanced medical diagnostic techniques.

4.2 Solution Concept and Scope

The main role of the tool is the integration and comparison of feature selection algorithms. The implemented software is intended to effectively extract relevant predictors as well as support medical data analysis, especially coming from targeted therapies used to treat patients suffering from HCL. That task is challenging as only about 2 % of leukemias are HCL, thus there is a limited amount of this type of clinical data [11]. Also, the trained models can be saved and used when required. The product is a web application with an easy-to-use graphical user interface. It is an important aspect of the project since the end-users are not expected to be familiar with the notion of programming.

4.3 Method and Implementation

The following section describes the applied method of solving the presented problem and applied technologies.

4.3.1 Architecture of the System

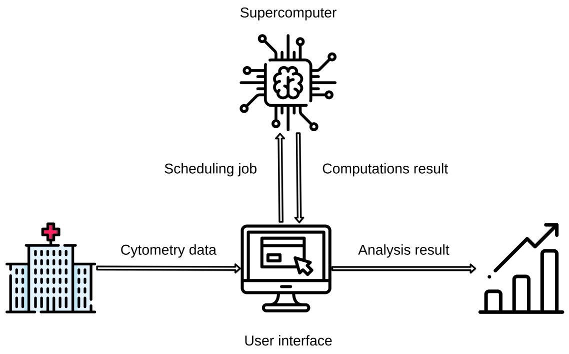


Figure 9. Conceptual diagram of the project

The system is based on the client application deployed with the use of Docker. A user places data and specifies the details of the requested computations via the user interface. The data and parameters are transferred via REST API to the configured supercomputer providing resources for training the model. When computations are successfully completed, the results of an analysis are available in an application tab in the form of diagrams, graphs and statistics. The conceptual diagram of the system is presented in Figure 9.

4.3.2 Applied Technologies

The backend is based on Django framework. Feature selection algorithms are implemented using Python libraries such as *numpy* v. 1.18.1, *pandas* v. 1.0.3, *matplotlib* v. 3.1.3., *scikit-learn* v. 0.22.1 and R *rmcfs* package v. 1.3.4.

The frontend is implemented using TypeScript and Angular framework. The application is containerized with Docker.

External HPC and storage resources are accessed via Rimrock REST API [1] and PLG-Data service [2].

4.3.3 Feature Selection and Classification Algorithms

Utilised feature selection algorithms are: Random Forest, Kendall or Pearson correlation coefficient based selector, Lasso Regression, Elastic Net Regularization, Chi-Square Test, Information-theoretical approach [10], Monte Carlo Feature Selection.

Utilised classification algorithms are: k-nearest neighbour, Random Forest, Support Vector Machine, Multi-layer perceptron classifier.

4.3.4 Methodology

The whole algorithm is similar to stratified k-fold cross-validation. For each partition i of k partitions of the dataset the feature selection is performed, taking partition i as test subset and the remaining $k-1$ partitions as training subset. The dimension of the training subset is reduced to the most relevant features and such reduced subset is used to train the ensemble of classifiers specified by the user. The averaged classification metrics obtained on test subset are used as weights for the features rank for a given partition. The process is illustrated in Figure 10.

After performing feature selection on each partition, all feature ranks with their corresponding weights are aggregated to build the final feature rank, which is a basis for selection of the most relevant features in the context of the whole dataset. The most important features are used to train the final models.

4.4 Project Results

The process of development was completed successfully and all requirements were met. In addition, the obtained results are biologically interpretable. The project allowed us to work on both engineering as well as research aspects. The source code is available at *github.com/pszemkor/FSTool.*

4.4.1 Evaluation of Feature Selection and Classification Models

The clinical data used to train the models was obtained from The Ohio State University College of Medicine, one of the most important research centres focused

[1] `https://submit.plgrid.pl/`

[2] `https://data.plgrid.pl/?locale=en`

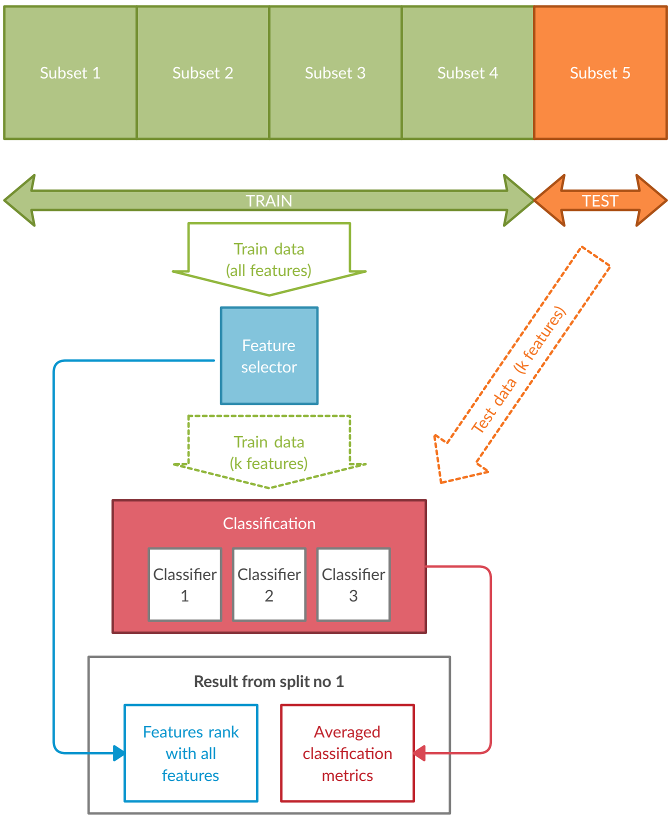


Figure 10. Diagram of feature selection and classification performed for one of the splits for given feature selector

on HCL. It consists of 145 features and 54 samples, where 17 of them belong to "control" and 37 to "case" classes.

The list of features selected by information-theoretical algorithm is presented in Table 1. The average accuracy is presented in Table 2.

4.4.2 Interpretation of the Results in the Biological Context

Numerous results achieved by these algorithms have an immediate biological interpretation. For instance, the CD19 positive B-cells are the main contributors to the tumour burden in HCL [12]. At the same time, the subsets of T-cells, both CD4 or CD8 positive, which were identified by selected methods are associated with the observed and previously described phenomenon of reduced immunity of HCL patients to bacterial infections, which in some cases leads to death. In particular, the CD27

Name	Final Score
CD19+/CD80-	499.36
CD19+	496.96
CD19+/CD80-/CD86-	489.93
CD27-/CD45RA+	487.70
CD19+/CD86-	484.44
CD8+/CD27-	479.55
CD4+/CD27-	475.24
CD8+/CD29+	473.75
CD27-/CD45RO+	472.13
CD8+/45RA+/CD27-	454.68

Table 1. Selected features

Classifier Name	Accuracy	Min Accuracy	Max Accuracy
KNN	0.76	0.5	0.91
MLP	0.55	0.3	0.73
Random Forest	0.81	0.7	0.91
SVM	0.82	0.73	0.91

Table 2. Average accuracy

marker is responsible for the immune signalling between B and T lymphocytes, and the expression of this marker is characteristic of HCL [13].

4.4.3 Future Work

Due to the increasing popularity of cloud computing, some future clients may want to take advantage of such resources instead of HPC clusters supporting SLURM. From the algorithmic side, one of the ideas for further development is to incorporate genetic algorithms as well as backward elimination.

Acknowledgements

The authors of this project would like to express their gratitude to Dr. Michał Seweryn, who provided meaningful support throughout the development of the project. This publication is partly supported by the EU H2020 grant Sano No. 857 533, and by the project Sano carried out within the International Research Agendas Programme of the Foundation for Polish Science, co-financed by the European Regional Development Fund. This research was supported in part by PLGrid Infrastructure.

5 SYSTEM FOR ANALYZING DAMAGE TO THE SURFACE OF AIRCRAFT STRUCTURES USING CONVOLUTIONAL NEURAL NETWORKS

The study aimed to create a system for classifying images of the surface of aircraft structures according to the occurrence of corrosion. Its goal is to improve the efficiency and efficacy of visual, non-destructive aircraft inspection (NDI) procedures. The system was developed owing to the database containing images of fragments of aircraft fuselages collected by the Air Force Institute of Technology (AFIT) in Warsaw. The heart of the system is a trained Convolutional Neural Network (CNN) model, which can be used via a web interface. The final product utilises a neural network based on the EfficientNetB0 architecture, which achieved 75 % accuracy on the test set, correctly classifying 90 % of images showing corroded surfaces.

5.1 Problem Description

During exploitation, the aircraft is prone to corrosion, which can lead to serious damage. The situation that made the public aware of the seriousness of that issue was the Aloha Airlines Flight 243 incident, caused by failure of the maintenance program, with corrosion being one of the incident's contributing factors. Therefore, both traditional and innovative inspection techniques are used to assure passengers' safety, which is paramount in aviation.

One of the main techniques to control the appearance of possible damages is visual inspection conducted by professionals. To enhance the visibility of damages, new methods were developed, such as D-sight [15]. It is used for processing fuselage images in the scope of DAIS (D-Sight Aircraft Inspection System) technology. However, the images in DAIS are still analysed and processed visually. To avoid human errors and make the procedure automatic, we propose to apply Deep Learning (DL) algorithms.

Current state-of-the-art architectures of Artificial Neural Networks (ANNs) for image classification are CNNs, containing convolutional layers capable of effective extraction of image features. Concerning previous CNN applications in NDI, a worth-mentioning approach utilises an SVM classifier, trained on features extracted by a CNN from images of airplanes' fuselages [16]. Some other prior works were focused on the application of single and ensemble CNN models. For example, in [17], the ensembles of CNNs were used to improve the classification performance on borescope images of aircraft surfaces and compared to single models.

In this paper, we demonstrate the efficacy of CNN architectures in corrosion detection. As the input, we use images of fragments of aircraft fuselages, initially preprocessed by the DAIS.

5.2 Solution Concept and Scope

The proposed system consists of an image classifier and a user interface in the form of a website. Pictures provided by the DAIS are analysed by a pretrained ANN. The main requirement for the system is an intuitive and easy-to-use classifier since the intended users are aircraft specialists and people interested in aviation. The classifier is to determine corrosion occurrence in a 2-point scale (*corrosion* vs *no corrosion*). The result contains the predicted class and a softmax value for each class. Information about the areas where damages were detected is also presented to the user. The classifier can be easily retrained or replaced.

5.3 Method and Implementation

The web application is implemented in a client-server architecture model and works as HTTP API. The user through a web browser sends POST request with all required information like the image for the classifier to predict, then the server processes the request and returns the result in the form of a text (prediction) or an image (heatmap). The server was built with Flask microframework due to its simplicity and flexibility, which allowed changes of technology decisions throughout the project duration. The client side was written in HTML, CSS, and JavaScript with AJAX requests for dynamic page experience.

The dataset used in this project consisted of 13 075 images of 37 aircraft structures divided into 5 classes: *no corrosion* – 6 431 samples, *soft corrosion* – 6 040 samples, *medium corrosion* – 578 samples, *hard corrosion* – 0 samples, *soft damage* – 26 samples. Because of the data being unbalanced, the decision was made that the appropriate task should be a binary classification: *no corrosion* (for not corroded surfaces) and *corrosion* (the rest). To ensure the closest simulation of the NN's behavior on new data, images in training, validation and test sets came from different aircrafts (training set – machines 1–30 – 10 343 samples, validation set – machines 31–34 – 1 448 samples, test set – machines 35–37 – 1 284 samples).

The core of the project was searching for a proper NN architecture able to achieve both high *accuracy* and *recall* (because of the *False Negatives* (FN) occurrence being a threat to the safety of passengers of examined aircrafts) on the DAIS dataset. During the experiments phase, the SOTA architectures (ResNet, Inception, EfficientNet) and various machine learning methods and preprocessing techniques (data standardization and data normalization, data augmentation – horizontal and vertical flips, ensemble learning, classification of the features returned by convolutional layers of neural networks by SVM, threshold manipulation) were tested. The results were also analysed using t-SNE method, by visualising the dataset after last convolutional layer to check classes separation in a 2-dimensional space. The experiments were conducted on PLGrid *Prometheus* supercomputer using Python language and TensorFlow framework.

One of the functional requirements was to present the pixels which are the most responsible for the result of prediction (probably representing corroded areas) by using a heatmap. This goal was achieved with the use of the Grad-CAM (Gradient-weighted Class Activation Mapping) [18] tool.

5.4 Project Results

The developed application delivered all functionalities specified at the beginning of the project. The user can upload images in jpg, jpeg, png or bmp file formats and apply different filters: Gaussian blur, Fourier transform, contouring, sharpening or grayscale transformation. Then the user can crop the image as the input for prediction. The results are shown on the frontend, including the values of neurons in the output layer for each class. The result includes Grad-CAM output in the form of a heatmap overlaying the image (Figure 11).

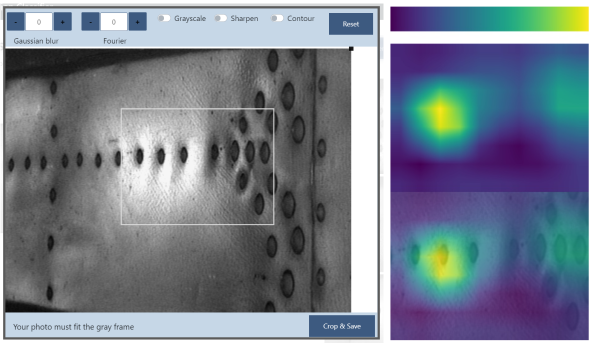


Figure 11. Window for image modifications with sample changes (left), the heatmap generated after presenting the result of prediction (damage-rusty rivet) (right)

The performance of each model was measured using the evaluation metrics presented in Table 3. However, the main focus was on *accuracy* and *recall* (inducing the lowest number of FNs). The best results were achieved using the network based on the EfficientNetB0 architecture (without the fully connected layer at the top of the network, followed by a GlobalAveragePooling2D and a Dense Layer with softmax activation) that was trained on the standardized data. The threshold separating classes was set to 0.3 (it is set to 0.5 by default), as it produced high *recall* while preserving acceptable *accuracy*. The number of FNs was reduced over two times with the use of the selected value threshold in comparison with the default value threshold usage. The issue with the split point manipulation was that only 57 % of pictures showing no corrosion were classified correctly, although it was acceptable as the focus of work was on the other class. The rest (FP) can be eliminated during visual inspection.

Threshold	Accuracy	TP	FP	TN	FN	Precision	Recall	F1
0.5	0.76	555	150	416	163	0.79	0.77	0.78
0.3	0.75	644	241	325	74	0.73	0.90	0.80

Table 3. Metrics computed on the test set for the chosen NN

It is also worth noting that 12 % of images featuring *soft corrosion* and only 1 % of those showing *medium corrosion* were classified incorrectly. That could mean that the former were more similar to not corroded ones from the network's perspective, which is a desirable and intuitive behaviour.

Analysing incorrectly classified images from the *no corrosion* class, one can conclude that the network makes choices similar to those made by nonprofessionals. The differences in light intensity and the aircraft surface waviness can produce a wrong impression of the surface being corroded (Figure 12). It should also be mentioned that the analysis of heatmaps obtained with the use of Grad-CAM shows that the network is focusing on areas near the aircraft rivets, which is desirable.

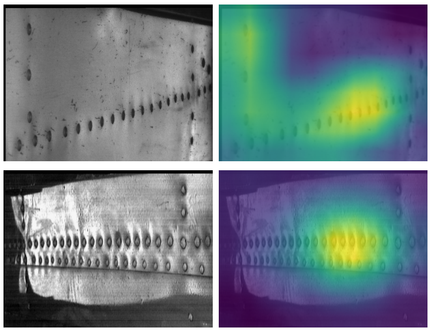


Figure 12. Sample images classified as *corroded* and corresponding heatmaps: examples of TP (top) and FP (bottom) predictions

5.5 Conclusions and Future Work

We have developed a web application for NDI of the aircraft fuselage based on DAIS images. We demonstrate that the system can be useful to predict the occurrence of fuselage corrosion. Moreover, convincing experiments were conducted, showing that using DL for classifying DAIS images for such prediction has potential and can be an effective tool in real, production systems. We expect that having the access to a bigger DAIS image database, even better results could be achieved. Simultaneously, further works regarding ensembles of classifiers based on the best-performing architecture could be executed to reach higher *accuracy*. Moreover, to obtain smaller networks that could be implemented in the DAIS hardware, we propose a multi-teacher knowledge distillation method [19]. Another interesting direction is carrying out extensive image preprocessing on DAIS images. On the web application side, further works could include adding more filters and providing an administrator panel which involves user authentication.

6 CONCLUSIONS

Based on the analysis of the presented theses, we can make observations in two different areas: teaching and research.

From the didactic point of view, we investigated such aspects as the size of the typical students team or preferred topic areas. There are clear trends that can be utilised by teachers to improve the process of managing and evaluating bachelor theses. Our research is narrowed to the software engineering field so it would be interesting to investigate other fields in a similar fashion.

In terms of the conclusions about students' research, presented theses demonstrated interesting results from both technological and conceptual perspectives. All theses delivered working software which solves some specific unique problem, often difficult and even scientific. The applied libraries testify to the good orientation of all the graduates in the modern technology stack. Judging by the analyzed works, it seems that in future we will see more and more theses solving complex problems by taking advantage of the existing machine learning libraries. Good examples are such libraries as Tensor Flow, Keras. This trend is also visible all around the world and is caused by the availability of complex algorithms provided in a moderately easy and approachable way.

It is also worth noting that bachelor theses in the computer science field are increasingly interdisciplinary works. Students are eager to solve real-world problems and have the proper tools to make it possible nowadays. This trend can be beneficial both in the teaching area because students can learn how to deal with non-superficial obstacles, and in the research area: the outcomes of students' projects can be used to develop real solutions.

Acknowledgements

This work was supported by the PLGrid Infrastructure. We would like to thank Professor Dr. Krzysztof Dragan (AFIT, Warsaw) for his contribution to this research.

Jacek Dajda and Michał Idzik are grateful for support from the subvention of Polish Ministry of Education and Science assigned to the AGH University of Science and Technology (Faculty of Computer Science, Electronics and Telecommunications).

REFERENCES

[1] Android Developers: Processes and Threads Overview. Last access: 26.09.2021, available online `https://developer.android.com/guide/components/processes-and-threads`.

[2] Moskala, M.: Effective Kotlin: Best Practices. Chapter 8, 2020, Item 51.

[3] International Norm IEC 60908 Version 2.0. 1999, Audio Recording – Compact Disc Digital Audio System.

[4] Smith III, J. O.: Physical Audio Signal Processing. 2010, [online], Last access: 26.09.2021, available online: `https://www.dsprelated.com/freebooks/pasp/`.

[5] Aldwell, E.—Schachter, C.—Cadwallader, A.: Harmony and Voice Leading. Cengage Learning, 2018.

[6] Cormen, T. H.—Leiserson, C. E.—Rivest, R. L.—Stein, C.: Introduction to Algorithms. MIT Press, 2009.

[7] Sikorski, K.: Harmony: A Collection of Exercises and Examples. PWM, 2003 (in Polish).

[8] HarmonySolver Musescore Plugin. Available at: `https://github.com/miksik98/HarmonySolverPlugin`.

[9] Musescore. Available at: `https://musescore.org`.

[10] Pietrzak, M.—Lozanski, G.—Grever, M.—Andritsos, L.—Blachly, J.—Rogers, K.—Seweryn, M.: On the Analysis of the Human Immunome via an Information Theoretical Approach. International Journal of Computational Biology and Drug Design, Vol. 13, 2020, No. 5-6, pp. 555–581, doi: 10.1504/ijcbdd.2020.113878.

[11] Troussard, X.—Cornet, E.: Hairy Cell Leukemia 2018: Update on Diagnosis, Risk-Stratification, and Treatment. American Journal of Hematology, Vol. 92, 2017, No. 12, pp. 1382–1390, doi: 10.1002/ajh.24936.

[12] Stetler-Stevenson, M.—Tembhare, P. R.: Diagnosis of Hairy Cell Leukemia by Flow Cytometry. Leukemia and Lymphoma, Vol. 52, 2011, No. sup2, pp. 11–13, doi: 10.3109/10428194.2011.570820.

[13] Forconi, F.—Raspadori, D.—Lenoci, M.—Lauria, F.: Absence of Surface CD27 Distinguishes Hairy Cell Leukemia from Other Leukemic B-Cell Malignancies. Haematologica, Vol. 90, 2005, No. 2, pp. 266–268.

[14] Robinson, P. N.: Deep Phenotyping for Precision Medicine. Human Mutation: Variation, Informatics, and Disease, Vol. 33, 2012, No. 5, pp. 777–780, doi: 10.1002/humu.22080.

[15] Komorowski, J. P.—Bellinger, N. C.—Gould, R. W.—Forsyth, D. S.—Eastaugh, G. F.: Research in Corrosion of Ageing Transport Aircraft Structures. CASI Journal, Vol. 47, 2001, No. 3, pp. 289–298.

[16] Malekzadeh, T.—Abdollahzadeh, M.—Nejati, H.—Cheung, N. M.: Aircraft Fuselage Defect Detection Using Deep Neural Networks. 2017, arXiv: 1712.09213.

[17] Ren, I.—Zahiri, F.—Sutton, G.—Kurfess, T.—Saldana, C.: A Deep Ensemble Classifier for Surface Defect Detection in Aircraft Visual Inspection. Smart

and Sustainable Manufacturing Systems, Vol. 4, 2020, No. 1, pp. 81–94, doi: 10.1520/SSMS20200031.

[18] Selvaraju, R. R.—Cogswell, M.—Das, A.—Vedantam, R.—Parikh, D.—Batra, D.: Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. International Journal of Computer Vision, Vol. 128, 2020, pp. 336–359, doi: 10.1007/s11263-019-01228-7.

[19] Zuchniak, K.—Dzwinel, W.—Majerz, E.—Pasternak, A.—Dragan, K.: Corrosion Detection on Aircraft Fuselage with Multi-Teacher Knowledge Distillation. In: Paszynski, M., Kranzlmüller, D., Krzhizhanovskaya, V. V., Dongarra, J. J., Sloot, P. M. A. (Eds.): Computational Science – ICCS 2021. Springer, Cham, Lecture Notes in Computer Science, Vol. 12747, 2021, pp. 318–332, doi: 10.1007/978-3-030-77980-1_25.

Jacek Dajda received his Ph.D. in 2008 in the area of agile software methods. Currently he works at the Department of Computer Science, AGH University of Science and Technology in Kraków. His research interests focus on software engineering, databases and analytical information systems. Chair of the competition committee.

Michał Idzik received his M.Sc. (2014) in the area of natural language processing at AGH University of Science and Technology, Kraków. He worked on access layer for high energy physics experiments database on internship (2012) at CERN (The European Organization for Nuclear Research). His research interests include evolutionary algorithms, multiobjective optimization, natural language processing and data visualization. He is a member of the competition committee.

Wojciech Kania graduated from the Wroclaw University of Science and Technology with thesis entitled "Intuitive Sound Processing Application" under supervision of Bogumiła Hnatkowska.

Mikołaj Sikora, Wiktor Pawłowski, Jakub Sroka graduated from AGH University of Science and Technology with thesis entitled "Rule-based system for solving functional harmony exercises" under supervision of Maciej Smołka.

Filip Ślazyk, Przemysław Jabłecki graduated from AGH University of Science and Technology with thesis entitled "A tool for comparison and integration of feature selection algorithms for modeling of response to targeted therapy for patients with hairy cell leukemia" under supervision of Maciej Malawski.

Emilia Majerz, Aleksandra Pasternak graduated from AGH University of Science and Technology with thesis entitled "System for analyzing damage to the surface of aircraft structures using convolutional neural networks" under supervision of Witold Dzwinel.

Joanna Świebocka-Więk, Wojciech Thomas, Andrzej Paszkiewicz are members of the competition committee.