GRAPHICAL USER INTERFACE (GUI) WORKBENCH FOR YOLOV2

Lazaros Totsas

The Maersk Mc-Kinney Moller Institute
The Technical Faculty
University of Southern Denmark

DECEMBER 2018

Committee

Torben Worm, University of Southern Denmark
Danish Shaikh, University of Southern Denmark

Colophon

Typeset by the author in Utopia 12pt
using LaTeX with the *memoir* class.

Abstract

With the increased use of Machine Learning products, a lot of demand in Machine Learning development arose leading to the need of incorporating new development tools. One of the main problems of addressing a Machine Learning task is the difficulty and the experience of the developer to overcome such a problem.

This report address that problem by suggesting a Graphical User Interface as Machine Learning Workbench. The product would try to investigate if such a solution can lift the need of experienced personnel for creating a Machine Learning model.

CONTENTS

CHAPTER

1 INTRODUCTION

Since 1956, when Artificial Intelligence (AI) first coined in the Dartmouth conference in 1956, there has been a huge increase of usage for machine learning (ML) algorithms[44]. During the 21st century though, AI led to applications which have great impact to our lives today. Such applications provide solutions that were nearly impossible before. The domains where AI is being used is not a small list as the usage has been spread in every possible domain. It might not be the case that the ML algorithm performs everything on its own, but nevertheless it provides an additional value to the existing product/service. Notable mentions are Google's deep neural networks which are used in a variety of their services (Google Translate, Google Now, etc)[21], AI assisted surgery where the surgeon is being assisted with AI to minimize errors and improve movement[7], self-driving car with Tesla delivering the first autonomous car[19], etc.

All ML algorithms are not of the same type, as each one performs better or worse in different tasks[45]. One of those types is Supervised learning which classifies images with computer vision. A Supervised algorithm is being trained by human experts by feeding the algorithm with training data. The training data contain input/predictors and the correct answers (output) is identified. From the training data, the algorithm should be able to learn the patterns in order to identify relationships and dependencies between the target prediction output and the input features. Computer vision is a specific field in ML which tries to gain a high-level of understanding in static images or videos.

A subcategory of computer vision and Supervised learning is object detection[32]. Object detection algorithms are a specific technology which creates a semantic over an image or video for a specific trained class (human, animal, etc) and one of the most broaden uses of such algorithms is face detection and pedestrian detection. However, developing such systems that detect faces or pedestrian can prove to be a complex task. Developers need to have skills and knowledge in computer science such as probability and statistics, data modeling & evaluation, machine learning algorithms and others. For that reason such skill set falls into a specific category of expert developers called Machine Learning Developers.

Such developers require a set of deep learning frameworks to develop a system, without reinventing the wheel. Those frameworks are essential for this level of skill set, as Machine Learning Developers are utilizing existing technologies where Computer Scientists main task is to innovate. One of the most used and well documented frameworks is Google's Tensorflow[42], available in various language but fully supported in Python, C++ and R, following with Caffe which is aiming for speed and

transposability[9], Microsoft Cognitive Toolkit[46], PyTorch[35] and others.

Each and every one of those deep learning frameworks require either utilizing through command line interface or developing the training and testing process through the framework's API. A process like this could be less approachable for someone that does not have experience in those frameworks, as also learning such framework could be inefficient for a small company. Of course for a field expert this could be more approachable as experience is a big factor. However, this does not mean that even experienced developers can not make mistakes using command line interfaces or APIs and would not profit from an automatized solution.

This leads into a question why ML development is still a difficult task that requires human experts, when there are many deep learning frameworks and cutting edge companies that actively develop and use such ML algorithms. In his article Eric Feuilleaubois [17] addresses the exact same concern arguing why and how such task are not approachable and provides suggestions that could have an impact on the effectiveness respectively. His suggestion adds an extra ingredient into ML, a graphical user interface (GUI), which provides an easier way to utilize such algorithms.

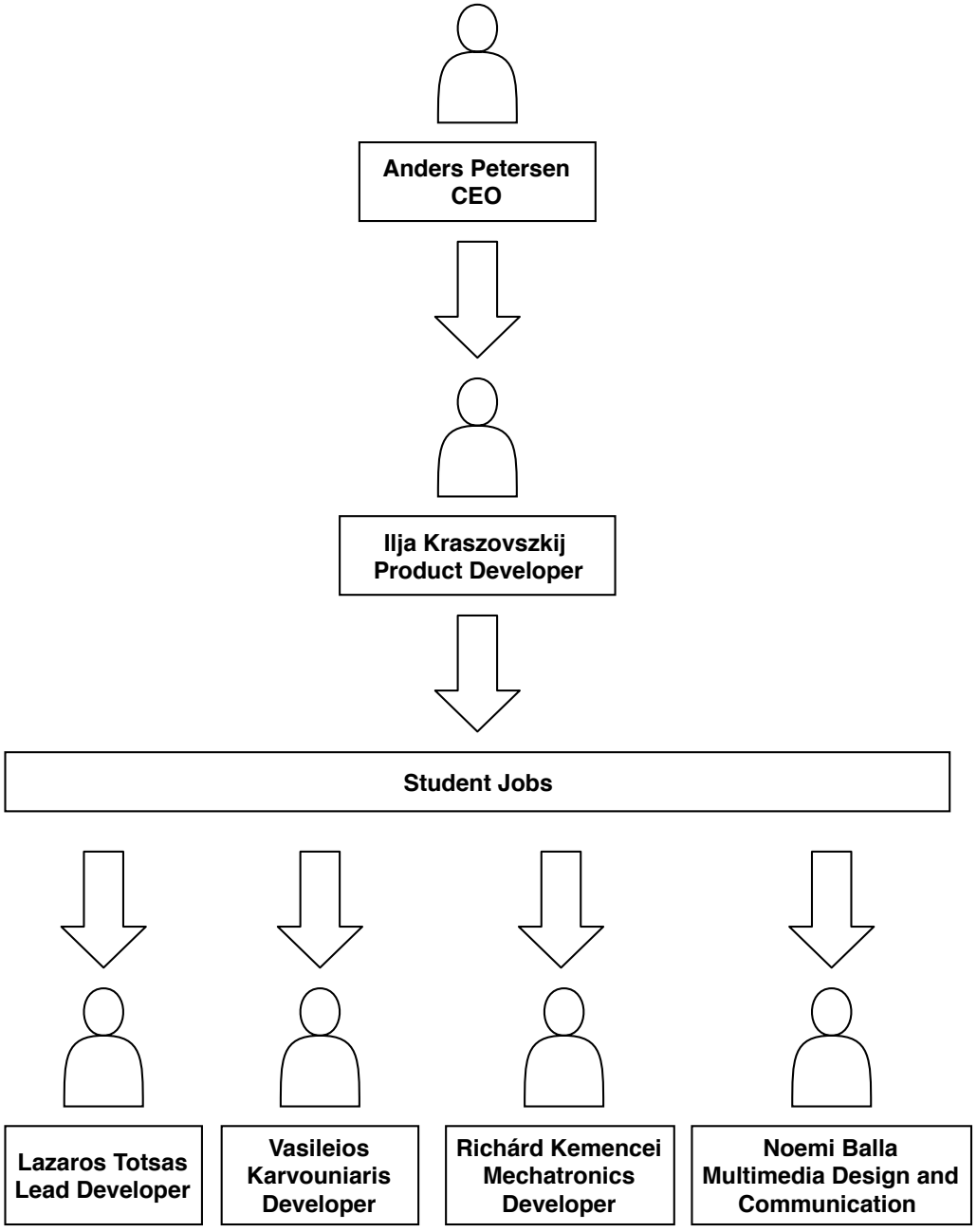
This report will analyze this problem of ML algorithms utilization and provide a solution after addressing the similar existing solutions. The problem is being addressed from the perspective of Ecobotix ApS, as a use case scenario. As a result a set of requirements will occur and will be analyzed in order to build an architecture design for the system, following with the implementation of the system. After the implementation a validation of the existing solution will be performed by using Ecobotix team as validation for the proposed solution. Lastly, a discussion and conclusion will be provided regarding the solution's requirements, followed by future work.

1.1 Use Case Scenario company

Ecobotix ApS is a small company founded in February 2015 by Anders Petersen. Its purpose is to commercialize original patentable ideas of Integrated Pest Management (IPM) by use of unmanned units in agriculture. Over the years Ecobotix has been funded by several programs, such as the Green Development and Demonstration Program (GUDP)[22] and InnoBooster[1], to continue innovating the agricultural domain. The core human resources of the company are Anders Petersen as CEO and Ilja Kraszovszkij as product designer and other student positions. For better understanding the structure of the company and their team position as of responsibilities see Figure 1.1.

We can understand that the company's resources are limited, especially if we think about the limited working hours each member can work due to the fact that the business relies on students. Such formation of a team requires good communication and fast iterations to accomplish the tasks with the minimum impact on the working hours. Therefore Ecobotix ApS works as a good example of a company that would profit from a ML algorithm automization.

Figure 1.1: Ecobotix Diagram



CHAPTER

2 ANALYSIS

In this chapter the problem statement will be analyzed in order to derive requirements. As a result, an early hypothesis will be formed as a proposed solution. The proposed solution and requirements will be followed by the Minimal Viable Product (MVP)[29] definition as a product with just enough features to verify or disprove the hypothesis and provide suggestions for future development.

2.1 Problem Statement

Developing a ML product currently requires the presence of someone with expertise in AI field. As a result this leads in increased expenses for a company, especially, a start up company like Ecobotix with limited resources. This automatically subtracts the opportunity for small companies to innovate. By bringing products with ML functionality, a company could increase the value of a product and as a result the value of the company itself.

Moreover, ML specialists lack a set of a complete toolkit to improve their productivity and functionality. At the moment, command line interfaces or APIs are the only options available for development. On top of that creating the workspace with all the necessary frameworks installed and tools provided is a challenging process.

Finally, the lack of a unified solution affects both sides, experienced users or not, leading to the reinvention of the wheel, even for simple tasks while decreasing productivity.

Unvalidated Hypothesis

As a result from the problem statement, this report proposes a Graphical User Interface (GUI) as a ML workbench, providing a set of tools to train and test a model. The unvalidated hypothesis is formulated with the research question:

"If a GUI workbench related to ML existed, then it would improve the performance of ML developers, experienced or not."

In order to validate the hypothesis a methodology will be formulated to address the problem. The methodology is based on the book "The Lean Startup" by Eric Ries where he explains that each claim which says that customers would use the product that you are developing, is in essence a hypothesis that needs to be validated[38]. To validate the hypothesis, his approach involves creating a Minimal Viable Product (MVP) and test it to relevant early customers. In our case the early customer is the

example company Ecobotix ApS. Based on Eric Ries, after testing for feedback the MVP you have to preserve or pivot the original assessment. The feedback will show if your hypothesis is valid and preserve or you need to pivot and change the solution.

In his book he describes a development methodology "The feedback loop", which is consisted by three steps, **Build, Measure and Learn**. The MVP is part of the first step - **Build**, the testing process of the product is the **Measure** and evaluating and the feedback is the **Learn** step which determinates whether to pivot or preserve.

2.2 Proposed Solution

The proposed solution is based on the problem statement and the unvalidated hypothesis. From the problem statement as it is being formulated a ML product is required to fill the missing gap in order to make ML development an easier process using a GUI as workbench. The validation methodology of the hypothesis first step is to Build a MVP. Thus, a more detailed analysis is required to identify the critical points. ML development includes three general steps which will be referred from now on as modules, dataset, training and testing.

In her article Yael Gavish refers to how to develop a ML model, by identifying the required steps[20]. The steps that she suggests add two additional steps by including the conceptualization of the idea and the productization. Those steps are not correlated with our problem statement though and therefore not used. Continuing the analysis, the MVP product will consist of those three modules in order for the customer to be able to provide a complete solution for training a ML model. As a MVP the solution will focus into a relevant area for Ecobotix. Such an area involves computer vision using a mounted camera into the Ecobotix's product in order to detect a certain type of plant. For this matter object detection will be used using supervised learning classification[10].

Each of the three modules will be developed in correlation of the later, leading to a MVP product that tries to validate the hypothesis by using Ecobotix as a test case scenario.

2.3 Related Work

The related work will focus into addressing related work similar to the proposed solution. The purpose of the related work is to investigate whether such a solution has been addressed already and to what extend. The research is performed with Google's search engine with the following keywords, using different combinations:

GUI, graphical, user, interface, machine, learning, algorithms, friendly, easy

The findings are presented below as a list categorized into experienced, inexperienced and both users in ML leaning. Each of the findings will be evaluated based on the most related category they belong to and how they address the problem statement.

- **ML workbench for experts**

1. **Azure Machine Learning service** [6]: Azure from Microsoft provides a comprehensive set of tools from training the model, testing it, even deploying it in production. The UI elements are just enough to provide a slight help with the development, but it stills needs great input from the user to perform a task. The user is able to write code and perform changes but it is not possible to use only the UI elements with a bit of code.
2. **Dataiku** [47]: Dataiku is a general platform related to big data. One of their features is machine learning, similar with Azure. The UI interactions provide great help as they include ready to use models without the need of coding. Although, you can not train your model from scratch without coding.

- **ML workbench for inexperienced**

1. **Lobe** [13]: Lobe acquired by Microsoft is a drag and drop tool, that requires no additional interaction to train, test and publish your model other than interacting with a UI. Currently it is in invitation beta phase and they support creating classification models using computer vision. Some of their examples are object detection, sign language translator, drone autopilot, etc. However, from their examples and demonstrations it seems that you are unable to select which algorithm would perform the classification.

- **ML workbench for both**

1. **Deep Cognition** [12]: Deep Cognition is a visual editor and an IDE for ML developers. It allows to train and test a model with the easiness of a UI and at the same time fine tune by coding. The idea is that whatever you can do while coding a ML algorithm, you can do it by using Deep Cognition visual editor, at least for the most parts. Either if you want to create model following the AI wizard for pretrained models or you want to build the model layer by layer, Deep Cognition allows you to do that via the visual editor or by coding it. It also supports running it as a desktop application or by a cloud instance. One thing that is not included is preprocessing a custom dataset. Although, you can use one made outside the platform or use existing ones.

2.4 Object detection algorithm

The purpose of this section is to address which algorithm will be used for the proposed solution. The selection must take into consideration the specific area in which the product as MVP is referring to. As already mentioned, the MVP will focus on using an algorithm that is able to detect objects, in particular plants related to Ecobotix product. For that matter it is required to understand how the object detection algorithm could be used on this product.

Ecobotix is using a drone to disperse biological material into fields with various plants using a dispenser. The dispersing process is being performed manually by the operator that controls the drone. As of the dispersing location is being done by

approximation where to drop the materials. Such an approach does not provide the best solution to this matter. For this reason, a ML model could be used while the drone is in flight mode to detect where to drop the material and be precise.

Such an approach requires the use of an algorithm that can perform classification in real time. Taking that into consideration Arthur Ouaknine performed a benchmark review of the current state-of-the-art object detection models[34]. In his review he addresses the subject with how an object detection algorithm is being used and what are the criteria to identify the best choice based on the use. The criteria that he mentions are speed and a performance metric. The speed is how fast the model can perform detection measured in Frames Per Second (FPS)[18] and how well it performs using mean Average Precision (mAP)[23]. The result of his tests are shown in Figure 2.1 and displaying the mAP and FPS performance respectively of the current state-of-art algorithms in PASCAL VOC[15] and COCO datasets[27].

Figure 2.1: Overview of the mAP scores on the 2007, 2010, 2012 PASCAL VOC dataset and 2015, 2016 COCO datasets.

Model	PASCAL VOC 2007	PASCAL VOC 2010	PASCAL VOC 2012	COCO 2015 (IoU=0.5)	COCO 2015 (IoU=0.75)	COCO 2015 (Official Metric)	COCO 2016 (IoU=0.5)	COCO 2016 (IoU=0.75)	COCO 2016 (Official Metric)	Real Time Speed
R-CNN	x	62.4%	x	x	x	x	x	x	x	No
Fast R-CNN	70.0%	68.8%	68.4%	x	x	x	x	x	x	No
Faster R-CNN	78.8%	x	75.9%	x	x	x	x	x	x	No
R-FCN	82.0%	x	x	53.2%	x	31.5%	x	x	x	No
YOLO	63.7%	x	57.9%	x	x	x	x	x	x	Yes
SSD	83.2%	x	82.2%	48.5%	30.3%	31.5%	x	x	x	No
YOLOv2	78.6%	x	x	44.0%	19.2%	21.6%	x	x	x	Yes
NASNet	x	x	x	43.1%	x	x	x	x	x	No
Mask R-CNN	x	x	x	x	x	x	62.3%	43.3%	39.8%	No

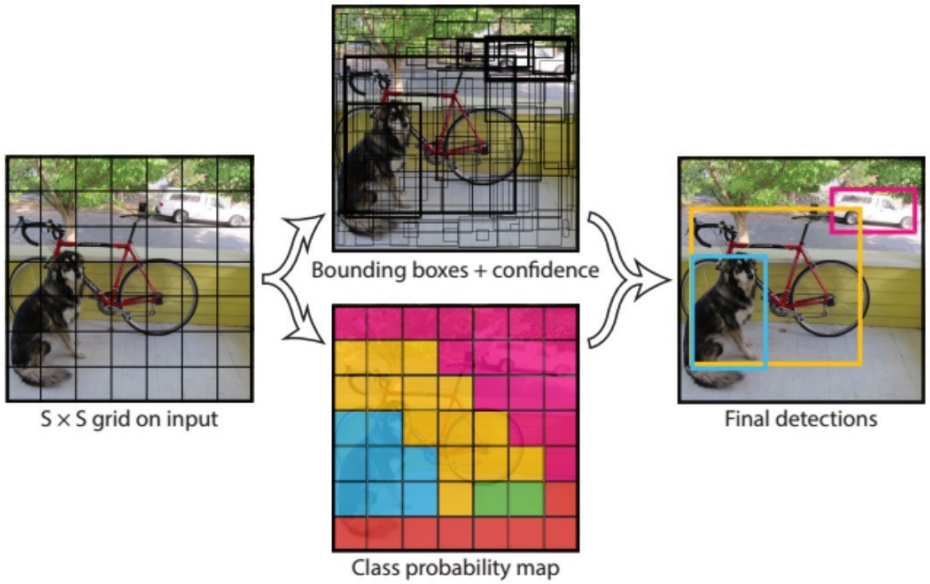
Yolov2

From his benchmark tests and the need of Ecobotix for a real time prediction model can be concluded that Yolov2 is the right choice to be used as the object detection algorithm. The Yolov2 is developed by Joseph Redmon and Ali Farhadi focusing on improving accuracy while still being a fast detector[36]. The architecture of Yolov2 is consisted by 24 convolutional layers followed by 2 fully connected layers, where the latest ones are responsible for predicting the anchor boxes. Yolov2 sees the detection as a regresion problem. It divides the input image into SxS grid where for each of the grid cell predicts a B bounding boxes, confidence for each box and C class probabilities as shown in Figure 2.3.

Figure 2.2: Real Time Systems on PASCAL VOC 2007. Comparison of speeds and performances for models trained with the 2007 and 2012 PASCAL VOC datasets.

Model	mAP	FPS	Real Time speed
Fast YOLO	52.7%	**155**	Yes
YOLO	**63.4%**	45	Yes
YOLO VGG-16	66.4%	21	No
Fast R-CNN	70.0%	0.5	No
Faster R-CNN VGG-16	73.2%	7	No
Faster R-CNN ZF	62.1%	18	No

Figure 2.3: Yolov2 confidence box scores prediction



CHAPTER

3 REQUIREMENTS

In this chapter a set of user stories and non-functional requirements will be mapped. The mapping of the requirements derive from proposed solution referred to the problem statement. Furthermore, from the related work and the object detection algorithm research a set of objectives and tools are identified. The result of that will be a product as MVP that tries to evaluate the unvalidated hypothesis.

Based on the methodology selected for the validation of hypothesis, Eric Ries states that the first step is to Build a MVP, the product must have the minimum features in order preserver or pivot after testing the product. The selection of User Stories and Nonfunctional Requirements for the MVP are based on MoSCoW method [30], which is a prioritization technique used in project management and software development. Each of the requirements are marked with a MoSCoW acronym showing the prioritized based on the MVP criteria :

- **M**ust : A critical requirement
- **S**hould : A high priority requirement
- **C**ould : A requirement not necessary
- **W**on't : A requirement that has the least priority, maybe for a later release

User Stories

- The user will be able to train a model using Yolov2 by interacting with a GUI. **[M]**
- The user will be able to test an existing Yolov2 model or a trained one with GUI. **[M]**
- The user will be able to import pre-trained models either for training or testing Yolov2. **[C]**
- The training process' parameters can be left as default or customized to the users need. **[M]**
- The user will be able to start and stop the process on demand. **[M]**
- During the training process the user will be able to inspect the process by observing visualized metric diagrams. **[M]**

- The user will be able to export the trained model on demand. **[C]**
- The user will be able to create or import a dataset in order to train or test a model on it. **[S]**
- The user will have the capability for the dataset, either created or imported to preprocess it. The preprocess will be related to Yolov2 requirements. **[M]**
- Whether the preprocessing is relevant to Yolov2 or not, the user should be able to review the preprocess after the process is finished. **[S]**
- The user will be able to test and review the performance of a trained or imported model. **[M]**
- The trained models will be saved and deleted on demand. **[S]**
- During testing the system will save the statistics of each test for later review. **[C]**
- The user will be able to test Yolov2 model on an image or a video. **[M]**
- The user will be able to create a new project and load it again with all the parameters and settings. **[M]**
- The user should be able to select any of the created projects, as also delete it. **[S]**

Nonfunctional Requirements

- The training process would be able to perform locally or on a server to utilize more processing power and avoid excess amount of processing power leading to unusable operating system. **[S]**
- The user must be able to perform all the task, from preprocessing, training and testing without the need/assistance of command line terminal or coding. **[M]**
- The UI must be simple and self explanatory, avoiding unnecessary buttons or interactions. **[S]**
- The platform should eliminate the requirement from the user to install various libraries and should be as easy as opening a desktop application. **[C]**
- The platform should provide a rest API for production use and integration of each model into other services/products. **[C]**
- The platform should be able to run the training process, utilizing CPU or GPU, without the need of installing additional libraries. **[C]**

CHAPTER

4 DESIGN

This chapter will describe the design architecture of the platform and the decisions that were taken to conclude into this design.

4.1 Application Type

The application type should be correlated with the proposed solution and the requirements mentioned in chapter 3. From the proposed solution analysis three essential steps were identified for the a development of a ML model. A dataset that includes the raw data, training the ML model using the dataset and testing it. As already mentioned, those three steps will be named as modules. In order to conclude what type of application fits the best for the product, three assessment categories were used inferred from requirements and proposed solution.

1. **Cross-Platform** The product must be available in every platform without the need of installing any additional libraries and always use the same dependencies. Platform availability includes Windows, Mac, Linux.

2. **Resource Utilization** During the training process the ML algorithm uses high levels of computing power. This high use of available resources could lead to unusable operating system. Such a problem can be solved by deploying the product into a server with higher computing power.

3. **Easiness** As the problem statement addresses the easiness of ML algorithms, the architecture design should also follow up. The product must be easy to use, but also easy to deploy it with the required dependencies.

Desktop Application

A desktop application can support cross-platform development using such featured frameworks. Such frameworks support the development of a product that is developed once and can be deployed across different platforms.

The second assessment category Resource Utilization implies the need of two separate implementations for the ML algorithms functionality. One being included inside the package of the desktop application and one being capable to be deployed on a server. Such a solution requires developing two APIs for communicating with the algorithms, thus such a solution will increase the complexity of the development.

Lastly, a desktop application has a high level of easiness especially for installing the application and run it. Without the need of installing separate libraries and dependencies. Such a case though, is not applicable for the proposed solution as the ML algorithm must not be embedded inside the application.

Web Application

A web application is originated to have a separation between presentation layer and application layer as the presentation layer sends content to browsers in the form of HTML/JS/CSS. By this design the requirement of running the training process remotely is fulfilled.

Secondly, a web application is unaware of the platform that is being used as the user access the presentation layer through the web browser. With that being said, it must be taken into consideration that the application layer is being hosted in an environment with installed required libraries/dependencies.

Lastly, referring to the Easiness assessment as mentioned above the application layer must be deployed into an environment that supports the overall functionality of the platform. Taking that into account either if the user is going to "install" the platform into his personal computer or into a more powerful computing wise server. The need of pre-installed libraries/dependencies remains.

Application Type Selection

Based on the mentioned criteria and the analysis of the two available application types a web based application is selected. It must be taken into consideration that such a choice is not the perfect solution for solving the problem statement because of the added complexity of invoking and utilizing the ML algorithm. This solution though provides great freedom of utilization the ML algorithms, as the functionality will not be embedded into the application but out of the scope of it. By that it increases the extensibility and keeps low coupling between the application layer and the ML algorithms. A further analysis will be performed in the next section 4.2.

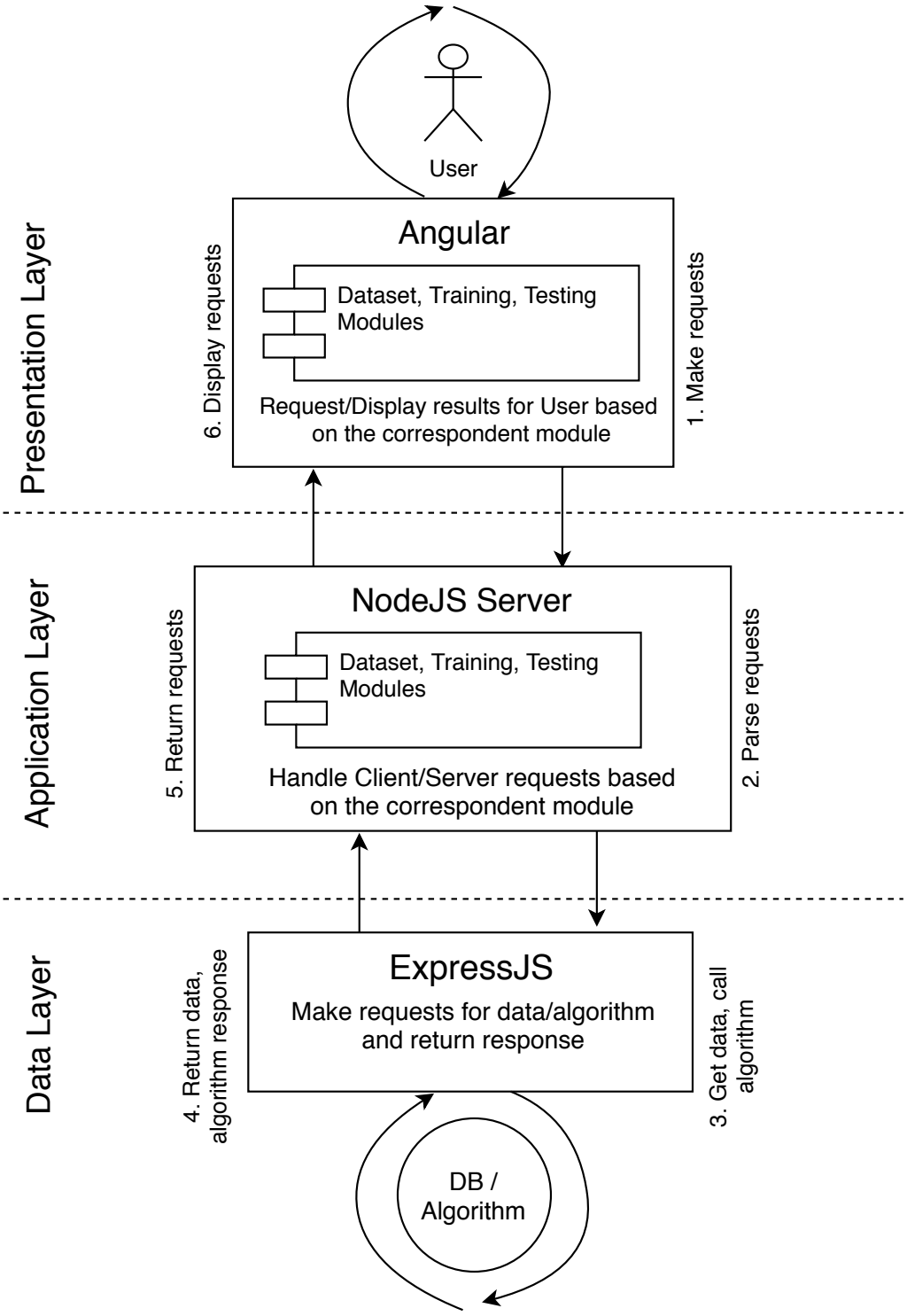
4.2 A 3-Tier Architecture

A 3-tier architecture is a type of software architecture for web applications based on client-server model[31]. A "tier" is referred as a "layer" which separates the User Interface, the Application Logic and Data Storage into Presentation Layer, Application Layer and Data Layer respectively. In this use of the 3-tier architecture the Data layer will also include the invocation of ML algorithms.

In Figure 4.1 the domain model of the proposed solution based on the 3-tier architecture is shown. It can be recognized there is one main actor of the system, the User. As the only benefactor of the system, the User is responsible to deploy it for use in any of the supported platforms. The system is being separated into 3 layers as of the 3-tier architecture, using a Representational State Transfer (REST)[37] Application Programming Interface (API)[5] for propagating the requests/responses. Both the presentation and application layer include a module. This module is a visualization of the three sub-systems referred as modules from the proposed solution. In the

presentation layer the modules are being used as web pages, with the specified functionality for each one of them. In the application layer though, each of the modules refers to the mapping of the API with the relevant handler as such. The handlers are implementations of each module either for performing CRUD[11] tasks on DB or for using the ML algorithm.

Figure 4.1: Domain Model



Presentation Layer

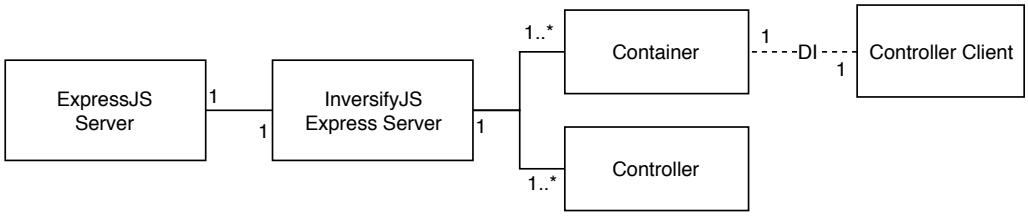
The presentation layer is using Angular which is a structural framework for dynamic web apps written in Typescript[4]. It uses HTML as template language and it extends the HTML's syntax to create application driven components called directives. These directives are being infused with extra functionality compared to regular Document Object Model (DOM) as data binding. The data binding is an asynchronous connection between the DOM and the component. In Angular the term services is being correlated with application logic. As the components are used for presenting the data and visualize them, a service is responsible for everything else, providing a decoupled connection for the data with the components and the view. This exact design is followed in the presentation layer of the product as shown in Figure 4.2.

In order for the User to use each of the three pages the User must create or select a project. The project selection is performed by the project service which create a call to the application layer via API. Each of the pages include components that are created in correlation with the page functionality. More details about each of the components and their functionality follows in the next sections by describing each module.

Application Layer

The application layer contains the domain logic of the product by replying to requests and return responses of operations and data between presentation and data layer. It is written in Typescript using ExpressJS[16] and InversifyJS[24] frameworks to create a server with REST API functionality. ExpressJS is being used to instantiate the server on a specific port and InversifyJS for populating the server with functionality. The API is invoked into the server by defining a class with the annotation **@controller**. Each controller belongs into a container assigned to a controller client class with Dependency Injection (DI)[14] by InversifyJS. By using this design the controller is unaware of the requested functionality as the client is responsible for doing so and by doing so the Separation of Concerns (SoC)[40] is followed. Lastly, the containers are consumed by the server creating an API knowing how to distribute each request among controllers. In Figure 4.3 is shown a representation of the mentioned design.

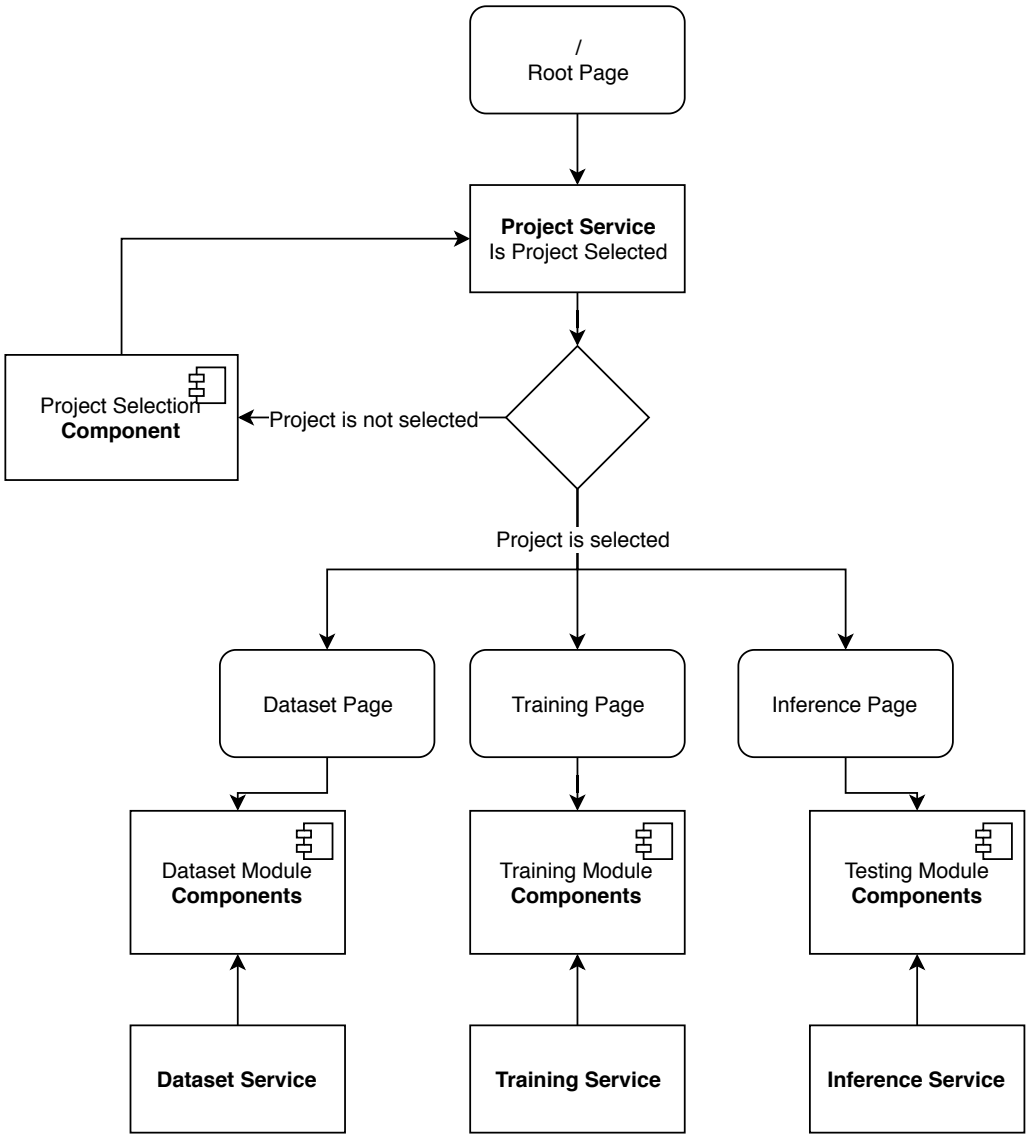
Figure 4.3: Application Layer



Data Layer

The data layer contains the implementation of the selected ML algorithm, the database for storing necessary data structures and the workspaces used as User projects. The data layer is responsible to resolve with content the requests performed from the presentation layer.

Figure 4.2: Presentation Layer



Yolov2

The data layer holds the implementation of Yolov2 as a separate service outside the boundaries of the system. Such a solution provides the ability of hot-swapping the algorithm without relatively changing the structure of the application layer. For the purpose of this report for using a ML algorithm as proof of concept an already ready to use repository have been used for Yolov2. The implementation of Yolov2 belongs to a Github user rodrigo2019[39]. The implementation is written in Python using a high-level neural networks API Keras[25] and Tensorflow[42] as a foundation library that can be used to create Deep Learning models. The repository splits the implementation into two parts, back-end and front-end as commonly called. The back-end is written using Tensorflow and includes the implementation of the convolutional layers of Yolov2. The front-end is written using Keras and includes the invocation of the back-end model. The front-end includes the overall functionality of how to prepare the dataset for the training, the training process and testing a model.

By separating the utilization and the actual implementation of Yolov2 room is created for using another ML algorithm instead. The back-end also includes other implementations which can be used, like "Tiny Yolo", "MobileNet", "SqueezeNet" and "Inception3". The selection of which algorithm is going to be used is done with a configuration file, which is using a config JSON file. The configuration is called from the front-end and includes a set of options how the algorithm is going to be used. Some of the options are selecting pretrained weights, number of epochs, batch size, learning rate, etc. A more detailed explanation is gonna be performed during the analysis of the modules later on, in this chapter.

Database

For storing data structures for each project LokiJS has been used. LokiJS is a NoSQL in-memory JavaScript Datastore with Persistence focused on performance[26]. The data structures are being saved and can be read from a file using JSON, without the need of additional support environment as it only requires Javascript. Each project includes a JSON file with all the uploaded images and the annotation classes. A JSON object of an image is shown in listing 4.1 .

Listing 4.1: Saved image as JSON DB object

```
{
    "fieldname": "file",
    "originalname": "IMG_20171004_213219.jpg",
    "encoding": "7bit",
    "mimetype": "image/jpeg",
    "destination": "projects/please/images/",
    "filename": "IMG_20171004_213219.jpg",
    "path":
       "projects/please/images/IMG_20171004_213219.jpg",
    "size": 99844,
    "$loki": 6,
    "annotation": [{
        "id": 0,
        "class": "lazaros",
        "xmin": 143.76638888888888,
        "xmax": 1114.2375,
        "ymax": 0,
        "ymin": 1129.6054166666668,
        "x": 143.76638888888888,
        "y": 0,
        "width": 970.4711111111111,
        "height": 1129.6054166666668
    }],
    "width": 1382,
    "height": 1843
}
```

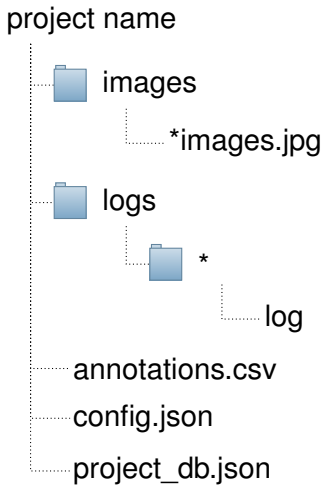
The images are saved as data entries of a collection named "images". Each entry includes the name of the file and the relative path of the image, as the images are

not being saved as files but as pointers to the files. Furthermore, LokiJS stores the index of the saved image as "$loki", in the given example the image has index 6 in the collection array. Lastly, the object includes each annotation with the details how it is mapped on the image.

Workspace

Each project the User is creating refers into a file structure, rather than saving it into the database. When the User creates a project, a new folder with the given name is created on the server inside a "Projects" folder. The reason of saving a project in that form is based on the types of the data and how a project is defined. In Figure 4.4 the representation of the project structure is shown. The "config.json" is already mentioned as a configuration file for the algorithm. The "project_db.json" is the database file, as referred previously, for storing image data structures. The "logs" folder and the "annotation.csv" will be mentioned further on, in the 4.4 section.

Figure 4.4: Project Structure as Workspace



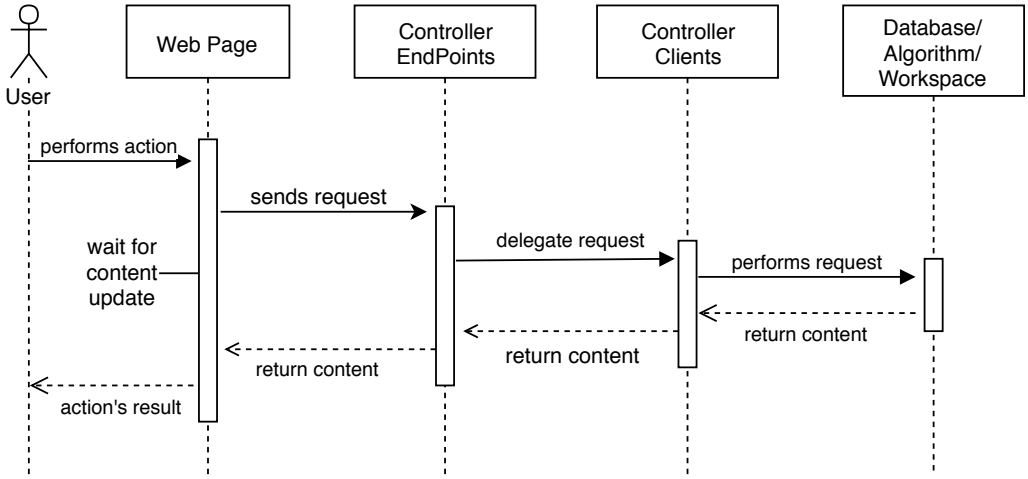
4.3 Communication Data Flow

In order for the presentation layer to communicate with the application layer for content/actions a REST API is build. As mentioned before in the application layer a class as **@controller** is being used for **EndPoint**. An endpoint in a communication channel is the one end from two systems trying to communicate. Typically for APIs an endpoint is part of the communication URL that is used. In this product four endpoints are being used defined by the **@controller** classes. Three ML algorithms based for handling Yolov2 functionality and one for handling project functionality.

- /image-management
- /prediction-management
- /training-management
- /project-management

Each of the endpoints include sub-endpoints that reflect the functionality of the main endpoint. When invoking a call the appropriate client takes place to handle the request from the **@controller**. The clients use the data layer to respond the request and pass it to the API. A general visualization is shown in Figure 4.5.

Figure 4.5: Generalization of a REST API call



The requested content could refer to any of the three data structures of the data layers, as the response could be data or the confirmation that an action is performed. It is important to mention that some calls require to wait for a response as they are directly linked with updating the UI.

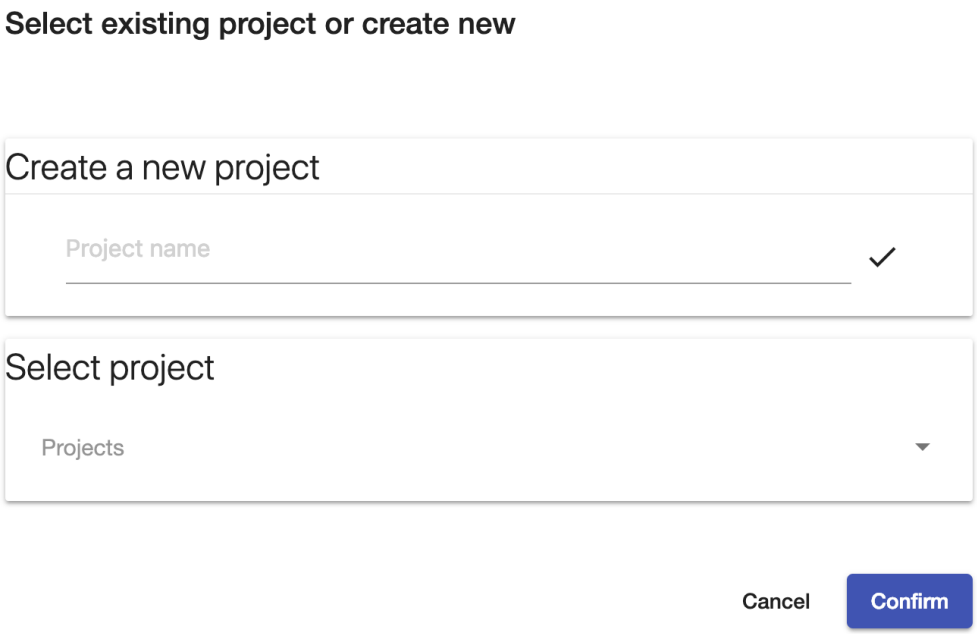
4.4 Conceptualization

This section will show a detailed analysis of the functionality from the product. The analysis will be performed from the perspective of the steps required to develop a model, as the problem solution states. In the analysis chapter 2 is mentioned that in order to develop a ML model, three steps are required referred as modules, **dataset**, **training** and **testing**. Each of these modules will be analyzed from the presentation layer to the data layer following a top-down breakdown, as the UI has significant role in the proposed solution. Before moving on the project creation/selection must be mentioned because it is linked with the functionality of each module.

Project

As mentioned before a project is defined by a folder which contains the necessary files that are required for a project. In order for the User to use any of the modules first a project must be selected. For that a "project" component was created with a "project.service" for communicating with the API as shown in Figure 4.6.

Figure 4.6: Project selection/creation UI



When the User opens the "project" component it requests a list of already created projects from the application layer by iterating the projects folder for directories in the data layer.

On project creation a new directory is created inside the projects folder with the default **config.json** and an empty images folder. Before creation it checks if a project with the same name exists and selects the project, otherwise it returns an error message.

The project selection is performed by selecting the project directory as a workspace using Singleton pattern[41]. Using a singleton pattern ensures that the application layer will have one instance as a workspace, as the product use is for single user. The class *WorkspaceSingleton* is responsible for instantiating the project based on the project path location. Furthermore, it exposes a static directory containing the selected project images.

Dataset Module

Starting from the dataset module Yolov2 requires a dataset where images are annotated with a class that indicates what is the annotated object. The dataset page includes three individual Angular components, images, image-annotation and upload.

Upload component

The upload component has been used as the name states to upload the images to the data layer. The User, by interacting with the UI as shown in Figure, 4.7 uploads the images in sequence one by one using the "upload.service". This way the UI can visualize the completion of each upload. In order to achieve this visualization the Observable pattern has been used[33]. In Angular, by assigning the type *observable* to a function or variable, you can *subscribe* for changes, when available.

When uploading the images, the application layer uses the correspondent image path based on the project singleton instance. As also, initialize the database by storing the image properties, if there is no available database by that moment.

Figure 4.7: Image uploading sequence UI

Images component

The images component is a grid list of images fetched with the "images.services". The *service* is requesting the images which are currently uploaded to the data layer and returns an array of type *Images* as model to the component. The template of the component iterates the array and renders each image, as shown in Listing 4.2. The *source* of each image is defined by the *url* and the image filename. The *url* is static link that dynamically changes content from the application layer based on the selected project. An example of how to access an image with url is as follows :

http://localhost:8001/static/IMG_20170630_145902.jpg

Using such a design for displaying images greatly decreases the network utilization as the REST API carries only properties of the images as text, not the image itself as Base64 encode. Also, in order to access an image the filename is needed and it is not possible to access the */static/* directory url for safety reasons.

Lastly, every image has a *(click)* action, which informs the *service* of the image-annotation component that an image is selected.

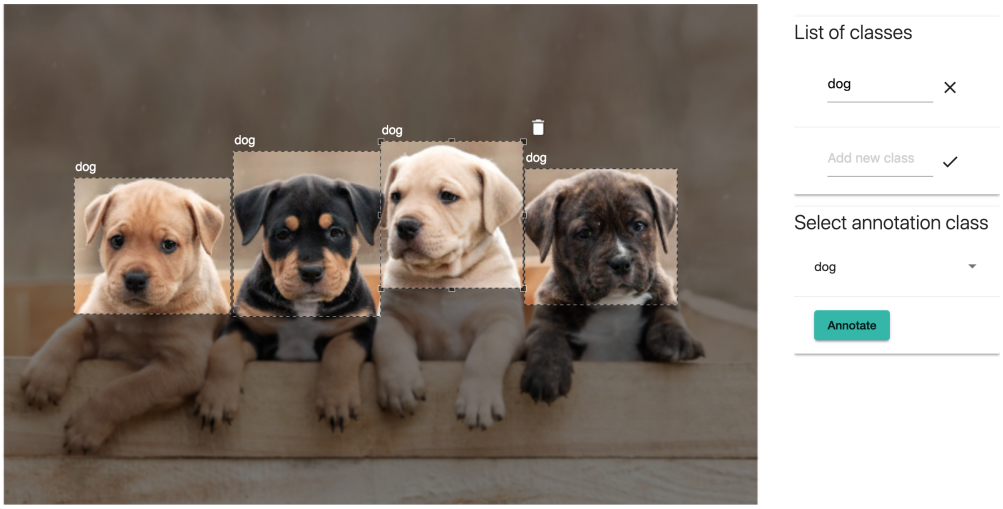
Listing 4.2: Images component DOM template

```
1 <div class="scroll-container">
2   <mat-grid-list cols="2" rowHeight="2:1">
3     <mat-grid-tile *ngFor="let img of images">
4       <img style="width: 100%;height: auto;"
            (click)="newImage(img.filename)"
            src={{url+img.filename}}>
5     </mat-grid-tile>
6   </mat-grid-list>
7 </div>
```

Image-annotation component

The lifecycle of the image-annotation component begins when an image is selected from the images component. In order for the two components to communicate with each other, a listener is created to the "image.annotation.service". A listener is instantiated in Angular as *BehaviorSubject*, which when combined with *Observable* type variable creates a message broker that can be "fired" on events. The image-annotation component on *ngOnInit()* subscribes to such events. On this subscription "fire" event it displays the image that the "image.service" sends and requests from the application layer that the already created annotation classes saved in the database. When the request is completed it updates the template class list of the component as shown in Figure 4.8.

Figure 4.8: Image-annotation component UI



As we can see the User can interact with the UI to create annotation classes, as also delete existing ones. This functionality is supported by the component's service which on UI action, sends POST calls on the "/project-management" endpoints with sub-endpoints "/class/new" or "/class/delete" respectively. Each POST includes a payload with *body* the annotation class name, either for deleting it or creating a new one.

Furthermore, the component's main functionality is to annotate the selected image. The annotation has two stages, the visualization of the already created annotations saved in the database and saving the newly created annotations inside the database. When the image is shown the component's service create a GET call request for the already available annotations of the image on the "/image-management" endpoint with sub-endpoint "/annotate/list/ + img". The *img* is for the image filename. The annotation process is performed by creating a rectangular via "drag and drop" action on the image. Then the User selects which annotation class represents the annotated rectangular. Lastly, the User interacts with the "Annotate" button to submit any change, by creating a POST call with sub-endpoints "/annotate/new" or "/annotate/update". There is also a sub-endpoint as "/annotate/delete" for deleting an annotation. The reason that the component service is using two different endpoints is because of the type of the problem that it tries to solve. The classes are

related to the project and the annotation is related to an image. That might be an indication that this component's functionality could be split in two separate components. The implementation of the component as also the two annotation steps will be analyzed further more in the chapter 5.1 describing which library have been used and how.

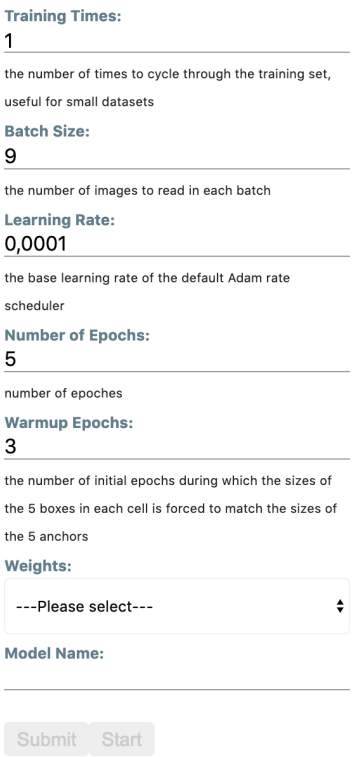
Training module

Following up after the creation of the annotated dataset is the training module. The training module functionality is in regards of training a Yolov2 model using the required dataset. The training module includes one component which is the "config-editor" and the Tensorboard for visualizing the training process metrics.

Config-editor component

In the analysis of Yolov2 at subsection 4.2 a config.json file that saves the configurations of Yolov2 and uses it for starting the training process is described. The "config-editor" component is in practice a UI form that is related to the configuration file. When the User enters the training page on *ngOnInit()* lifecycle performs a GET call for retrieving the configurations and assigns them into a form group of variables. The component is being support by the "training.service" for communicating with the application layer. The ML properties that have been selected for configuration are the most used and easy to understand, as the product is a MVP.

Figure 4.9: Config-editor component UI



In Figure 4.9 the available options as ML properties to configure for Yolov2 are shown, as also their default values. The form requires to fulfill all the required fields in order to start the training process after sending the new values to the data layer. In order to do so the POST call is using the "/training-management" with sub-endpoint "/config/update" and the GET call "/config" for the retrieval.

The "Weights" field refers to the selection of existing pretrained weights or starts from scratch. For better understanding an example is given in regards of the configuration process and the training process.

The User who is going to train a model for the first time selects the option "Start clean" and names the model "dogs". During the training process on every epoch two metrics are used in order to evaluate the model that needs to be saved. The first metric is **Average loss** and the second is **mAP**. Every time the **Average loss** is lower than the previous time the model is saved as "dogs.h5" as Keras model. When the **mAP** is higher than the previous, a new model is saved as "dogs_bestMap.h5". In both cases the new models overwrite the previous ones.

In the second phase the User wants to retrain the model, because some changes were performed on the dataset. The User can choose the pretrained weights option as "Previous highest mAP".

The reason that there is only a mAP selection is because object detection is the most accurate metric. Sometimes it takes time for the mAP metric to start as the algorithm does not detect any object from the validation dataset. In that case the average loss model is selected automatically.

On form submission the application layer needs to make the appropriate changes in order to prepare the file structure of the project. First, the *training.client* saves the form values to the config.json file with full path. Doing so ensures that during the training process Yolov2 will be able to find the required files. One step before the training process is creating the training and validation dataset by separating the original dataset. This is done automatically from the Yolov2 implementation that is being used with 85% as training set and 15% as validation set. The split of the dataset is performed when each image from the database with the annotations is being parsed to create the "annotations.csv" file. The "annotations.csv" includes all the images with their annotations following the structure :

'path', 'xmin', 'ymin', 'xmax', 'ymax', 'class'

Each line of the csv file includes the image name, the coordinates of one annotation and the class. If an image has more than one annotation another row is created with the same image name and the coordination of the annotation.

Tensorboard

Tensorboard is part of the Tensorflow library and is being used for visualizing the training metrics using a log file that is being generated every time the training process is started. In order to use Tensorboard an *iframe* is being used with the url of the Tensorboard instance currently running. The *iframe* url source always points to localhost with port 6006, as the application layer will always use that url.

Each project includes a logs folder where the Tensorboard is looking for available logs. Each time the User trains the model the "training.client" creates a sub-folder

inside the logs folder and names it incrementally as +1 of the previous train time. Following that design when a User has trained the model two times there will be two different folders names, 1 and 2, as also the logs of each training as show in Figure 4.10. The invocation of the Yolov2 for training it and of Tensorboard will be described

Figure 4.10: Example of multiple logs for one model

in chapter 5.2 with more details.

Testing module

The testing module wraps the performance of a created model testing it into an image. The inference page does not have any components as it is used only for uploading a test case image for inference using a POST call to "/prediction-management" endpoint with sub-endpoint "/predict_yolo". The POST call uses a Promise to ensure that the call will be handled when the response will arrive from the "detector.client" and annotate the image with the detected object. The design flow of the testing module is performed into 6 steps.

1. Upload the image using the Inference page
2. Temporarily saving the image into the data layer
3. Invoke Yolov2 for testing the image based on the project configuration file and weights
4. Parse the output text into an array with the coordinates and class per detected object
5. Respond to the request by sending back the array to the presentation layer
6. Annotate the image displaying the annotated objects and their confidence

The third, fourth step are described in chapter 5.2 and the sixth step is been described also in chapter 5.1.

CHAPTER

5 IMPLEMENTATION

This chapter will address in detail some aspects of the product implementation where the Design chapter did not. Also, encountered issues will be addressed and how they were tackled with the current implementation. Lastly, a review of the overall implementation will be performed in regards of the fulfillment of the product's requirements.

5.1 Image annotation

In order to perform the image annotation task on the presentation layer a JQuery plugin was used called jQuery-select-areas, as there was no alternative for typescript [2]. Such solution is not optimal as it requires to load first JQuery as global library and second the interaction with Angular can be performed in the lower level in Javascript. Although, this is not optimal, another solution would be to recreate the library into a typescript library, which is not the case.

In order to use the library, it must be done through the declared variable as JQuery object. The initialization of the annotation tool for an image starts when an image appears in the "image-annotation" component when on *(load)* it calls the function initAnnotation with argument the image as object as shown in Listing 5.1. Starting from line 7 the library is being destroyed and reset again on the next line when a new image is loaded. Next the saved annotations on the data layer are being called and used to recreate if already created annotations exist by calling

jQuery(img).selectAreas('add', areaOptions);

The *areaOptions* is an array of representations with annotations loaded to the library. Doing so it automatically displays the annotations on top of the image. On line 30 the library is been instantiated to be called every time an annotation is changed even though the function is not been called. That is because the *jQuery* variable is declared global. When the User selects an annotations this function returns a promise which ID of the annotations is selected. The reason for doing so is because the library has an array that controls the existing annotations on the image but there was no way to interfere with that array, so I created a duplicate of it and always knew which area id is selected. Important aspect of this function for later use is the image width and height, as also the visualization ratio based of the original dimensions of the image. Storing these measurements is important because when the annotations

are stored in the database, they are stored as measurements of the original size of the image, not the representational which in every case can be different. The same applies for the visualization as it is the reverse process, from original dimensions to representational.

Listing 5.1: Initialization of JQuery-select-areas on image

```
initAnnotation(img) {
    this.currentIMG = img;
    this.imageOriginalWidth = img.naturalWidth;
    this.imageOriginalHeight = img.naturalHeight;
    this.ratioX = this.imageOriginalWidth / img.width;
    this.ratioY = this.imageOriginalHeight / img.height;
    jQuery(img).selectAreas('destroy');
    jQuery(img).selectAreas('reset');
    this.loadListOfAnnotations(this.imageName, img).then((imageAnnotations) => {
      this.annotations = imageAnnotations;
    });
    jQuery(img).on('changed', (event, id, areas) => {
      return new Promise((resolve, reject) => {
        resolve(id);
        reject(0);
      }).then(() => {
        this.currentAreaId = id;
        this.areas = areas;
      }).catch(() => {
        this.currentAreaId = 0;
      });
    });
}
```

Continuing with the functionality for storing the annotation from the image into the data layer. As said when a User selects an annotation the ID is stored and from the UI the User can click on the "annotate" button to save the annotation. In that case the annotate function is called as shown in Listing 5.2. As already described about the inconsistencies that are caused from using a JQuery library, the first 5 line are trying to match the current ID of the library with the current ID of the created annotation array. Those IDs could be different because the library ID will always start from zero and increment, but the stored array due to deletions might not have the zero key of the array and start from one.

Next is the creation of the annotation array object with the properties for recreating it upon retrieval from the database. As we can see the areas array has been used, which is the library's array with the matched ID that was found in the beginning. The x,y,width and height are being multiplied with the ratio for storing the original placement in pixels. Lastly, by searching if the runtime stored annotations array length is zero or it includes the currentAreaId, which is the selected annotation, we update or create a new annotation in the database.

Listing 5.2: Saving the annotation into the database

```
annotate() {
    for (let i = 0; i < this.areas.length; i++) {
      if (this.areas[i].id === this.currentAreaId) {
        this.annotationsId = i;
      }
    }
    jQuery(this.currentIMG).selectAreas('setTag', this.currentAreaId,
        this.annotationClass);
    const annotationObject: Annotations = {
      id: this.currentAreaId,
```

```
10          class: this.annotationClass,
11          x: (this.areas[this.annotationsId].x * this.ratioX),
12          y: (this.areas[this.annotationsId].y * this.ratioY),
13          width: (this.areas[this.annotationsId].width * this.ratioX),
14          height: (this.areas[this.annotationsId].height * this.ratioY),
15          imgHeight: this.imageOriginalHeight,
16          imgWidth: this.imageOriginalWidth
17        };
18        if (!this.annotations.some(e => e.id === this.currentAreaId) ||
              this.annotations.length === 0) {
19          this.annotations.push(annotationObject);
20          this.imageAnnotationService.newAnnotationAPI(
21            this.annotations[this.annotations.findIndex( e => e.id ===
                  this.currentAreaId)],
22            this.imageName);
23        } else {
24          this.annotations[this.annotationsId] = annotationObject;
25          this.imageAnnotationService.updateAnnotationAPI(
26            this.annotations[this.currentAreaId],
27            this.imageName);
28        }
29    }
```

5.2 Invoking Python

As it is already mentioned in the Analysis chapter, Python is the most prominent language for developing ML solutions. For that reason the implementation also of Yolov2 is in Python, leading us with the problem of finding a solution on how to invoke Python scripts through Node.js. In order to solve this problem a native solution was used from Node.js called **Child Processes** [8].

The requirements that must be met from this functionality is, passing arguments to the child process and being able to read the child process output in real time. For those reasons, the **Spawn Child Process** is used as the other child process options does not include real time output by default.

Listing 5.3: Invoking Yolov2 for prediction

```
1   const buffer = new Array<any>(imageData);
2   const configFile = WorkspaceSingleton.getInstance().currentProject +
        CONFIG_FILE;
3   const weights = await projectWeights();
4   const predictResult: Promise<any> = new Promise<any>((resolve, reject) => {
5       const child = childProcess.spawn(`${YOLO}/external_process.sh`,
6           ['-n', 'predict', '-c', `${path.resolve(configFile)}`, '-p', `${YOLO}`,
                '-w', `${weights}`, '-i', `${buffer}`]);
7       child.stdout.on('data', (data) => {
8           resolve(this.parseDetections(data.toString()));
9       });
10      child.stderr.on('data', (data) => {
11          console.log(`stderr: ${data}`);
12      });
13  });
14  return predictResult;
```

Listing 5.3 shows how the "detector.client" in the application layer creates a process of Yolov2 for predicting an image. The input of this function is the *imageData* as buffer. Following the configuration file of the project and the weights currently available (either as lower validated loss or highest mAP if exists) as absolute path

strings respectively. In order to ensure that a response will follow after the prediction a Promise is used for the invoked process. Yolov2 is not being directly called, as it requires a special environment for fulfilling each requirements. For that reason a *Bash* script was created that activates the required environment, see Listing 5.4.

Listing 5.4: Invoking Python scripts through bash script

```
#!/bin/bash
POSITIONAL=()
while [[ $# -gt 0 ]]
do
key="$1"
case $key in
        -l|--logs)
    LOGS="$2"
    shift # past argument
    shift # past value
    ;;
        -p|--yolo)
    YOLO="$2"
    shift # past argument
    shift # past value
    ;;
    -c|--config)
    CONFIG="$2"
    shift # past argument
    shift # past value
    ;;
    -w|--weights)
    WEIGHTS="$2"
    shift # past argument
    shift # past value
    ;;
    -i|--image)
    IMAGE="$2"
    shift # past argument
    shift # past value
    ;;
    -n|--name)
    NAME="$2"
    shift # past argument
    shift # past value
    ;;
    --default)
    DEFAULT=YES
    shift # past argument
    ;;
    *)    # unknown option
    POSITIONAL+=("$1") # save it in an array for later
    shift # past argument
    ;;
esac
done
set -- "${POSITIONAL[@]}" # restore positional parameters
if [ $NAME == 'train' ]; then
        /bin/zsh -c ". /anaconda3/bin/activate tensor; exec python 
            ${YOLO}/train.py -c ${CONFIG}"
elif [ $NAME == 'predict' ]; then
        /bin/zsh -c ". /anaconda3/bin/activate tensor; exec python 
            ${YOLO}/predict.py -c ${CONFIG} -w ${WEIGHTS} -i ${IMAGE}"
elif [ $NAME == 'tensorboard' ]; then
        /bin/zsh -c ". /anaconda3/bin/activate tensor; exec tensorboard 
            --logdir ${LOGS}"
else
        echo "No fileName man!!"
```

The script starts with the input arguments that have been invoked and as we see in Listing 5.3, line 6 there are 6 parameters as arguments. Use Yolov2 for prediction, the absolute path of the config file, the absolute path of Yolov2 folder, the weights and the image for prediction. Then the **Bash** script calls the appropriate function based on the arguments and creates a call in an activated environment shell. In our case that is an Anaconda environment [3][28]. The output of the invoked script for the prediction is caught by the function parseDetections with data as string, for parsing the output buffer to an array of annotations and return it back to the application layer.

In the Training module the invocation of Yolov2 is also used , as also Tensorboad. Both have such an option for that in the **Bash** script. The difference though, with invoking Yolov2 for prediction or training is that the later might need to stop on demand. The same applies for the Tensorboard as it would like to resource consumption. For that reason an implementation is created where, whenever a process is being spawned a *process ID (PID)* is assigned to it as an identification. The application layer though does not recognize which process was created before, as it might be overwritten as value by another invocation. The solution for this problem is to return to the presentation layer the PID, as a response and stop the training or the Tensorboard instance on demand by this PID.

5.3 Tensorflow Serving

One of the most significant issues I encountered is in regards of the Testing module. Originally, for the prediction Tensorflow Serving (TFS) would be used. "TensorFlow Serving is a flexible, high-performance serving system for machine learning models, designed for production environments." as of each information web page [43]. The main idea is that any Tensorflow model can be deployed for production, meaning fast predictions because the model weights have already been exported as variables and invoked from TFS.

As of this product, it would be used on top of the existed REST API for predictions. It must be stated as the information web page says "out-of-the-box integration with TensorFlow models, but can be easily extended to serve other types of models and data"[43]. At this state using Yolov2 as the selected ML algorithm to solve the problem statement with TFS had none unsolvable problem. Problems such as the created weights should be transformed to TFS format or removing the prediction layer of Yolov2, were solved after proper research.

Yolov2 though, for prediction requires some preprocessing for the input image, which leads to the unsolvable problem. As of the original decision for using Node.js for the application layer, it was not possible (with the time constraints) to perform the required preprocessing using typescript. Therefore, TFS was not used and in place was implemented the Python invocation also for the prediction as described in section 5.2.

5.4 Requirements fulfillment

From the specified requirements in chapter 3 based on the MoSCoW prioritization technique, all the User Stories and NonFunctional Requirements assigned as Must were fulfilled. Partially though, for two User Stories they were not fully implemented. For testing a trained model it was only implemented for using an image and it was also required for videos. Also the import functionality of a dataset or a model is not implemented. Both of the missing Must requirements though can be considered as supplementary features as it was required as an extra option of the main requirement. Lastly, the development included some Should and Could features to complement Must features.

CHAPTER

6 EXPERIMENTAL VALIDATION

In this chapter the platform will be assessed on its performance and how well it answers the hypothesis. The experimental validation falls under the Measure step of "The feedback loop" methodology. Based on this, two assessments have been made, one for creating a model using the platform and the other for analyzing the feedback from UX test interviews. Lastly, the Learn step will be followed based on the Measure step findings.

6.1 Model creation via platform

For the assessment of the platform a model should be created, from creating the dataset up to testing the performance of the model. Initially, as Ecobotix been used as a test case scenario, the problem solving for object detection is related. Although, due to some inconsistencies in the company the dataset for creating the model was never created. Therefore, I decided to create a model for detecting me as face using Yolov2 algorithm.

For the dataset 478 images used and annotated with one class as "lazaros". The images include myself in different lighting environments as also facial expressions. Also, the selected images include other people in order for the algorithm not to converge very fast. The training and validation set was automatically created as 406 and 72 respectively. For the training different configurations were used in order to find the most optimal.

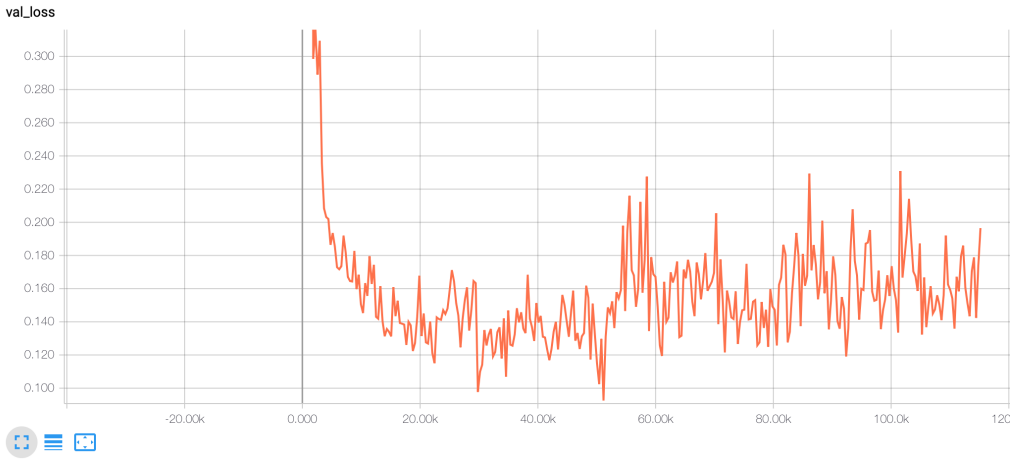
The results were inconsistent without being able to find any correlation on each model performance. It seems that the model was overfitting right from the start of the training, as the validated loss was getting values below 0.1 pretty fast, see Figure 6.1. This is an indication that the dataset is small and the model has no further training to do.

The created model's performance could be characterized as mediocre. The best performed model had mAP 0.2, with validated accuracy 0.602 and validated loss 0.041. It must be stated that the performance of the model is inconsistent, as the detections could be wrong as for example detecting plants as me or detecting another faces as also me. Most of the times though the model would detect me on an image, but at the same time it would also detect something else as me.

After some research, I realized that face detection is a sensitive category in ML, as it requires further development to detect facial features on top of an existing ML algorithm. Furthermore, from testing the platform a lot of features were missing for improving my workload and to make it easier. Such features are cross-fold validation,

further configurations for the algorithm, live observation of the training phase and better connection between the application layer with the presentation layer.

Figure 6.1: Example of overfitting



6.2 Qualitative UX interviews

The survey was conducted through preformed questionnaire evaluating various aspects of the platform and its processes, followed by a short interview of the people involved for better understanding of the feedback provided. The objective of this survey is to gain knowledge by analyzing others' user experience when using the platform. The survey is performed on the Ecobotix team which includes experienced and non experienced personnel in ML.

The survey started after the surveyed people were asked to create a new project,use the platform to upload photos for the training process and create new annotations as classes. The platform allowed them to select the desired configuration, to start and stop the training process at any given moment, while observing the statistics depicted by the graphs.

The Ecobotix members taking part in the survey, provided useful data for better improvement and evolving of the platform, as it is worth noting that people with no experience on AI, gave uniform answers to the ones that had experience. The platform enabled them to use it as efficiently as possible. Positive responses were given to the ease of use of the user interface to complete any task, while all of them, found the overall functionality efficient for the purpose of the MVP presented. All three processes, annotation, training and inference were rated highly, although training was rated the lowest.

Some of the comments were related to the GUI and possible additions that could be made, such as feedback to the user regarding the training process or the confidence levels while training. Beneficial, according to the answers, would also be the addition of extra parameters to the algorithm's configuration, more user outputs or examples of use. Overall, the designed purpose of the platform was understandable and clear to the users in regards of the MVP state.

Finally, only one negative answer was given for increasing the productivity and the data indicate it came from the most experienced ML user. On the contrary, the rest of the users saw an increase to their productivity using the platform.

6.3 Learning phase

As the last part of the **Build, Measure, Learn** loop the Learn step includes the decision whether the project must pivot or preserve the MVP and continue developing the rest of the requirements. The review of the Measure step is being separated into two categories, interaction with the platform and ML model creation effectiveness.

Although, the sample size is small it provides great feedback how well the UI is designed, which is part of the problem statement to provide a solution for easy interaction with ML tasks. Of course there were some problems that cause inconsistency in the development flow of a model, but the overview is that the UI is supporting well the ML development. With that said the first category can be flagged as preserve and move to further requirements development related to UI.

On the other hand though, the creation of a model could not be characterized as sufficient. Of course we must take into consideration that the product is in MVP state, but that does not mean that conclusions can not be drawn. The lack of certain functionalities in regards of the model creation led to under-performing models. That is understandable as certain ML model types require certain functionalities and the created model required further facial recognition capabilities. That bad selection of problem provides an insight of the general notion.

Building a UI to support ML development can be combined, but up to a certain level. If the ML development will be focused on a specific task and only that then product can be characterized as preserve. If though the ML development would be more broad then the product must pivot and incorporate further, more essential capabilities.

CHAPTER

7 DISCUSSION

This report is originated by my experience utilizing Machine Learning algorithms and the struggle of using them. It is not only a hard task to understand them, but also to implement a ML model that actually does perform well. My research started with Ecobotix. As a small team with minimum working hours for the student workers, i wanted to deliver a service to the Ecobotix product that includes a ML model. As I did not want to spend a lot of hours on a project that is not part of my job, I searched for ML development platforms, that would make my life easier. Unfortunately, I was unsuccessful, but it led me to formulate and choose this as my Master Thesis.

The proposed solution from the beginning was hard to address, as from one side the goal of this Thesis is not to create a ML algorithm, but on the other side I needed to use such an algorithm into my platform. This led me into the choice of using the algorithm in each most supported language, which is Python and if needed make changes into the implementation.

The selection of desktop or web application had also some complications in relation of which language should be selected. As a desktop application it would require a lot more development to conclude into what is achieved as web application in such a short time. It must be taken into consideration that the available time for developing the product was not enough for a desktop application.

The selection of using Angular and Node.js is based also on the fact that they both are written in the same language and would greatly improve my development speed, as also my capabilities in Typescript. Although, as of the Experimental Validation it would be better to use Python for the back-end, rather than Node.js as it would be the same language as the algorithm implementation.

Lastly, by addressing the problem's requirements I concluded into answering my initial Hypothesis. Based on the Hypothesis and my conclusions, the answer can only come into two conditions. The performed solution addresses on solving a specific task that was researched and identified as object detection. There were some inconsistencies along the way, but the conclusion is that it can increase the performance of the ML development, if the task is specific into one ML area. If otherwise, the Hypothesis answer is that a product can not fully address broaden areas of ML using UI. Of course, such a statement can be flipped if the product is developed by a huge team and the team incorporate fully support for various ML problem areas.

CHAPTER

8 CONCLUSION

This report addresses how Machine Learning algorithms are being utilized nowadays.

In the Analysis it focuses on identifying the underlying problem by formulating the Problem Statement and creating a Hypothesis that would try to solve the problem.

"If a GUI workbench related to ML existed, then it would improve the performance of ML developers, experienced or not."

In order to validate the Hypothesis a certain methodology was selected, as **Build, Measure and Learn**. For the proposed solution that would try to validate the Hypothesis a Minimal Viable Product is identified in correlation with **Ecobotix ApS** as test case scenario. The solution addresses the problem by creating a Graphical User Interface for training and testing a ML algorithm without the need of writing code. Furthermore, related products/services have been found in order to have a better understanding of the missing gap. Lastly, by combining the purpose of Ecobotix as company and how a ML model could be used for the company's benefit, an object detection algorithm as ML type was selected, using **Yolov2** as the most prominent solution for the matter.

For the Requirements a set of User Stories and Nonfunctional requirements have been mapped that would try to solve the Hypothesis by creating a product that follows these requirements. The requirements have been prioritized based on MoSCoW method for creating the MVP.

In the Design chapter the primary task is to identify the proper architecture and analyze it based on the proposed solution. As the type of the application, a **Web Application** was selected with the **3-Tier Architecture**. Next, the analysis of each layer of the 3-Tier architecture was performed and provide an estimation of what would be included on each layer. Also, a generalization of the communication data flow was performed in order to understand how data are flow between Presentation layer and Application layer. Lastly, the conceptualization of the overall design in detail is performed, oriented by Yolov2's three steps of development, creating a dataset, train the model and test the performance of the created model.

The Implementation chapter address issues and important decisions that were taken during the product development. These decisions refer to the dataset creation and the ML algorithm invocation. Also, related to the testing process of Yolov2 an issue is addressed of how the original decision change to the current implementa-

tion. Lastly, the requirements fulfillment is addressed, stating the requirements level of completion.

Lastly, the Experimental Validation falls under the **Measure and Learn** steps of the selected methodology. Two performed measurements were made to validate the proposed solution, by creating a ML model using the platform and by qualitative User Experience interviews with the Ecobotix team. In the end of the chapter the learning step is addressed which lead us to the answer of the Hypothesis.

Based on the **Learn** outcome and the Hypothesis, the answer of the Hypothesis must be separated into two parts.

"The Hypothesis has been confirmed, but in one condition. In order to address such a problem, the solution must be specific into on machine learning target area."

CHAPTER

9 FUTURE WORK

Future work of the product include the stated requirements. Must be taken into consideration though whether the product will pivot into something broaden or preserve and focus on the development of a specific case of ML. Nevertheless, additional required features can be identified in either case.

9.1 Back-end, Front-end interactions

Some parts of the product require better visualization and representation of what is happening in the background. For example during prediction process represent into the UI that the back-end is busy and the UI is waiting for a response.

9.2 Implementing sockets

During the training process, after the experimental validation it was seen that users wanted to know live what is happening with the algorithm and not only from the Tensorboard metrics visualizations. Such a case requires socket for constant connection.

9.3 Tensorflow Serving

A feature as Tensorflow Server will provide extra value for such a tool, as the user will be able it with the REST API for production, as it is fast.

9.4 Dockerize the product

As of now some parts of the product must be done manually and this increases the complexity. Such a problem could be solved by making the product one-click execution using docker as solution.

BIBLIOGRAPHY

[1] URL: https://innovationsfonden.dk/en/programmes/innobooster.

[2] 360Learning. *360Learning/jquery-select-areas*. Dec. 2016. URL: https://github.com/360Learning/jquery-select-areas.

[3] Anaconda. *Home*. Dec. 2018. URL: https://www.anaconda.com/.

[4] *Angular*. URL: https://v2.angular.io/docs/ts/latest/guide/architecture.html.

[5] *Application programming interface*. Dec. 2018. URL: https://en.wikipedia.org/wiki/Application_programming_interface.

[6] *Azure Machine Learning service*. URL: https://azure.microsoft.com/en-us/services/machine-learning-service/.

[7] Phil Britt. *How AI-Assisted Surgery Is Improving Patient Outcomes*. June 2018. URL: https://www.roboticsbusinessreview.com/health-medical/ai-assisted-surgery-improves-patient-outcomes/.

[8] Samer Buna. *Node.js Child Processes: Everything you need to know*. June 2017. URL: https://medium.freecodecamp.org/node-js-child-processes-everything-you-need-to-know-e69498fe970a.

[9] *Caffe*. URL: http://caffe.berkeleyvision.org/.

[10] *Computer vision*. Dec. 2018. URL: https://en.wikipedia.org/wiki/Computer_vision.

[11] *Create, read, update and delete*. Nov. 2018. URL: https://en.wikipedia.org/wiki/Create,_read,_update_and_delete.

[12] *Deep Cognition*. URL: https://deepcognition.ai/.

[13] *Deep Learning Made Simple*. URL: https://lobe.ai/.

[14] *Dependency injection*. Dec. 2018. URL: https://en.wikipedia.org/wiki/Dependency_injection.

[15] M. Everingham et al. "The Pascal Visual Object Classes Challenge: A Retrospective". In: *International Journal of Computer Vision* 111.1 (Jan. 2015), pp. 98–136.

[16] *Express - Node.js web application framework*. URL: https://expressjs.com/.

[17] Eric Feuilleaubois. *Making Deep Learning User-Friendly, Possible? – Towards Data Science*. Apr. 2018. URL: https://towardsdatascience.com/making-deep-learning-user-friendly-possible-8fe3c1220f9.

[18] *Frame rate*. Dec. 2018. URL: https://en.wikipedia.org/wiki/Frame_rate.

[19] Suhasini Gadam. *Artificial Intelligence and Autonomous Vehicles – Data Driven Investor – Medium*. Apr. 2018. URL: https://medium.com/datadriveninvestor/artificial-intelligence-and-autonomous-vehicles-ae877feb6cd2.

[20] Yael Gavish. *Developing a Machine Learning Model From Start to Finish*. July 2017. URL: https://medium.com/@yaelg/product-manager-guide-part-3-developing-a-machine-learning-model-from-start-to-finish-c3e12fd835e4.

[21] *Google AI*. URL: https://ai.google/.

[22] *Green Development and Demonstration Programme*. URL: https://innovayt.eu/gudp/.

[23] *Information retrieval*. URL: https://en.wikipedia.org/w/index.php?title=Information_retrieval&oldid=793358396#Average_precision.

[24] Remo H. Jansen. *InversifyJS*. URL: http://inversify.io/.

[25] *Keras: The Python Deep Learning library*. URL: https://keras.io/.

[26] *Lightweight javascript in-memory database: LokiJS*. URL: http://lokijs.org/.

[27] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.

[28] *Managing environments*. URL: https://conda.io/docs/user-guide/tasks/manage-environments.html.

[29] *Minimum viable product*. Nov. 2018. URL: https://en.wikipedia.org/wiki/Minimum_viable_product.

[30] *MoSCoW method*. Sept. 2018. URL: https://en.wikipedia.org/wiki/MoSCoW_method.

[31] *Multitier architecture*. Dec. 2018. URL: https://en.wikipedia.org/wiki/Multitier_architecture.

[32] *Object detection*. Nov. 2018. URL: https://en.wikipedia.org/wiki/Object_detection.

[33] *Observer pattern*. Dec. 2018. URL: https://en.wikipedia.org/wiki/Observer_pattern.

[34] Arthur Ouaknine. *Review of Deep Learning Algorithms for Object Detection*. Feb. 2018. URL: https://medium.com/comet-app/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852.

[35] *PyTorch*. URL: https://pytorch.org/.

[36] Joseph Redmon and Ali Farhadi. "YOLO9000: Better, Faster, Stronger". In: *arXiv preprint arXiv:1612.08242* (2016).

[37] *Representational State Transfer*. June 2016. URL: https://fr.wikipedia.org/wiki/Representational_State_Transfer.

[38] Eric Ries. *The lean startup: how todays entrepreneurs use continuous innovation to create radically successful business*. Crown Business, 2011.

[39] rodrigo2019. *keras-yolo2*. Nov. 2018. URL: https://github.com/rodrigo2019/keras-yolo2.

[40] *Separation of concerns*. Apr. 2018. URL: https://en.wikipedia.org/wiki/Separation_of_concerns.

[41] *Singleton pattern*. Dec. 2018. URL: https://en.wikipedia.org/wiki/Singleton_pattern.

[42] *TensorFlow*. URL: https://www.tensorflow.org/.

[43] *TensorFlow Serving | TensorFlow*. URL: https://www.tensorflow.org/serving/.

[44] *The Emergence of the Age of AI*. Aug. 2018. URL: https://www.bbvaopenmind.com/en/the-emergence-of-the-age-of-ai/.

[45] *Types of Machine Learning Algorithms You Should Know*. June 2017. URL: https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861.

[46] *Unlock deeper learning with the new Microsoft Cognitive Toolkit*. URL: https://www.microsoft.com/en-us/cognitive-toolkit/.

[47] *Visual Machine Learning and Modeling in Dataiku*. URL: https://www.dataiku.com/dss/features/machine-learning/.

List of Figures

APPENDIX

9.5 Survey

Questions

1. Do you have experience in AI field?
2. Do you find the idea of developing an AI model through Graphical User Interface useful?
3. How would you rate the difficulty of using an AI algorithm?
4. How would you rate the difficulty of the User Interface to complete any task?
5. How would you rate the annotation process?
6. How would you rate the training process?
7. How would you rate the inference process?
8. Based on the concept of MVP do you find efficient the overall functionality?
9. Based on the completion of this product as MVP, describe what would be beneficial to be included or changed.
10. Would you use such a tool to increase your productivity?

Responses

1. Four responded YES and two NO
2. Four responded YES and two NO
3. From low to high scaling, two responded three, three responded four and one responded five
4. From low to high scaling, two responded three, three responded four and one responded five, one responded one, four responded two and one responded three
5. From low to high scaling, one responded four and five responded five

6. From low to high scaling, two responded three, three responded four and one responded five
7. From low to high scaling, two responded four and four responded five
8. Everybody responded YES
9. Text responses

When the user presses a button and it takes a minute for the UI to react e.g. to start up Tensorboard, the user needs to know that something is happening in the background. So in general, more outputs to the user in these cases.
Perhaps adding the possibility for more advanced model manipulation, also feedback to the user regarding the training stage or confidence levels while training.
option for selection of second or third algorithm, what is Yolo, tool tips, clarification of which algorithm is being used, the purpose of the software / GUI, case studies, examples of use, future use cases and planned suggested future works in terms of development.
As inexperienced in Machine Learning i can't propose any changes. Though the use of the platform is sufficient and understandable in what is suppose to do. It seems that it covers a part of the problem though.
The configuration of the algorithm for training the model seems to be mediocre as in some cases it needs additional parameters or even change the algorithm based on the corresponded task.

10. Four responded YES, one MAYBE and one NO