Università
della
Svizzera
italiana

Faculty
of Informatics

Bachelor Thesis

January 28, 2020

AITIA

Artificial Intelligence Tool for Image Augmentation

Sergiu Gheorghita

Abstract

In this report, we propose to use Artificial Intelligence techniques and in particular Neural Networks to augment an image dataset by predicting relevant features, in the form of tags to be set by users, for new unseen images. To do so, we start with using image segmentation and image classification NNs on existing images and we retrain them on our custom dataset. In order to create our custom dataset, we build a Web based application which facilitates image capturing, labelling, storing and retrieving an image dataset.The tool enables children to browse an image dataset, augment it with tags and then observe how they can help AI to learn from tags they did and make more precise predictions on new images added to the same dataset. The aim of this report is to deliver a simple and effective game like Web application to teach AI's principles to primary school children. The key contribution of the application is to afford non-technical users the possibility of using and exploiting advanced state-of-the-art artificial intelligence methods. We make this possible such that a user does not have to write any code to use this powerful AI.

Advisor
Prof. Monica Landoni
Assistant
Andrea Gallidabino

Advisor's approval (Prof. Monica Landoni): Date:

Acknowledgements

I would like to express my special thanks of gratitude to my Advisor Monica Landoni for always helping me out when needed. I want to give my special thanks to Andrea Gallidabino, the co-advisor who was always to keep me down to earth and spurred me at the same time. I'm really grateful and glad for working with this team, they literally made me grow during this period. I want to give my special thanks to professor Mauro Pezzè, this bachelor project experience is something I will keep close to me forever, it made me realize all the dynamics concerning a project, the report, and how they work together. I'm grateful to my friends who supported me during the developing and reporting time. Thanks to Lorenzo Ferri, from who I learned almost everything I know in terms of full-stack development. Thanks to Manav Choudhary, the person who witnessed with me the moment of the first results of AITIA and YOLO combined. He suggested me to use YOLOv4 Tiny and that changed a lot of things. Thanks to Lazar Najdenov, Vlad Scobioala, and Andrea Perozziello who supported in all the ways they could. Thanks to Lorenzo Verniaghi, Alessio Giovagnini and all the people who helped me in some way to grow during these years. I'm glad for being a student of USI Informatics, I feel lucky. Thanks to my Family who always pushes me and thanks to them in particularly because I was able to access USI university.

Contents

1 Application Architecture **5**
1.1 Application Technologies . . . 5
1.2 The Web Server . . . 6
1.2.1 Web Server Architecture . . . 6
1.2.2 Web API Routing . . . 6
1.2.3 Controller . . . 6
1.2.4 Models . . . 7
1.3 The Web Application . . . 7
1.3.1 Frontend Architecture . . . 7
1.3.2 Services . . . 7
1.3.3 React Components . . . 7
1.4 The Database . . . 8
1.5 The Disk Storage . . . 8
1.6 YOLO - FCNN . . . 9
1.6.1 YOLOV4 Tiny . . . 10

2 Application Tour and Sub-Processes **11**
2.1 Trailer . . . 12
2.1.1 Trailer Technologies . . . 12
2.1.2 Trailer Architecture . . . 12
2.2 HomePage . . . 13
2.2.1 Resources Recovery and Polishing . . . 14
2.3 Videos Interface . . . 15
2.3.1 Videos Upload - Backend process . . . 15
2.3.2 Video Fragmentation . . . 16
2.4 Collections Interface . . . 17
2.5 Dataset Interface . . . 18
2.5.1 Image Selection Process . . . 19
2.5.2 Data Categorization and Dataset Initilization . . . 20
2.5.3 Datasets and Categories . . . 21
2.5.4 Multiple datasets . . . 22
2.6 Multiple Classification . . . 23

3 User Evaluation and Feedback **24**
3.1 observational study architecture . . . 24
3.2 observation results . . . 25
3.3 Discussion . . . 26

4 YOLO Artificial Intelligence Model **27**
4.1 YOLO - Input Dataset paramenters . . . 27
4.2 YOLO - training . . . 28
4.3 Experimental Results . . . 28
4.4 YOLO - testing . . . 29

5 The State of the Art **31**
5.1 Desktop GUIs . . . 31
5.2 Web Applications . . . 32

6 Future Work and Goals **33**
6.1 Full Ecosystem and Native Application . . . 33
6.2 User Friendly . . . 33
6.3 Test Driven Developing . . . 34
6.4 Retrieve datasets online . . . 34
6.5 Dataset sharing and user population . . . 34

7 Possible Applications **35**
7.1 Stereotypes detection on digital artifacts 35
7.2 Disorders detection 35
7.3 Courses for kids 35
7.4 Labelling Tool for training CNNs 35
7.5 Third Party Devices 35

Introduction

Artificial intelligence nowadays represents the implementation of software with the power to learn human abilities, flavors, and preferences to assist them during everyday life. We are still far away from the true definition of an Artificial Intelligence (AI), i.e., a system that can have a creative conversation with humans and can think and develop new ideas. Nowadays most of the world's population uses Artificial Intelligence in their routine without even knowing. AIs solve different complex tasks in an efficient way and are already used in multiple industries such as healthcare, entertainment, finance, education, data analysis, data security, gaming, social media, driving, etc. We can use it for object recognition as we do in this report. Object recognition is the combination of two powerful components also known as Image Classification (labeling) and object localization (bounding box). There are already several CNN models nowadays, but the one we used to test if this application is possible is called YOLO (You Look Only Once), an open-source CNN. However, it is not simple, or straightforward to configure, train and use it. Hence, our report bridges the gap between powerful object recognition AI and non-technical users. An important assumption that we made before starting was, if we can make primary and middle school children understand the full ecosystem behind a CNN model, then we can assume it will also work for all users, technical and non-technical. Artificial Intelligence is no longer only for big companies and technical users but gives a starting point, a top-down approach and an idea about AI and Machine Learning. On this assumption, we have to develop the application constantly thinking about our target and his interaction with our tool. The technologies we used to implement AITIA are Express TypeScript for the backend Web server, JavaScript and React for the frontend Web application, MongoDB database to store all the information needed, a static storage which contains all the datasets for the training tasks and all the media files used for producing datasets. On top of that, we will try to insert an object detection neural network inside this environment to see if the tool works, if we can proceed on this direction, if it is user friendly, and if kids are attracted by the tool itself. AITIA is structured in 4 steps or pages (up-to-date): trailer and introduction, resources retrieval, and DataSet Building. Output Data Visualization Interface is still missing as are the training interface and the testing Interface. The The Introduction trailer was made with Movavi Video Editor Plus 2020, is divided into 4 main steps: Introduction, what Artificial Intelligence means nowadays, YOLO introduction and characteristics (labeling, bounding box, real-time), use all of these features on cartoons like Shrek. The This video approach is made for a specific target, children of primary and middle school, since the application will be tested as a course for primary and middle school students. We will be able to make this course thanks to Ated4Kids ITC Ticino which beliefs in the same idea as us: children are the main characters and active players of the digital revolution happening right now. Our goal is to let children express themselves in this new upcoming world and then as a side effect previous generations will also benefit from AITIA.

In section number 1 we present our application and its architecture, section 2 is in charge of the application tour and subprocesses, the 3^{rd} section shows the observational study we did on children to test our app. Section 4 has as central topic YOLO, dataset, training, and testing. Section 5 compares our application with existing, the 6^{th} is represented by the future work and goals. The last chapter argues about possible use cases of AITIA now and in the future.

1 Application Architecture

Our app consists of 5 main components. The first is called Web server, responsible for all the interactions between the other components. The second component is the Web Application Interface, responsible for the User interaction within the application. The third component, the object-oriented database is in charge of storing objects created by the user. The fourth is the disk storage within the app, which is the "physical" storage in our file system. The fifth component is the Convolution Neural Network whose duty is object Recognition.

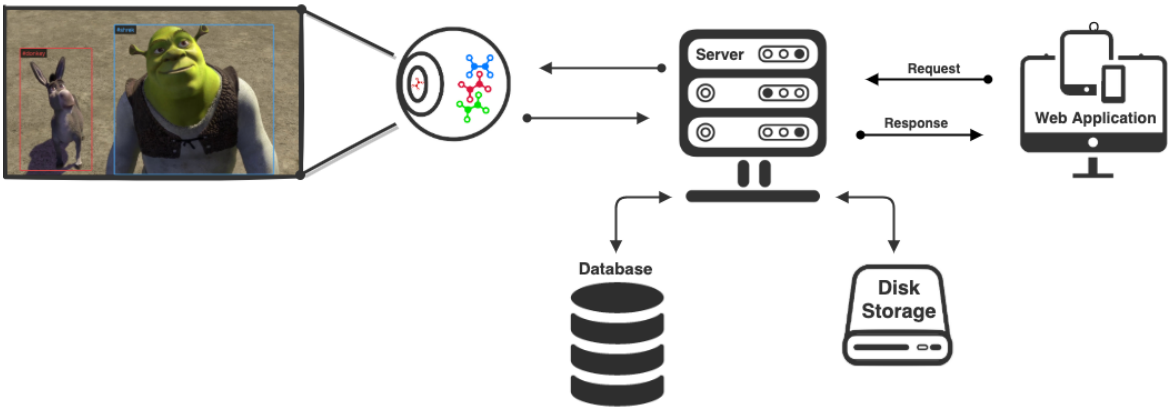

Figure 1. Application architecture.

1.1 Application Technologies

The Application uses Javascript and React for the frontend duties, in the backend architecture we used NodeJS with Express as framework and TypeScript as programming language.
The database for this architecture is MongoDB. We think that from a full-stack point of view is a good architecture, allowing you to use only one programming language. Objects created on the Web app side are stored exactly as they are in JSON format in the database. The Web Server maps all the disk storage into the database. By using JavaScript with React gives you a lot of help for the frontend implementation; in terms of resources, you can find all the components and information you need for your app. We also think using TypeScript in the frontend is better than using Javascript as we did. By definition, “TypeScript is JavaScript for application-scale development.

1.2 The Web Server

As explained above the application was developed using TypeScript as programming language which guarantees to the developer good maintainability of complex projects. Almost all the backend functions are asynchronous, allowing developers to produce promise-based code as synchronous, without blocking the main thread and making the code much easier to write and read.

1.2.1 Web Server Architecture

The backend is built on 3 main components. The first is represented by the list of express routes which maps a given request to the main component aka Controller. The second component is the Controller itself, in charge of getting the requested data from model schemas and then produces a response for the frontent. The third component, a list of model schemas, are the "blueprints" of the real data inside the database.

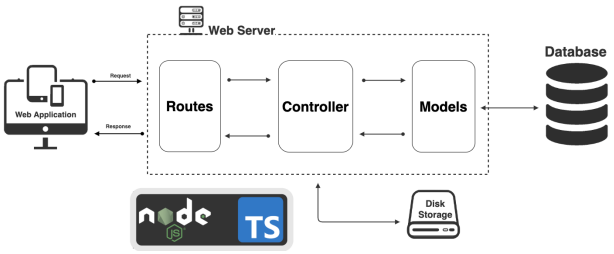

Figure 2. Web server architecture.

1.2.2 Web API Routing

Our Application Programming Interface has 5 main routes. Each one of them has the CRUD functions implemented, i.e., create, read, update, and delete. Routes are in charge to deal with all requests and responses related to a particular Object. We have 5 main objects and we provided a route for each one:

1. **Videos Route** - handles request and response for videos.
2. **Folder Route** - handles request and response for folders of images.
3. **Images Route** - handles request and response for images which will be used to train the AI.
4. **Categories Route** - handles request and response for categories which are populated by labels and each label/tag is linked to a one or more categories.
5. **Cropped Card Route** - handles request and response for cropped images which are needed to implement data visualization and to stores their location in normalized % coordinates within the full image.

1.2.3 Controller

The Controller is the brain of our application, is responsible for all tasks and calculations given a request or a response, it connects all components and makes possible the interaction between them. There are 2 macro-structures in our controller:

- The first manages the database.
- The second one manages the disk storage.

These 2 macro-structures work in parallel, each one of them is structured in 5 smaller sub-structures (video, folder, images, category, cropped card). The first one manages data virtually and the second one manages data physically.

1.2.4 Models

There are 5 model schemas in total, they represent the logic structures of the data stored inside the database. Schemas allows the application to have a well defined and structured data. The controller interacts through schemas with the database.

1.3 The Web Application

For the front end implementation, we used JavaScript and React. An important thing we want to achieve in the future is native application and React Native is one of the best JavaScript frameworks in terms of mobile apps. Hence, we have choosen React library as starting point.

1.3.1 Frontend Architecture

The frontend architecture has 3 main elements, the first is called Services which manages all the requests from the frontend to the Web Server. The second element is represented by React components, they are independent and reusable. The third element is the Web Router, which allows the user to navigate through the application and redirects React components.

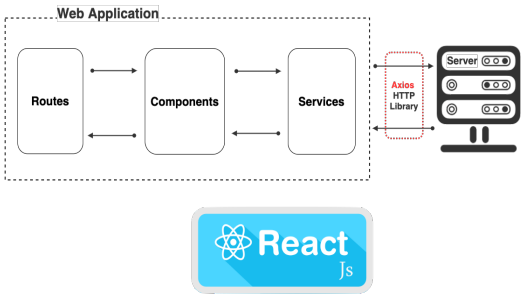

Figure 3. Frontend architecture.

1.3.2 Services

Services are responsible for invoking the API. For a clean and well-built frontend service a good choice could be to group invocation methods in one JavasScript file or few.

1.3.3 React Components

React has 2 types of components: class and function components and both are interchangeable. We used both, but we highly prefer the function component approach.

1.4 The Database

MongoDB is an object-oriented database and is quite the right choice for this kind of application which is built on top of objects and components.

Database Architecture

The database has 5 main schemas which are: videos, folders, images, categories, and cropped cards. All of them are needed for multi-dataset production.

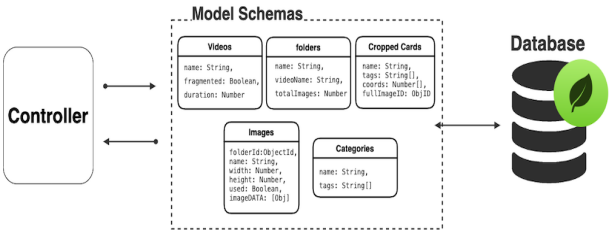

Figure 4. Database Management System graph.

1.5 The Disk Storage

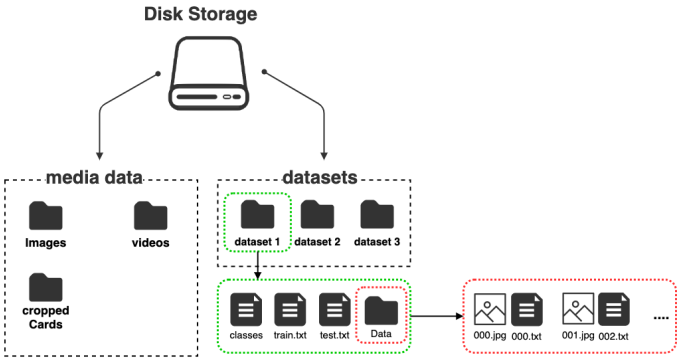

Figure 5. Disk storage graph.

Disk Storage Architecture

The phisical storage has 2 main tasks:

- The first task is to store all the media data uploaded or created by the user.
- The second task is to store datasets which are implemented gradually following YOLO input data requirements.

1.6 YOLO - FCNN

"You only look once (YOLO) is a state-of-the-art, real-time object detection system".

YOLO original paper [6]

We decided to use YOLO architecture as the Artificial Intelligence model inside the AITIA's ecosystem since is a very powerful design which achieved state-of-the-art results in object detection and classification. Also, there is a lightweight implementation of YOLO referred to as Tiny-YOLO which compromises a bit on accuracy, for huge gains in computational time. Tiny-YOLO has considerably less learnable parameters than YOLO. We can run the Tiny version on our mobile devices and, since mobile device is our direction, we considered Tiny-YOLO as the best fit model for our application.

YOLO Architecture

YOLO architecture consists of 24 convolutional layers followed by 2 fully connected layers. It uses 1 × 1 reduction layers and 3 × 3 convolutional filters in 1 layer. (6). Tiny-YOLO on the other hand, has only 9 convolutional layers.

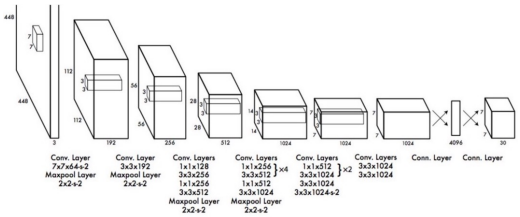

Figure 6. YOLO Architecture design, Image Credits: [6]

YOLO architecture is a fully convolutional neural network (FCNN). This neural network can be used for the following 4 tasks:

1. **Object Classification:** Classifies the recognized object within an image by labeling.
2. **Object Detection:** Classifies the recognized object within an image by bounding the object within a box.
3. **Object Recognition:** Combines the first 2, gives a bounding-box to the recognized object and also assigns a label.
4. **Object Segmentation:** Classifies every pixel within an image as belonging to a particular class. This essentially finds the exact boundary of the object and not just a bounding box around it.

YOLO takes the full image as input along with a text file which describes the coordinates of the bounding boxes present in this image and their corresponding class number. It should be noted that it does not need a cropped image of the bounding box as input. It is trained to optimize the detection performance. It formulates the object detection problem as a regression problem.

YOLO processes the entire image in both training and testing phases. It implicitly encodes contextual information about objects and their appearance. Also, it is important to note that it recognize all detected objects within the image all at once. Hence, it looks at the input image only once to make all the predictions about the image in parallel.

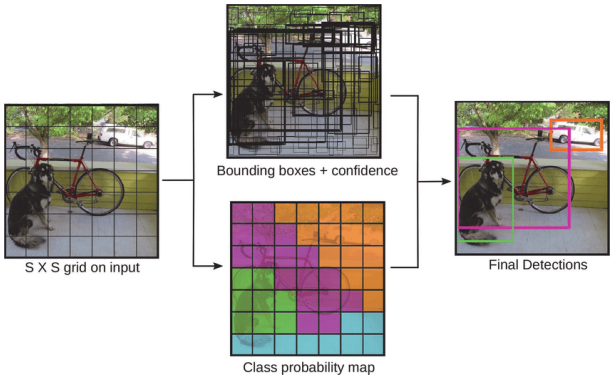

Figure 7. Regression Process, Image Credits: [6]

Detection's output

Each bounding box is defined by a tuple of 5 values: {

$$x\_center,$$
$$y\_center,$$
$$width,$$
$$height,$$
$$confidence$$

}

1.6.1 YOLOV4 Tiny

Joseph Redmon, the creator of YOLO, has also created a lightweight version called Tiny-YOLO. The memory size of this smaller version is around 20 MB, while the size of the larger version is around 250 MB. The tiny version has just 9 convolution layers in comparison to the 24 convolution layers for the larger version. The tiny version also uses the same 1×1 reduction layers and 3×3 convolutional filters. The tiny version is very fast to train and is also very fast in the inference phase compared to the larger version.

All of the above features make it an ideal model for embedded-devices such as phones, tablets, etc. During inference phase Tiny-YOLO v4 is 8 times faster than the larger version, YOLO v4. The acccuracy of Tiny-YOLO is about 2/3 of the larger version on the MS COCO dataset.

2 Application Tour and Sub-Processes

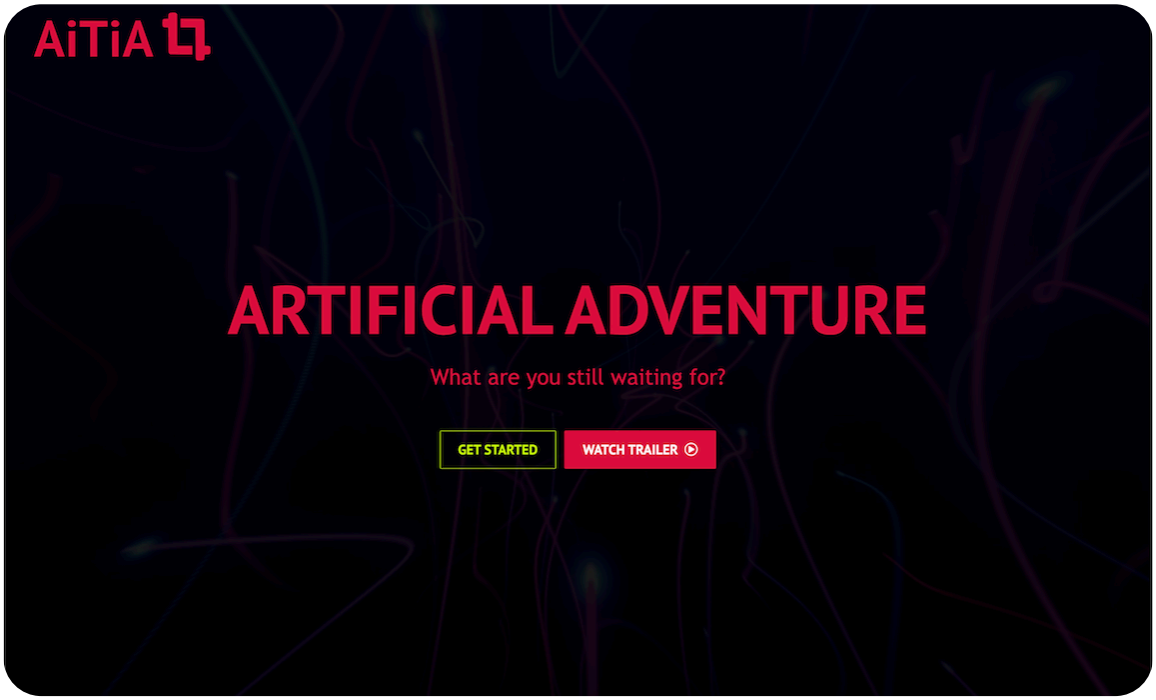

Figure 8. Welcome page.

Why a Web application?

When we started, we used a different approach and different technologies, our goal was to build a Python GUI which allows the user to produce a dataset. The GUI was implemented with the last version of Python and the main library we used to plot the images was MatPlotLib. The progress was slow and MatPlotLib is good at plotting graphs but is not for the development of complex systems which goal isn't just plotting. GUI's performance confirmed at 100% that we were using the wrong tools. Another issue was that our graphical user interface wasn't even good as other GUIs already existed and there were no benefits in that direction anymore.

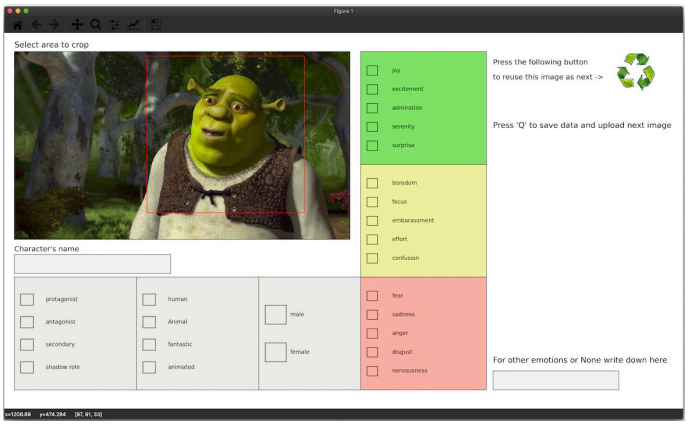

Figure 9. First version of the data-setting tool.

This was our Python GUI's result. Is not clear and nor user-friendly. There weren't big advantages for the user and in order to use it the client has to download it from Github and configure some parameters. In order to accomplish the user-friendly goal the Web app was the best solution.

Another simple and important reason is that within a Web app we can keep a clean environment for user interaction, he doesn't have to download the app and doesn't even have any configuration tasks. To enable the user to save his datasets online and to share them with other users we understood that another key component was missing, the database. From that moment our direction was the one we are still in.

2.1 Trailer

Video introduction makes the user feel conformable about learning something new, we start by giving them all the pieces of information needed to understand the general idea and purpose, we did it through this video trailer. The user starts to understand the concept by learning as a passive watcher.

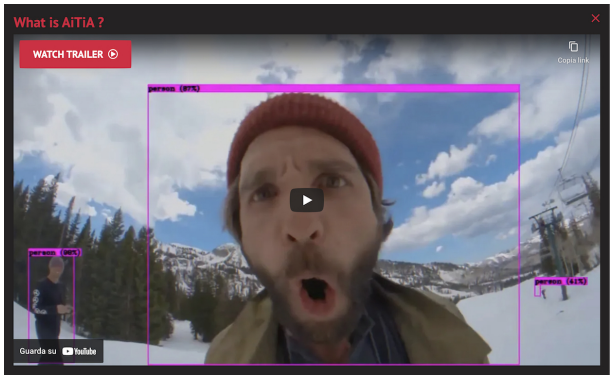

Figure 10. Video trailer preview.

2.1.1 Trailer Technologies

There are several tools for video making and all of them have different features and different structures inside. We choose the one was fitting our personal user-friendly standards and we didn't invest time in comparing video maker apps. MOVAVI Video Editor Plus is the one we used for the implementation of the video trailer.

2.1.2 Trailer Architecture

The video is structured on 3 main components plus the introduction. We start by introducing the basic idea of AI through a video made for kids by Registro.it[4] then we show some similar results of YOLO real-time detection to make users aware of what they are going to do. The last step is where we show them the full process of retrieving, labeling, training, and testing.

Figure 11. Intoduction.

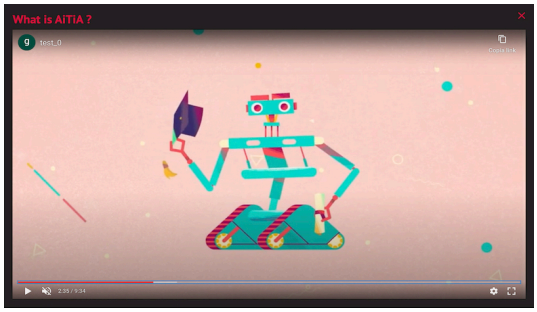

Figure 12. What is an AI nowadays?

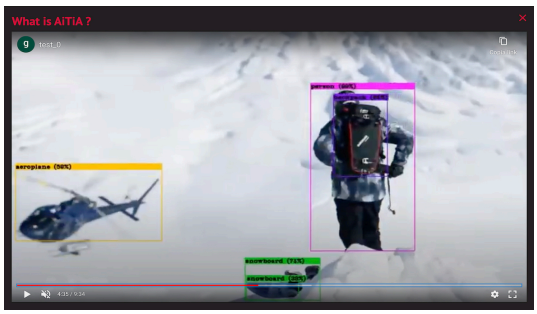

Figure 13. YOLO introduction and explanation.

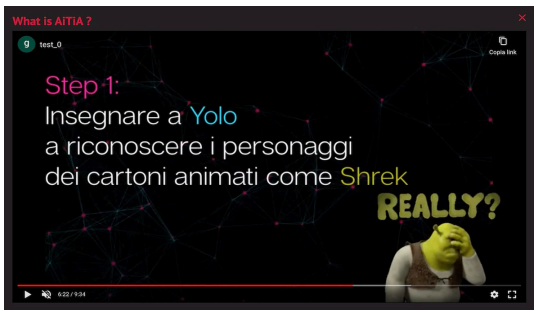

Figure 14. What we can do with AITIA.

2.2 HomePage

The home page of AITIA is simple, structured in 4 main components, and a tutorial button that introduces all of them. Only the first two are available, the app is still in a development phase.

1. On the top left square we find the apposite space for retrieving resources.
2. The top right square is where the process for building datasets starts and ends.
3. On the left bottom square we have the space dedicated for training YOLO.
4. The last square has the role to deploy results.

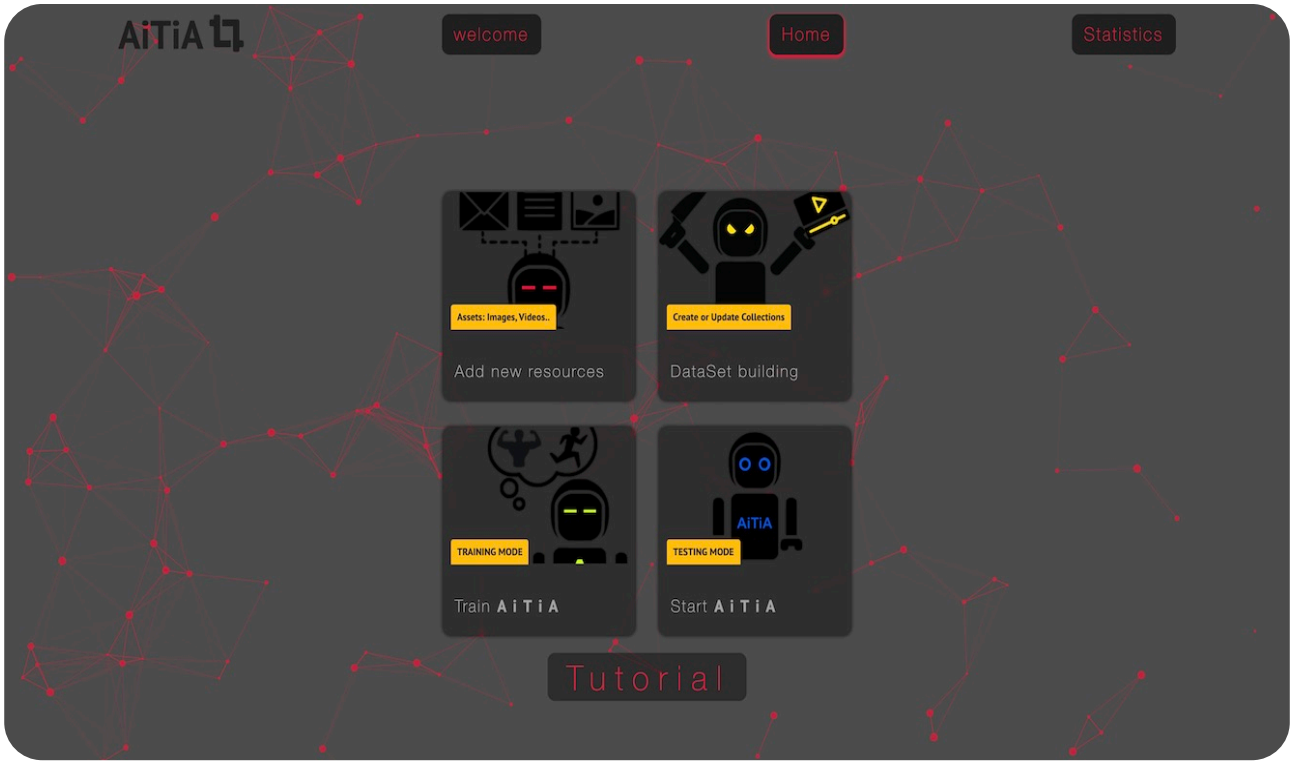

Figure 15. Main Page Visualization.

The Tutorial is a React component we implemented because of its utility and because it's user-friendly. Reactour (trailer components) is a tour inside the application, allows the user to move between all the tour steps by clicking the arrows on the display or by using the keyboard arrows. Example:

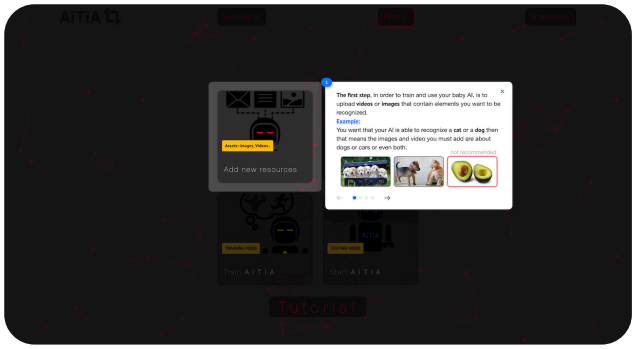

Figure 16. first tutorial element.

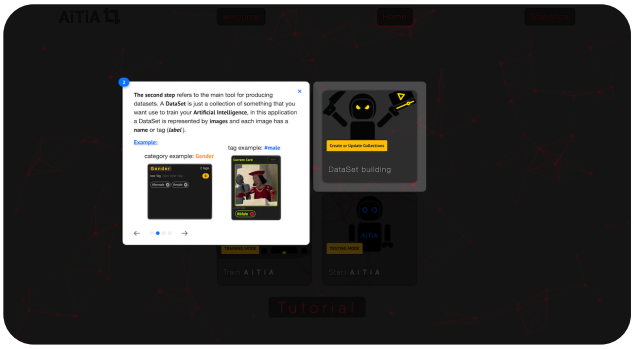

Figure 17. second tutorial element.

Just by using the arrow, the tutorial component will take the user to the next step. This allows us high control on elements, facts, and theory that a client needs for understanding all processes.

2.2.1 Resources Recovery and Polishing

Add new resources component allows user to upload files of image and video type and allows video fragmentation which will be useful in the classification process.

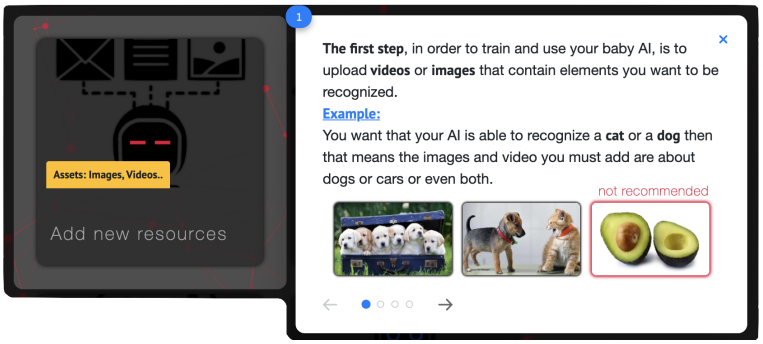

Figure 18. Main page - tutorial first step.

Tutorial Main Page - Step 1

In the tutorial step, we try to make simple the idea about filtering the right resources for the right prediction the user will do at the end of the full process. The concept itself is really simple "add what you want to be recognized".

Tutorial Main Page - Step 2

Here, we introduce the concept of data categorization, specify the type of datasets we'll implement in the next steps and we give them the general idea of associating an image to a label. We add image examples because they are a powerful tool and help users to associate the idea to real facts experience.

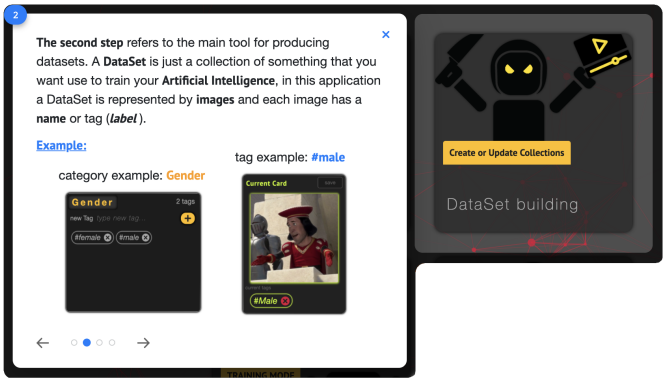

Figure 19. Main page - tutorial second step.

2.3 Videos Interface

This is the first place where the user starts to interact actively with the Web server, here the user deals with almost all requests and responses for the videos routes.

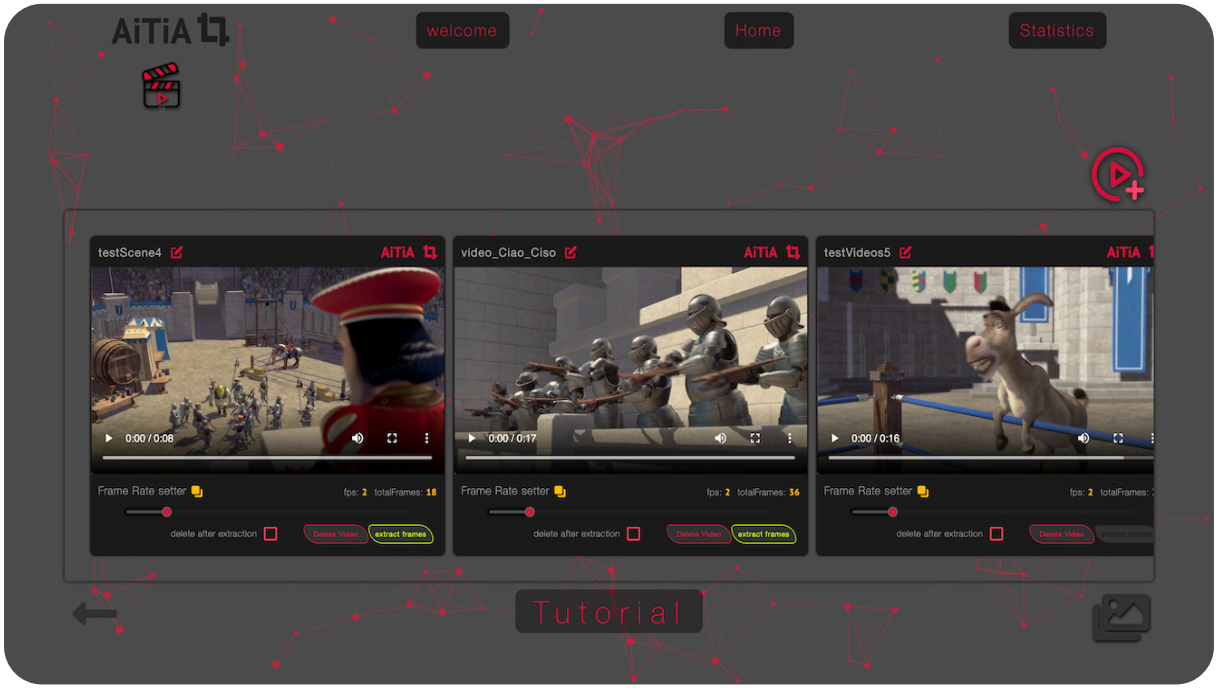

Figure 20. Video Interface.

What a user can do:

- Add single or multiple videos at time and check preview of upload videos.
- Delete a video if not necessary.
- set the frame rate for fragmentation.
- start video fragmentation.

Video Upload - Frontend process

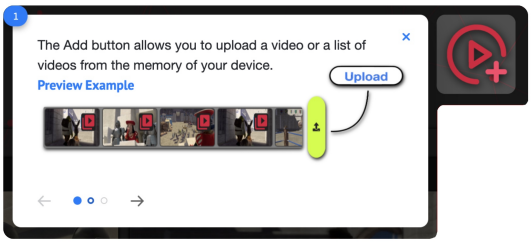

Figure 21. Video tutorial step 1.

The tutorial step show a preview of multi-upload, the user is allowed to upload 10 videos per time and can't upload a video he already uploaded until the name of the first one is not changed.

2.3.1 Videos Upload - Backend process

To implement upload functionalities, we used Multer a node.js middleware for handling multipart/form-data.

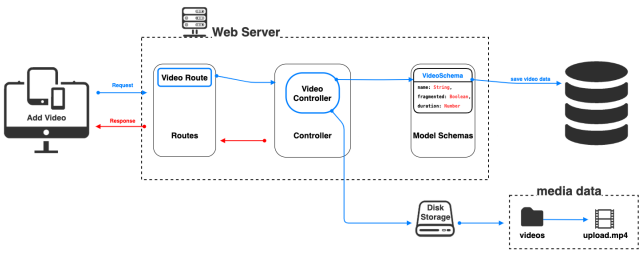

Figure 22. Full process of uploading videos

As we can see from the graph, the data relative to the video is stored in our database and the video itself is stored on the disk storage.

In order to perform the video fragmentation task we implemented a python script that does that for us, the script takes three input parameters, i.e., *video path*, *output path* and *frame rate*. The output is stored on the disk storage and the data relative to the new folder are saved inside the database. To run our python script from the application we used the exec.js library.

2.3.2 Video Fragmentation

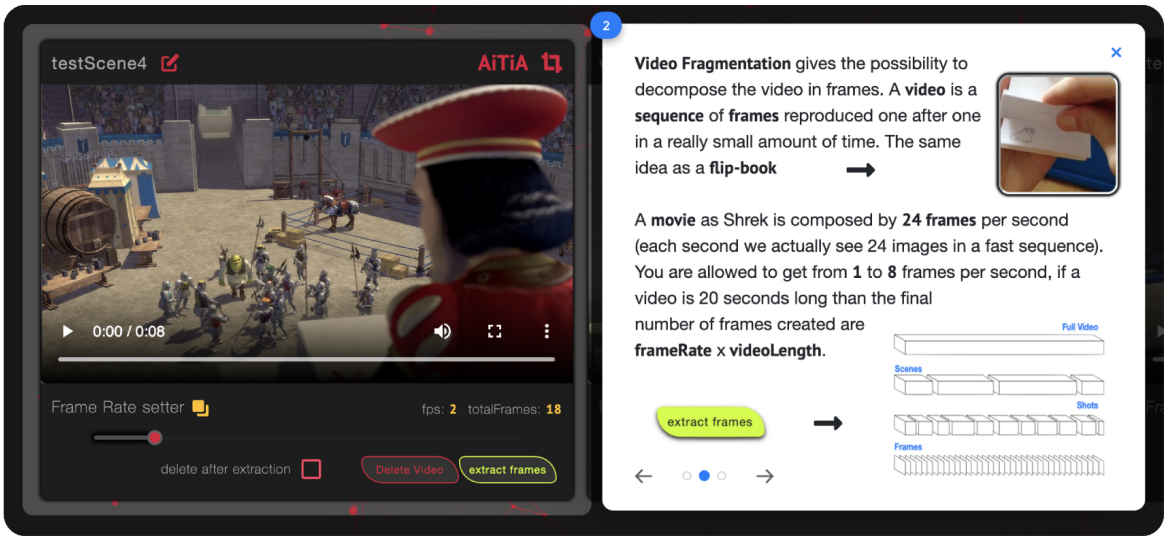

Figure 23. Video tutorial step 2.

The tutorial step explains what is video fragmentation, we give an example of a flip-book gif to make the idea, then we start to introduce frames, what fps means and how is calculated.
We also added an undirected graph that shows what happens in the backend during fragmentation.

As we can see during this process we deal with 3 different model schemas, add a new folder into the disk storage, and many images as fps · video.length.

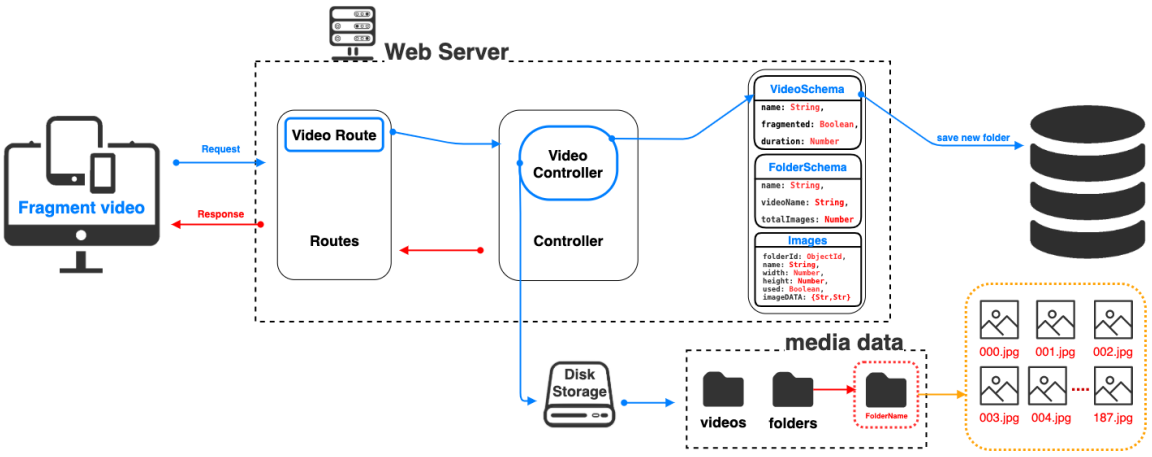

Figure 24. Video Fragmentation - background process.

2.4 Collections Interface

This interface allows the visualization of all collections a user has, it is a show case from where the user can remove images which are not relevant for the classification process.

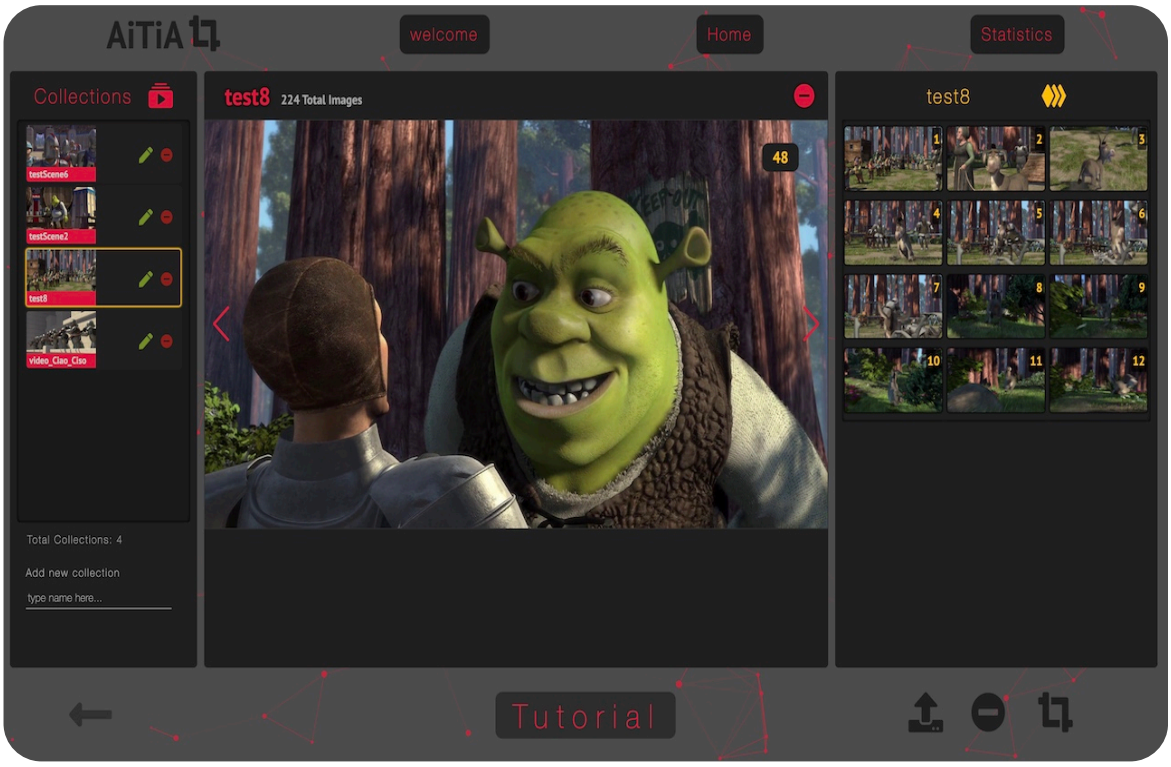

Figure 25. Tutorial steps of Collection interface.

Here the delete functions allow the user to delete a collection (left bar), delete a single image(center), delete images by picking(right bar). The tutorial is implemented in a way that features and ways for interaction are explained.

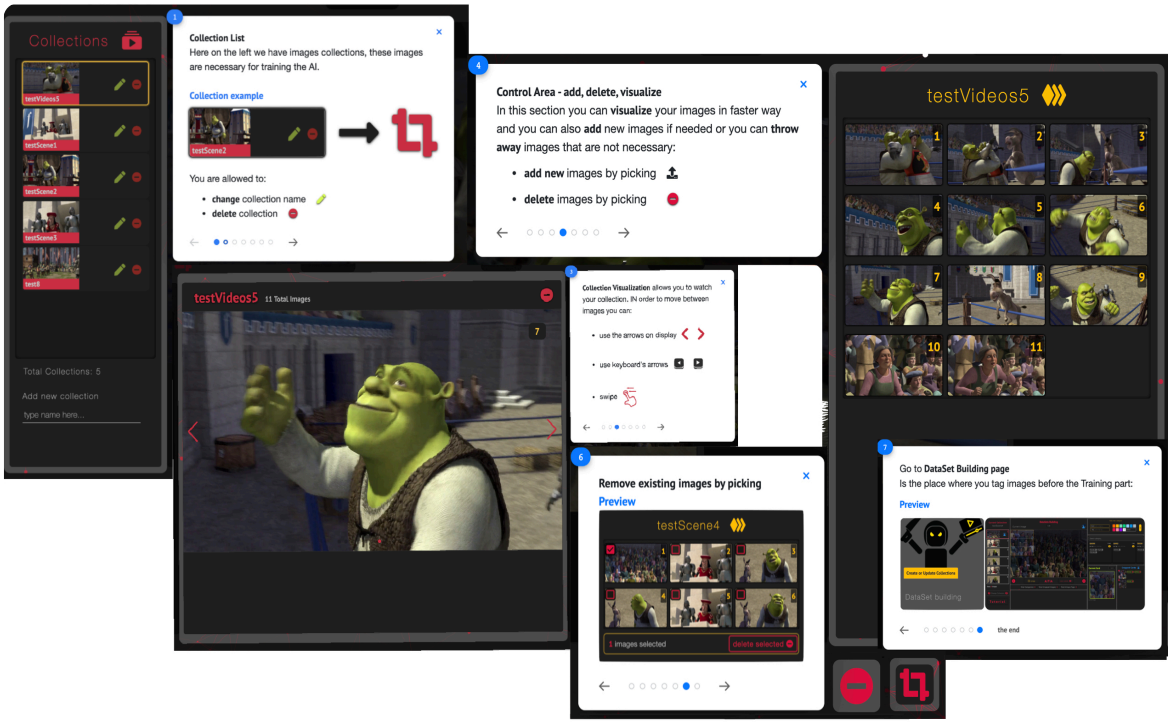

Figure 26. Tutorial steps of Collection interface.

2.5 Dataset Interface

The following interface is the core component of our application, is where users interact with the main tasks for dataset production.

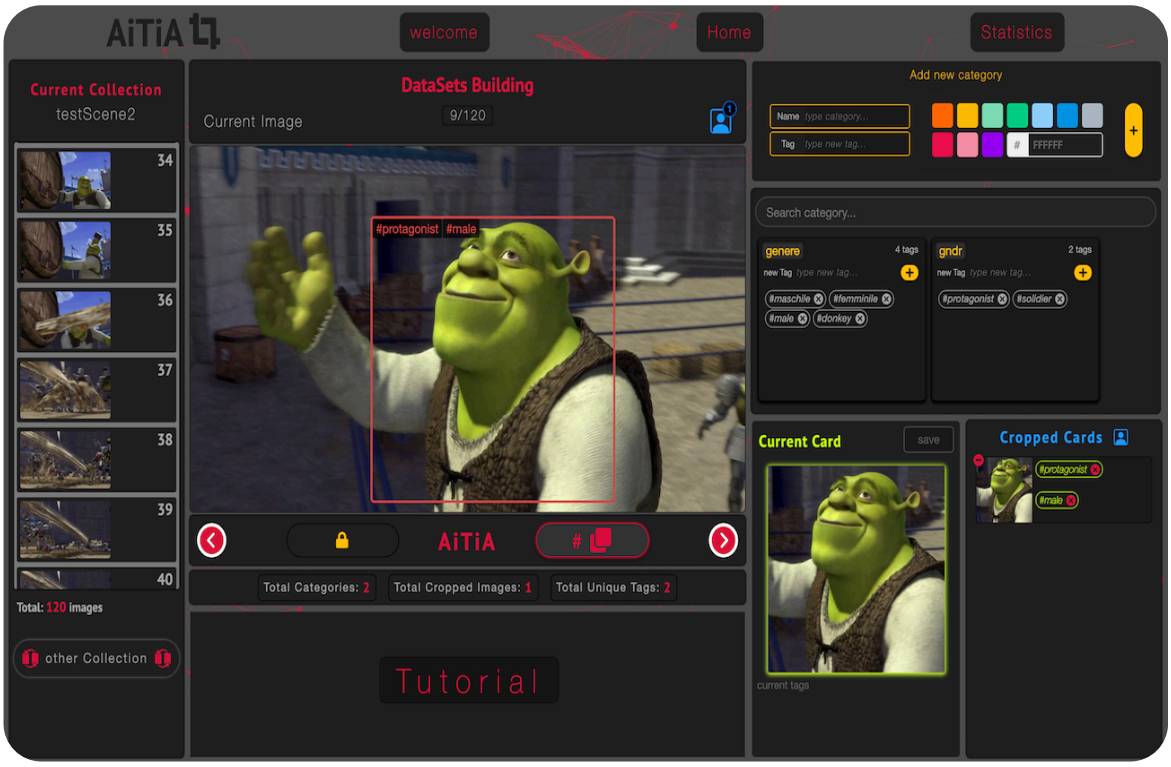

Figure 27. Dataset building interface.

Starting from the left we can visualize all images inside a collection or change it if needed. At the center we have the main component in charge of cropping the selected area, store the coordinates of the bounding box and create a visualization of this bounding-box as well. On the top right, the Add Category component allows to add a new category and assign it a visual id (color). On the center-right all existing categories are displayed, each category can have multiple tags. Tags are the main key, the user assigns tags to cropped images. On the bottom right these last two components play the roles of saving the card with the given tags and display the result.

2.5.1 Image Selection Process

Our core component takes as input a collection of images, displays them, and uses them for multi-dataset building. Here, the tutorial hits the points, allows the user to understand gradually from left to right all the single components of the page

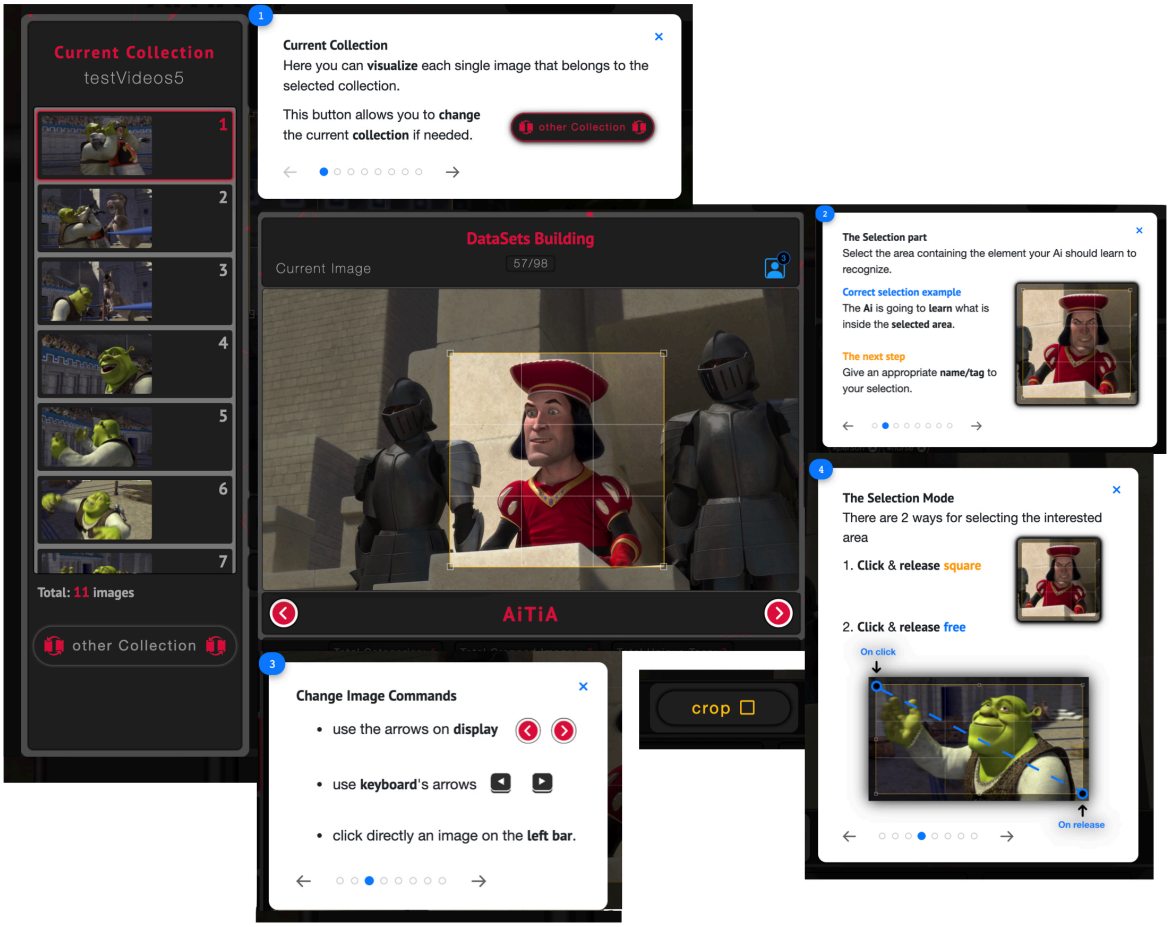

Figure 28. Shows tutorial steps, from 1 to 4.

When an area is selected AITIA automatically normalizes the coordinates concerning the full image and calculates the center (x,y) of the current bounding-box. The coordinates YOLO needs are in the following format:

$$[centerX, centerY, width, height]$$

We implemented two types of crop methods, the first one is a blocked square selection, width and height have the same value while increasing or decreasing, the second one is the unblocked selection which allows to performs a better selection on the target.

2.5.2 Data Categorization and Dataset Initilization

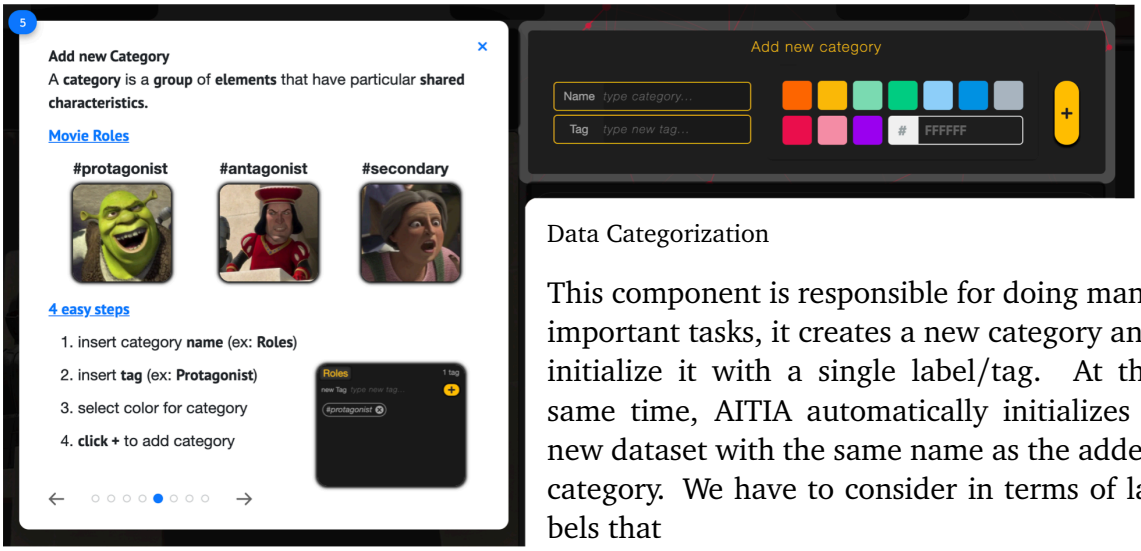

Figure 29. tutorial step number 5.

Data Categorization

This component is responsible for doing many important tasks, it creates a new category and initialize it with a single label/tag. At the same time, AITIA automatically initializes a new dataset with the same name as the added category. We have to consider in terms of labels that

$$Dataset's\ labels \subseteq Category's\ labels$$

The following figure shows how to work the server side after the add category button is pushed.

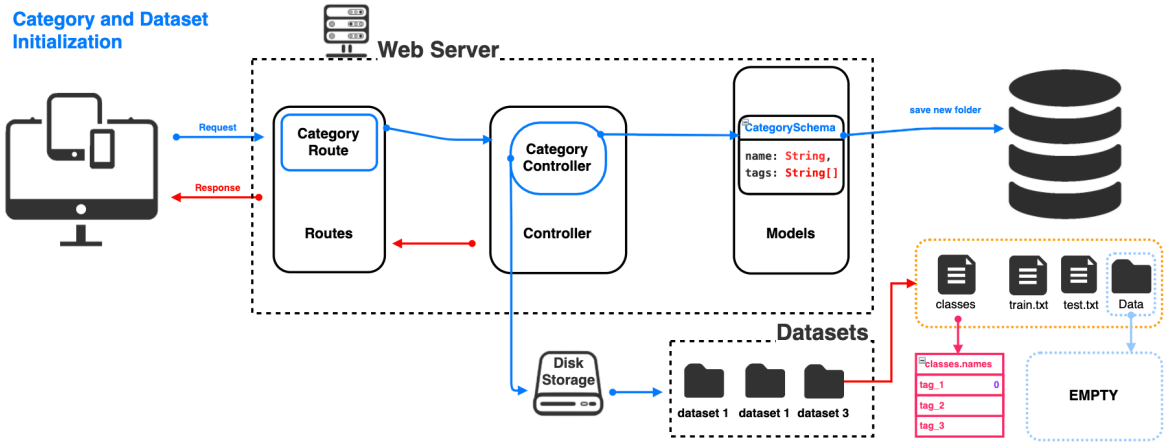

Figure 30. Add New Category and Dataset initialization.

Through the category route, the category controller is activated and uses a Category schema to create the object in the database and also initialize physically a Dataset on the disk.

2.5.3 Datasets and Categories

In this section, we show and explain how a category and its tags are related to a dataset and its labels.

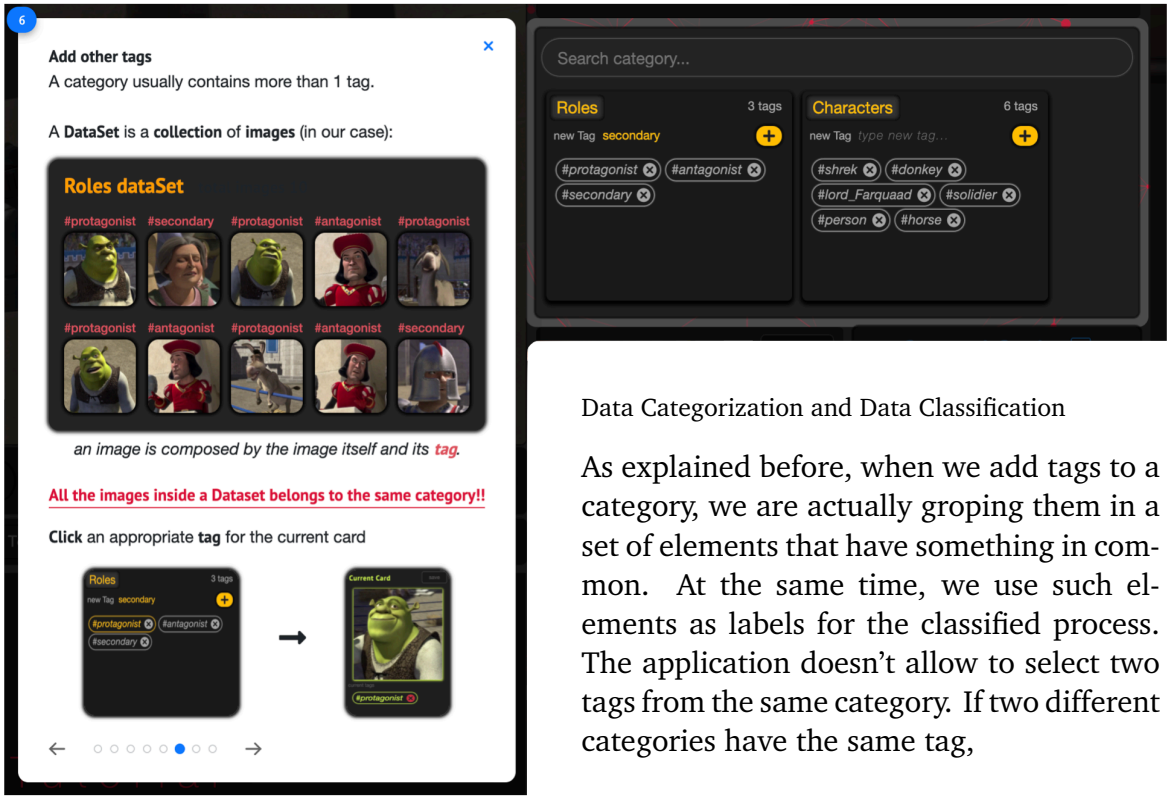

Figure 31. shows tutorial step number 6.

Data Categorization and Data Classification

As explained before, when we add tags to a category, we are actually groping them in a set of elements that have something in common. At the same time, we use such elements as labels for the classified process. The application doesn't allow to select two tags from the same category. If two different categories have the same tag,

Example:

$$National\_Football\ Teams = \{Spain, Italy, France, ...\}$$

$$Europian\_Countries = \{Spain, Italy, France, ...\}$$

the tag is not doubled but acquires on more ID of a different category. In this way, we have a many to many relationships and avoid tag duplication.

2.5.4 Multiple datasets

When it comes to save the current selection with the current tags many things start to work at the same time. From a frontend point of view, the tag is added to the right components, i.e., cropped cards. All states in the parent component which are related to a tag or a cropped image are updated.

This interface is built on functional components that work directly on props values, this allows us to have displayed data changing dynamically.

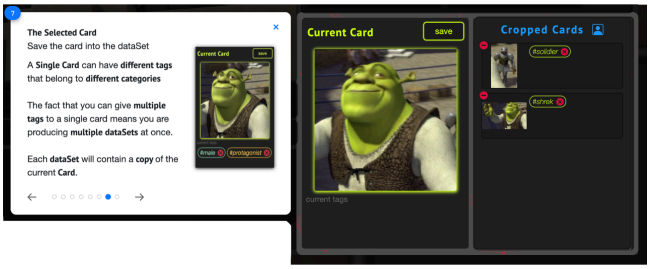

Figure 32. tutorial step 7.

The following graph shows what happens on the server-side when a new card is saved. The Cropped Card Controller once received the data, saves the cropped image in media data for fronted visualization, starts the physical process of multi-dataset building, and saves the properties of the cropped card in the database.

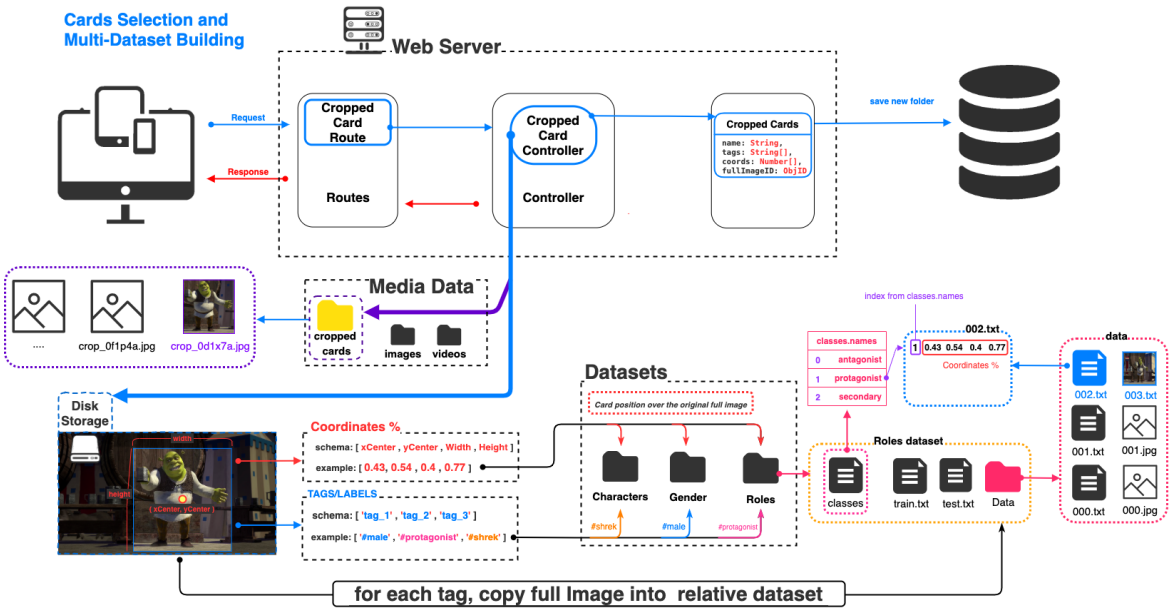

Figure 33. Backend process when new card is saved.

For each card's tag the controller component updates the related dataset, saves a copy of the original full image and creates a new file of .txt format which contains coordinates and object's name (index value of classes.names file).

2.6 Multiple Classification

After an image receives bounding-box and label a user can visualize all the tags that he added to the full image. During Visualization is not possible to select new elements.

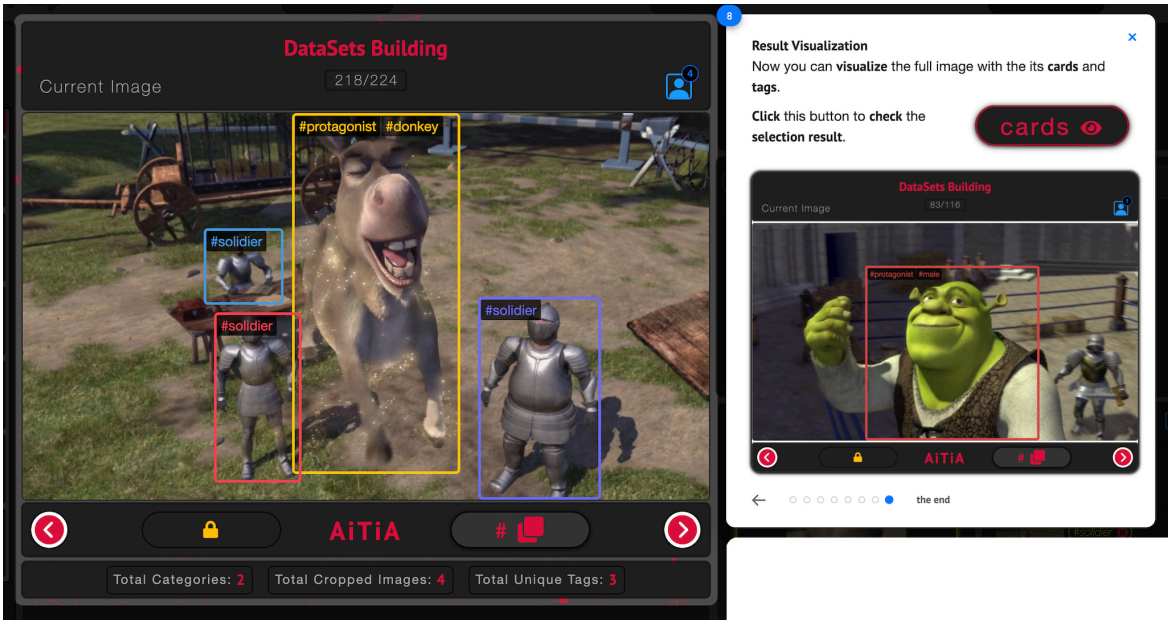

Figure 34. Tutorial step 8.

End of the chapter

With this last feature, we showed almost each part of the app implemented until now. In the next chapter, we argue the observational study we did on kids of middle school to test the impact, the interest, and problems in terms of user-friendly application.

3 User Evaluation and Feedback

3.1 observational study architecture

We want to observe the interest that children might show in the application and the field of informatics especially in artificial intelligence. Feedback will allow us to improve the application, test and improve its friendliness, and also we can use new input from users to add and extend features.

Design

To obtain information and feedback we organized an observational study.

Participants

Children in the ages between 11 - 13.

Materials

The device used in both cases was a MacBook Pro 16-inch 2018 with the incorporated trackpad. Children were not accustomed to it and had some problems at the beginning with the MAC OS trackpad.

Procedure

Step 1
We showed them the application video trailer while explaining some logic behind YOLO and in general Object Detection Methods.

Step 2
We prepared a paper where children had to choose the target for data-setting i.e. the category they wanted to train on, and 7 hints to do that, starting from video upload and segmentation.

Step 3
Following the test, we asked them to answer an online questionnaire.

Feedback skeleton

Feedback was implemented with google forms. Here are the questions asked:

1. What have you thought about the app at the first sight?
2. Describe this experience with one single word.
3. How much did you like the app from 1 to 10?
4. Explain which part was not clear?
5. Do you have any ideas on how to improve the app?

3.2 observation results

Users own words:

- "The app may be useful for a lot of purposes"
- "I want to help you in the future by testing new features"
- "I would like the app in a way where you do not have to train it, I would like it to recognize an element right away, writing the categories and tagging it"
- "Please, insert full keyboard's commands control"

All the users requested to have the application translated in a language where they feel more comfortable since the test required to learn new material. Some of them proposed to implement a tutorial in text so they could take their time to understand instead of having only a video as a tutorial.
Children gave a mark of 9/10 on average, which is yes a good grade but it wasn't the best for a kid, not in that moment.

We gathered some feedback by observing the interaction between the user and the application.

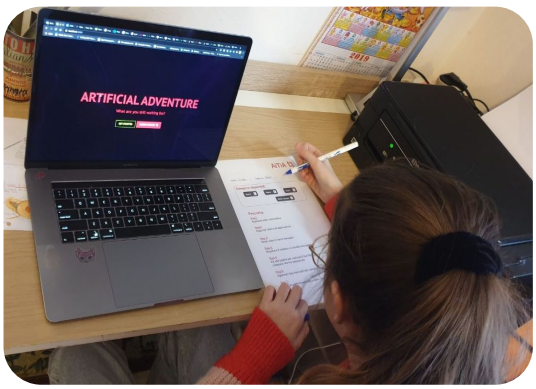

Figure 35. Step 2 Started.

Observation Process

During the test we were aiming for feelings and reactions, we observed kids directly. We focused on the first impact, to understand if AITIA woke up any kind of interest for them at the first appearance. Children seemed to be really interested and showed an increased level of enthusiasm towards our work, it seems that the words "Artificial Intelligence" awaken something in children's mind.

Figure 36. Kid 1 while building a dataset.

Figure 37. Kid 2 while building a dataset.

Children struggled sometimes with English language by not understanding some concepts and words. However, the language barrier was passed with my intervention. I explained to them the terminology of categorization, classification and presented the available configurations. A very important thing we noticed is that children learned really quickly the full process from retrieving resources to actually building a DataSet. Children were happy to be involved in technology and at the end of the experiment they look more confident with the application and with themselves. Despite all the problems found in the application we are satisfied with the results of the feedback as it gave us some critical review upon which we must work and improve. There are many things needed to be done and everybody agreed about it. The thing that we enjoyed the most is that kids after processing what is this application started to give new hints and possible use cases for the future. We were affected by the enthusiasm and discussions that took place after the testing, it proved to us that the direction we are taking is valid and that we are developing a significative artefact.

3.3 Discussion

The application manages to stimulate interest from the users giving them the possibility to interact with it and to discover the word of artificial intelligence.
In the future it might be interesting to further develop the app by implementing the features requested by the users and to repeat a similar experiment using a larger set of subjects. The application could also be used as a didactic tool to introduce a person to the world of artificial intelligence in a dynamic and interesting way and to also develop their interest in the subject.

4 YOLO Artificial Intelligence Model

We build our dataset in accordance with the data format defined by YOLO. Our application allows the user to generate multiple datasets at once. To do so, we give to the user the possibility to assign multiple labels to a single element/bounding-box in the image.

4.1 YOLO - Input Dataset paramenters

The input dataset to YOLO has the following 4 components:

1 - **classes.names** : It defines the name of the classes for the Image Classification problem. As defined above, a category is a group of elements and a subset of these elements are used here as class names.

$$Classes.names \subseteq Category.tags$$

2 - **train.txt** : It contains the paths of the images used to train the model. Generally, we split the complete dataset, such that the training set contains 75% images and the test set contains about 25% images.

3 - **test.txt** : It contains the paths of images used for testing once the training is over.

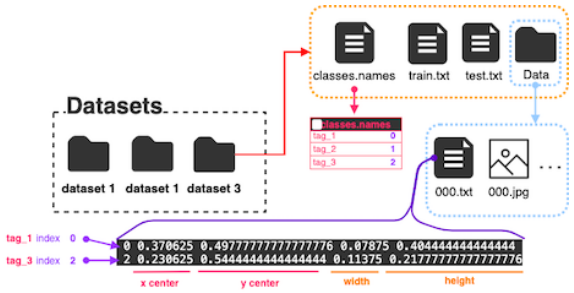

Figure 38. Example of YOLO's input data.

4 - **data** : It contains pairs of images and .txt files:

- The original images are resized to $416x416$ pixel and are then used in this data. The files' name works as a counter of all the full images the user used.
- The .txt files contain 5 numbers in each row. The first number is the index of the class of the object, the second and the third numbers, (x_Center, y_Center) are the coordinates of the box. The last two numbers represent the width and height of the relative bounding-box.

Training Data Preparation

We used our application on some scenes of Shrek Movie, we followed the complete process from fragmenting the scenes to using them as collections of images. We then created the ground truth for these images using AITIA by defining the bounding boxes and labeling them.

Total Images : 128

train.txt: Paths to 110 images → 85% of our Dataset

test.txt: Paths to 18 images → 15% of our Dataset

Average selections (objects) per full image : 3.2

4.2 YOLO - training

Training YOLO requires a GPU, so we decided to use Google Colab as the GPU resource.

GPU, Graphical Processing Unit, are powerful computing machines, which were hyper-optimized for rendering graphics in the gaming industry. Most of the graphics processing is a form of matrix multiplication. Hence GPU is hyper-optimized for dealing with large matrix multiplications. Now, a neural network is also, at a fundamental level, a sequence of transformations that can be represented by matrix multiplications. Therefore, GPUs are very well suited to accelerate the implementation of neural network training.

It should be noted here that the large YOLO version requires a lot of GPU training time and a lot of GPU RAM compared to the Tiny-YOLO. This was another motivation for us to choose Tiny-YOLO over the larger version.

4.3 Experimental Results

Configuration:

- Batch Size : 20'000
- MAX BATCHES : 20'000
- Training time: 30min
- classes: 5
- *shrek, lord_farquaad, donkey, secondary, solidier*
- Images processed: 16'0000

Results:

- Last accuracy = 36.62%
- Best Accuracy = 42.93 %

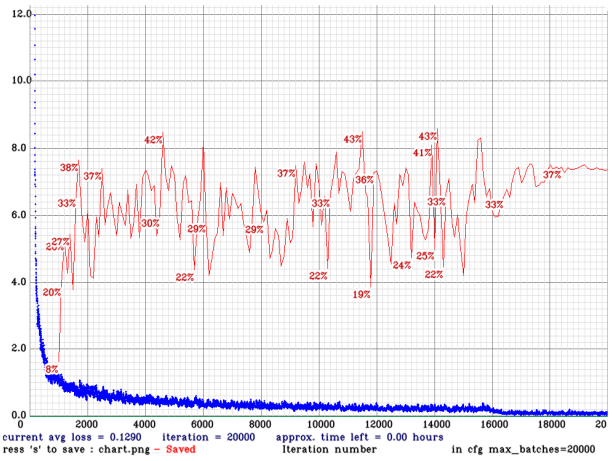

Figure 39. graph representation of training task.

4.4 YOLO - testing

We tested YOLO on our custom dataset, especially we tried all 18 testing images with a threshold of 0.9, which means that the minimum confidence score for YOLO on a bounding box must be at least 0.9 out of 1.0. For this threshold, out of total 18 images, Tiny-YOLO was able to successfully predict 10 images.

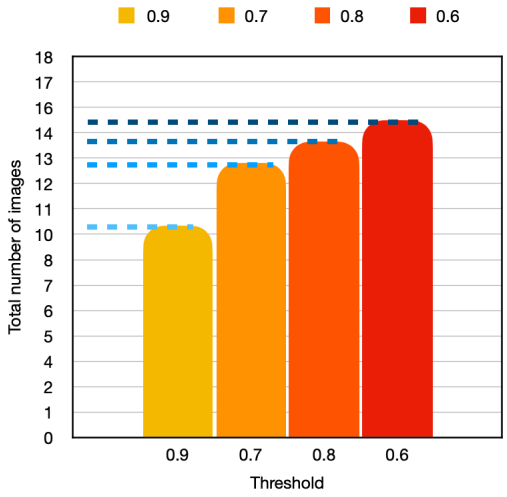

Figure 40. detection on different thresh.

The purpose of this exercise was to perform an end-to-end test of the complete integration of our application with the AI methodology's training and testing. Hence, we chose to work with a relatively smaller dataset. We were able to confirm that our application produces a logically consistent dataset, which works seamlessly with the training and testing of the Artificial Intelligence methodology i.e Tiny-YOLO.

As part of the future work we would aim to experiment with larger datasets.

It is useful to note here that for our testing data out of 8 images that were not recognized properly, 6 belonged to the label: "secondary". The label "secondary" had the fewest training example images corresponding to it. This leads to a 'class-imbalance' problem and that explains the sub-optimal performance for this label.

Figure 41. Good prediction example.

Good Detection Example

test file: 0000.jpg
Predicted in: 3.143 ms

Object Predicted: SHREK
confidence score: 97%

Bad Detection Example

test file: 0009.jpg
Predicted in: 5.180 ms

Object Predicted: SECONDARY
confidence score: 61%

Figure 42. Bad prediction example.

VIDEO Predictions

We tried to detect Characters on video scenes and surprisingly it worked beautifully.

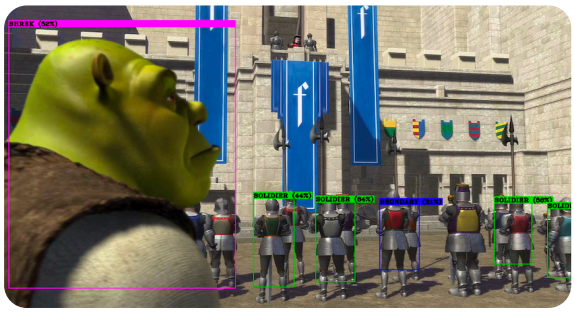

Figure 43. #Shrek,#Secondary, #Soldier x 4.

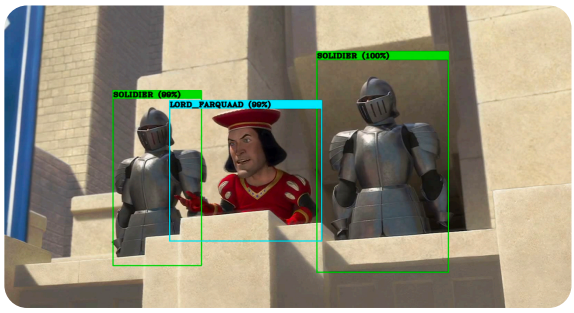

Figure 44. #Lord Farquaad,#Soldier x 2.

The video detection

We can confirm that the dataset built with AITIA works and respects all YOLO's input dataset needs.
The next step (untraveled road yet) is to study how to do a proper Dataset to produce a balanced one and test different approaches on building and testing datasets.

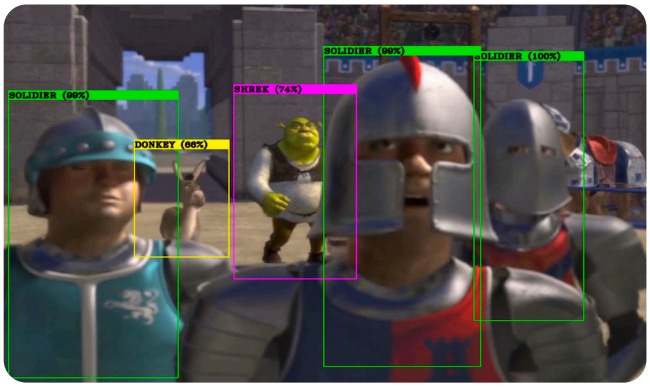

Figure 45. #Shrek,#Donkey, #Soldier x 3.

YouTube Link - First Testing Approach of AITIA - Video Object Recognition

5 The State of the Art

There are several tools for producing datasets and each of them have their advantages and disadvantages. We divided them into 2 categories, desktop GUIs and Web application:

5.1 Desktop GUIs

Yolo_label - is a sensitive image-labelling tool for object detection, its design refers to the original Web site of Josep Redmon, creator of Yolo. Yolo_label needs 7 pre-configuration tasks only to start using the tool, the tutorial is available on the apposite GitHub page but isn't inside the tool itself and doesn't allow to extract images from video in the environment.

Figure 46. Yolo Label interface.

Figure 47. OpenLabelling interface.

OpenLabeling - is a python GUI which a user can download from GitHub and use it to build a custom dataset. As other GUIs, it needs some pre-configuration and the tutorial is not inside the program but is available on GitHub. There few interesting tools inside as resizing images, zoom in, zoom out and edge sensitivity.

Yolo_mark - allows the user to build a custom dataset from scratch as the other tools but its user target is technical users. It runs on both Windows and Linux both need some configurations that are different for both Windows and Linux. This program doesn't allow direct extraction of frames from a video.

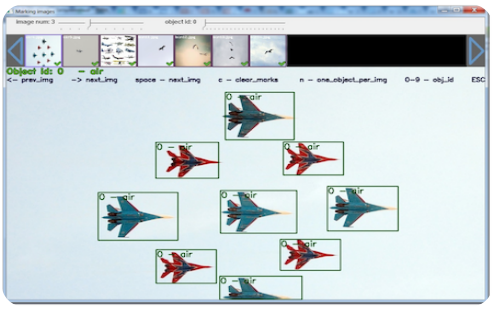

Figure 48. Yolo Mark interface.

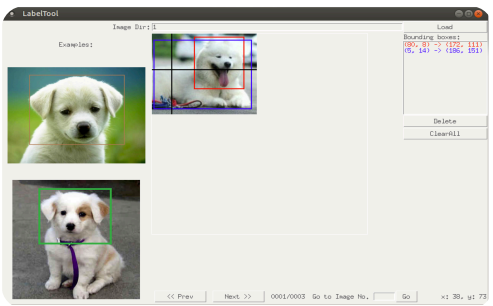

Figure 49. BBox-Label-Tool interface.

BBox-Label-Tool - was implemented with Python Tkinter (our second choice before deciding to build a Web application). This GUI as well needs to be downloaded from GitHub and a set of python configuration before starting using the tool. His small tutorial is available on the apposite space on GitHub and its target is technical users. It doesn't allow to extract images from videos.

Observations - Desktop GUIs
All of these GUIs accomplish their task, but from a user's point of view aren't user-friendly and easily intuitive. Their target is technical users and their bigger limitation is they are a "gear" of the full AI ecosystem we want to create. From a non-technical user point of view, these tools are hard to understand, they need to be downloaded and need many configurations before starting. Let's assume that a persistent non-technical user succeeds in the dataset building task then he needs to find somewhere else where find out what to do next and so on. All these GUIs don't have an internal way for frames extractions and the tutorial is not available inside the program but is on GitHub. We try to help non-technical users by implementing all the configuration tasks and by providing a user-friendly interface for technical and non-technical people.

5.2 Web Applications

Roboflow - is beautiful from a user-friendly point of view, is innovative clean, and allows a full environment for everything a user needs for resources retrieval, training, testing, data, and progress visualization. From a technical point of view is the best for in terms of what a user can ask. We will study how Roboflow is done since many things can suggest to us the best approach to proceed from now on.

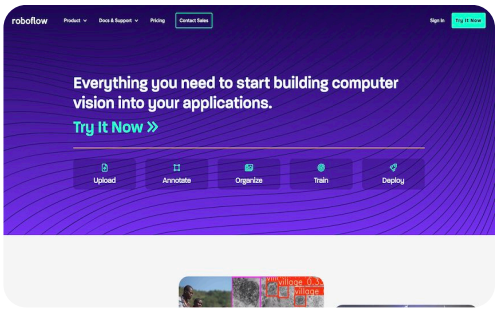

Figure 50. Roboflow interface.

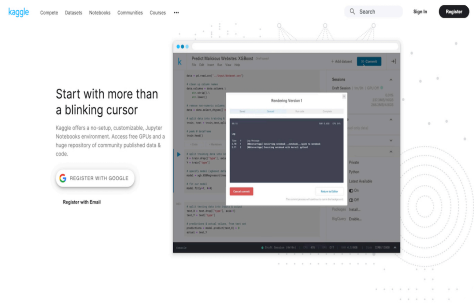

Figure 51. Kaggle interface.

Kaggle - is already implemented as a social platform where people can post and check other people's work, find job offers and post their results or interact with others. Its community has many relevant topics as computer vision, GPU, TPU, NN, etc... Kaggle also is amazing but is for non-technical users but there are a lot of things for beginners inside.

Observations - Web Apps
These 2 online tools are a better solution for producing custom datasets, they are user-friendly, and have many features inside, Kaggle allows also multi-labeling. What hits the point in terms of the AI Ecosystem is RoboFlow, it allows users to upload data, create custom datasets, train and deploy results. Kaggle's power is the social aspect because is a place where share informations, results news, and so on. Both Kaggle and Robofow have as a target technical users, they were designed for developers. From a non-technical point of view, they don't have a context or non-technical guides and explanations. Roboflow ecosystem is really powerful, but their goal is to provide everything a user needs to test an AI. However, it is not designed as an application that can be used during everyday life, by everyone, and without difficulties. The same goes for Kaggle which is more centered on the social aspect. We want to use Kaggle and Roboflow to build our own app, with aspects of both but we'll try to do more by allowing also non-technical users to enjoy the same AI experience as technical users with less effort.

6 Future Work and Goals

6.1 Full Ecosystem and Native Application

For now, the application can only produce an effective dataset in a discrete amount of time.

The first step is to implement all the new functionalities and missing user stories which can mislead and make users lose focus as side effect.

The second step is to make AITIA a native application. A user should be able to access it from a phone, tablet, computer or all other devices with display and connection to internet.

6.2 User Friendly

An application with an effective User-Friendliness reduces at minimum the communication gap between the user and the app itself. Our design must be simple, clean, intuitive, and reliable. We must anticipate users' needs and the solution is to implement a Recommender system based on collaborative Filtering(CF). CF makes filtering decisions for an individual user based on the judgments of other users, infers an individual's interests or preferences from that of other similar users. Since the idea is to make it social, to fill the net with sharable datasets. The best solution of course is to use the collaborative filtering approach to anticipate what users need by providing assistance in order to make the user comfortable with the app.

An other step is to map the application interactions completely on the keyboard's commands, this will help people that have a certain amount of confidence as programmers, gamers, etc. to feel comfortable using the application and also will help people with less confidence to learn step by step the commands they want for the given tasks.

One important feature we must implement is the language choice. From the Observational study with kids, we confirmed this feature is indispensable for a better understanding of the applications and of the logic behind AI.

The tutorial is what makes the difference in a user-friendly context, in the report we showed its first version, but it needs more study to know exactly what pieces of information and where to place them.

Data Visualization: user is able to see his progress while building the dataset and visualize all kind of data within the ecosystem by using charts.

6.3 Test Driven Developing

For complex applications likes this one, we must use the Test Driven Development approach where test cases are developed to specify and validate what the code will do. We have to design and develop tests for every small functionality of the application. The desired result is to write new code only if an automated test fails. We avoid code duplication and in general, you have overall control in the application architecture, processes, and errors.

6.4 Retrieve datasets online

As the next step of AITIA, we want to implement a search engine that crawls images directly from Google or other search engines. If that is not possible and not worthy then plan B is to use the Bing Image Search API which allows a controlled amount of images given a query. Option 2 seems to be easy to use but we'll investigate both of them. Our idea is to allow a user to retrieve resources not only from the own disk memory but also from the web. The same goes for videos, in fact, we want also to give to users the choice between personal video and video from YouTube or other video-sharing platforms.

6.5 Dataset sharing and user population

A further step is the implementation of users' profiles with a social area and community-integrated where people can relate on a third kind of resources, the shared ones. If we construct such a system, is possible that is going to work on the same principles social media are based on. Users will interact with each other and the number of datasets can have an exponential growth way allowing users to directly choose datasets instead of building them.

7 Possible Applications

7.1 Stereotypes detection on digital artifacts

One possible future direction of research concerns the use of AI to detect gender stereotypes in digital artifacts[5]. In particular, it would be interesting to extend the work by Rubegni et al, (2019) that proposes the use of five lenses to detect gender prototypes in multimedia stories created by children [1]. A first attempt in this direction has been taken by a colleague student, who submitted at the end of her Bachelor project a report on "Exploiting AI for automatic gender stereotypes detection in Disney movies" (Sassone, 2019) [7]. In her work, Gloria developed a system to automatically classify male and female characters in Disney movies. The work presented here could go a step further and support the automatic detection of Roles and Agency, two important lenses to enable the detection of gender stereotypes. Besides, by having children to train the system we will be able to detect tehri biases too while helping them to reflect on them.

7.2 Disorders detection

Autism Spectrum Disorders[3]

This prototype could help the detection of early signs and symptoms of Autism Spectrum Disorders. In line with work by Abbas et al. (2017). By looking at the selections and tags assigned by individual children experts could recognize anomalies and decide whether an early intervention is useful.

Dyslexia and Dysgraphia disorders[2]

AITIA can be used to detect dysgraphia, dyslexia, and dyspraxia disorders. Our prototype can be used to detect this kind of disorders, how? In order to do so, the implementation of a dataset is needed, the dataset is made by handwritten text pictures. There are 2 labels for this dataset, standard children and dyslexic and/or dysgraphic children. At the end of this process, we can use AITIA app for object classification and to teach YOLO differences between the two test subjects. With a good dataset and YOLO trained at its full potential we can make some important steps in disorders detection.

7.3 Courses for kids

In the future, it might be used as a course for students of primary and middle school. As we observed in chapter 3, kids from middle school were interested in our application idea and were emotionally touched. We think that this prototype can arouse interest in Artificial Intelligence and informatics fields not only from kids but also from high school students and in general non technical users.

7.4 Labelling Tool for training CNNs

Technical people can enjoy our prototype app which allows an easy, guided, and straightforward way for producing datasets (up-to-date), a way to train and deploy.

7.5 Third Party Devices

As we explained above in Chapter 4, YOLO Tiny can run on mobile's GPU, hence all the other third-party devices can use our application. Drones have the power to retrieve resources for classification and are able to classify objects at the same time.

References

[1] A. D. A. L. J. Elisa Rubegni, Monica Landoni. Detecting gender stereotypes in children digital storytelling. 2019.

[2] M. S. Gilles Richard. Dyslexia and dysgraphia prediction: A new machine learning approach. 2019.

[3] A. I. C. P. S. A. S. D. G. T. F. Lonnie Zwaigenbaum, Jessica A. Brian. https://www.cps.ca/en/documents/position/asd-early-detection, journal = Computer Vision and Pattern Recognition, year = 2018.

[4] LReegistro.it. https://www.nic.it/en. 2018.

[5] E. P. Paula Kusumaputri. Classifying disney characters from commercial merchandise using convolutional neural networks. 2015.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2017.

[7] G. Sassone. Exploiting ai for automatic gender stereotypes detec- tion in disney movies. 2019.