**Athens University of Economics and Business**
M.Sc. Program in Business Analytics (PT)

Machine Learning and Content Analytics | September 2023

# "Art Unveiled: Enhancing Access to Arts for the Visually Impaired"

*Team Name: Tango*
*Students: Panagiotis Vaidomarkakis, Alexandros Lemonidis, Katerina Gioupai*
*Student I.D.s: p2822203, p282214, p282208*

## Table of Contents

# 1. Introduction

> *"Even though people born blind have never seen that bananas are yellow, a remarkable discovery by Johns Hopkins University researchers has revealed that, like sighted individuals, those born blind can grasp the concept that two bananas are likely to be the same color and understand why."*

Ref. => https://hub.jhu.edu/2021/08/17/blind-people-understand-color/

While searching and brainstorming for the project subject, we came up to the aforementioned article and as it seems it served as the wellspring of inspiration for our undertaking, ***"Art Unveiled: Enhancing Access to Arts for the Visually Impaired"***.

# 2. Project Description

Art Unveiled is a visionary project born from the desire to extend the boundaries of art appreciation to those with visual impairments, opening for them new avenues for cultural enrichment.

*Problem Statement:*
Millions of visually impaired individuals around the world are denied the opportunity to experience the visual beauty and cultural significance of art. The existing methods of art appreciation do not cater to their needs, leaving a significant portion of the population excluded from this enriching aspect of human culture.

At its core, this project seeks to address a profound question: How can we make art more inclusive and accessible to individuals who are unable to perceive it visually?
It is driven by the ambition to harness the power of machine learning by crafting a system capable of narrating intricate art pieces to those who cannot visually perceive them.

# 3. Vision/Goals

*Vision:*

In pursuit of our mission, Art Unveiled envisions a future where art knows no boundaries and can be experienced by everyone and at any time or place, regardless of their visual capabilities.
Our goal is the creation of a system that can provide vivid and insightful descriptions of artworks (paintings), enabling individuals with visual impairments to appreciate the beauty and significance of art in a meaningful way.

*Goals:*

- **Inclusivity**: To break down the barriers that separate art and its enthusiasts by enabling the visually impaired to fully experience and appreciate artworks.
- **Empowerment:** To empower users with visual impairments to explore and interpret art on their terms, fostering a sense of independence and confidence.
- **Enrichment**: To enhance the lives of individuals with visual impairments by granting them access to the profound beauty and cultural significance of art.

*Benefits:*

- **Accessibility**: Art Unveiled will provide a unique platform for individuals with visual impairments to access and enjoy art, promoting inclusivity.
- **Empowerment**: Users will gain autonomy in their art exploration, promoting independence and self-confidence.
- **Enrichment**: The project will bring cultural enrichment, emotional resonance, and personal growth to users' lives.
- **Education**: Art Unveiled can be used as an educational tool, broadening horizons and fostering creativity.
- **Awareness**: The project raises awareness about the needs of individuals with visual impairments and highlights the potential for technology to address these needs.

*Stakeholders:*

- **Visually Impaired Individuals:** The primary beneficiaries of the project.
- **Art Institutions and Museums:** Potential partners and beneficiaries as Art Unveiled can enhance the accessibility of their collections.
- **Educational Institutions:** Can utilize the platform for educational purposes.
- **Accessibility Advocates and Organizations:** May support or collaborate on the project.

*Risks:*

- **Technical Challenges:** Challenges related to image recognition, natural language generation, and accessibility implementation.
- **User Adoption:** Ensuring visually impaired users are aware of and comfortable using the platform.
- **Data Privacy:** Handling user data and ensuring privacy and security.

# 4. Methodology

To start implementing the desired idea into reality, the following steps will be followed:

1. **Data Acquisition:** Various digital art images of famous painters will be collected and downloaded from the internet.
2. **Data Annotation/Labeling:** The art images dataset will be imported to an annotation/labeling software, in order to annotate each image with bounding boxes to identify the regions of interest within the artwork.
3. **Data Augmentation:** In order to artificially increase the training set by creating modified copies of the existing dataset, we will use the "Albumentations" Python library.
4. **Deep Learning Framework for Object Detection:** Leveraging state-of-the-art deep learning frameworks such as YOLOv8, multiple models will be used to perform object detection on the annotated art pieces will be trained, then hyperparameters will be fine-tuned based on specific metrics and finally retrain the best performing model, resulting to the best performing one in terms of prediction.
    a. Train
    b. Finetune
    c. Retrain
    d. Predict
5. **Text to Audio Tool**: gTTS (Google Text-to-Speech) Python library will be used for transforming textual content produced from the deep learning model into spoken words.

With this multifaceted approach, Art Unveiled aims to build an application that will act as a bridge for the gap between art and accessibility, enriching the lives of visually impaired individuals by granting them a unique and enriching art experience.

# 5. Data Collection

### 5.1 Data Sources

The scope of this project dictates that the desired model should be able to recognize various art styles and be able to produce accurate labels that could then be used to provide the most representative description possible. In order to achieve this goal, various art images representing various art movements were collected. Emphasis was given to more art images adjacent to more realism-movement like styles, as they were deemed easier for a model to interpret and their description could more easily be envisioned from vision impaired people, in comparison to more abstract art images.

In total, a set of 125 different images were chosen (mainly from https://www.wikiart.org/) to go through data preprocessing and augmentation in order to produce a dataset capable of being used by the YOLOv8 algorithm.

*Image 1 - Preview of Dataset used.*

***A brief synopsis of the images in the collection:***

- *Human beings*: to capture the human element in art, we basically included artworks featuring people.
- *Animals*: Artworks depicting basic animal forms were also integrated.
- *Nature backgrounds*: Images with natural backgrounds added depth and context to our dataset.
- *Assorted Objects*: A variety of other objects were included to diversify our dataset further.
- All images were of type .jpg

### 5.2 Data Preprocessing

In order for the collection of images to be accessible to the various object detection models, they must first be annotated so that each of the images have the necessary labels and bounding boxes coordinates. The resulting dataset containing the images and their annotation, must be exported into a YOLO format file, which in turn can be utilized by the object detection models.

The tool used to annotate the images, produce and export the necessary dataset is called Label Studio, a web based tool with the capability to export the datasets in various formats, including the YOLO style format necessary.

In order to add annotation to each of the 125 images in the dataset, bounding boxes were drawn by hand in order to highlight the most representative aspects of the artwork. Key attributes of our dataset include image resolutions ranging from high to moderate, varying numbers of objects per image, and a diverse distribution of labels reflecting different art elements. For each bounded box drawn in an image, a label was added to each box with a representative title. All of the labels were added either prior to the drawing of the bounded box or synchronously depending on the need.

An example of an image with drawn bounded boxes and labels imported in Label Studio, can be seen in the image below. Below the image, available annotations configured are listed. Indicative ones are highlighted in colorful circles.


*Image 2 - Painting "Grande Odalisque" by Jean Auguste Dominique Ingres.*

### 5.3 Data Augmentation

As the dataset consisted of 125 images, it was necessary to use data augmentation techniques in order to have a greater variety of images available for the training of the various models. The goal was to create a pool of around 1000 images to assist in the better training of the object detection model.

For the augmentation of the images in the dataset, the 'albumentations' library was used, publicly available in Python.

The augmentation of the images consisted of the following steps:

1) Resizing: Crop each of the available images in the dataset with a random probability produced from a specified seed and set width and height parameters.
2) Random Scaling: Rescale each of the available images in the dataset with a random probability of 1-p, produced from a specified seed. The decision to set the probability to 1-p stems from the fact that it was deemed better to not crop and scale an image to the same extent as it would not produce the desired results.
3) Rotation: Randomly rotate each of the available images in the dataset with a random probability from a specified seed. The range of angles for the rotation was set to (-90, 90) degrees.
4) Contrast: Modify the contrast of each of the available images in the dataset
5) Snow Effect: A snow effect was added to each of the available images in the dataset with a random probability produced from a specified seed. This effect was viewed as an interesting way of adding variation to the images.
6) Rain Effect: A rain effect was added to each of the available images in the dataset with a random probability produced from a specified seed. This effect was added in the same vein as the snow effect mentioned above, but with the random probability set to 1-p in order for the rain and snow effect to be complementary to each other as otherwise the results could be of worse quality than the desired.

### 5.4 Final Dataset

The resulting dataset of images after the augmentation steps consisted of 1000 images and was split into training, validation and prediction sets. The sets were placed in respective subfolders in the directory with the following structure:

1) dataFolder/train/images
2) dataFolder/train/labels
3) dataFolder/val/images
4) dataFolder/val/labels
5) dataFolder/predict

The reason behind the directory structure above is that the object detection models can only utilize the files when these are placed in folders with specific names (train/images, train/labels etc.)

The predict folder contains only the images required to be used by the chosen final model in order to produce description of the images, later used by NLP and text to audio tools to provide audio description.

# 6. Application Architecture

### 6.1 Architecture Overview

The high level architecture of the designed applications is as follows:

1) Utilization of an optimal object detection model that is the result of the training of various different object detection models with a dataset of labeled and annotated art images. This model will be used to output labels for the images provided to it as per the application's functional requirements.
2) Utilization of an NLP tool in order to transform the labels provided from the object detection model to a cohesive and meaningful text description that can provide sufficient context to the art image.
3) Utilization of a text to audio tool in order to transform the text description into audio and thus convey the context of the image in the most representative way to a visually impaired person.

### 6.2 Object Detection Models

6.2.1 Chosen Deep Learning Framework

As the main goal of the application is to provide an audio description of an art image to visually impaired people, the most important aspect would then be to utilize a model that could accurately recognize the various entities depicted in an art image and output the key labels for it, that could then be shaped into a concise description.

Therefore, it was decided to utilize an object detection model as the foundation of the application and would provide the predicted labels for every art image that is 'fed' to it. After careful consideration and research, the YoloV8 model was chosen as it offers the following benefits:

1. **Efficient architectural composition:** The architectural composition of YOLO variants is compatible for one-stage detection and classification, making it computationally lightweight with respect to other detectors. (11)
2. **Speed & Simplicity:** It is fast and efficient, making it an excellent choice for real-time object detection tasks. Additionally, its ability to function effectively with minimal training data simplifies its implementation and adaptability for new tasks. (12)
3. **Optimized Accuracy:** YOLOv8 has outperformed other detectors in both accuracy and speed. Small targets in various complex scenes were easier to capture. (13)
4. **Variety of Pre-trained Models:** YOLOv8 offers a range of pre-trained models to cater to various tasks and performance requirements, making it easier to find the right model for specific use cases. (14)

On top of the aforementioned benefits of the YOLOv8 model, it is worth mentioning that the model is offered in 5 variations, based on the number of parameters employed for the training of the offered model. The variations are shown below:

| Model | size (pixels) | mAP$^{val}$ 50-95 | Speed CPU ONNX (ms) | Speed A100 TensorRT (ms) | params (M) | FLOPs (B) |
|---|---|---|---|---|---|---|
| YOLOv8n | 640 | 37.3 | 80.4 | 0.99 | 3.2 | 8.7 |
| YOLOv8s | 640 | 44.9 | 128.4 | 1.20 | 11.2 | 28.6 |
| YOLOv8m | 640 | 50.2 | 234.7 | 1.83 | 25.9 | 78.9 |
| YOLOv8l | 640 | 52.9 | 375.2 | 2.39 | 43.7 | 165.2 |
| YOLOv8x | 640 | 53.9 | 479.1 | 3.53 | 68.2 | 257.8 |

*Table 1 - Different YoloV8 Models Comparison.*

In the end, the YoloV8 nano & medium models were selected to be employed as the larger models had too many parameters for the size of the dataset that was created through the preprocessing (labeling and augmentation). On top of that, it would take extensive resources to even run these models properly, so it was decided that the most sensible approach would be to use the model with the most parameter that is actually possible to run properly.

6.2.2 Object Detection Model Training & Parameters Tuning

As mentioned in 8.1.1, the 2 models chosen to be trained for the application were the YoloV8 nano and YoloV8 medium variations. Besides that, these 2 models employ a very high number of hyperparameters that can be used to tailor the behavior of each model in order to produce the maximum result in terms of performance.

All possible combinations of different parameters were employed in the various training runs of the models, so that it could be possible to then compare the results from the various combinations and evaluate which model had the best performance.

The parameters modified along with the options tested are the following:

- Activation function: Various activation functions were chosen in the different runs, in order to examine a wide variety of results.

  The different functions used:

  - LeakyReLU: This activation function is very useful when used after feature extraction, like in this application's case. In comparison to the ReLu function, which is negatively influenced by negative values can occur after feature extraction causing neutrons in the deep learning model to may never activate (as a result of a negative bias), LeakyReLu allows a small negative slope for the values of X and in that way all neurons in the network can contribute to the output. (1,2)

    - SiLU: A variant of the ReLu function that is weighted by the input and performs better on reinforcement learning tasks. (3,4)
    - Swish: Considered an improvement to the ReLu function, as in comparison with the ReLu it adds a sigmoid in order to address the problem of 'dead' neurons (neurons that never activate due to their inputs). (6)
- Optimizer: The following optimizers were used in order to examine the different results:
    - SGD: Stochastic Gradient Descent (SGD) is a variant of the Gradient Descent algorithm, used for optimizing deep learning by using smaller batch sizes and thus improving calculating efficiency. (7)
    - AdamW: AdamW is a variation of the Adam algorithm that is considered generally as the 'go-to' for deep learning models. However, AdamW introduces a regularization function in order to address the drawback of the way Adam decays the weights and causes a problem in the generalization of the models. (8,9)
    - RMSProp: This optimizer algorithm keeps the moving averages of the squared gradients for each weight, resulting in the acceleration of the learning to the desired direction. (10)
- Epochs: The different epochs utilized were 30 and 50.

Beyond providing a list of different values for the various parameters of the models, some extra parameters were set to specific values in order to assist in the training process:

- Batch size: The batch size is the number of images provided to the model during the training process and can be modified to speed up the process accordingly. The value was set to 75 in order to provide a sensible number to the model in accordance with the GPU's capabilities.
- Seed: The seed was set to a specified random number (42) in order to obtain reproducible results
- Task: Set to detect, as required
- Mode: Set to train, as required
- Val: Set to false, in order to not slow down the training process since the validation process would be performed afterwards.
- Cache: Set to True instead of False, in order to load the image into the provided RAM in order to speed the training process by not transferring extra load to the GPU.
- Exist_Ok: Set to true in order to overwrite the models.
- Classes: Set to 87, as many as the number of labels in the available dataset.

It is also worth noting that it was not possible to test various loss functions, since the task was about object detection. Instead, the mAP metric was utilized which essentially measures the ratio to which a bounded box covers the image that is provided for prediction. The exact method utilized to evaluate the mAP metrics is described in the model evaluation section.

6.2.3 Different Models Evaluation

All available trained models (36 in total) were evaluated based on the following metrics:

1) Mean Recall:

   Recall is defined as: (True Positives) / (True Positives + False Negatives)

   True Positives (TP): The number of correctly detected objects.

   False Negatives (FN): The number of objects that are present in the ground truth but were not detected by the model.

   Mean Recall essentially accounts for different object classes or categories in the dataset. It calculates the Recall separately for each object class and then computes the mean (average) Recall across all classes

2) Mean Precision:

   Precision = (True Positives) / (True Positives + False Positives)

   True Positives (TP): The number of correctly predicted positive instances.

   False Positives (FP): The number of instances that were predicted as positive but are actually negative.

   Mean Precision is then defined as: $(1 / N) * \Sigma$ Precision_i

   It provides an overall assessment of how well the model performs in terms of making accurate positive predictions across all classes.

3) mAP@0.5: This metric extends the concept of Average Precision to evaluate a model's performance across multiple classes or categories.

   It essentially calculates the AP individually for each class and then computes the mean (average) of these AP values.

   It is defined as:

   $mAP = (1 / N) * \Sigma AP_i$

   N: The total number of classes or categories.

   AP_i: The Average Precision for class i.

4) mAP@0.5-0.95: mAP over IoU thresholds from 0.5 to 0.95, with a step of 0.05. All values from each level (0.5, 0.55, 0.60 etc) are pooled together and then their average is taken.
5) Mean mAR: It is the average value of all the above metrics (mean recall, mean precision, mAP 0.5, mAP 0.5-0.95).

The comparison for each model was based on the Mean mAR value in order to keep the best model and proceed with its finetuning.

The following table features the summarized result of the evaluation process for all 36 models:

| Model Name | Mean Precision | Mean Recall | mAP@0.5 | mAP@0.5-0.95 | Mean mAR |
|---|---|---|---|---|---|
| mod_yolov8m_afLeakyReLU_optSGD_epochs50 | 0,885757916 | 0,751785629 | 0,8489507 | 0,648409535 | 0,783725945 |
| mod_yolov8m_afSiLU_optSGD_epochs50 | 0,885757916 | 0,751785629 | 0,8489507 | 0,648409535 | 0,783725945 |
| mod_yolov8m_afSwish_optSGD_epochs50 | 0,885757916 | 0,751785629 | 0,8489507 | 0,648409535 | 0,783725945 |
| mod_yolov8m_afSwish_optSGD_epochs30 | 0,770150722 | 0,771091692 | 0,826689438 | 0,611578535 | 0,744877597 |
| mod_yolov8m_afLeakyReLU_optSGD_epochs30 | 0,903874852 | 0,610607098 | 0,753446736 | 0,54658031 | 0,703627249 |
| mod_yolov8m_afSiLU_optSGD_epochs30 | 0,876146668 | 0,476999139 | 0,596177063 | 0,432502402 | 0,595456318 |
| mod_yolov8n_afLeakyReLU_optSGD_epochs50 | 0,738791233 | 0,508777796 | 0,565415443 | 0,400975774 | 0,553490062 |
| mod_yolov8n_afSiLU_optSGD_epochs50 | 0,738791233 | 0,508777796 | 0,565415443 | 0,400975774 | 0,553490062 |
| mod_yolov8n_afSwish_optSGD_epochs50 | 0,738791233 | 0,508777796 | 0,565415443 | 0,400975774 | 0,553490062 |
| mod_yolov8m_afLeakyReLU_optAdamW_epochs50 | 0,754729815 | 0,471268363 | 0,558790795 | 0,389228612 | 0,543504396 |
| mod_yolov8m_afSiLU_optAdamW_epochs50 | 0,754729815 | 0,471268363 | 0,558546498 | 0,389186114 | 0,543432697 |
| mod_yolov8m_afSwish_optAdamW_epochs50 | 0,756375527 | 0,414749438 | 0,531744361 | 0,361974374 | 0,516210925 |
| mod_yolov8n_afLeakyReLU_optAdamW_epochs50 | 0,706846606 | 0,433548702 | 0,539367752 | 0,355783615 | 0,508886669 |
| mod_yolov8n_afSiLU_optAdamW_epochs50 | 0,706846606 | 0,433548702 | 0,539367752 | 0,355783615 | 0,508886669 |
| mod_yolov8n_afSwish_optAdamW_epochs50 | 0,706846606 | 0,433548702 | 0,539367752 | 0,355783615 | 0,508886669 |
| mod_yolov8n_afSiLU_optSGD_epochs30 | 0,661118302 | 0,303223472 | 0,397849145 | 0,27584779 | 0,409509677 |
| mod_yolov8n_afSwish_optSGD_epochs30 | 0,661118302 | 0,303223472 | 0,397849145 | 0,27584779 | 0,409509677 |

| | | | | | |
|---|---|---|---|---|---|
| mod_yolov8n_afLeakyReLU_optSGD_epochs30 | 0,656854398 | 0,298266555 | 0,388076678 | 0,271374225 | 0,403642964 |
| mod_yolov8m_afSwish_optAdamW_epochs30 | 0,612408534 | 0,306149214 | 0,381789327 | 0,246942034 | 0,386822277 |
| mod_yolov8m_afLeakyReLU_optAdamW_epochs30 | 0,619053785 | 0,303627874 | 0,375488604 | 0,244263229 | 0,385608373 |
| mod_yolov8n_afLeakyReLU_optAdamW_epochs30 | 0,638361414 | 0,285844386 | 0,374103769 | 0,234967935 | 0,383319376 |
| mod_yolov8n_afSiLU_optAdamW_epochs30 | 0,638361414 | 0,285844386 | 0,374103769 | 0,234967935 | 0,383319376 |
| mod_yolov8n_afSwish_optAdamW_epochs30 | 0,638361414 | 0,285844386 | 0,374103769 | 0,234967935 | 0,383319376 |
| mod_yolov8m_afSiLU_optAdamW_epochs30 | 0,609784669 | 0,297509341 | 0,372662784 | 0,242424801 | 0,380595399 |
| mod_yolov8n_afLeakyReLU_optRMSProp_epochs50 | 0,010524004 | 0,152619296 | 0,020178062 | 0,008692249 | 0,048003403 |
| mod_yolov8n_afSiLU_optRMSProp_epochs50 | 0,010524004 | 0,152619296 | 0,020178062 | 0,008692249 | 0,048003403 |
| mod_yolov8n_afSwish_optRMSProp_epochs50 | 0,010222253 | 0,145052641 | 0,020040192 | 0,008597798 | 0,045978221 |
| mod_yolov8n_afLeakyReLU_optRMSProp_epochs30 | 0,003008209 | 0,109251406 | 0,006311139 | 0,00239296 | 0,030240929 |
| mod_yolov8n_afSiLU_optRMSProp_epochs30 | 0,003008209 | 0,109251406 | 0,006311139 | 0,00239296 | 0,030240929 |
| mod_yolov8n_afSwish_optRMSProp_epochs30 | 0,003008209 | 0,109251406 | 0,006311139 | 0,00239296 | 0,030240929 |
| mod_yolov8m_afLeakyReLU_optRMSProp_epochs30 | 0 | 0 | 0 | 0 | 0 |
| mod_yolov8m_afLeakyReLU_optRMSProp_epochs50 | 0 | 0 | 0 | 0 | 0 |
| mod_yolov8m_afSiLU_optRMSProp_epochs30 | 0 | 0 | 0 | 0 | 0 |
| mod_yolov8m_afSiLU_optRMSProp_epochs50 | 0 | 0 | 0 | 0 | 0 |
| mod_yolov8m_afSwish_optRMSProp_epochs30 | 0 | 0 | 0 | 0 | 0 |
| mod_yolov8m_afSwish_optRMSProp_epochs50 | 0 | 0 | 0 | 0 | 0 |

*Table 2 - Different Metrics Values for all 36 models tried.*

From the above table, it is evident that 50 epochs and SGD optimizer are some of the parameters that produce the best results.

Therefore, it was decided to keep those values and tune all the other hyperparameters. As it would be required to perform over 300 steps for a proper tuning but there is a lack of resources for that, only 25 steps were performed eventually.

### 6.2.4 Model FineTuning

Now, it is time to write a new yaml file, containing both the location of train and validation datasets, number of classes and the classes as far as the tuned hyperparameters.

In order to perform the fine tuning step, it was necessary to create a new .yaml file that contains the following information:

- Train dataset location
- Validation dataset location
- Number of classes
- Tuned hyperparameter values

Image 3 below showcases how the best model is stored. For each run, the algorithm outputs a fitted value based on the values of parameters in each run and after all the runs chooses the parameters that result in the best fitted value.

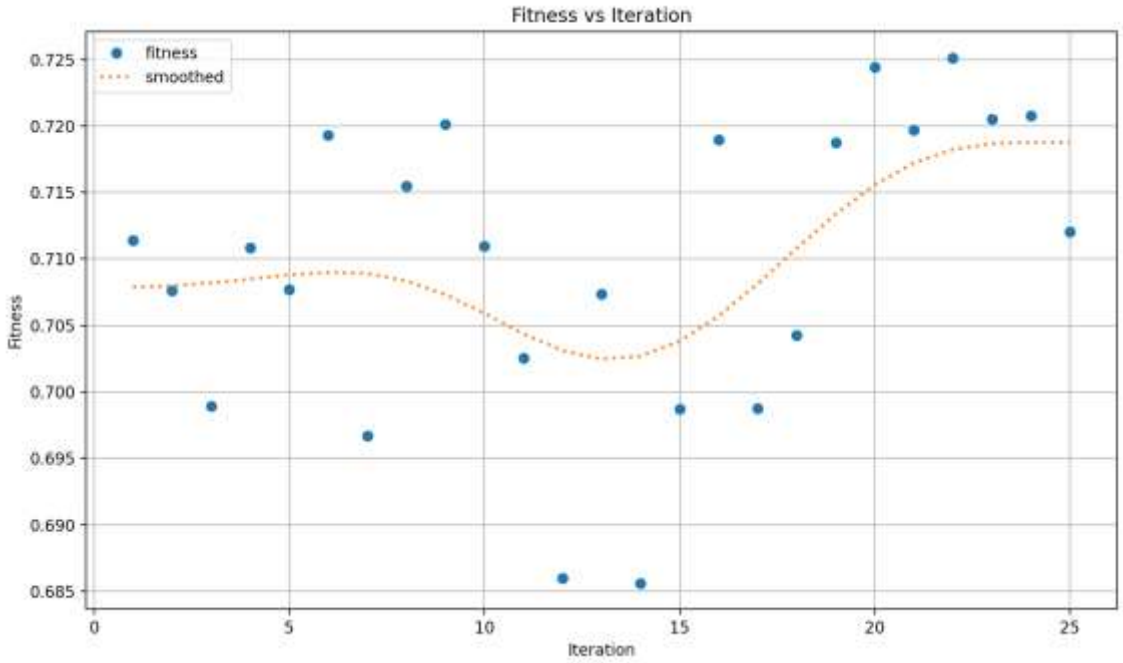


*Image 3 - Hyperparameter evaluation - Fitted values plot.*

Then, the model and its parameters with the best fitted value is stored as the best one.

6.2.5 Model ReTraining

As the 3 best resulting models from the initial training steps had the same scores, it was decided to retrain all of them using the resulting hyperparameter values from the fine tuning step.

The hyperparameters used and their respective values are shown in the list below:

- lr0: 0.01915
- lrf: 0.01034
- momentum: 0.94117
- weight_decay: 0.00032
- warmup_epochs: 2.3912
- warmup_momentum: 0.55326
- box: 7.95282
- cls: 0.58789
- dfl: 0.96978
- hsv_h: 0.01366
- hsv_s: 0.52124
- hsv_v: 0.35768
- degrees: 0.0
- translate: 0.0659
- scale: 0.27582
- shear: 0.0
- perspective: 0.0
- flipud: 0.0
- fliplr: 0.44663
- mosaic: 0.98286
- mixup: 0.0
- copy_paste: 0.0

6.2.6 Model Selection

The retraining of the best 3 models with the best hyperparameters values, resulted in the following models with parameters shown in Table 3 below:

| **Model Name** | **Mean Precision** | **Mean Recall** | **mAP@0.5** | **mAP@0.5-0.95** | **Mean mAR** |
|---|---|---|---|---|---|
| final_mod_yolov8m_afSiLU_optSGD_epochs50 | 0,886240302 | 0,770729013 | 0,872777745 | 0,647865438 | 0,794403125 |
| final_mod_yolov8m_afSwish_optSGD_epochs50 | 0,874672583 | 0,737006283 | 0,842548664 | 0,643817356 | 0,774511222 |
| final_mod_yolov8m_afLeakyReLU_optSGD_epochs50 | 0,906068394 | 0,710403903 | 0,814814968 | 0,596735161 | 0,757005606 |

Table 3 - Best 3 Models Metrics

The final best model has the following characteristics:

1) Model version: **YOLOv8m** (medium)
2) Activation function: **SiLU**
3) Optimizer algorithm: **SGD**
4) Number of epochs: **50** epochs.

Since a best model was finally selected, the next step was to use that model to predict and evaluate the results.

Image 4 below shows the Precision-Recall curve:

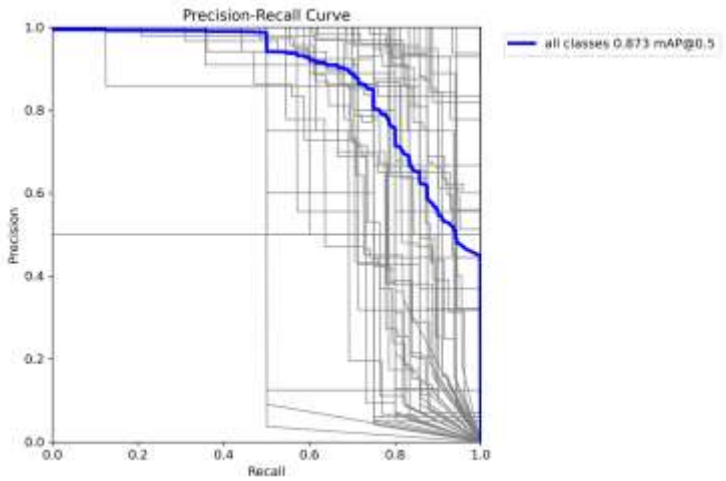


*Image 4 - Precision/Recall Curve.*

It is evident from Image 4 that the Precision-Recall curve follows a somewhat expected behavior where the precision and recall begin to deteriorate after the 0.5 probability mark.

Image 5 below shows the resulting labeled image with bounded boxes from the model:

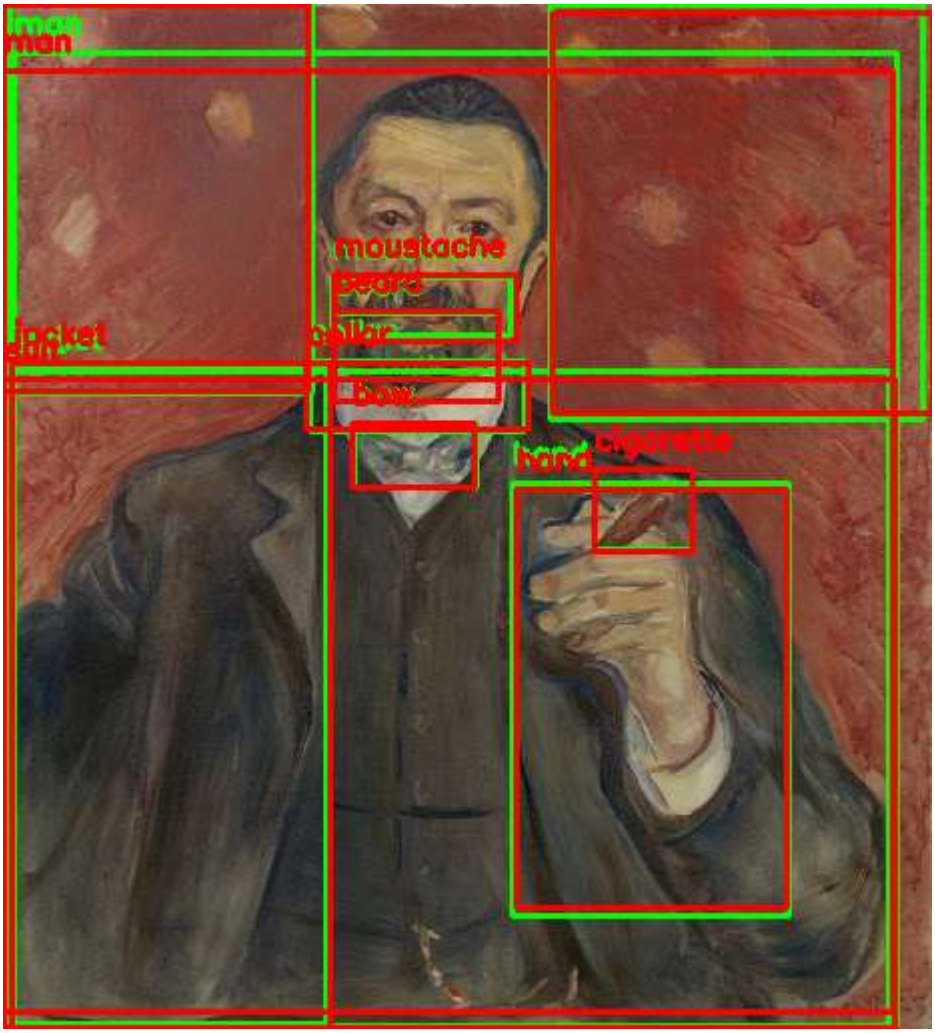


*Image 5 - Painting "Portrait of Felix Auerbach" by Edvard Munch.*

All red bounding boxes are the true bounding boxes, while all green bounding boxes are those predicted by our model.

For the above image, we get the following labels:

**cigarette** [319.93572998046875, 252.6908721923828, 48.428314208984375, 40.384368896484375]

**jacket** [81.81804656982422, 345.4493103027344, 158.9314422607422, 327.6328125]

**suit** [223.13778686523438, 347.5521545410156, 439.12060546875, 328.89569091796875]

**wall** [79.1089096069336, 97.4341049194336, 151.5585479736328, 194.8682098388672]

**wall** [366.7974548339844, 104.30216979980469, 186.01318359375, 207.354248046875]

**collar** [207.094482421875, 195.85699462890625, 107.11737060546875, 34.32856750488281]

**bow** [206.021240234375, 225.54232788085938, 61.33436584472656, 32.65068054199219]

**man** [225.69436645507812, 267.30352783203125, 440.88726806640625, 486.01171875]

**mustache** [209.26239013671875, 150.90036010742188, 88.54649353027344, 29.251602172851562]

**hand** [323.3612365722656, 347.7980651855469, 138.68701171875, 216.00816345214844]

**beard** [206.28912353515625, 176.6114501953125, 81.72068786621094, 44.689056396484375]

A more challenging image (with more labels to predict) with the corresponding results is shown below:

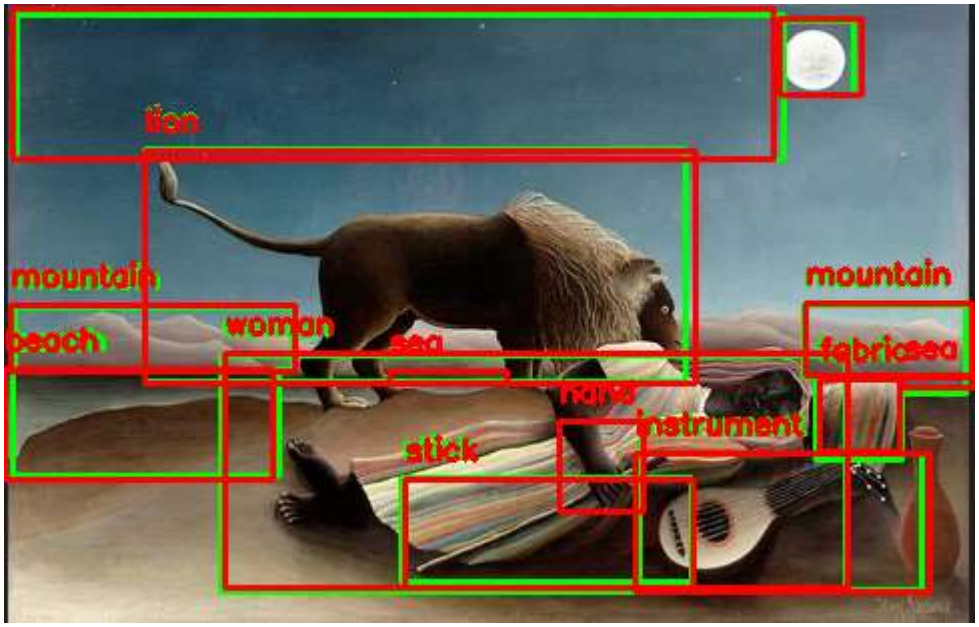


*Image 6 - Painting "The sleeping Gypsy" by Henri Rousseau.*

All red bounding boxes are the true bounding boxes, while all green bounding boxes are those predicted by our model.

For the above image, we get the following labels:

**lion** [205.20472717285156, 130.57998657226562, 270.0575256347656, 115.50276947021484]

**mountain** [439.6497802734375, 168.33013916015625, 80.96676635742188, 35.86024475097656]

**woman** [264.6780090332031, 234.62149047851562, 312.2446594238281, 117.95729064941406]

**sky** [197.7666015625, 41.26252365112305, 383.191650390625, 72.3783950805664]

**fabric** [426.78515625, 207.3651885986328, 42.59698486328125, 39.29779052734375]

**beach** [70.83949279785156, 208.77940368652344, 133.21792602539062, 53.262969970703125]

**mountain** [74.72250366210938, 166.93804931640625, 139.37814331054688, 31.24957275390625]

**stick** [271.055419921875, 262.72296142578125, 145.3410186767578, 51.84429931640625]

**moon** [404.85845947265625, 26.839141845703125, 38.74517822265625, 36.563812255859375]

**hand** [298.219482421875, 231.07574462890625, 41.149505615234375, 45.43208312988281]

**instrument** [387.622314453125, 258.1241455078125, 140.94747924804688, 67.85720825195312]

**sea** [464.91082763671875, 190.761962890625, 31.362274169921875, 7.8019866943359375]

## 6.3 NLP Tool

6.3.1 Scope

The next step in the application is where the labels that are the output of the object detection model can be easily transformed into a cohesive description of the art image that can then be conveyed via audio to the end user.

In order to achieve this goal, it was decided to connect to Google's Bard API, which is an NLP model that can provide a text description based on the provided labels.

Google Bard provides a very fast and efficient way of meeting the application's goal of providing an audio description, without requiring the development load of a new NLP model since the focus of the development of the application is an object detection model and not an NLP or text to audio model.

6.3.2 Implementation

Connection to the Google Bard API was easy enough through the library available for Python, however it was necessary to obtain a token for using not through a subscription but rather than the session available in the browser (essentially, it was necessary to have an open Bard session in the browser to obtain the token and use it while running the code).

Sample image:

*Image 7 - Painting "Portrait of Felix Auerbach" by Edvard Munch.*

Resulting description:

```
A man is wearing a jacket, suit, and tie. He has a mustache, beard,
and cigarette in his hand. His collar is turned up and he has a bow
tie on. His beard is neatly trimmed. He is standing in front of a
wall.
```

The description above seems to be really close to the actual art image and thus, this validates that our model works in a satisfactory way.

Another Sample image (*more demanding with more labels):

*Image 8 - Painting "The sleeping Gypsy" by Henri Rousseau.*

Resulting description:

A woman is standing on a beach, playing a musical instrument. She is facing the ocean, which is behind her. The sky is blue and clear, and the moon is visible in the upper left corner of the image. In the foreground of the image, there is a lion lying on the beach. The lion is facing the woman, and its tail is curled around its body. To the right of the lion, there is a mountain range. The woman is wearing a long dress, and she has her hair tied back in a ponytail. She is holding a stick in her right hand and an instrument in her left hand. The beach is sandy and has a few palm trees scattered around. The water in the ocean is blue and calm.

As it seems, there are some minor details that google Bard has wrongly added, which do not exist in the art image provided such as:

- **standing** on a beach
- She is **facing** the ocean
- lion **lying** on
- its tail is **curled** around
- hair tied back in a **ponytail**
- a few **palm** trees scattered

The above mistakes are not a result of the object detection model but rather they have been added by google Bard to enhance the description.

## 6.4 Text to Audio Tool

6.4.1 Scope

The final step in the application is to deliver an audio recording of the text description provided by Google Bard API to the end user, which is after all the main goal of the application.

The tool selected for the aforementioned task is the gTTS python library, as it provides a fast and easy integration with the rest of the architecture of the application.

6.4.2 Implementation

A simple import of the gTTS library in the python script is enough to initialize an 'audio' object which takes as arguments the text descriptions from the previous step.

# 7. Notes and Future Work

- Additional Datasets

While our primary dataset is integral to our project, we acknowledge the potential value of larger datasets as well as supplementary datasets with a larger variety of art movements as well as containing textual descriptions of artworks, historical context, and audio descriptions. These additional resources could further enrich our analysis and broaden the scope of accessibility.

- Further model finetuning

The final proposed model is evidently not in the optimal state, which would require training and fine tuning to a much larger extent. The main focus would be (along with a much larger dataset) a possible freezing of a certain percent of layers in order to retrain some of the model layers specifically on the characteristics of art images, further improving the quality of the predictions of the model. This would of course lead to a much higher quality of audio descriptions to the end user, greatly improving his experience.

- Mobile Application Development

To elevate our project and lead the way for a successful market launch, our focus is on two essential objectives: making our model accessible to end users and enhancing their engagement with artworks.

An application that could be utilized by end users would follow the following architecture:

1) Develop a REST API that is capable of receiving an image and returning an audio description
2) Host the application's codebase (Python script that includes final model and API to utilize it) on a server.
3) Develop a mobile application that consumes the aforementioned API

In conjunction with a targeted marketing strategy that capitalizes on service opportunities and reach of our intended audience, it could lead to an application that could be used in a wide variety of art related spaces such as museums, art galleries etc.

# 8. Members/Roles

1. Panagiotis Vaidomarkakis
    a. Role: Code Development
    b. Responsibility: Leading the development of the project's codebase.
2. Alexandros Lemonidis
    a. Role: Code Development and Technical Documentation
    b. Responsibilities: Contributing to the code development efforts and codebase, collaborating with Panagiotis Vaidomarkakis. Leading the creation of technical documentation to ensure comprehensive understanding and usability of the codebase.
3. Katerina Gioupai
    a. Role: Dataset Preparation and Report
    b. Responsibilities: Managing and leading the data preprocessing phase, ensuring data quality and relevance. Along with Panagiotis Vaidomarkakis and Alexandros Lemonidis wrote the report documenting project details and outcomes.

Collaboration and communication among us was a key to project deliverable.

# 9. Time Plan

**Weeks 1,2:** Pre-study Phase
- Business Case Definition/Development
- Technical Feasibility and Approach Study

**Weeks 3,4:** Design Phase
- Architecture Design
- Tool and Libraries Selection
- Methodology Definition

**Week 5:** Content Phase
- Dataset collection & preparation

**Weeks 6-8:** Development Phase
- Deep learning algorithm for object detection implementation
    - Training
    - Fine Tuning
    - Retraining
    - Prediction
- Supplementary tools and libraries integration

**Weeks 9,10:** Final Work
- Final Report Deliverable

# 10. Bibliography


1) Reluplex made more practical: Leaky ReLU, Jin Xu; Zishan Li; Bowen Du; Miaomiao Zhang; Jing Liu
2) Target Recognition Based on CNN with LeakyReLU and PReLU Activation Functions, Tongtong Jiang; Jinyong Cheng
3) Analyzing the impact of activation functions on the performance of the data-driven gait model, Bharat Singh, Suchit Patel, Ankit Vijayvargiya, Rajesh Kumar
4) SinLU: Sinu-Sigmoidal Linear Unit, by Yiting Li, Qingsong Fan, ORCID, Haisong Huang, Zheng gong Han, Qiang Gu
5) Enhancement of Deep Learning in Image Classification Performance Using Xception with the Swish Activation Function for Colorectal Polyp Preliminary Screening, Natinai Jinsakul, Cheng-Fa Tsai, Chia-En Tsai, Pensee Wu
6) A Brief Review of the Most Recent Activation Functions for Neural Networks, Marina Adriana Mercioni; Stefan Holban
7) Improving Generalization Performance by Switching from Adam to SGD, Nitish Shirish Keskar, Richard Socher
8) Large-batch Optimization for Dense Visual Predictions, Zeyue Xue;Jianming Liang; Guanglu Song;Zhuofan Zong;Liang Chen;Yu Liu;Ping Luo
9) Why AdamW matters: https://towardsdatascience.com/why-adamw-matters-736223f31b5d
10) Improving the Rprop Learning Algorithm, Christian Igel;Michael Hüsken
11) "*YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection"*, Muhammad Hussain, University of Huddersfield https://www.researchgate.net/publication/371875367_YOLO-v1_to_YOLO-v8_the_Rise_of_YOLO_and_Its_Complementary_Nature_toward_Digital_Manufacturing_and_Industrial_Defect_Detection
12) "*YOLO: Algorithm for Object Detection Explained [+Examples]*", Rohit Kundu, Ph.D. in the Electrical and Computer Engineering department of the University of California, Riverside. https://www.v7labs.com/blog/yolo-object-detection
13) "DC-YOLOv8: Small-Size Object Detection Algorithm Based on Camera Sensor" https://www.mdpi.com/2079-9292/12/10/2323
14) Ultralytics YOLOv8 Docs https://docs.ultralytics.com/models/yolov8/#overview