AUTOMATED IMAGE TAGGING USING YOLOV5

by

Harrison Hall

A project submitted in conformity with the requirements
for the degree of Bachelor of Science in Computer Science
Western Governors University

Contents

Part A

Letter of Transmittal

June 30, 2024

Jane Doe
StockX
4001 S 700 E #300
Millcreek, UT 84107

Dear Ms. Doe,

I am writing to propose a project that addresses a significant challenge faced by StockX: the manual tagging of images. This process is not only time-consuming but also prone to inconsistencies and errors, which negatively impact the user experience and search functionality of our platform.

Our current system requires extensive manual effort to tag each image with relevant metadata, resulting in delays and inefficiencies. This manual process leads to inconsistencies in tags, making it difficult for users to find the images they need quickly and accurately. To address this problem, I propose the implementation of an automated tagging system using a pre-trained YOLOv5 model. This machine learning model is capable of detecting objects within images and generating appropriate tags automatically. The system will process images, detect

objects, and generate tags, which will be stored in our database.

The proposed solution will bring several benefits to StockX: Automating the tagging process will significantly reduce the time and effort required to tag images, allowing our staff to focus on more strategic tasks. The use of a machine learning model will ensure consistent and accurate tagging, improving the search functionality and user experience on our platform. The system will be capable of handling large volumes of images, supporting the growth of our image library without compromising on tagging quality.

The project will be completed in four phases over three months: Planning (2 weeks), Development (6 weeks), Testing (2 weeks), and Deployment (2 weeks). The initial development cost is estimated at $17,000, with an annual maintenance cost of approximately $1,800 for server hosting and updates. The model will utilize publicly available datasets such as COCO and VOC, ensuring a diverse and comprehensive training dataset. There are no significant ethical concerns, as the data used is publicly available and properly annotated. We will ensure data privacy and security throughout the project.

With a solid background in machine learning and software development, I am confident in my ability to successfully implement this solution. My experience includes developing similar automated systems and integrating machine learning models into existing workflows. I believe this project will provide substantial improvements to our image tagging process, enhancing the overall efficiency and user satisfaction of our platform. I look forward to discussing this proposal further and obtaining your approval to proceed.

Sincerely,

Harrison Hall,
Software Engineer

Part B

Project Proposal Plan

1. Project Summary

1.1. The Problem

The client StockX (a stock photo website), currently relies on manual tagging of images to categorize and manage their extensive image library. This process is time-consuming, prone to errors, and inconsistent, resulting in inefficiencies and poor search results for users.

1.2. Client's Needs

Given these challenges, the client requires a more efficient and accurate solution to manage their image tagging such as Automation. Automating a process would not only ensure consistency in tagging but also significantly increase the efficiency of the image tagging process and reduce the manual workload of their staff. This could ultimately lead to an overall better user experience by providing more relevant search results and quicker access to desired images.

1.3. Deliverables

The project will include several key deliverables to ensure a comprehensive solution to the client's needs:

Automated Tagging Application. A Python-based server that uses a pre-trained YOLOv5 model to detect objects in uploaded images and generate tags automatically.

User Guide. Comprehensive documentation that includes setup instructions, usage guidelines, and examples of how to use the application.

Visualization Outputs. Generated visualizations, including images with bounding boxes, confidence distribution bar plots, and class distribution pie charts, to provide insights into the tagging process.

1.4. Benefits to the Client

The automated tagging application will significantly reduce the time and effort required for manual tagging, increase the accuracy and consistency of image tags, and enhance the overall user experience on the stock photo website. By automating the tagging process, the client can focus on other critical tasks, improve the search functionality for users, and increase user satisfaction.

2. Project Summary

2.1. Source of Data

The raw data used to train the YOLOv5 model comes from publicly available datasets, such as COCO (Common Objects in Context) and VOC (Visual Object Classes). These datasets contain annotated images for various object categories and are widely used in the computer vision community.

2.2. Data Processing and Management

Throughout the application development life cycle, the data will be processed and managed as follows:

Design. Preprocess the raw datasets to standardize image sizes and annotations. Split the data into training, validation, and test sets.

Development. Train the YOLOv5 model using the preprocessed data. Apply data augmentation techniques to improve the model's robustness.

Maintenance. Store the trained model and processed datasets in a structured format. Regularly update the datasets and retrain the model as needed.

Others. Evaluate the model using a separate test set to assess performance. Integrate the trained model into the application for deployment.

2.3. Data Justification and Handling Anomalies

The COCO and VOC datasets are comprehensive and widely recognized in the industry for training object detection models. These datasets are well-annotated and contain a diverse range of objects, ensuring the model is robust and accurate. Any data anomalies, such as outliers or incomplete data, will be addressed through preprocessing steps, including data cleaning and augmentation.

2.4. Ethical and Legal Concerns

There are no ethical or legal concerns regarding the use of COCO and VOC datasets, as they are publicly available and commonly used in research and commercial projects. Proper attribution will be given to the sources of these datasets.

3. Implementation

3.1. Methodology

The project will follow the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology, which includes the following phases:

Business Understanding

Define project objectives and requirements.

Data Understanding

Collect and preprocess data.

Data Preparation

Clean and transform data for modeling.

Modeling

Train and evaluate the YOLOv5 model.

Evaluation

Assess the model's performance and validate results.

Deployment

Implement the model into the application and deploy.

3.2. Implementation Plan

The project will be implemented through the following steps:

Define Requirements

Understand the client's needs and define project goals.

Application Development

Develop the Python server and integrate the pre-trained model.

Testing and Validation

Test the application and validate the model's performance.

Deployment

Deploy the application and provide user documentation.

Maintenance and Updates

Monitor the application, update datasets, and retrain the model as needed.

4. Timeline

The timeline below outlines the duration of each implementation phase:

Milestone	Start Date	End Date
Define Requirements	Day 1	Day 7
Application Development	Day 8	Day 35
Testing and Validation	Day 36	Day 49
Deployment	Day 50	Day 56
Maintenance and Updates	Ongoing	Ongoing

5. Evaluation Plan

5.1. Verification Methods

To ensure the successful implementation of the project, we will employ the following

verification methods:

Design Stage. Conduct peer reviews of project requirements and design documents to ensure completeness, accuracy, and feasibility within the project scope.

Data Collection and Preparation Stage. Perform data quality checks to ensure datasets are complete, accurate, and free of anomalies. Verify that data preprocessing steps (e.g., resizing, augmentation) are correctly implemented and do not introduce errors.

Model Training Stage. Use cross-validation techniques to ensure the model is not overfitting and performs well on unseen data. Track key performance metrics such as precision, recall, and mean Average Precision (mAP) during training to monitor model performance.

Application Development Stage. Write and execute unit tests for individual components of the application to ensure they function correctly. Perform integration testing to verify that different components of the application work together seamlessly.

Testing and Validation. Conduct functional testing to ensure the application meets all specified requirements and performs the intended tasks correctly. Engage end-users in testing the application to validate that it meets their needs and expectations.

5.2. Validation Methods

For validation, the following methods will be used upon completion of the project:

End-to-End Testing. Perform end-to-end testing of the entire application to ensure all components work together as expected and the system performs efficiently.

Performance Evaluation. Validate the performance of the YOLOv5 model on a separate test set to ensure it meets the desired accuracy and robustness criteria.

Client Feedback. Gather feedback from the client and make necessary adjustments to ensure the application fully addresses their needs and provides the expected benefits.

6. Resources and Costs

6.1. Hardware Costs

The project requires the following hardware costs:

Hardware	Cost
Development Machine	$1,500 (One-time)
Server for Hosting	$50/month

6.2. Software Costs

The software required for the project includes various open-source tools, all of which are free of charge:

Software	Cost
Python and Libraries	Free
Flask	Free
YOLOv5 Model	Free
OpenCV	Free
Matplotlib and Seaborn	Free

6.3. Estimated Time and Labor Costs

The project involves the efforts of a project manager, data scientist, and software developer. The table below provides an estimate of the labor time and associated costs:

Role	Rate	Hours	Total Cost
Project Manager	$50/hour	50	$2,500
Data Scientist	$60/hour	60	$3,600
Software Developer	$55/hour	70	$3,850

6.4. Total Cost Overview

The table below provides a summary of the initial development costs and the annual maintenance costs:

Cost Type	Amount
Initial Development Cost	$17,000
Annual Maintenance Cost	$1,8000/year

Part C

Application

1. Project Summary

1.1. The Problem

The client StockX (a stock photo website), currently relies on manual tagging of images to categorize and manage their extensive image library. This process is time-consuming, prone to errors, and inconsistent, resulting in inefficiencies and poor search results for users.

Part D

Post-implementation Report

1. Solution Summary

1.1. Problem and Solution Summary

The stock photo website faced a significant challenge with manually tagging images, which was time-consuming and error-prone. This inefficiency led to poor search results and user experience. The solution was to develop an automated tagging system using a pre-trained YOLOv5 model. This system detects objects within images and generates tags automatically, streamlining the tagging process and improving the accuracy and consistency of image categorization.

1.2. Application as a Solution

The application processes images uploaded to a Python server, uses the YOLOv5 model to detect objects, and generates relevant tags. It saves the processed images with bounding boxes, along with confidence distribution and class distribution visualizations. This automation reduces the manual workload, enhances tagging accuracy, and improves search functionality on the website.

2. Solution Summary

2.1. Source of Data

The raw data used to train the YOLOv5 model came from publicly available datasets such as COCO (Common Objects in Context) and VOC (Visual Object Classes). These datasets include annotated images across various object categories, collected from the internet and labeled by human annotators.

2.2. Data Processing and Management

Throughout the application development life cycle, the data was processed and managed as follows:

Design. The raw datasets were preprocessed to standardize image sizes and annotations. The data was split into training, validation, and test sets to ensure robust model training and evaluation.

Development. The preprocessed data was used to train the YOLOv5 model. Data augmentation techniques, such as rotation, flipping, and scaling, were applied to enhance the model's robustness. The trained model was then integrated into the application.

Maintenance. The trained model and processed datasets were stored in a structured format, with regular backups to ensure data integrity. The model was periodically retrained with updated datasets to maintain performance.

Evaluation. The model's performance was evaluated using a separate test set, tracking key metrics like precision, recall, and mean Average Precision (mAP). These metrics were used to fine-tune the model and ensure its accuracy.

3. Machine Learning

3.1. Method Identification (The "What")

The method employed is a supervised machine learning technique using the YOLOv5 (You Only Look Once) object detection model. This model detects and classifies objects within images, drawing bounding boxes around detected objects and assigning labels to them.

3.2. Method Development (The "How")

The YOLOv5 model was pre-trained on large-scale datasets such as COCO and VOC. These datasets provided a diverse range of annotated images, which were used to train the model. Data augmentation techniques, such as rotation, flipping, and scaling, were applied to enhance the model's robustness. The model was trained using supervised learning to recognize and classify objects in new images.

3.3. Method Justification (The "Why")

The YOLOv5 model was selected for its high accuracy and real-time performance capabilities. Its ability to detect multiple objects within a single image and draw bounding boxes with high precision made it an ideal choice for automating the tagging process. The pre-trained nature of the model reduced development time, allowing for quick deployment and integration into the application.

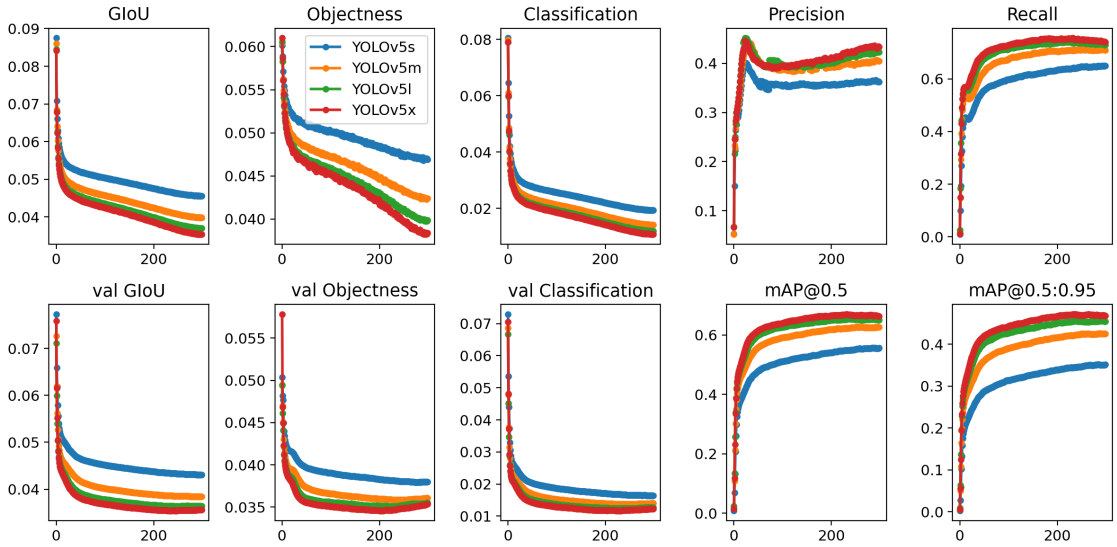
4. Validation

4.1. Validation Method

The validation method used for the YOLOv5 model included tracking key performance metrics such as GIoU, objectness, classification loss, precision, recall, and mean Average Precision (mAP) during training to ensure the model's accuracy and robustness.

4.2. Validation Results

The graph below, sourced from the YOLOv5 GitHub page, visually represents the performance metrics.



The following table summarizes the results of the validation method applied to the YOLOv5l (large) model from the graph above:

Metric	Result
GIoU Loss	Approx. 0.04
Objectness Loss	Approx. 0.04
Classification Loss	Approx. 0.04
Precision	Approx. 0.04
Recall	Approx. 0.7
Validation GIoU Loss	Approx. 0.04
Validation Objectness Loss	Approx. 0.035
Validation Classification Loss	Approx. 0.015
mAP@0.5	Approx. 0.06
mAP@0.5:0.95	Approx. 0.040

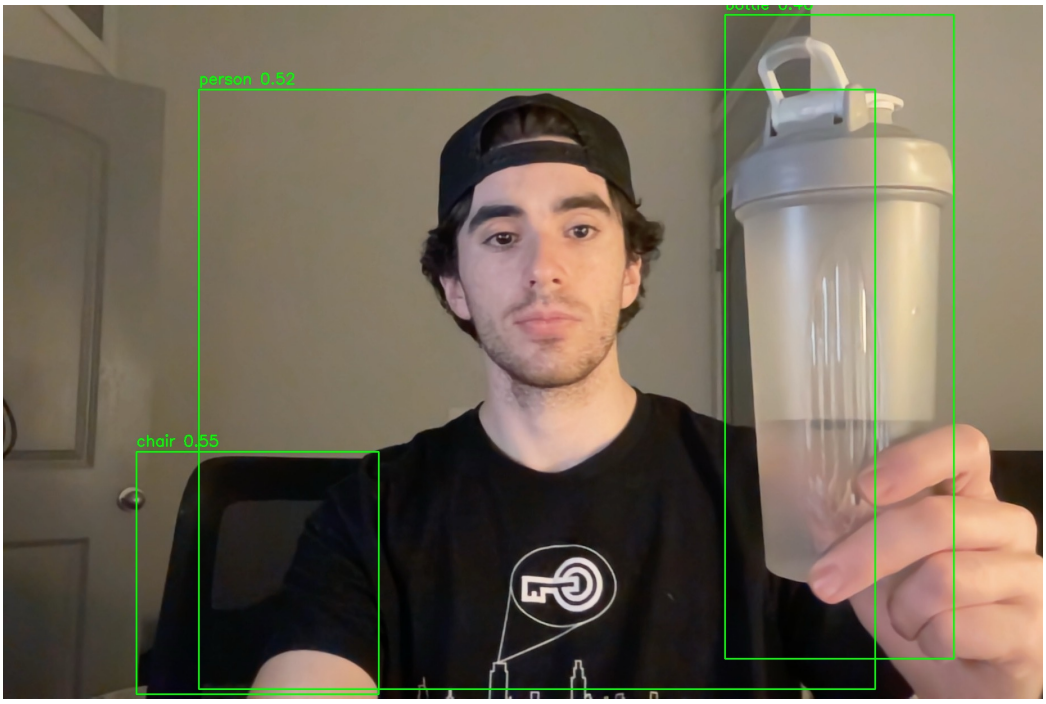
5. Visualizations

5.1. Visualization Location

The visualizations are primarily located and can be found within the webpage application's user-flow, specifically in the second step of the Image Tagging process titled "View Results". This is where a user can view their provided results (i.e. generated tags) and visualizations.

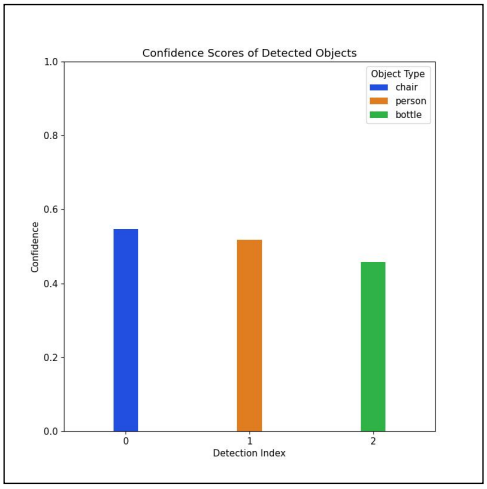
5.2. Sample Visualizations

Below are examples of the visualizations included in the webpage application. These visualizations provide insights into the detected objects within the images.

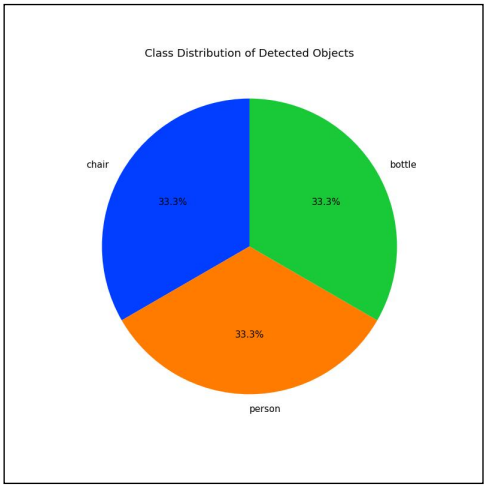


(a) An example of the sample image with bounding boxes applied on the detected objects

(a) **Sample Image with Bounding Boxes.** This visualization shows the detected objects within an image, with each object highlighted by a bounding box and labeled with its confidence score.



(b) An example of the Confidence Distribution



(c) An example of the Class Distribution

(b) **Confidence Distribution Bar Chart.** The bar chart above displays the confidence scores of the detected objects, providing a visual representation of the model's certainty in its detections.

(c) **Class Distribution Pie Chart.** The pie chart above illustrates the distribution of different classes of detected objects, giving an overview of the types of objects present in the image.

6. User Guide

6.1. Validation Method

The validation method used for the YOLOv5 model included tracking key performance metrics such as GIoU, objectness, classification loss, precision, recall, and mean Average Precision (mAP) during training to ensure the model's accuracy and robustness.

References

Jocher, G. (2020). YOLOv5 by Ultralytics (Version 7.0) [Software]. Zenodo. https://doi.org/10.5281/zenodo.3908559. Retrieved from https://github.com/ultralytics/yolov5

Jocher, G. (2020). YOLOv5 by Ultralytics (Version 7.0) [Software]. Zenodo. https://doi.org/10.5281/zenodo.3908559. Retrieved from https://github.com/ultralytics/yolov5

Jocher, G. (2020). YOLOv5 by Ultralytics (Version 7.0) [Software]. Zenodo. https://doi.org/10.5281/zenodo.3908559. Retrieved from https://github.com/ultralytics/yolov5