INTERNSHIP REPORT

B.Tech in Computer Science Engineering

DIVISION OF COMPUTER SCIENCE ENGINEERING

SCHOOL OF ENGINEERING

CUSAT

CATHERIN MARIA JACOB

20223039

Internship Details:

Wrench Solutions Pvt Ltd

R&D Center ,#16A, B block, Cochin Special Economic Zone

Kochi 682037

Mentor Details:

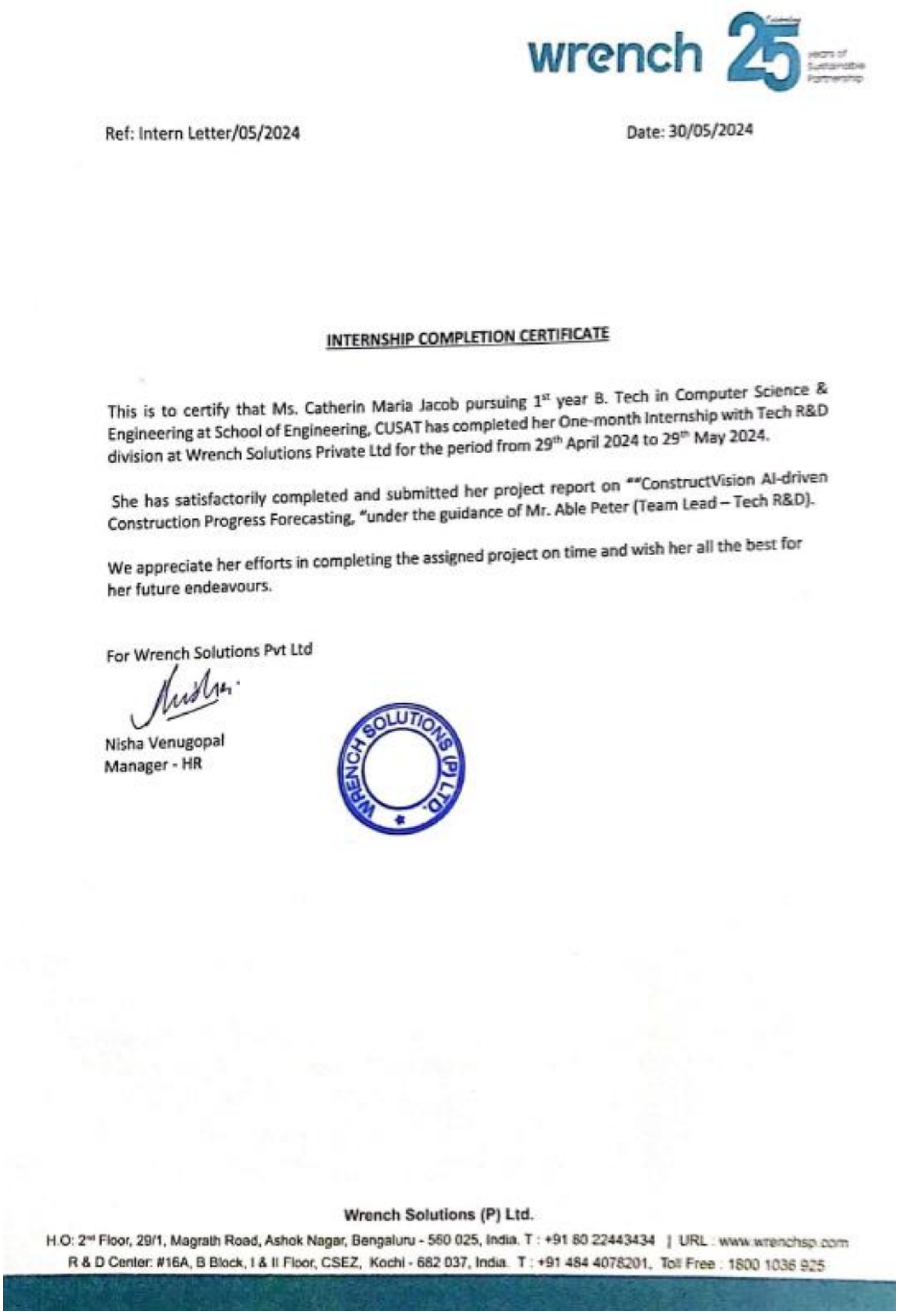
Mr. Able Peter

Technical Lead

able.m@wrenchsolutions.com

+91 94000 18834

MAY 2024



wrench 25 years of Sustainable Partnership

Ref: Intern Letter/05/2024

Date: 30/05/2024

INTERNSHIP COMPLETION CERTIFICATE

This is to certify that Ms. Catherin Maria Jacob pursuing 1st year B. Tech in Computer Science & Engineering at School of Engineering, CUSAT has completed her One-month Internship with Tech R&D division at Wrench Solutions Private Ltd for the period from 29th April 2024 to 29th May 2024.

She has satisfactorily completed and submitted her project report on ""ConstructVision AI-driven Construction Progress Forecasting, "under the guidance of Mr. Able Peter (Team Lead – Tech R&D).

We appreciate her efforts in completing the assigned project on time and wish her all the best for her future endeavours.

For Wrench Solutions Pvt Ltd

[illegible]

Nisha Venugopal
Manager - HR

WRENCH SOLUTIONS (P) LTD.

Wrench Solutions (P) Ltd.

H.O: 2nd Floor, 29/1, Magrath Road, Ashok Nagar, Bengaluru - 560 025, India. T : +91 80 22443434 | URL : www.wrenchsp.com
R & D Center: #16A, B Block, I & II Floor, CSEZ, Kochi - 682 037, India. T : +91 484 4078201, Toll Free : 1800 1036 925

DIVISION OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF ENGINEERING

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

CERTIFICATE

Certified that this is a bonafide record of the internship

done by

CATHERIN MARIA JACOB (20223039)

of III Semester, Computer Science and Engineering in the year 2024 in partial fulfilment requirements for the award of Degree of Bachelor of Technology in Computer Science and Engineering of Cochin University of Science and Technology.

Dr. Pramod Pavithran	Dr .Sudheep Elayidom
Head of Department	Class Coordinator

Acknowledgement

I would like to express my heartfelt gratitude to all those who have supported me throughout my internship project on the ConstructVision: AI-driven Wall Construction Progress Forecasting Web Application. First and foremost, I would like to thank my teachers for their invaluable guidance and encouragement, which inspired me to explore the depths of artificial intelligence, machine learning and web development.

I extend my sincere appreciation to my family for their unwavering support and belief in my abilities. Their constant motivation and encouragement have been instrumental in helping me navigate the challenges of this project.

I am especially grateful to Mr. Able M Peter and Mr. Anuroop Raphael from Wrench Solutions for their exceptional support and insights. Their expertise and encouragement provided me with a deeper understanding of artificial intelligence & machine learning and its challenges, significantly enriching my learning experience.

Thank you all for being an integral part of this journey. Your support has made this project not only possible but also deeply fulfilling.

Abstract

This report details the internship undertaken at Wrench Solutions (CSEZ), Kakkanad, from 29-04-2024 to 29-05-2024, during which a project titled *"Constructvision"* was developed and implemented. This project aims to revolutionize construction progress monitoring by addressing the inherent challenges of traditional inspection methods, which are often labour-intensive, time-consuming, and prone to inaccuracies. This initiative utilizes advanced computer vision and machine learning technologies to automate the prediction of construction stages, thereby enhancing the efficiency and accuracy of project management in the construction industry. By harnessing Convolutional Neural Networks (CNNs) and the YOLOv5 object detection model, ConstructVision seeks to analyse construction images in real-time, offering valuable insights into various stages of wall construction, such as foundation laying, brickwork, plastering, and final completion. The project aims to bridge the gap between conventional practices and modern technological advancements, ultimately empowering stakeholders with timely and reliable data for informed decision-making. ConstructVision's innovative approach not only enhances operational efficiency but also contributes to reducing delays and optimizing resource allocation in construction projects. This project stands as a significant step toward integrating artificial intelligence into construction management, promoting smarter, more effective practices that align with the industry's evolving needs.

CONTENTS

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

The Constructvision internship project addresses a crucial challenge in the construction industry: accurately assessing project progress using AI. Traditional monitoring methods rely heavily on manual inspections and reports, which are time-consuming, labour-intensive, and prone to human error. Such limitations can lead to inaccurate progress assessments, potentially causing delays and inefficiencies. To tackle this, the Constructvision project utilises advanced computer vision and machine learning techniques, specifically Convolutional Neural Networks (CNNs) and the YOLOv5 object detection model, to automate the prediction of construction stages through image analysis. By examining labelled images of wall construction at various stages, the system can predict completion percentages, delivering near real-time insights into project status. This automation allows construction managers to make data-driven decisions, optimise resources, and proactively manage timelines. By bridging the gap between traditional methods and modern technology, Constructvision enhances operational efficiency and accuracy in construction management, leading to smarter, more effective project oversight.

CHAPTER 2 : ORGANIZATION OVERVIEW

2.1 BRIEF OVERVIEW OF INDUSTRY

Wrench Solutions, headquartered in the Cochin Special Economic Zone (CSEZ), Kochi, offers advanced software solutions for the construction industry. Their platform integrates core aspects such as project management, procurement, resource management, and safety, using cutting-edge technologies like AI, cloud computing, and data analytics. This comprehensive solution enhances efficiency and collaboration by providing real-time insights, automated workflows, and centralized project tracking. Wrench Solutions empowers construction teams to monitor progress, manage risks, and ensure compliance, ultimately reducing delays and cost overruns. With its scalable and flexible architecture, the platform is suitable for projects of all sizes, enabling seamless coordination between stakeholders and improving overall project delivery. Wrench Solutions is a trusted partner in modernizing and optimizing construction management processes.

2.2 OBJECTIVE ABOUT THE STUDY/PROJECT

The primary objective of this study was to develop an AI-driven solution that leverages convolutional neural networks (CNNs) to accurately predict construction progress. The focus was on utilizing advanced object detection techniques to classify different stages of construction work based on real-time images, specifically targeting wall construction. By automating the analysis of these images, the model aimed to identify and categorize various construction stages, allowing for the tracking of progress over time. This AI-powered approach not only improves the accuracy of construction monitoring but also provides timely insights for project managers to make informed decisions, optimize resource allocation, and anticipate potential delays. The ultimate goal was to demonstrate how AI can revolutionize traditional construction management practices by providing a reliable, scalable, and automated method for tracking and forecasting progress across construction projects.

CHAPTER 3 PROJECT OVERVIEW

3.1 PROBLEM STATEMENT

The aim of the project is to utilise a CNN model to predict the progress of constructing a wall using images captured during the construction process.

3.2 Purpose of Application

The purpose of the *ConstructVision* project is to revolutionise construction progress forecasting by employing artificial intelligence to automate the tracking of wall construction stages. By utilising labelled images taken at various phases—from foundation to completion—the project aims to deliver precise predictions of construction progress. This allows construction managers to make informed, data-driven decisions, optimise resource allocation, and mitigate delays. The ultimate goal is to enhance project efficiency, reduce costs, and ensure timely delivery of construction projects.

The key objectives are:

- Automate tracking of construction stages.
- Provide accurate progress predictions.
- Enable data-driven decision-making.
- Optimise resource allocation.
- Minimise delays and enhance efficiency.

3.3 Steps Taken

The first step in the project was the collection of wall images at various stages of construction. The dataset was created using the Bing Image Downloader Python library, which allowed for the retrieval of diverse images showing walls in different phases of construction. The images were manually labelled using the LabelImg tool, which assigned labels based on the construction progress, such as:

1. 20%: Foundation stage
2. 40%: Initial bricklaying
3. 60%: Completion of bricklaying
4. 80%: Wall plastering
5. 100%: Completed and painted wall

The annotations were stored in XML format using PASCAL VOC for compatibility with object detection models.

3.3.2 Data Preprocessing & Splitting

For data preprocessing and splitting, Roboflow played a key role in transforming and optimizing the dataset to enhance the YOLOv5 model's training effectiveness. First, the images were uploaded to Roboflow, where various preprocessing techniques were applied. This included resizing images to a uniform 640x640 resolution, which ensures compatibility with YOLOv5's input requirements. Additionally, transformations like grayscaling, auto-orienting, and horizontal flipping were used to diversify the dataset, improving the model's robustness by allowing it to recognize construction stages under varied conditions. Roboflow also facilitated the dataset splitting into training, validation, and testing sets, which is crucial for effectively evaluating model performance. Through these preprocessing steps, Roboflow streamlined dataset preparation, ensuring both consistency and quality in the data fed to the model.

3.3.3 Model Selection & Training

The study utilised YOLOv5 (You Only Look Once), a state-of-the-art object detection model, as the core technology for predicting wall construction progress. YOLOv5 was selected for its high accuracy and speed, along with its capability to detect objects in images in real-time. The model is available in various sizes—small, medium, large, and extra-large—where the medium version was chosen to achieve a balance between accuracy and computational efficiency. During training, the model was fed labelled images, enabling it to predict construction progress based on predefined categories. The training process was implemented using the PyTorch framework, which provided extensive libraries for machine learning, allowing for efficient development and fine-tuning of the model to enhance its performance in accurately forecasting construction stages.

3.3.4 Image Processing Pipeline & Progress Calculation

A website was developed using Flask, HTML and CSS which features a streamlined image processing pipeline that allows users to upload construction images for analysis. Upon uploading, each image undergoes several preprocessing steps, and OpenCV is utilised to further process the images. After preprocessing, the images are passed through the YOLOv5 model, which analyses the visual data to identify the current construction stage, such as foundation, 25% brickwork, or 50% brickwork. The model outputs a prediction, which is then displayed on the website for the user. This user-friendly interface ensures that anyone can easily upload an image and receive immediate feedback on the construction progress stage without the need for tracking or historical data.

CHAPTER 4: TOOLS AND TECHNOLOGIES USED

4.1 YOLOv5: Model and Its Suitability for Object Detection

YOLOv5 (You Only Look Once version 5) is a state-of-the-art object detection model known for its speed and accuracy. Unlike traditional object detection methods that apply a sliding window approach to search for objects in an image, YOLOv5 treats object detection as a single regression problem, predicting bounding boxes and class probabilities directly from full images in one evaluation. This results in significantly faster inference times, making it ideal for real-time applications such as construction progress monitoring. YOLOv5 also offers various model sizes (small, medium, large, and extra-large), allowing users to choose the most suitable version based on the specific requirements of accuracy and computational resources. Its architecture leverages advancements like CSPNet for better gradient flow and model performance, making YOLOv5 a robust choice for accurately detecting construction features across different stages.

4.2 OpenCV: For Image Preprocessing

OpenCV (Open Source Computer Vision Library) is an essential tool for image processing and computer vision tasks. It provides a wide range of functionalities that are critical for preprocessing images before they are fed into the object detection model. In the context of the *ConstructVision* project, OpenCV is utilized for tasks such as image resizing, normalization, and transformation. By resizing images to a consistent size, OpenCV ensures compatibility with the YOLOv5 model. Additionally, OpenCV facilitates converting images to grayscale to reduce computational complexity and enhance feature extraction. Other preprocessing capabilities, such as image filtering, rotation, and flipping, allow for augmenting the dataset, which can improve the model's robustness and performance.

4.3 Roboflow: For Image Augmentation and Dataset Management

Roboflow is a powerful tool designed to simplify the process of managing and preparing datasets for machine learning projects. In the *ConstructVision* project, Roboflow is leveraged for image augmentation, which involves generating variations of the training images to increase the dataset's diversity and improve the model's generalization capabilities. By applying transformations like rotation, scaling, and flipping, Roboflow helps mitigate overfitting and enhances the model's performance on unseen data. Additionally, Roboflow provides an intuitive interface for dataset management, allowing users to easily organize, annotate, and preprocess images while ensuring consistency across the training dataset, which is vital for effective model training.

4.4 Flask: For Web App Development

Flask is a lightweight web framework for Python that is designed for building web applications quickly and efficiently. It follows a micro-framework architecture, meaning it provides the essential features needed for web development without the overhead of additional components. In the *ConstructVision* project, Flask is used to handle server-side logic, manage routing, and facilitate communication between the client and the YOLOv5 object detection model. Its simplicity allows developers to create RESTful APIs that enable users to upload images and receive real-time predictions. Flask's extensibility also means that additional functionalities, such as database integration and user authentication, can be added as needed, making it a flexible choice for the project's requirements.

4.5 Labelling: Annotation Tool Used for Labelling Images

LabelImg is an open-source graphical image annotation tool widely used for creating training datasets for object detection tasks. In the *ConstructVision* project, LabelImg plays a crucial role in preparing the dataset by allowing users to annotate images of wall construction stages with bounding boxes and corresponding class labels. This tool supports various annotation formats, including PASCAL VOC and YOLO, making it compatible with the YOLOv5 model. By facilitating precise labelling of images, LabelImg ensures that the model is trained on accurately annotated data, which is essential for achieving reliable predictions. The ease of use and flexibility offered by LabelImg make it a vital component in the dataset preparation process for the project.

4.6 HTML: For Structuring the Web Application

HTML (HyperText Markup Language) serves as the backbone of the *ConstructVision* web application, providing the structural framework for the user interface. It defines the layout and elements of the application, such as forms for image upload, buttons for submitting data, and sections for displaying results. By utilizing semantic HTML, the development team ensures that the application is accessible and easy to navigate. HTML also allows for the incorporation of multimedia elements, such as images and videos, enhancing user engagement. Overall, HTML is crucial for creating a well-organized and intuitive user interface that effectively supports the application's functionality.

4.7 CSS: For Styling the User Interface

CSS (Cascading Style Sheets) is utilized to enhance the visual presentation of the *ConstructVision* web application. It allows developers to apply styles, such as colors, fonts, spacing, and layout designs, to the HTML structure, ensuring a polished and aesthetically pleasing user experience. Through CSS, the application can achieve responsiveness, making it accessible on various devices and screen sizes. This adaptability is important for users who may access the application from different platforms. By creating a visually appealing and user-friendly interface, CSS contributes significantly to the overall usability and functionality of the *ConstructVision* web application, encouraging user interaction and engagement with the model's features.

4.8 Bing Image Downloader

Bing Image Downloader is a useful Python library for gathering large datasets of images, particularly for training machine learning models. It automates the process of downloading images from Bing search results, allowing users to quickly collect relevant data for training CNNs. In the context of construction progress forecasting, Bing Image Downloader can be used to gather images of various construction stages, which can then be fed into the image processing pipeline for model training and validation. This tool saves time and effort by streamlining the data collection process, ensuring a diverse and representative dataset.

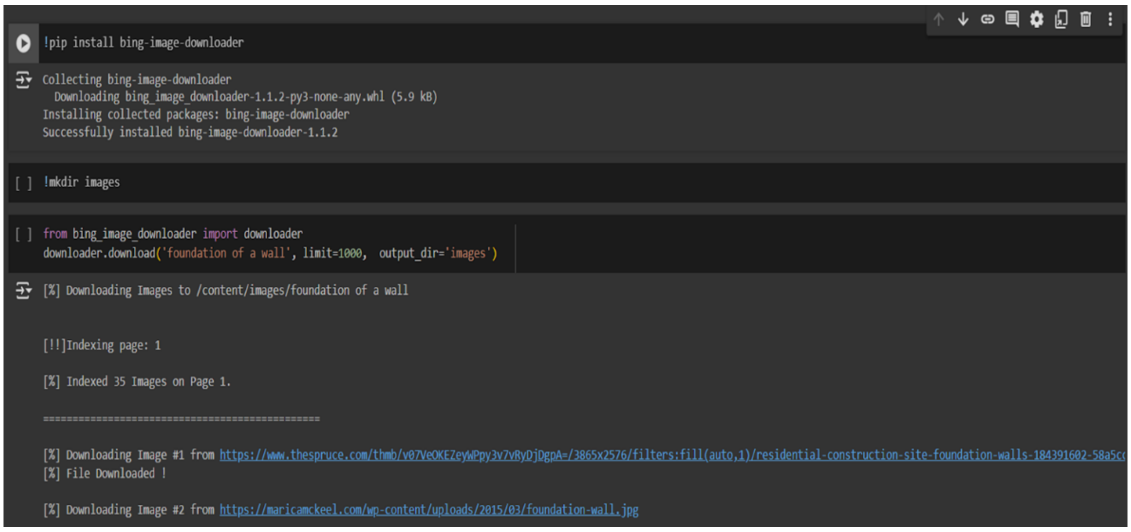
4.9 VS Code: for Website Development and Google Colab for Model Processing

VS Code (Visual Studio Code) is an excellent tool for web development, offering features such as a powerful code editor, integrated terminal, and live preview capabilities. It's ideal for building and deploying web applications, like those used for integrating image processing models. Google Colab, on the other hand, is a cloud-based platform that facilitates the seamless execution of machine learning models, including those for image processing. Colab provides free access to GPU resources, making it perfect for training and testing Convolutional Neural Networks (CNNs) used in construction progress forecasting. By combining VS Code for front-end and back-end development with Colab for model processing, developers can establish an efficient workflow that takes advantage of the strengths of both platforms.

Chapter 5: Dataset Preparation and Preprocessing

5.1 Data Collection

The data collection process involved gathering a diverse set of images that represent various stages of wall construction. Using the Bing Image Downloader, a Python library for bulk image collection, images were efficiently sourced from online databases. This tool facilitated the bulk downloading of construction images by allowing searches tailored to each construction stage. By using specific keywords, the tool quickly amassed a large dataset for labelling and model training, expediting the data collection phase. This approach provided a broad array of construction images, forming a representative dataset that supports the training of an accurate and reliable model.



```
!pip install bing-image-downloader

Collecting bing-image-downloader
  Downloading bing_image_downloader-1.1.2-py3-none-any.whl (5.9 kB)
Installing collected packages: bing-image-downloader
Successfully installed bing-image-downloader-1.1.2

[ ] !mkdir images

[ ] from bing_image_downloader import downloader
    downloader.download('foundation of a wall', limit=1000,  output_dir='images')

[%] Downloading Images to /content/images/foundation of a wall


[!!]Indexing page: 1

[%] Indexed 35 Images on Page 1.

===============================================

[%] Downloading Image #1 from https://www.thespruce.com/thmb/v07VeOKEZeyWPpy3v7vRyDjDgpA=/3865x2576/filters:fill(auto,1)/residential-construction-site-foundation-walls-184391602-58a5c
[%] File Downloaded !

[%] Downloading Image #2 from https://maricamckeel.com/wp-content/uploads/2015/03/foundation-wall.jpg
```

Fig 5.1 Screenshot: Bing Image Downloader

5.2 Annotation Process

Each collected image was manually labelled to represent a specific construction stage. This annotation process was done using the LabelImg tool, where bounding boxes were drawn around relevant portions of the pictures to label the wall construction progress. The stages were classified with the following labels:

- 20%: Foundation stage

- 40%: Initial bricklaying

- 60%: Completion of bricklaying

- 80%: Wall plastering

- 100%: Completed and painted wall

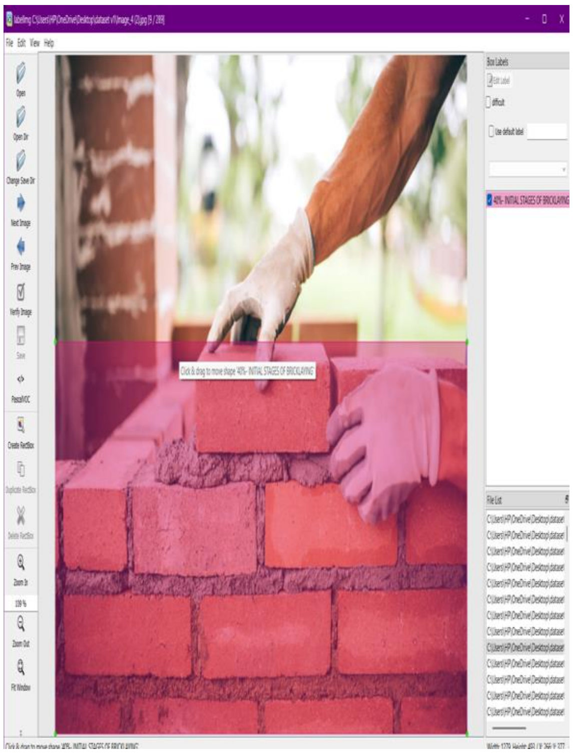
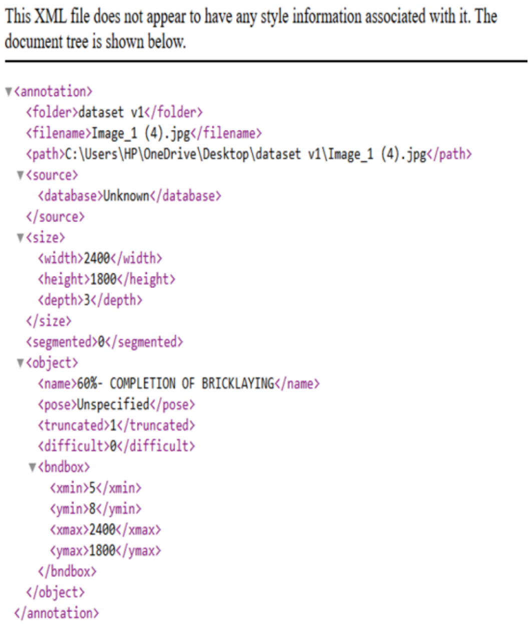


Fig 5.2.1 Screenshot: LabelImg tool



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<annotation>
   <folder>dataset v1</folder>
   <filename>Image_1 (4).jpg</filename>
   <path>C:\Users\HP\OneDrive\Desktop\dataset v1\Image_1 (4).jpg</path>
  ▼<source>
     <database>Unknown</database>
   </source>
  ▼<size>
     <width>2400</width>
     <height>1800</height>
     <depth>3</depth>
   </size>
   <segmented>0</segmented>
  ▼<object>
     <name>60%- COMPLETION OF BRICKLAYING</name>
     <pose>Unspecified</pose>
     <truncated>1</truncated>
     <difficult>0</difficult>
    ▼<bndbox>
       <xmin>5</xmin>
       <ymin>8</ymin>
       <xmax>2400</xmax>
       <ymax>1800</ymax>
     </bndbox>
   </object>
 </annotation>
```

Fig 5.2.2 Screenshot: Pascal VOC XML File

The annotations were saved in XML format using the PASCAL VOC standard, a widely recognized format for image data in machine learning. This XML format preserves key information, including bounding box coordinates and class labels, which are essential to create structured, standardised data, critical for training the model to detect each phase accurately.

5.3Data Pre-processing and Splitting :Using Roboflow for Dataset Transformation

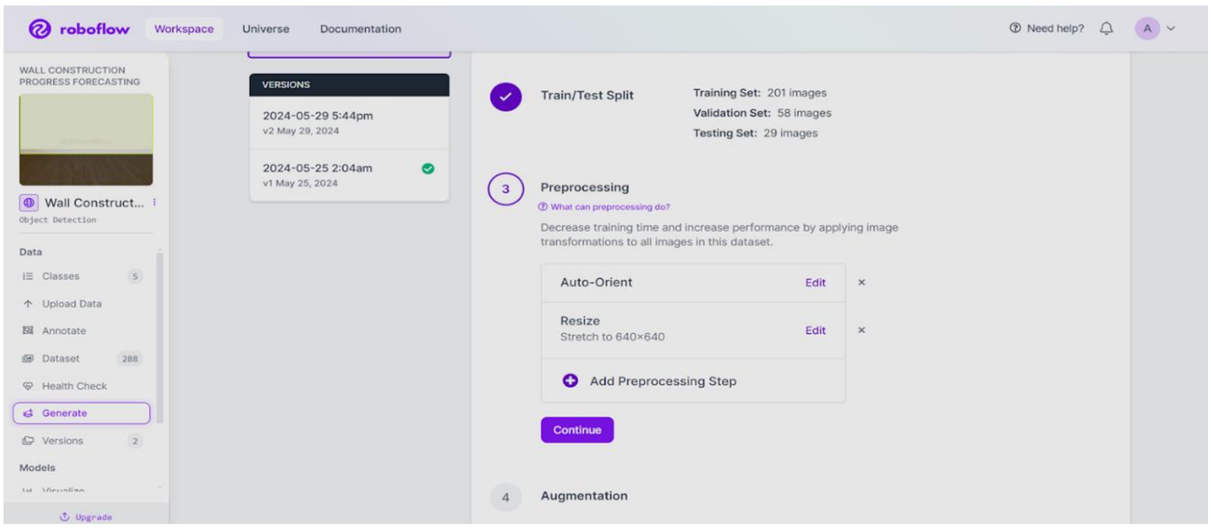
To ensure effective training of the model, the dataset was systematically divided into three distinct sets:

- **Training Set (70% of the data)**: Used to teach the model by exposing it to various construction images and their corresponding progress annotations.

- **Validation Set (20% of the data)**: Serves to fine-tune model parameters and prevent overfitting by evaluating performance on unseen data during training.
- **Test Set (10% of the data)**: Reserved for final evaluation, allowing for an unbiased assessment of the model's predictive capabilities once training is complete.

Preprocessing was a critical step in preparing the dataset for model training. This involved several key actions, such as resizing the images to a uniform dimension, which ensures that all input data has the same shape, facilitating efficient processing during training. Converting images to grayscale helped reduce computational complexity while retaining essential features necessary for the model to learn effectively. Additionally, modifying the orientation of images was performed to standardise the dataset, ensuring that all images were presented consistently, which is vital for accurate learning.

To streamline these preprocessing tasks, **Roboflow** was utilised, a powerful tool that allows for efficient image transformations. Roboflow enabled the application of consistent transformations across the dataset, including resizing, normalisation, and augmentation techniques, which enhance the model's robustness. By employing these preprocessing steps, the dataset was optimised for training, leading to improved model performance and reliability in predicting construction progress.



2024-05-29 5:44pm
v2 May 29, 2024

2024-05-25 2:04am
v1 May 25, 2024

Train/Test Split
Training Set: 201 images
Validation Set: 58 images
Testing Set: 29 images

3 Preprocessing

Decrease training time and increase performance by applying image transformations to all images in this dataset.

Auto-Orient

Resize
Stretch to 640×640

4 Augmentation

Fig 5.3 Screenshot: Data Preprocessing and Splitting using Roboflow

Chapter 6: Model Training and Development

6.1 CNN Model Architecture Selection

YOLOv5 was selected as the core architecture for ConstructVision's construction progress forecasting due to its strengths in real-time detection, accuracy, and efficiency. For a project focused on real-time analysis of construction images, YOLOv5's fast inference is essential, allowing ConstructVision to process images continuously without delay. This speed is matched by YOLOv5's strong object localization and classification accuracy, crucial for precisely identifying construction stages, such as foundation and partial brickwork. Its ability to learn distinct features of each stage makes YOLOv5 ideal for delivering reliable progress predictions, which are essential for effective monitoring.

The model's efficiency further enhances its suitability for ConstructVision. Unlike heavier models, YOLOv5's lightweight structure allows it to run smoothly on standard hardware, keeping the application responsive even when processing multiple images. This efficient operation ensures ConstructVision is practical and scalable in real-world settings, where speed and resource economy are essential. Additionally, YOLOv5's adaptability to custom datasets supports its role in this project. With minimal fine-tuning, it can be trained on ConstructVision's unique dataset of labelled construction stages, learning to detect subtle visual differences relevant to progress tracking. YOLOv5 also supports transfer learning, allowing ConstructVision to benefit from pre-trained weights that speed up training and improve detection accuracy on limited data.

Furthermore, YOLOv5's ease of integration into web applications was critical. Its modular structure and compatibility with libraries like OpenCV made it straightforward to create a pipeline from image upload to prediction display. This integration supports a user-friendly experience where users can upload images and receive progress updates instantly. Finally, YOLOv5's large community and extensive documentation provided resources that streamlined model setup and troubleshooting. This support facilitated rapid implementation and ensured ConstructVision leveraged best practices in model training and deployment.

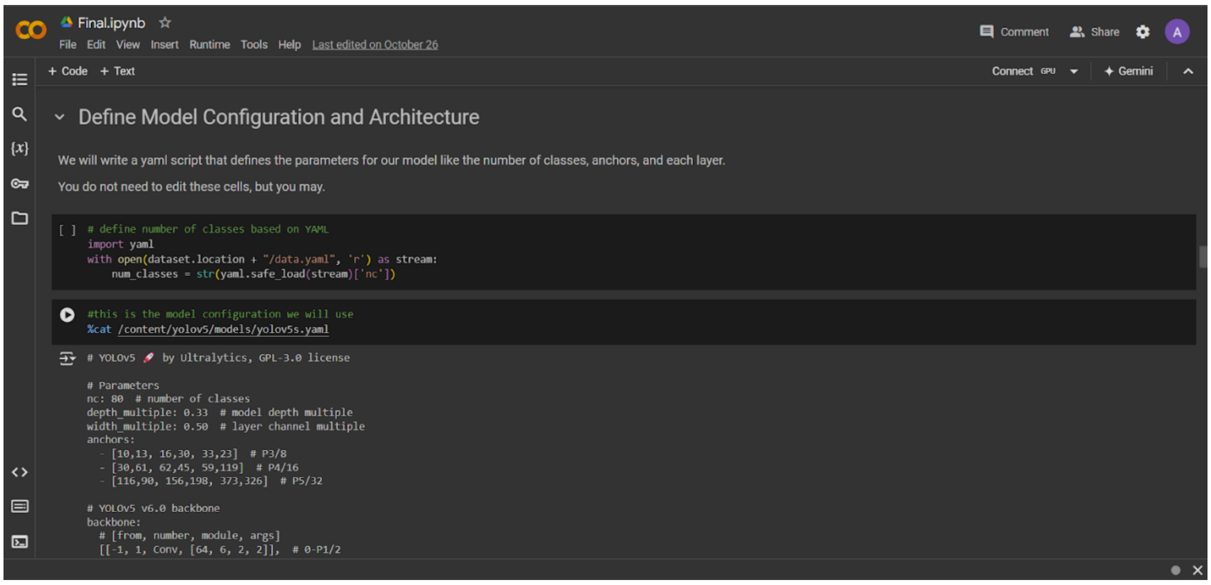
6.2 Training Process

6.2.1. Dataset Preparation & Augmentation

The YOLOv5 training process began with assembling a labelled dataset featuring various construction stages, such as foundation and brickwork. Data augmentation techniques, including rotation, flipping, and scaling, were applied to enhance the dataset's diversity. This step ensured the model could adapt to different angles and lighting conditions commonly encountered on construction sites, ultimately improving its robustness and performance in real-world scenarios.

6.2.2. Model Initialization & Transfer Learning

Training commenced by cloning the YOLOv5 repository and utilizing a pre-trained model with weights from the COCO dataset. This transfer learning approach provided a solid foundation of features, allowing the model to train more efficiently on the construction dataset. By fine-tuning the pre-trained model, we aimed to enhance its ability to recognize specific visual cues pertinent to the construction progress stages.


Final.ipynb
File Edit View Insert Runtime Tools Help Last edited on October 26
Comment Share
+ Code + Text
Connect GPU
Gemini

Define Model Configuration and Architecture

We will write a yaml script that defines the parameters for our model like the number of classes, anchors, and each layer.

You do not need to edit these cells, but you may.

```
# define number of classes based on YAML
import yaml
with open(dataset.location + "/data.yaml", 'r') as stream:
    num_classes = str(yaml.safe_load(stream)['nc'])
```

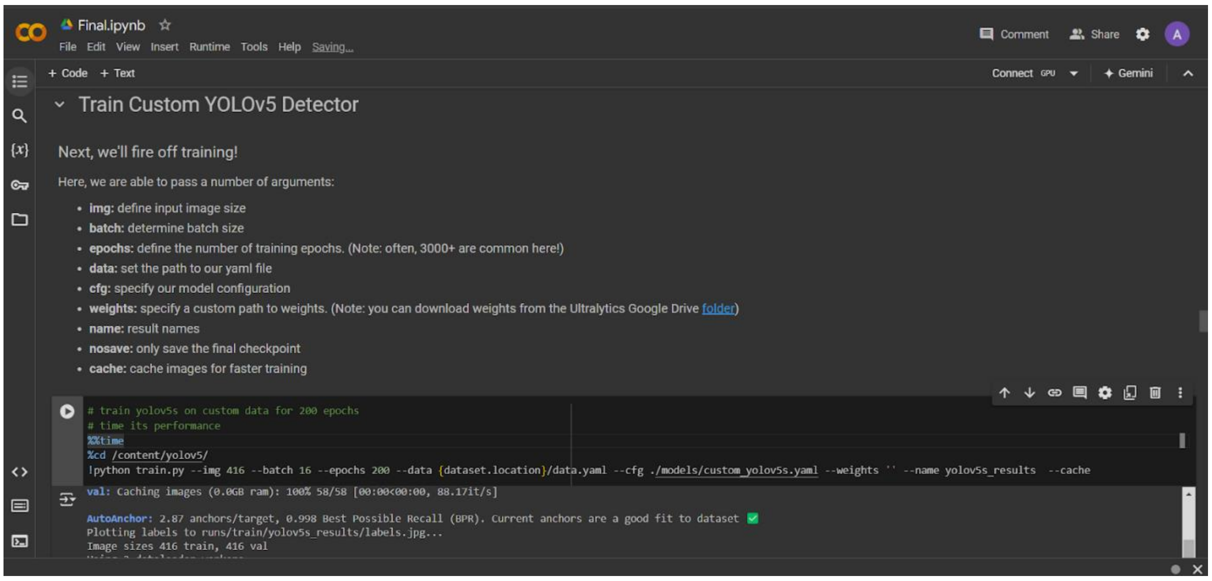
```
#this is the model configuration we will use
%cat /content/yolov5/models/yolov5s.yaml
```

```
# YOLOv5 🚀 by Ultralytics, GPL-3.0 license

# Parameters
nc: 80  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 v6.0 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Conv, [64, 6, 2, 2]],  # 0-P1/2
```

Fig 6.2.2 Screenshot: Model Initialization

6.2.3. Configuring Key Training Parameters

Key training parameters were configured to optimize performance. A batch size of 16 was selected to balance memory usage and training efficiency, while the number of epochs was set to 100 to ensure thorough learning from the dataset. The initial learning rate was established at 0.01, adjusted gradually during training, and the image size was standardized at 640x640 pixels to maintain detail without overloading computational resources.



Train Custom YOLOv5 Detector

Next, we'll fire off training!

Here, we are able to pass a number of arguments:

- **img:** define input image size
- **batch:** determine batch size
- **epochs:** define the number of training epochs. (Note: often, 3000+ are common here!)
- **data:** set the path to our yaml file
- **cfg:** specify our model configuration
- **weights:** specify a custom path to weights. (Note: you can download weights from the Ultralytics Google Drive folder)
- **name:** result names
- **nosave:** only save the final checkpoint
- **cache:** cache images for faster training

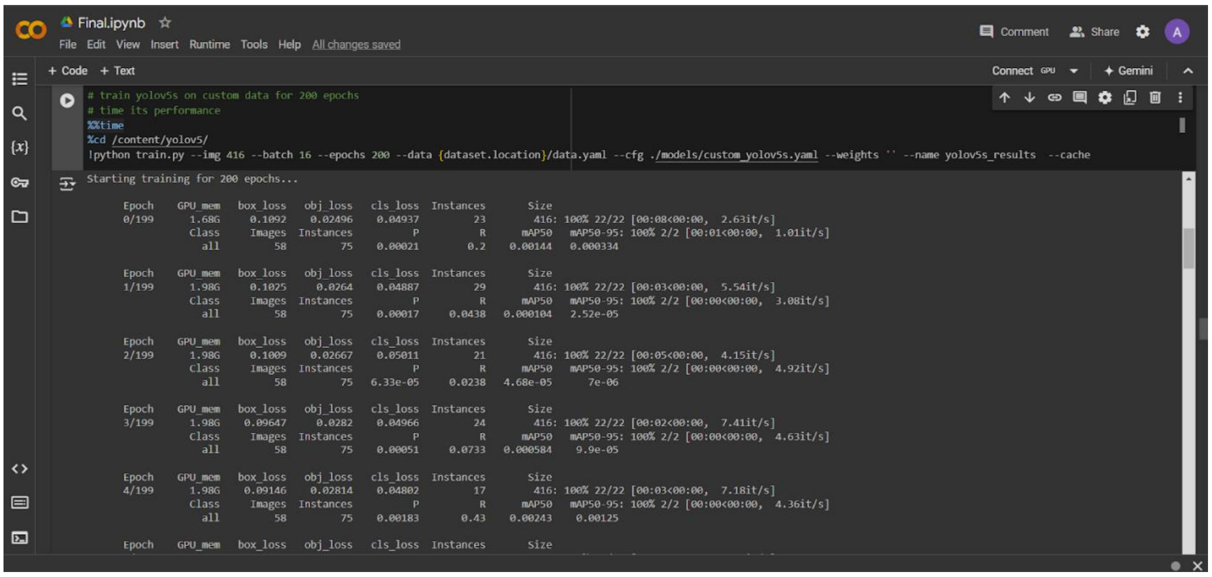
```
# train yolov5s on custom data for 200 epochs
# time its performance
%%time
%cd /content/yolov5/
!python train.py --img 416 --batch 16 --epochs 200 --data {dataset.location}/data.yaml --cfg ./models/custom_yolov5s.yaml --weights '' --name yolov5s_results  --cache
```

```
val: Caching images (0.0GB ram): 100% 58/58 [00:00<00:00, 88.17it/s]

AutoAnchor: 2.87 anchors/target, 0.998 Best Possible Recall (BPR). Current anchors are a good fit to dataset ✅
Plotting labels to runs/train/yolov5s_results/labels.jpg...
Image sizes 416 train, 416 val
```

Fig 6.2.3 Screenshot: Model Training Parameters

6.2.4. Model Training Execution

Training involved processing images in batches, where the model adjusted weights through gradient descent to minimize prediction errors. Performance was evaluated on a validation set after each epoch, ensuring that the model effectively learned the relevant features of construction stages and preventing overfitting. This iterative process led to improved accuracy in stage classification.



```
# train yolov5s on custom data for 200 epochs
# time its performance
%%time
%cd /content/yolov5/
!python train.py --img 416 --batch 16 --epochs 200 --data {dataset.location}/data.yaml --cfg ./models/custom_yolov5s.yaml --weights '' --name yolov5s_results  --cache
```

```
Starting training for 200 epochs...

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      0/199      1.68G     0.1092    0.02496    0.04937         23        416: 100% 22/22 [00:08<00:00,  2.63it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 2/2 [00:01<00:00,  1.01it/s]
                   all         58         75    0.00021        0.2    0.00144   0.000334

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      1/199      1.98G     0.1025     0.0264    0.04887         29        416: 100% 22/22 [00:03<00:00,  5.54it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 2/2 [00:00<00:00,  3.08it/s]
                   all         58         75    0.00017     0.0438   0.000104   2.52e-05

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      2/199      1.98G     0.1009    0.02667    0.05011         21        416: 100% 22/22 [00:05<00:00,  4.15it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 2/2 [00:00<00:00,  4.92it/s]
                   all         58         75   6.33e-05     0.0238   4.68e-05      7e-06

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      3/199      1.98G    0.09647     0.0282    0.04966         24        416: 100% 22/22 [00:02<00:00,  7.41it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 2/2 [00:00<00:00,  4.63it/s]
                   all         58         75    0.00051     0.0733   0.000584    9.9e-05

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      4/199      1.98G    0.09146    0.02814    0.04802         17        416: 100% 22/22 [00:03<00:00,  7.18it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 2/2 [00:00<00:00,  4.36it/s]
                   all         58         75    0.00183       0.43    0.00243    0.00125

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
```

Fig 6.2.4 Screenshot: Model Training Execution

6.2.5. Evaluation Metrics and Monitoring

The model’s performance was monitored using metrics like mean Average Precision (mAP), precision, and recall. These metrics provided insights into the model's accuracy and reliability in detecting construction stages. The F1 score, representing a balance between precision and recall, was also tracked to ensure consistent performance throughout the training process.

6.2.6. Checkpointing & Weight Saving

A checkpointing mechanism was established to save the best-performing model weights based on validation metrics, specifically mAP. This approach ensured that the optimal weights were preserved for deployment, enhancing the model's accuracy in real-world applications. The final selected model was thus refined and ready for practical use.

6.2.7. Final Validation & Testing

Upon completing the training, the model was validated on a hold-out dataset to assess its generalizability. This final testing confirmed that the model could accurately classify construction stages across various conditions, ensuring its effectiveness and reliability for integration into ConstructVision for real-time progress forecasting.

6.3 Testing and Evaluation: Evaluation of model performance

6.3.1 Evaluation Dataset

The testing and evaluation phase of the YOLOv5 model involved a systematic assessment of its performance in predicting construction stages from images. A hold-out dataset, which included diverse images of various construction stages and was not used during training, ensured an unbiased evaluation of the model's generalizability and accuracy. This comprehensive approach aimed to gauge the model's effectiveness in real-world applications.

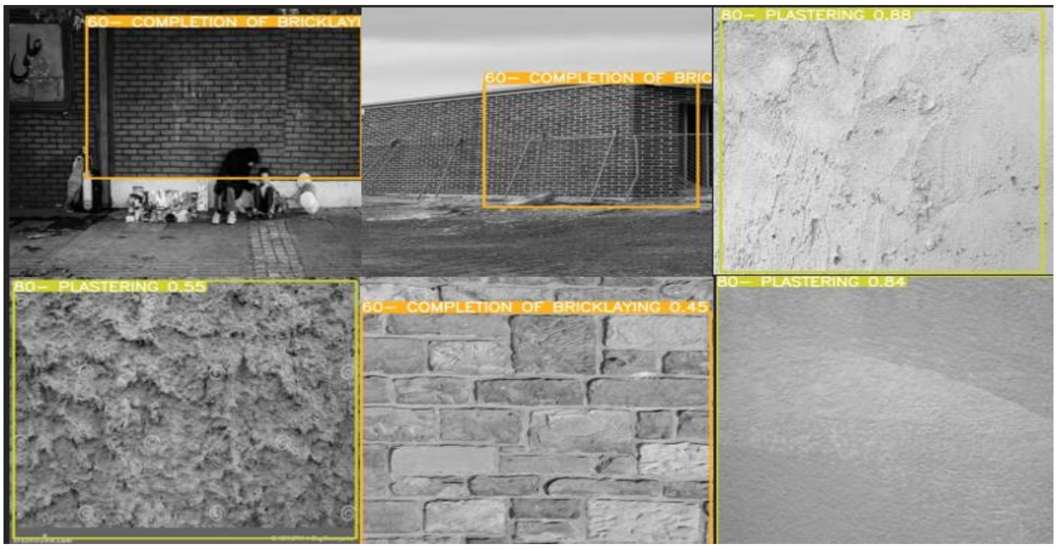


Fig 6.3.1 Test Set Detections After Model Training

6.3.2 Performance Metrics

Several key metrics were employed to evaluate the model's performance rigorously. The primary metric used was Mean Average Precision (mAP), which measures the model’s ability to accurately predict bounding boxes and classifications for each construction stage. In addition to mAP, precision and recall metrics were monitored to assess the model's effectiveness in minimizing false positives and false negatives in its predictions. The F1 score, which balances both precision and recall, was also calculated to provide a comprehensive overview of the model’s reliability in stage classification.

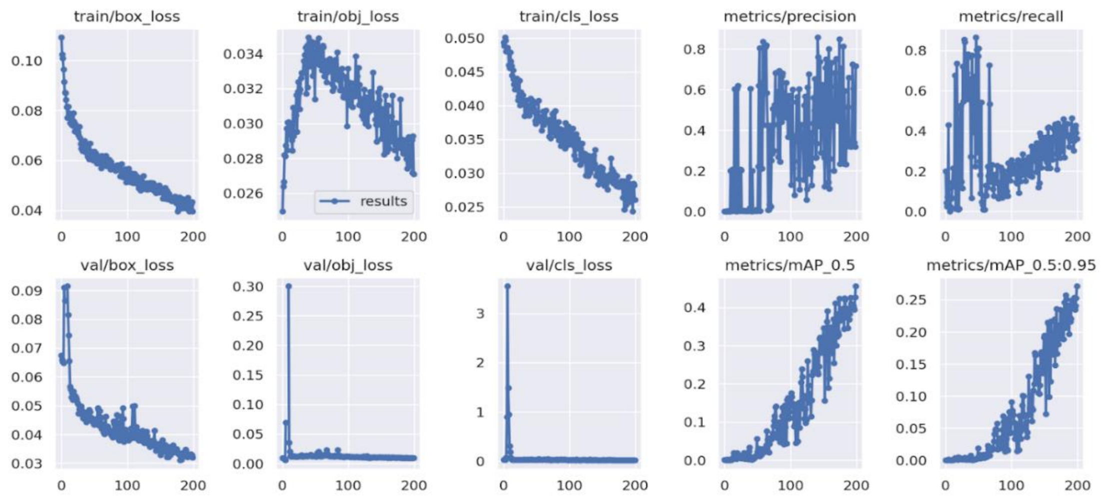


Fig 6.3.2 Model Performance Metrics Graphs

6.3.3 Comparative Analysis

The evaluation process involved comparing the model's predictions against the ground truth labels in the hold-out dataset. Each evaluation metric was calculated to gauge the accuracy of the predictions. This detailed analysis helped identify specific areas where the model performed well, as well as areas needing improvement. By understanding the strengths and weaknesses of the model, insights were generated that could inform further refinements in both the model architecture and preprocessing steps.

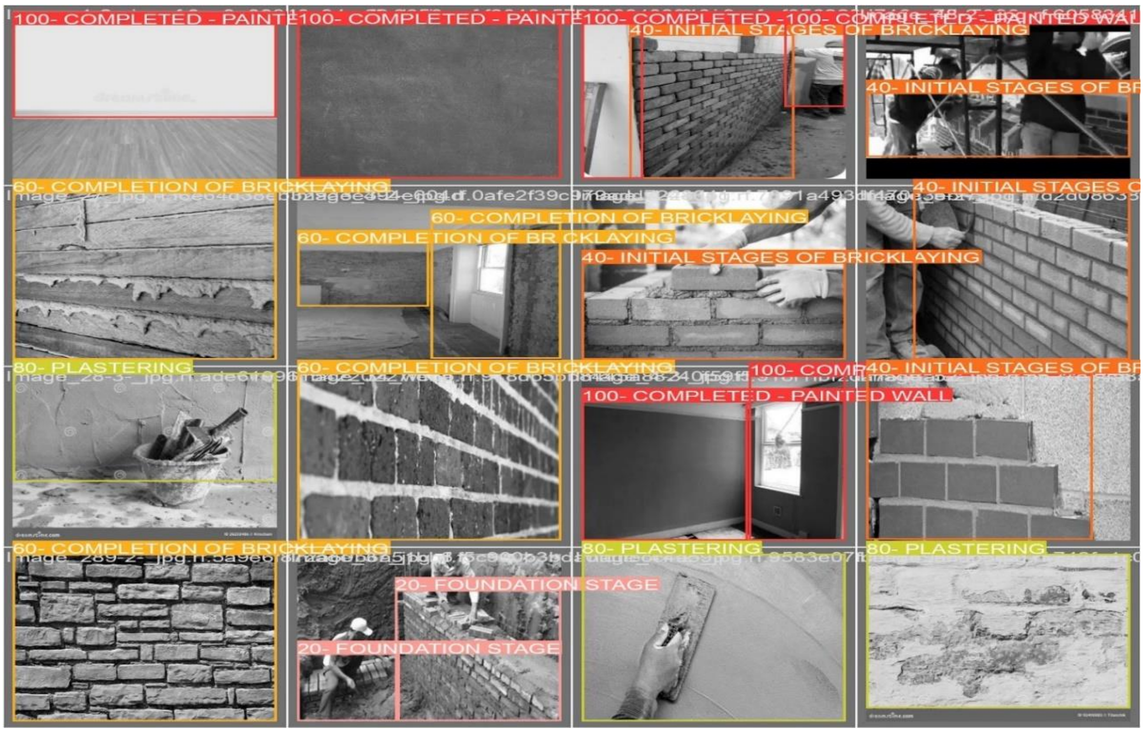

100- COMPLETED - PAINTED WALL
40- INITIAL STAGES OF BRICKLAYING
60- COMPLETION OF BRICKLAYING
80- PLASTERING
20- FOUNDATION STAGE

Fig 6.3.3 Validation Set

6.3.4 Results Overview

This phase of testing and evaluation not only ensured that the YOLOv5 model was accurately classifying construction stages but also provided valuable insights into its performance. The findings from this evaluation guided future enhancements to the model, aiming to improve its effectiveness for practical deployment in construction progress forecasting. By iteratively refining the model based on evaluation results, the system can achieve higher accuracy and reliability in real-world construction scenarios.

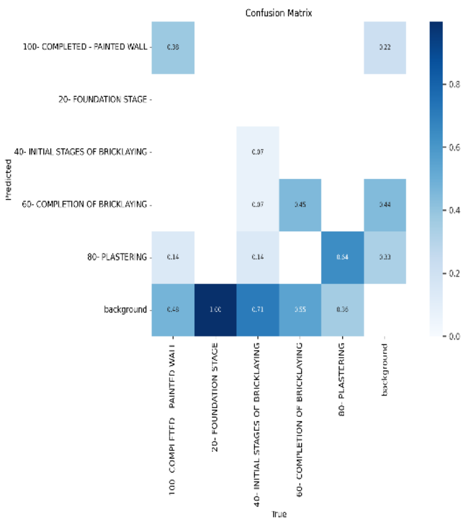


Fig 6.3.4.1 Confusion Matrix

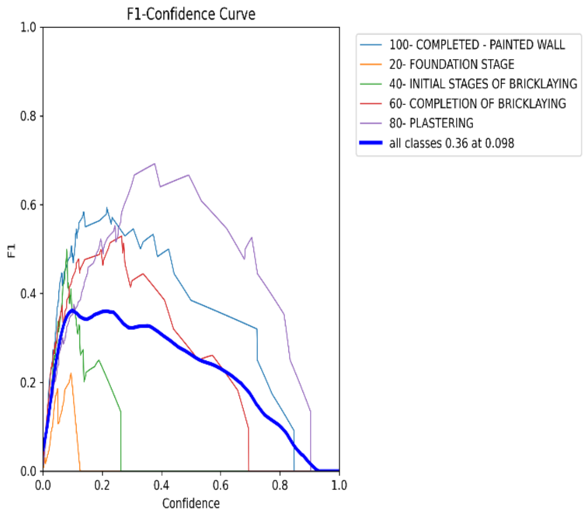


Fig 6.3.4.2 F1-Confidence Curve

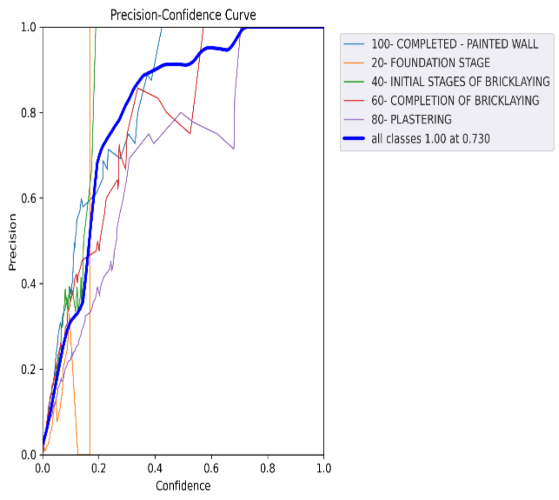


Fig 6.3.4.3 Precision-Confidence Curve

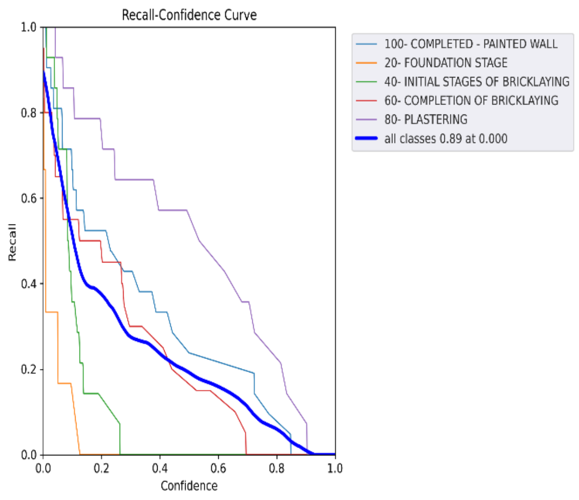


Fig 6.3.4.4 Recall-Confidence Curve

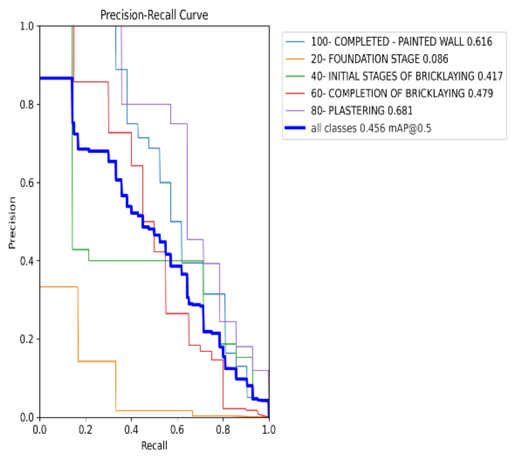
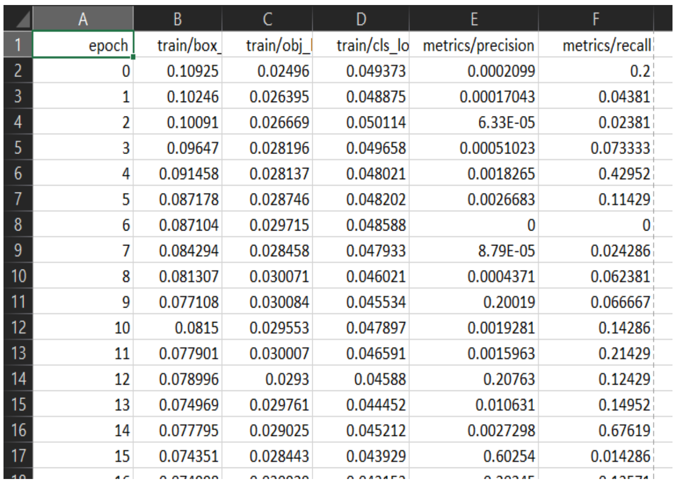


Fig 6.3.4.5 Precision-Recall Curve



	A	B	C	D	E	F
1	epoch	train/box_	train/obj_l	train/cls_lo	metrics/precision	metrics/recall
2	0	0.10925	0.02496	0.049373	0.0002099	0.2
3	1	0.10246	0.026395	0.048875	0.00017043	0.04381
4	2	0.10091	0.026669	0.050114	6.33E-05	0.02381
5	3	0.09647	0.028196	0.049658	0.00051023	0.073333
6	4	0.091458	0.028137	0.048021	0.0018265	0.42952
7	5	0.087178	0.028746	0.048202	0.0026683	0.11429
8	6	0.087104	0.029715	0.048588	0	0
9	7	0.084294	0.028458	0.047933	8.79E-05	0.024286
10	8	0.081307	0.030071	0.046021	0.0004371	0.062381
11	9	0.077108	0.030084	0.045534	0.20019	0.066667
12	10	0.0815	0.029553	0.047897	0.0019281	0.14286
13	11	0.077901	0.030007	0.046591	0.0015963	0.21429
14	12	0.078996	0.0293	0.04588	0.20763	0.12429
15	13	0.074969	0.029761	0.044452	0.010631	0.14952
16	14	0.077795	0.029025	0.045212	0.0027298	0.67619
17	15	0.074351	0.028443	0.043929	0.60254	0.014286

Fig 6.3.4.6 Screenshot: CSV file of training results of each epoch

Chapter 7: Web Application Features

7.1 Image Upload Interface

The image upload interface allows users to easily submit construction images for progress analysis. This interface is built with a simple HTML form, featuring a file input field and an upload button. Users can select an image file, which is validated to ensure it is a suitable image type, and then upload it directly to the server by clicking the submit button. The layout, styled using CSS, emphasises user accessibility, while the design aligns with the overall dark theme of the application, providing a user-friendly experience.

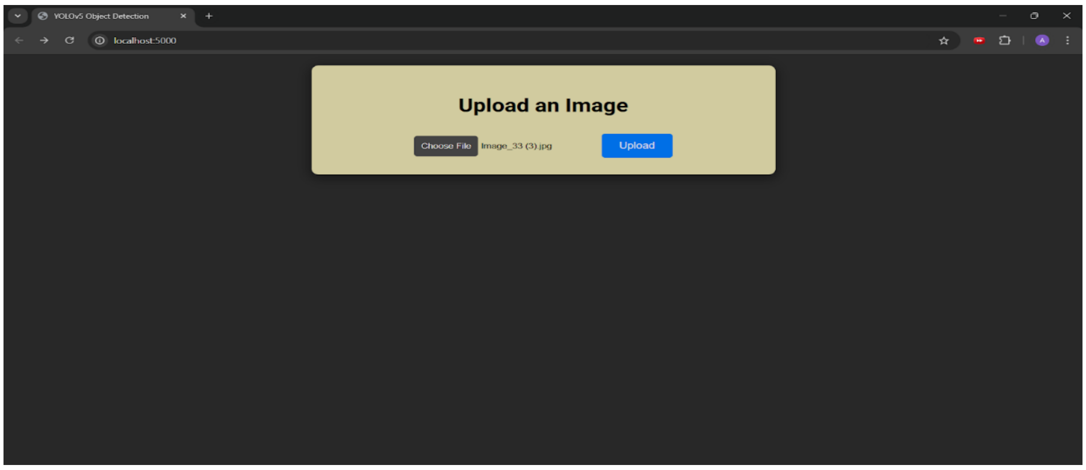


Fig 7.1 Screenshot: Web Application's Image Upload Interface

7.2 Prediction Display

After an image is uploaded, the application processes it through the YOLOv5 model, which detects the construction stage. Another HTML template displays this prediction visually. Once YOLOv5 processes the uploaded image, the bounding boxes and labels are drawn on the image to indicate the detected construction stage. This processed image, saved in the uploads folder, is rendered in the HTML user interface, along with a button for users to return to the main upload page. This setup enables quick viewing of results with the option to upload additional images.

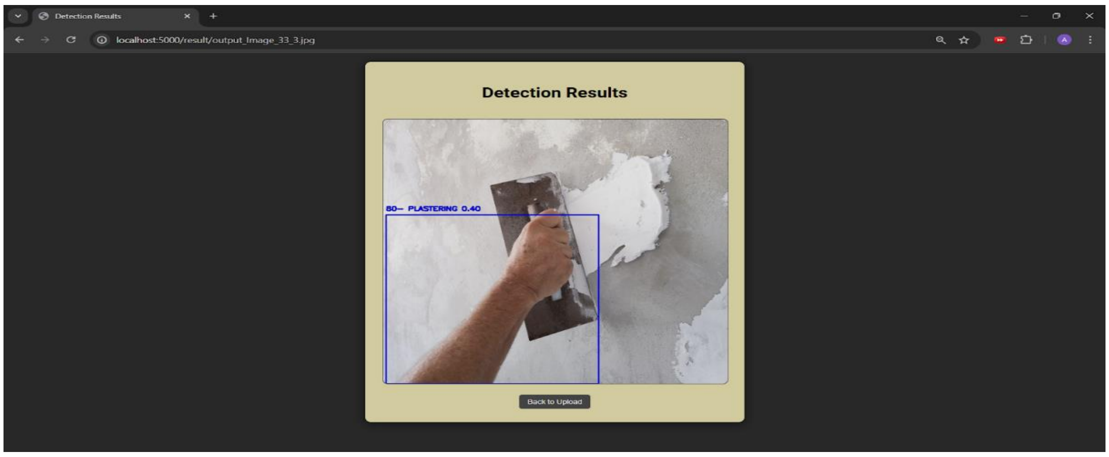
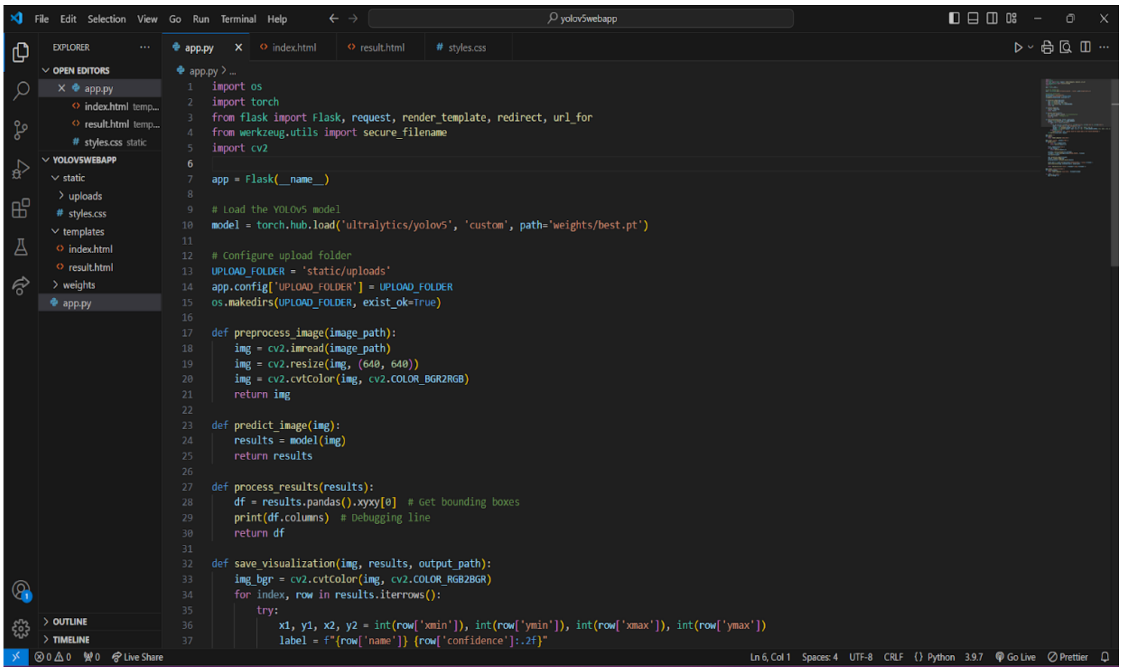


Fig 7.2 Screenshot: Web Application's Prediction Display

7.3 Flask Backend Integration

The backend, created with Flask, serves as the central controller for handling image uploads, preprocessing, and model inference. When the app launches, YOLOv5 is loaded with custom weights obtained from the previously trained model. A Python file sets up routes for the main upload page, processes images upon upload, and displays the results. Uploaded images are saved, pre-processed with OpenCV, and passed through YOLOv5 to predict the construction stage. Flask facilitates seamless communication between the front end and the model, providing predictions in real-time as users upload their images.



```python
import os
import torch
from flask import Flask, request, render_template, redirect, url_for
from werkzeug.utils import secure_filename
import cv2

app = Flask(__name__)

# Load the YOLOv5 model
model = torch.hub.load('ultralytics/yolov5', 'custom', path='weights/best.pt')

# Configure upload folder
UPLOAD_FOLDER = 'static/uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

def preprocess_image(image_path):
    img = cv2.imread(image_path)
    img = cv2.resize(img, (640, 640))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    return img

def predict_image(img):
    results = model(img)
    return results

def process_results(results):
    df = results.pandas().xyxy[0]  # Get bounding boxes
    print(df.columns)  # Debugging line
    return df

def save_visualization(img, results, output_path):
    img_bgr = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
    for index, row in results.iterrows():
        try:
            x1, y1, x2, y2 = int(row['xmin']), int(row['ymin']), int(row['xmax']), int(row['ymax'])
            label = f"{row['name']} {row['confidence']:.2f}"
```

Fig 7.3 Screenshot: Code Snippet of Web App Development

Chapter 8: Challenges and Solutions

8.1 Data Quality and Quantity:

Challenge: Acquiring a diverse and high-quality dataset was a major hurdle. The construction progress images needed to encompass various stages, environmental conditions, and materials to ensure the model could generalize well. However, obtaining sufficient high-quality images that accurately represented these variations proved difficult.

Solution: To address this issue, several preprocessing techniques were applied. Auto-orientation ensured that images were consistently aligned, while resizing standardized them to a uniform input size suitable for model training. Additionally, the dataset was regularly reviewed and cleaned to remove low-quality images and irrelevant data, enhancing the overall quality of the dataset. By leveraging tools like Bing Image Downloader for mass image collection, the project team also increased the quantity of data, which contributed to better model performance.

8.2 Data Labelling:

Challenge: Manually labelling the images was a time-consuming and labor-intensive process. This method was prone to human error, which could lead to inconsistencies in the labelled dataset. Accurate labelling was critical, as the performance of the AI model heavily depended on the quality of the labelled data.

Solution: To mitigate the challenges associated with manual labelling, AI tools were explored to automate parts of the labelling process. These tools utilized existing models to suggest labels for the images, significantly reducing the time required for manual annotation. By employing semi-automated labelling, the project improved accuracy and consistency, allowing the team to focus on refining and verifying labels rather than starting from scratch.

8.3 Model Overfitting:

Challenge: During initial training, the model demonstrated excellent performance on the training dataset but performed poorly on the validation set, indicating overfitting. This issue suggested that the model had learned to memorize the training data rather than generalize from it, leading to poor predictions on unseen data.

Solution: To combat overfitting, the project team implemented various data augmentation techniques. This included image transformations such as translation, flipping, and rotation, which artificially increased the diversity of the training dataset. By presenting the model with varied versions of the same images, it became more robust and better able to generalize to new, unseen images. Additionally, the team monitored validation loss during training and employed techniques like early stopping to prevent overfitting.

Chapter 9: Learning and Experience

9.1 Key Technical Skills Acquired

9.1.1 AI/ML Skills

- **Understanding CNN Models:** Gained experience with Convolutional Neural Networks (CNNs), particularly focusing on training models to analyze and predict wall construction progress using image data. Explored the implementation of YOLOv5 and fine-tuning pre-trained models like VGG16 and ResNet50.
- **Model Training & Evaluation:** Learned how to train machine learning models using labelled datasets, focusing on image-based progress prediction. Gained experience in evaluating model performance and optimizing it through iterative training cycles.

9.1.2 Computer Vision

- **Image Preprocessing:** Acquired hands-on experience in image preprocessing techniques such as resizing, normalization, and grayscale conversion using libraries like OpenCV to enhance model performance.
- **Object Detection & Annotation:** The user became proficient with tools like LabelImg for annotating images and creating labelled datasets. Additionally, explored techniques in object detection using YOLOv5 to recognize and track different stages of construction.
- **Image Augmentation:** Using Roboflow, improved the dataset through augmentation techniques to create a more diverse and robust training set. This enhanced your understanding of data quality improvement.

9.1.3 Web App Deployment

- **Flask Web Development:** Designed and developed a web application using Flask, implementing features for uploading images and predicting construction progress based on your trained model. HTML and CSS were utilized to create a user-friendly interface, highlighting your ability to integrate AI models into web-based platforms.
- **Model Integration & Automation:** The user demonstrated skills in integrating trained models with web applications to automate predictions, making the solution accessible to users through a streamlined upload-and-predict approach.

9.1.4 Dataset Handling

- **Data Collection & Curation:** Worked on gathering a representative set of construction images using Bing Image Downloader and carefully labelled them to create a high-quality dataset. This involved organizing the dataset with appropriate progress stages and ensuring diversity in image samples.
- **Data Annotation & Transformation:** Through LabelImg and Roboflow, images were effectively with bounding boxes and applied various transformations such as rotation, flipping, and scaling to enhance the dataset. This experience deepened my understanding of the importance of high-quality, labelled data for successful model training. These key technical skills demonstrate a comprehensive understanding of AI/ML techniques, computer vision, and web application deployment, showcasing my ability to work with complex projects and contribute meaningfully to the field of AI-driven solutions.

9.2 Real-World Application of AI in Construction

Applying AI in construction forecasting brings significant improvements to project management and progress tracking. By using AI models like CNNs and object detection techniques such as YOLOv5, construction progress can be monitored more accurately and consistently, reducing manual efforts and human error. This real-time analysis supports data-driven decisions, enhancing efficiency and providing better project timeline estimates.

A key takeaway is the importance of data quality and diversity. Building a well-annotated and representative dataset is crucial for training reliable models that generalize across different projects and conditions. Additionally, AI fosters technological innovation, paving the way for future advancements in automated safety monitoring, cost estimation, and quality control.

Overall, the application of AI in construction forecasting has the potential to revolutionize traditional practices, making progress evaluation more accurate, automated, and consistent.

Chapter 10: Conclusion

The ConstructVision project successfully demonstrated the potential of AI in forecasting construction progress using image analysis. By employing Convolutional Neural Networks (CNNs) and object detection techniques like YOLOv5, we developed a robust system that accurately predicts construction stages based on visual data. The meticulous process of dataset creation, including image collection, annotation, and augmentation, was crucial in ensuring the model's reliability and effectiveness.

This project not only highlights the importance of leveraging advanced AI technologies in construction management but also showcases the significant impact of automation on improving efficiency and reducing manual oversight. The web application developed for real-time image uploads and predictions exemplifies how AI can streamline construction progress tracking, making it accessible to project managers and stakeholders.

Overall, the successful implementation of this AI-driven solution paves the way for future advancements in the construction industry, promising enhanced project monitoring, timely decision-making, and a more data-driven approach to construction management.

REFERENCE

1. Ultralytics. (n.d.). YOLOv5 documentation. Retrieved from https://docs.ultralytics.com/yolov5/

2. Roboflow. (n.d.). Roboflow documentation. Retrieved from https://docs.roboflow.com/

3. Viso.ai. (n.d.). LabelImg for image annotation. Retrieved from https://viso.ai/computer-vision/labelimg-for-image-annotation/

4. Flask. (n.d.). Flask documentation. Retrieved from https://flask.palletsprojects.com/en/stable/

5. OpenCV. (n.d.). OpenCV documentation. Retrieved from https://docs.opencv.org/4.x/index.html

6. PyTorch. (n.d.). PyTorch documentation. Retrieved from https://pytorch.org/docs/stable/index.html

7. Bing Image Downloader. (n.d.). Python Bing Image Downloader documentation. Retrieved from https://pypi.org/project/bing-image-downloader/

8. Labelling. (n.d.). LabelImg tool for annotations. Retrieved from https://github.com/tzutalin/labelImg

9. PASCAL VOC. (n.d.). PASCAL VOC Dataset. Retrieved from http://host.robots.ox.ac.uk/pascal/VOC/