EFFICIENT COIN DETECTION IN BOARD GAMES: YOLO, OPENCV

A PROJECT REPORT

Submitted by

ARIVUMATHI D (1920110003)

SAISHREE R (1920110039)

SOORIYA M S (1920110048)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

SONA COLLEGE OF TECHNOLOGY, SALEM-5

(Autonomous)

ANNA UNIVERSITY: CHENNAI 600 025

November 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**EFFICIENT COIN DETECTION IN BOARD GAMES: YOLO, OPENCV.**" is the bonafide work of **"ARIVUMATHI D (1920110003), SAISHREE R (1920110039), SOORIYA M S (1920110048)"** who carried out the project work under my supervision.

SIGNATURE

Dr. J. Akilandeswari

HEAD OF THE DEPARTMENT

Professor

Department Of Information Technology

Sona College of Technology

Salem- 636 005.

SIGNATURE

Dr. K. Thangaraj

SUPERVISOR

Associate Professor

Department Of Information Technology

Sona College of Technology

Salem- 636 005.

Submitted for Project viva voce examination held on_____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Submitted for project viva voce examination held from 16.11.2023 to 18.11.2023

ACKNOWLEDGEMENT

First and foremost, we thank to **power of almighty** for showing us inner peace and for all blessings. Special gratitude to our **parents**, for showing their support and love always.

We express our sincere thanks to chairman **Sri.C.VALLIAPPA** and principal **Dr.S.R.R.SENTHILKUMAR** for providing adequate facilities to complete the project.

We are immensely grateful to our Head of Information Technology Department, **Dr.J.AKILANDESWARI** for the continue encouragement to complete the project.

We express our heartfelt thanks to our project supervisor **Dr.K. THANGARAJ** for his valuable guidance and fruitful discussions throughout the course of the project work.

We also thank our project coordinator **Prof. C. SANTHOSH KUMAR** for giving valuable insights and guidance to make this project journey more exciting and knowledgeable.

We feel proud in sharing this success with all our department Faculty, Staff members and friends who helped directly and indirectly to completing this project successfully.

ABSTRACT

This project is centred around the integration of advanced computer vision techniques to enhance board game experiences. The primary objectives involve leveraging the power of YOLOv8, an advanced object detection algorithm, and OpenCV, a versatile computer vision library, to achieve critical task: precise coin detection in board games.

The fusion of YOLO and OpenCV facilitates live game monitoring, enriching the gaming encounter. This work demonstrates the synergy between computer vision methods, enabling effective object tracking and streamlined data handling for diverse gaming applications. This project not only enriches the gaming experience but also has potential applications in various industries.

The project showcases the versatility and adaptability of these methods, which can be applied to a wide array of gaming scenarios, as well as other fields such as inventory management and security surveillance. Traditional board games have taken on a new dimension with this technology, offering players a heightened gaming experience.

The objective of this research is to precisely identify the coin pieces from the real-time images and videos from various board games. This project aims to construct a Deep Learning model that combines the Yolo technique and OpenCV.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVE OF THE PROJECT

Board games have long been a source of entertainment and intellectual challenge for people of all ages. In recent years, technological advancements have significantly transformed the way we engage with these timeless pastimes. This project represents a cutting-edge fusion of computer vision techniques that promises to revolutionize the traditional board game experience.

This project combines YOLO algorithm Technique along with OpenCV to accurately recognize coins in various kinds of board games.The core objective of this research is to enhance the board games and to precisely identify the coin pieces from real-time images and videos from various board games.

In order to enable real-time object detection(i.e. coins that correspond to the appropriate board games),this project aims to construct a Deep Learning model that combines the YOLO technique and OpenCV. Furthermore, the potential applications of these technology extend far beyond the realm of board games.

Specifically, we harness the capabilities of YOLO for precise coin detection on the game board. This technology ensures that coins are accurately identified , bringing a new level of precision to the gaming experience. As we delve deeper into the project, we will explore the technical details, implementation, and the far-reaching possibilities that this synergy of YOLOv8 and OpenCV can unlock.

1.2 LITERATURE SURVEY

Computer vision and robotics have made significant strides in recent years, offering innovative solutions for automating various tasks, including board game analysis and object tracking. In the context of the proposed project for enhancing the Shogi board game using YOLOv8 and OpenCV, it is essential to explore relevant literature and research that underpins this initiative.

- **Chess-Playing Robots**

The field of chess-playing robots has a long history, with notable examples like IBM's DeepBlue defeating Garry Kasparov in 1997. Chess-playing robots are robots designed to replicate human-like chess playing, requiring robust algorithms for vision and movement. While not specific to Shogi, the general principles of chessboard detection, piece identification, and movement tracking can be applied to our project.

- **Vision Systems for Chess Robots**

A chess-playing robot called MarineBlue, where they divided the architecture into three parts: Chess Engine, Robot Control, and Vision. This division is valuable as it ensures each component works robustly. The vision system is crucial for recognizing the chessboard and pieces, a fundamental aspect applicable to our Shogi board game project.

- **Challenges in Chessboard Detection**

Detecting a chessboard and its pieces is not a trivial task due to material and design diversity. Vision systems for chess-playing robots must be adaptable to different chessboards and pieces. A good vision system can accurately identify the chessboard and track piece movements, much like our objectives in enhancing the Shogi experience.

- **Object Tracking in Board Games**

Object tracking is a crucial aspect of our project, ensuring that piece movements are monitored accurately. The research in object tracking techniques, such as the use of contour detection, as outlined in your project's abstract, is valuable in understanding how similar tasks have been approached in the past.

- **PGN (Portable Game Notation) Integration**

PGN is a standard format for recording chess games, and similar standards can be applied to Shogi. Research on converting detected piece movements into PGN format is relevant for translating tracked movements into a format that the robot can understand and use for gameplay.

- **Material Recognition and Diversity**

The ability of vision systems to recognize diverse chessboard materials and piece designs is crucial. Research in material recognition and design diversity adaptation can provide insights into making the system robust across different Shogi boards.

- **Vision System Challenges**

Research works focusing on the challenges of vision systems for chess-playing robots can provide guidance on dealing with similar challenges in the context of Shogi. Vision system development for chess robots is an ongoing research field, and understanding the latest advancements is essential for your project's success.

In conclusion, the literature survey highlights the relevance and challenges of vision systems for board game-playing robots, drawing from the rich history of chess-playing robots. While specific to chess, the principles and techniques discussed in the literature are adaptable to the enhancement of Shogi using YOLOv8 and OpenCV. It is essential to build upon the knowledge and

experiences shared in these research works to create an effective and adaptable vision system for your project.

1.3 TECHNOLOGY USED

YOLOV8

YOLOv8, short for "You Only Look Once version 8," is a cutting-edge object detection algorithm. It is a new state-of-the-art computer vision model built by Ultralytics, the creators of YOLOv5. This model contains out-of-the-box support for object detection, classification, and segmentation tasks, accessible through a Python package as well as a command line interface.

YOLOv8 is renowned for its exceptional speed and accuracy in identifying and locating objects within images or video frames. It employs a single neural network to simultaneously predict object classes and their precise bounding boxes. This streamlined approach allows for real-time object detection, making it an ideal choice for a wide range of applications. Its robust performance and efficiency have established enabling rapid and precise analysis of visual data. YOLO (You Only Look Once) is a popular set of object detection models used for real-time object detection and classification in computer vision.

Originally developed by Joseph Redmon, Ali Farhadi, and Santosh Divvala, YOLO aims to achieve high accuracy in object detection with real-time speed. The model family belongs to one-stage object detection models that process an entire image in a single forward pass of a convolutional neural network (CNN).

The key feature of YOLO is its single-stage detection approach, which is designed to detect objects in real time and with high accuracy. Unlike two-stage detection models, such as R-CNN, that first propose regions of interest and then classify these regions, YOLO processes the entire image in a single pass, making it faster and more efficient.

EVOLUTION OF YOLO TO YOLOV8

- **YOLOv1** was the first official YOLO model. It used a single convolutional neural network (CNN) to detect objects in an image and was relatively fast compared to other object detection models. However, it was not as accurate as some of the two-stage models at that time.

Fig 1.1 - Yolo Evolution

- **YOLOv2** was released in 2016 and made several improvements over YOLOv1. It used anchor boxes to improve detection accuracy and introduced the Upsample layer, which improved the resolution of the output feature map.
- **YOLOv3** was introduced in 2018 with the goal of increasing the accuracy and speed of the algorithm. The primary improvement in YOLOv3 over its predecessors was the use of the Darknet-53 architecture, a variant of the ResNet architecture specifically designed for object detection. YOLO v3 also improved the anchor boxes, allowing different scales and aspect ratios to better match the size and shape of the detected objects. The use of Feature Pyramid Networks (FPN) and GHM loss function, along with a wider range of object sizes and aspect ratios and improved accuracy and stability, were also hallmarks of YOLO v3.
- **YOLOv4**, released in 2020 by Bochkovskiy et. al., introduced a number of improvements over YOLOv3, including a new backbone network, improvements to the training process, and increased model capacity. YOLOv4 also introduced Cross mini-Batch Normalization, a new normalization method designed to increase the stability of the training process.
- **YOLOv5**, introduced in 2020, builds upon the success of previous versions and was released as an open-source project by Ultralytics. YOLOv5 used the EfficientDet architecture, based on the EfficientNet network, and several new features and improvements, to achieve improved object detection performance. YOLOv5 became the world's state-of-the-art repo for object detection back in 2020 given its flexible Pythonic structure and was also the first model we incorporated for model-assisted learning at Encord.
- **YOLOv6** focused on making the system more efficient and reducing its memory footprint. It made use of a new CNN architecture called SPP-Net (Spatial Pyramid Pooling Network). This architecture is designed to handle

objects of different sizes and aspect ratios, making it ideal for object detection tasks.

- **YOLOv7** was introduced in 2022. One of the key improvements in YOLOv7 is the use of a new CNN architecture called ResNeXt. YOLOv7 also introduces a new multi-scale training strategy, which involves training the model on images at multiple scales and then combining the predictions. This helps the model handle objects of different sizes and shapes more effectively. Finally, YOLOv7 incorporates a new technique called "Focal Loss", which is designed to address the class imbalance problem that often arises in object detection tasks. The Focal Loss function gives more weight to hard examples and reduces the influence of easy examples.

- **Why YOLOv8?**
  A few of the main reasons you should consider using YOLOv8 in your next computer vision project are:
    - YOLOv8 has better accuracy than previous YOLO models.
    - The latest YOLOv8 implementation comes with a lot of new features, we especially like the user-friendly CLI and GitHub repo.
    - It supports object detection, instance segmentation, and image classification.
    - The community around YOLO is incredible, just search for any edition of the YOLO model and you'll find hundreds of tutorials, videos, and articles. Furthermore, you can always find the help you need in communities such as MLOps Community, DCAI, and others.
    - Training of YOLOv8 will be probably faster than the other two-stage object detection models.
- Use the Python Package

To use the Python CLI, first import the "ultralytics" package into your code. Then, you can use the package to load, train, and use a model.

YOLOV8 ARCHITECTURE

YOLOv8 is based on a deep convolutional neural network (CNN) architecture. The CNN consists of three main components:

Backbone: The backbone is the main part of the CNN, and it is responsible for extracting features from the input image. YOLOv8 uses a new backbone architecture called CSPNet, which is more efficient and accurate than previous backbones.

Neck: The neck is a component that is inserted between the backbone and the head. It is responsible for aggregating features from different levels of the backbone. YOLOv8 uses a new neck architecture called FPN+PAN, which better aggregates features from different levels of the backbone.

Head: The head is the final component of the CNN, and it is responsible for making predictions. YOLOv8 uses a new head architecture called PANet, which is more robust to occlusion and scale variations.

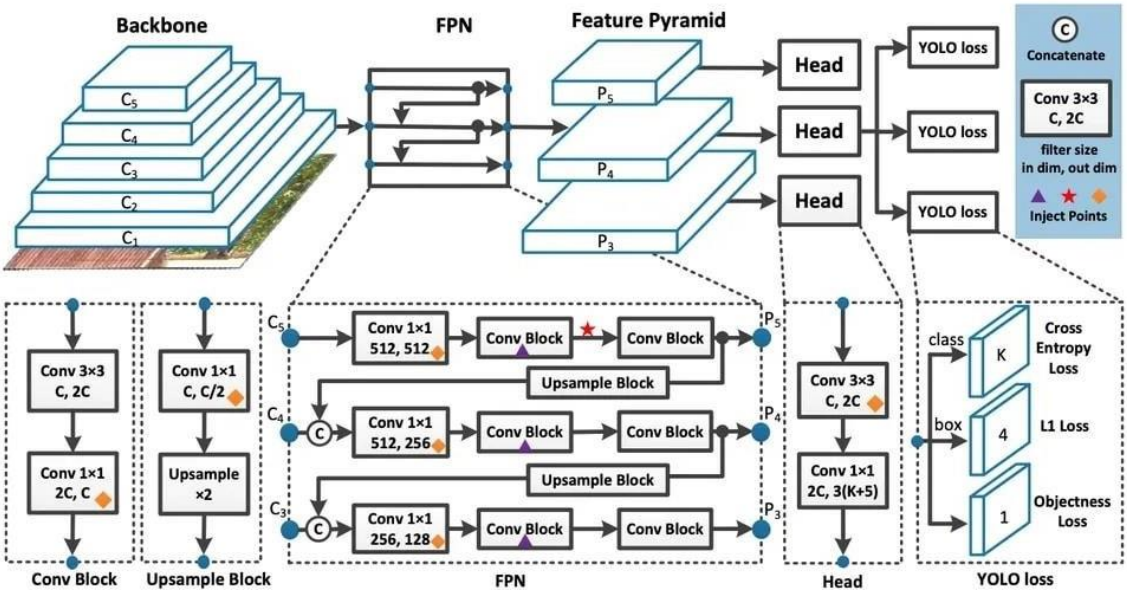

Fig 1.2 – YoloV8 Architecture

YOLOV8 WORKING

YOLOv8 works by first dividing the input image into a grid of cells. For each cell, YOLOv8 predicts a set of bounding boxes, along with the class probabilities for each bounding box.

YOLOv8 uses an anchor-free detection approach, which means that it does not use predefined anchor boxes. Instead, YOLOv8 predicts the center of an object directly. This approach is more accurate and efficient than traditional anchor-based detection approaches. To train YOLOv8, a supervised learning approach is used. The model is trained on a large dataset of images that have been labeled with the bounding boxes of the objects in the images. Once YOLOv8 is trained, it can be used to detect objects in real-time. To do this, the model is simply given an input image and it will predict the bounding boxes and class probabilities for each object in the image.

ADVANTAGES OF YOLOV8

YOLOv8 has several advantages over previous versions of YOLO, including:

- Improved accuracy: YOLOv8 is more accurate than previous versions of YOLO, especially for small and occluded objects.
- Increased efficiency: YOLOv8 is more efficient than previous versions of YOLO, which means that it can run faster on devices with limited resources.
- Anchor-free detection: YOLOv8 uses an anchor-free detection approach, which is more accurate and efficient than traditional anchor-based detection approaches.
- Improved robustness: YOLOv8 is more robust to occlusion and scale variations than previous versions of YOLO.

APPLICATIONS OF YOLOV8

YOLOv8 can be used for a variety of object detection tasks, including:

Self-driving cars: YOLOv8 can be used to detect pedestrians, vehicles, and other objects on the road.

Security and surveillance: YOLOv8 can be used to detect intruders, weapons, and other suspicious objects in security footage.

Retail: YOLOv8 can be used to detect products on shelves and to track customer movement in stores.

Manufacturing: YOLOv8 can be used to detect defects in products and to ensure that products are assembled correctly.

Overall, YOLOv8 is a powerful and versatile object detection algorithm that can be used for a variety of applications. It is the latest version of the popular YOLO family of algorithms, and it offers improved accuracy and efficiency over previous versions.

Open CV

- OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code.
- The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from

an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

- Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDAand OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

CHAPTER 2

HARDWARE AND SOFTWARE REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

1) Jetson Orin Nano Module

Jetson Orin Nano is a compact edge AI board. It has an 8GB module with an Ampere architecture GPU, a 6-core ARM CPU, and 1024 NVIDIA CUDA cores. Has up to 40 TOPS AI performance and a rich set of Ios.

2) Jetson Nano Camera

High-quality webcams or video cameras for capturing video data containing coin pieces for testing and real-time applications.

3) HDMI cable

HDMI means High-Definition Multimedia Interface, a standard for simultaneously transmitting digital video and audio from a source.

4) High-Performance Computing

A powerful computer or server with a multicore CPU and a good amount of RAM (e.g., 16GB or more) for training deep learning models efficiently.

5) Graphics Processing Unit (GPU)

A dedicated GPU, preferably NVIDIA CUDA-enabled, to accelerate the training of deep neural networks. GPUs significantly speed up the training process.

6) Storage

Sufficient storage space for storing large video datasets and model checkpoints. Fast solid-state drives (SSD) are recommended for better data access speeds.

7) SD Card

Optimized for mobile gaming, this card has maximum read speeds of up to 190 MB/s and maximum write speeds of up to 130 MB/s**.**

2.2 SOFTWARE REQUIREMENTS

1) Operating System

A Linux-based operating system (e.g., Ubuntu) is often preferred for deep learning development due to better compatibility with GPU libraries and tools. Windows or macOS can also be used. (preferably Windows 7 or later).

2) Python

Python is the primary programming language for deep learning. Ensure we have Python installed (preferably version 3.7 or later).

3) Deep Learning Frameworks

Install deep learning frameworks like TensorFlow, PyTorch, or Keras for building and training neural networks.

4) OpenCV

OpenCV (Open Source Computer Vision Library) is essential for video data preprocessing and computer vision tasks.

5) Video Processing Libraries

Libraries like FFmpeg can be useful for video data manipulation and conversion.

6) Dependencies and Libraries

Install any other project-specific libraries or dependencies that may be required for our specific implementation.

2.3 TOOLS

1) IDE (Integrated Development Environment)

Choose an IDE that we are comfortable with for coding, such as PyCharm, VSCode, or Jupyter Notebook.

2) ROBOFLOW

A computer vision developer framework that helps developers build and deploy computer vision models. Roboflow provides tools for Data collection, Preprocessing, Model training, Building datasets, Annotation, Deployment.

3) Jupyter Notebooks (Optional)

Jupyter Notebooks are helpful for interactive development and experimentation. We can also go with other notebooks like VS Code notebook, etc.

4) CUDA Toolkit (for NVIDIA GPUs)

If we are using NVIDIA GPUs, we will need to install the CUDA Toolkit to leverage GPU acceleration.

5) CuDNN (for NVIDIA GPUs)

The CuDNN library is also essential for deep learning on NVIDIA GPUs, as it provides optimized implementations of deep neural network operations.

Ensure that we keep all software and libraries up-to-date and properly configured for our project's needs. This is a basic list, and the specific requirements may vary based on the complexity of our project and our preferred technology stack.

CHAPTER 3

PROJECT DESCRIPTION

3.1 EXISTING SYSTEM

The proposed project involves the utilization of object detection and recognition techniques for the identification of shogi coins, a distinct aspect of the traditional Japanese board game. This innovative endeavor is characterized by a dearth of existing works available online. Unlike the well-established applications in chess games, the specific use of YOLO (You Only Look Once) algorithm for shogi coin detection remains uncharted.

Shogi, with its unique board layout and coin designs, presents a notable departure from conventional chess. The shapes and movement patterns of shogi coins differ significantly, and the absence of readily accessible resources and models for coin recognition within shogi underscores the novelty of this project. To execute this project, a custom dataset comprising shogi board images is to be created, annotated with precision using the Roboflow tool, and subsequently utilized to train a YOLOv8 algorithm. The chosen YOLO framework is well-suited for real-time object detection and can be tailored to this distinctive task with thoughtful parameter adjustments. By specifying parameters such as the number of epochs and batch size, the model can be fine-tuned to optimize accuracy and performance.

The potential implications of this project extend beyond the scope of gameplay analysis in shogi. It opens avenues for the creation of educational tools, automated game analysis, and enhancements to the overall shogi experience. Furthermore, it paves the way for the development of artificial intelligence-driven shogi strategies, which could aid players in their training and gameplay enhancement.

In summary, this project represents an innovative application of object detection and recognition techniques to the realm of shogi, an area that has hitherto received limited attention in the domain of computer vision and artificial intelligence. The absence of pre-existing works in this particular domain underscores the pioneering nature of this endeavor, making it a significant contribution to the shogi community and a promising avenue for the development of AI-driven applications in board games.

3.2 PROPOSED SYSTEM

The proposed system aims to develop an advanced solution for Shogi coin detection using state-of-the-art object detection and recognition techniques. Shogi is a traditional Japanese board game with historical significance, and accurately

detecting and recognizing Shogi coins on the board is crucial for enhancing player experience, facilitating research, and providing assistance to players. This project leverages computer vision and deep learning technologies to automate the process of coin identification, making it faster, more accurate, and adaptable to various applications.

- **Data Collection and Annotation**

To kickstart the project, a robust dataset of Shogi board images is collected, covering a wide range of board setups, lighting conditions, and coin orientations. To train the model effectively, each coin and square on the board is meticulously annotated using the Roboflow tool. Annotations include the coin's class (e.g., pawn, rook, knight) and precise location within the image. This annotated dataset will serve as the foundation for training the object detection model.

- **YOLOv8 Algorithm Implementation**

The heart of our system lies in the YOLOv8 (You Only Look Once) algorithm, a cutting-edge object detection model renowned for its speed and accuracy. We utilize YOLOv8 as the core engine for recognizing Shogi coins in images and videos. This algorithm is chosen for its ability to detect multiple objects in real-time, making it ideal for our application.

- **Model Training and Optimization**

Using the annotated dataset, the YOLOv8 model is trained to recognize Shogi coins effectively. Hyperparameters such as the number of epochs and batch size are fine-tuned to achieve optimal results. The model undergoes extensive training to improve its ability to identify coins under various conditions, including different board setups, coin colours, and lighting conditions. This ensures that our system can adapt to the dynamic nature of Shogi games.

- **Testing and Evaluation**

After model training, rigorous testing is conducted to evaluate its performance. The system is assessed on a test dataset to measure its accuracy, precision, recall, and F1-score. Additionally, the system is tested on newly supplied videos, simulating real-time Shogi games. The goal is to ensure that the system accurately identifies and tracks coins as they move during a game.

- **Real-World Applications**

The proposed system has a wide range of applications. It can be used to assist Shogi players by providing suggestions based on the current board state. Additionally, it can serve as a tool for researchers and historians interested in studying Shogi games and strategies. The system can also be incorporated into online Shogi platforms, offering real-time analysis and commentary during live matches.

The proposed system for Shogi coin detection using object detection and recognition leverages advanced computer vision techniques, YOLOv8 algorithm,

and a meticulously annotated dataset to automate the process of identifying and tracking Shogi coins. This project not only enhances the Shogi gaming experience but also offers valuable tools for researchers and historians.

CHAPTER 4

DESIGN AND IMPLEMENTATION

4.1 ARCHITECTURE DESIGN

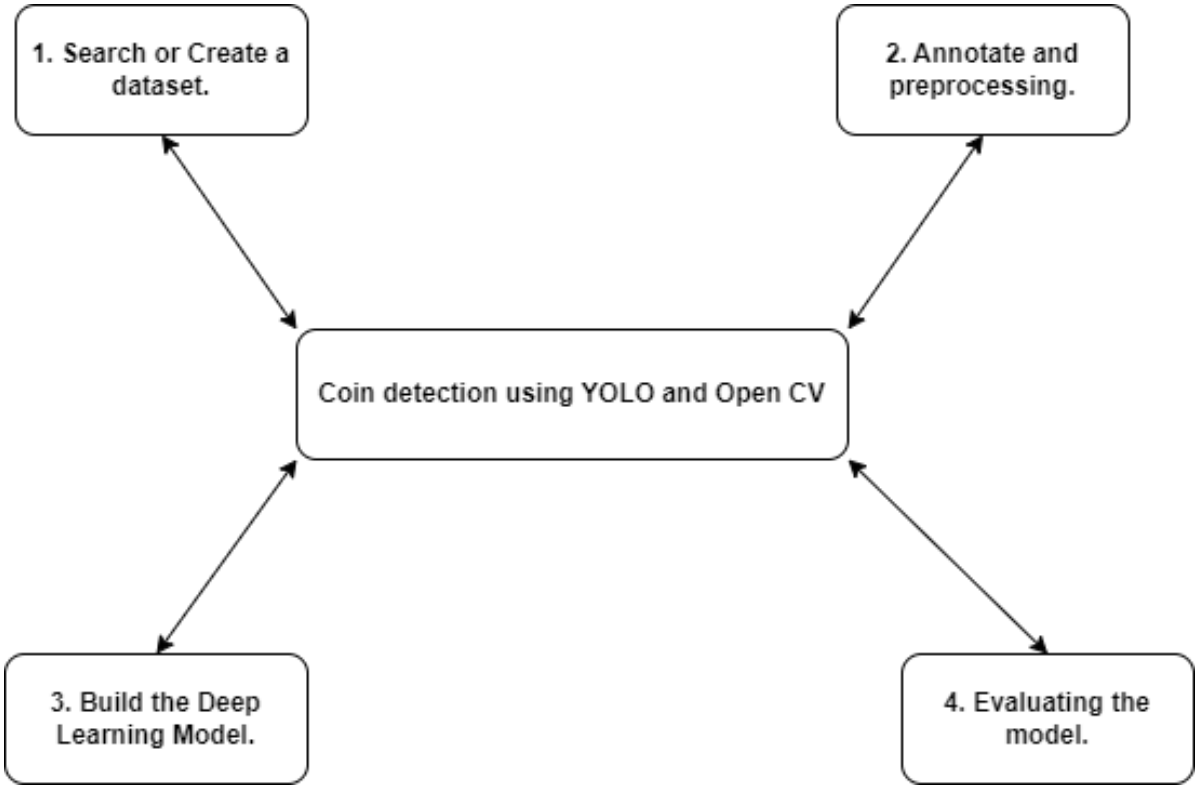

Fig 4.1 – Design Architecture

4.1.1. Search or Create a Dataset

- **Define the Purpose:** Clearly articulate the goal of the dataset, which is to support Shogi coin detection using object detection and recognition.
- **Data Sources:** Identify potential sources for Shogi board images and videos, such as online repositories, video recordings, or creating your own data.
- **Data Collection:** Describe the process of collecting data, including the number of images and videos needed for a representative dataset.
- **Data Diversity:** Emphasize the importance of having a diverse dataset that covers various Shogi board setups, coin classes, lighting conditions, and perspectives.

4.1.2. Annotating and Preprocessing

- **Annotation Tool:** Specify the tool you plan to use for annotating the dataset, such as the Roboflow tool, and discuss its features.
- **Annotation Guidelines:** Develop clear guidelines for annotators to ensure Consistent labeling, including defining annotation classes (e.g., pawn, rook, knight) and the format of annotations (bounding boxes).
- **Preprocessing:** Describe data preprocessing steps, including resizing images, normalizing pixel values, and data augmentation techniques to enhance the dataset's quality and diversity.

4.1.3. Build the Deep Learning Model

- **Model Selection:** Justify the choice of YOLOv8 as the object detection algorithm due to its real-time capabilities, and explain how it aligns with the project's goals.
- **Architecture Overview:** Provide an overview of the YOLOv8 architecture and its key components, including the backbone network, detection layers, and the use of anchor boxes.
- **Training Data Preparation:** Explain the process of converting annotated data into a format suitable for model training.
- **Hyperparameters:** Define the hyperparameters that can be adjusted during model training, such as the number of epochs, batch size, learning rate, and optimizer.
- **Model Training Pipeline:** Outline the steps involved in training the model, including data splitting into training and testing sets and the use of data loaders for efficient training.
- **Model Evaluation:** Discuss the choice of evaluation metrics (e.g., accuracy, precision, recall, F1-score) for measuring the model's performance.

4.1.4. Evaluating the Model

- **Testing on Test Dataset:** Explain how the model is tested on a dedicated test dataset to assess its performance objectively.
- **Real-time Video Analysis:** Detail the procedure for evaluating the model's performance on newly supplied videos to ensure that it effectively tracks and identifies Shogi coins in real-time.
- **Fine-Tuning:** Address the possibility of fine-tuning the model based on the evaluation results to improve its accuracy.
- **Future Scalability:** Mention plans for scalability and future enhancements, including potential integration with Shogi platforms and additional features such as move prediction.

This architecture design content provides a structured approach for your Shogi coin detection project, ensuring that each phase, from data collection to model evaluation, is well-defined and organized.

4.2 MODULES

Building a project for Shogi coin detection using object detection and recognition involves several key modules and components. Here are the primary modules you would need for such a project:

4.2.1 Data Collection

Module 1: Image and Video Collection: Gather a diverse dataset of Shogi board images and videos, including various board setups and lighting conditions.

4.2.2 Data Annotation

Module 2: Data Annotation: Annotate the collected images and videos to label the Shogi coins, specifying their class (e.g., pawn, rook, knight) and location within each image. You mentioned using the Roboflow tool for this task.

4.2.3 Deep Learning Model

Module 3: YOLOv8 Implementation: Implement the YOLOv8 (You Only Look Once) algorithm, a robust choice for object detection, and integrate it into your project.

4.2.4 Model Training

Module 4: Dataset Preprocessing: Prepare the annotated dataset for training, including data augmentation, normalization, and splitting it into training and testing sets.

Module 5: Model Training: Train the YOLOv8 model on the annotated dataset, specifying parameters like the number of epochs, batch size, learning rate, and optimization algorithms.

4.2.5 Model Evaluation

Module 6: Testing and Validation: Evaluate the model's performance using the test dataset to measure accuracy, precision, recall, and F1-score.

Module 7: Video Analysis: Test the model on newly supplied videos to ensure real-time Shogi coin detection and tracking.

4.2.6 Real-World Applications

Module 8: Integration with Shogi Platforms: If applicable, integrate the model into online Shogi platforms or mobile apps to provide real-time analysis and commentary during live matches.

4.2.7 Future Enhancements

Module 9: Machine Learning for Move Prediction: Implement machine learning techniques to predict the next moves in a Shogi game, enhancing the system's capabilities.

Module 10: Mobile App Development: Develop a user-friendly mobile application for wider accessibility and usability.

Each of these modules plays a critical role in building your Shogi coin detection project. Depending on your specific project requirements, you may also need additional modules for data management, user interfaces, and deployment. It's important to plan and execute each module systematically to ensure the success of your project.

4.3 DESCRIPTION OF PROJECT MODULES

4.3.1 Dataset Acquisition

Creating or acquiring a dataset is a crucial initial step in developing a deep learning model for precise coin detection in board games. The quality and diversity of the dataset directly impact the model's performance, and in this context, it is essential to have a well-annotated dataset representing various board game scenarios, camera angles, and lighting conditions. Let's delve deeper into the dataset acquisition process and its significance within the project context.

1) **Identifying Data Sources**

   To kickstart the dataset acquisition process, the first step is to identify suitable data sources. These sources can be either internal or external. Internal sources may include images or videos collected by your organization specifically for the project. External sources might involve publicly available datasets, third-party resources, or even web scraping from websites featuring board game images. It's important to determine the right mix of sources to ensure diversity in your dataset.

2) **Data Collection**

   Once you've identified the data sources, the next step is to collect data from these sources. This can be done through various means, such as manual data entry, automated web scraping tools, accessing APIs, or connecting to databases. In the context of board games, you may need to collect images or videos capturing different game scenarios, including different types of board games and their variations.

3) **Data Sampling**

   Depending on the scale and computational resources available for your project, you may need to decide on the size of your dataset. Data sampling can involve selecting a subset of data from large sources to ensure that the dataset

remains manageable. It's crucial to balance the size of the dataset with the capabilities of your computing infrastructure and the specific requirements of your project.

4) **Data Cleaning**

Raw data often comes with its share of challenges, including missing values, inconsistencies, errors, or outliers. Data cleaning is a critical step to handle these issues. It involves tasks like imputing missing values, correcting errors, and removing outliers to ensure data quality. In the context of coin detection in board games, ensuring that annotated coin positions are accurate and reliable is of utmost importance.

In this project, the dataset acquisition process is not just about gathering images of board game scenarios; it's also about collecting data that includes annotated coin positions. This annotation process plays a pivotal role in training the deep learning model. Precise annotation allows the model to learn the spatial characteristics of coins in the game, which is essential for.

4.3.2 Annotating and Preprocessing

Annotating and preprocessing the acquired dataset are essential steps in the development of a deep learning model for precise coin detection in board games. These steps not only enhance the quality of the dataset but also prepare it for effective model training and ultimately, the successful deployment of the model in real-world scenarios. In this process, Roboflow plays a crucial role in annotating the data, while preprocessing techniques are employed to ensure that the dataset is well-structured and ready for training. Let's delve into these steps in greater detail.

1) **Annotation with Roboflow**

Annotation is the process of marking the location of coins or game pieces in the images, providing the model with valuable ground truth information. This step is fundamental for the deep learning model to learn the spatial characteristics of these objects accurately. In the context of board games, this means precisely identifying and labeling the position of coins in various game scenarios, whether it's chess, checkers, or custom game pieces.

Roboflow is a robust and efficient tool for this annotation process. It simplifies the task of marking objects in images, making it accessible to both novice and experienced annotators. The platform provides various annotation options, such as bounding boxes or polygons, allowing for flexibility in the annotation process. Furthermore, Roboflow can handle large datasets with ease, streamlining the annotation process and saving valuable time.

2) **Labeling Game Pieces**

In addition to marking the location of coins, it's essential to label different types of coins or game pieces used in board games. This labeling ensures that the model can distinguish between various objects, even when

they share similar spatial characteristics. For instance, if you're working with chess, each type of piece (king, queen, pawn, etc.) should be labeled accordingly. This enables the model to provide precise identification of the objects it detects.

3) P**reprocessing Techniques**

Once the dataset is annotated, it undergoes preprocessing to ensure that it is well-structured and optimized for model training. Preprocessing involves several key techniques:

a) **Data Augmentation:** Data augmentation is the process of artificially increasing the size of the dataset by applying various transformations to the images. This can include techniques such as rotation, flipping, zooming, and adjusting brightness or contrast. Data augmentation helps the model generalize better and handle variations in camera angles and lighting conditions.

b) **Resizing:** Standardizing the size of images in the dataset is crucial. Resizing ensures that all images have the same dimensions, which is a requirement for most deep learning models. This step simplifies the computational aspects of training and helps the model process input data consistently.

c) **Normalization:** Normalization involves scaling pixel values to a standard range (typically between 0 and 1 or -1 and 1). Normalizing the data ensures that the model is more robust to variations in pixel intensity and can accelerate training.

The combination of YOLO (You Only Look Once) and Roboflow is particularly effective in this project. YOLO is a real-time object detection algorithm known for its efficiency in detecting objects in images. By utilizing YOLO, the deep learning model can efficiently and accurately detect multiple coins or game pieces simultaneously in a single pass, making it well-suited for tracking objects in board games.

In conclusion, annotating and preprocessing the dataset are pivotal steps in the development of a deep learning model for coin detection in board games. Roboflow simplifies the annotation process, ensuring that the model is provided with accurate and reliable ground truth information. Furthermore, preprocessing techniques, including data augmentation, resizing, and normalization, prepare the dataset for efficient model training. This process sets the stage for the subsequent module of deep learning model construction, where the model learns to identify and locate coins or game pieces accurately, enhancing the gaming experience by providing real-time monitoring of board game scenarios.

4.3.3 Deep Learning Model Construction

The Heart of Coin Detection in Board Games

The core of the project lies in the development of a deep learning model for precise coin detection in board games. In this section, we'll delve into the details of this crucial module, focusing on the utilization of the YOLO (You Only Look Once) algorithm as the backbone for object detection, the model training process, optimization of parameters, and the remarkable capabilities of YOLO in the context of tracking coins and game pieces in real-time.

1) **YOLO as the Object Detection Backbone**

   The YOLO algorithm is at the heart of the project's object detection capabilities. YOLO is renowned for its efficiency in real-time object detection and is ideally suited for this application. What sets YOLO apart is its ability to detect and locate multiple objects in a single pass through an image, making it well-suited for scenarios where quick and accurate object detection is essential, such as tracking coins and pieces during board games.

2) **Model Training on Annotated Data**

   The success of the deep learning model hinges on the quality of the annotated dataset we've acquired and the precision of the annotations made using tools like Roboflow. The annotated data serves as the training ground for the model, teaching it to recognize and pinpoint the location of coins or game pieces in images.

During the training process, the model is exposed to a variety of board game scenarios, each with its unique spatial characteristics, lighting conditions, and camera angles. This diversity in the training data allows the model to learn to handle real-world variations and make accurate detections regardless of the gaming context. The quality of the annotations is essential here, as any inaccuracies in the annotated data can lead to suboptimal model performance.

3) **Optimizing Model Parameters and Hyperparameters**

   Model training is not just a matter of feeding data to the algorithm. It involves careful tuning and optimization of model parameters and hyperparameters to achieve optimal performance. This process can be iterative and involves multiple aspects:

   a) **Model Architecture:** Choosing the right architecture for the deep learning model is a crucial step. YOLO itself comes in different versions, and selecting the most appropriate one for the project's requirements is essential.
   b) **Loss Function:** Defining an appropriate loss function is critical. It quantifies the error between the predicted outputs and the ground truth annotations and guides the model to improve its predictions.
   c) **Hyperparameters:** Parameters like learning rate, batch size, and the number of training epochs need to be fine-tuned to ensure the model converges effectively.
   d) **Regularization Techniques:** To prevent overfitting, regularization techniques like dropout and weight decay may be applied.

e) **Data Augmentation:** Techniques introduced during data preprocessing, such as data augmentation, continue to play a role in training, further enhancing the model's ability to generalize to different scenarios.

4) YOLO's Real-time Object Detection Capability

One of YOLO's defining features is its ability to perform real-time object detection, making it highly efficient for tracking coins and game pieces during board games. YOLO processes images swiftly and accurately, delivering precise object detection results with low latency. This is especially crucial in gaming scenarios where real-time monitoring and feedback are essential.

The Deep Learning Model Construction module is the heart of the project, where the YOLO algorithm takes center stage in enabling precise coin detection in board games. The model's training process, with careful optimization of parameters and hyperparameters, is a critical phase in ensuring the model's accuracy and robustness. YOLO's real-time object detection capabilities make it the ideal choice for tracking different coins or pieces simultaneously, enhancing the gaming experience by providing real-time monitoring of board game scenarios. The successful implementation of this deep learning model, built upon the foundation of annotated and well-preprocessed data, paves the way for innovative solutions for board game enthusiasts, ensuring precise and automatic monitoring of in-game events.

4.3.4 Model Evaluation

Ensuring Precision and Real-time Detection in Board Games

In the realm of developing a deep learning model for precise coin detection in board games, the Model Evaluation module plays a pivotal role in gauging the model's performance, its ability to accurately detect coins or game pieces, and its suitability for real-time application in gaming scenarios. This phase involves a comprehensive assessment of various metrics, including accuracy, precision, recall, F1 score, and the seamless integration of OpenCV for contour detection, which enables effective tracking of piece movement.

1) Assessing Model Accuracy

Accuracy is a fundamental metric in evaluating the performance of the deep learning model. It measures the model's overall ability to correctly identify and locate coins or game pieces within board game scenarios. High accuracy signifies that the model is making the right predictions consistently, contributing to a successful gaming experience.

2) Precision and Recall

Precision and recall are two critical metrics that offer deeper insights into the model's performance. Precision quantifies the model's ability to make accurate positive predictions, while recall measures its capacity to correctly identify all relevant instances. In the context of board games, precision ensures

that the model doesn't produce false positives, and recall guarantees that it doesn't miss any coins or pieces during detection.

3) **F1 Score**

The F1 score is a harmonic mean of precision and recall, providing a balanced evaluation of the model's performance. It is particularly useful when precision and recall need to be balanced, as it offers a single metric to assess both aspects. Achieving a high F1 score indicates that the model is effective at both making accurate positive predictions and capturing all relevant instances.

4) **Real-time Detection and OpenCV Integration**

Real-time detection is a key requirement for board games, as it ensures that the model can provide immediate feedback during gameplay. This is where the integration of OpenCV (Open Source Computer Vision Library) comes into play. OpenCV complements the YOLO model by facilitating contour detection. By tracking contours, OpenCV can monitor the movement of coins or game pieces on the board in real-time.

This integration allows for the logging of relevant data whenever a coin shifts during gameplay. Such data can be invaluable, not only for tracking the progress of the game but also for enhancing the gaming experience by providing insights, statistics, and real-time analysis. The synergy between YOLO and OpenCV in this context is essential to ensure that the model can adapt to dynamic gaming scenarios and continuously provide accurate information.

5) **Project Success Metrics**

The success of the project should be measured against specific criteria. These may include:

a) The model's ability to accurately detect coins or game pieces in various board game scenarios.
b) Low latency in detection, ensuring that the model operates in real-time.
c) false positives and false negatives in the detection process.
d) The effectiveness of OpenCV in tracking piece movement and capturing relevant data.

6) **Iterative Improvement**

Model evaluation isn't a one-time process; it's iterative. This means that as the model is deployed and used in real-world board game scenarios, continuous evaluation and refinement are necessary. Feedback from users and the model's performance in diverse gaming contexts should guide further model improvement and fine-tuning.

The Model Evaluation module is the final and crucial step in ensuring that the deep learning model for coin detection in board games meets the project's objectives. It focuses on a thorough assessment of accuracy, precision, recall, and the F1 score to gauge the model's performance. Real-time detection is paramount, and the integration of OpenCV for contour detection enhances the model's ability to track

piece movement during gameplay. The project's success metrics should be clear, and the process should be iterative to continually enhance the model's capabilities. This synergy between YOLO, OpenCV, and effective model evaluation ultimately leads to a cutting-edge solution that enhances the gaming experience by providing real-time monitoring of board game scenarios, precise detection of coins, and the capture of relevant data during gameplay.

The project's primary goal is to enhance the gaming experience by providing real-time monitoring of board game scenarios. It showcases the synergy between computer vision methods, where YOLO's object detection capabilities and OpenCV's contour tracking are combined to enable effective object tracking and streamlined data handling for a wide range of gaming applications. The fusion of these technologies opens the door to innovative solutions for board game enthusiasts, ensuring precise and automatic monitoring of in-game events.

CHAPTER 5

RESULTS

5.1 GRAPHICAL CHARTS

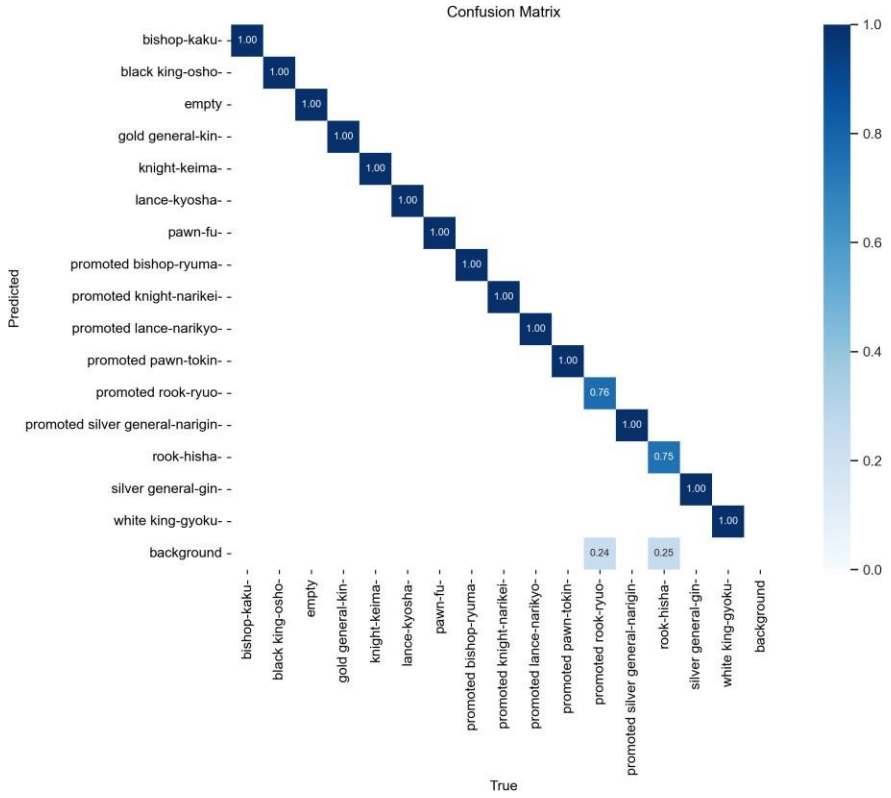

Fig 5.1 – Confusion Matrix

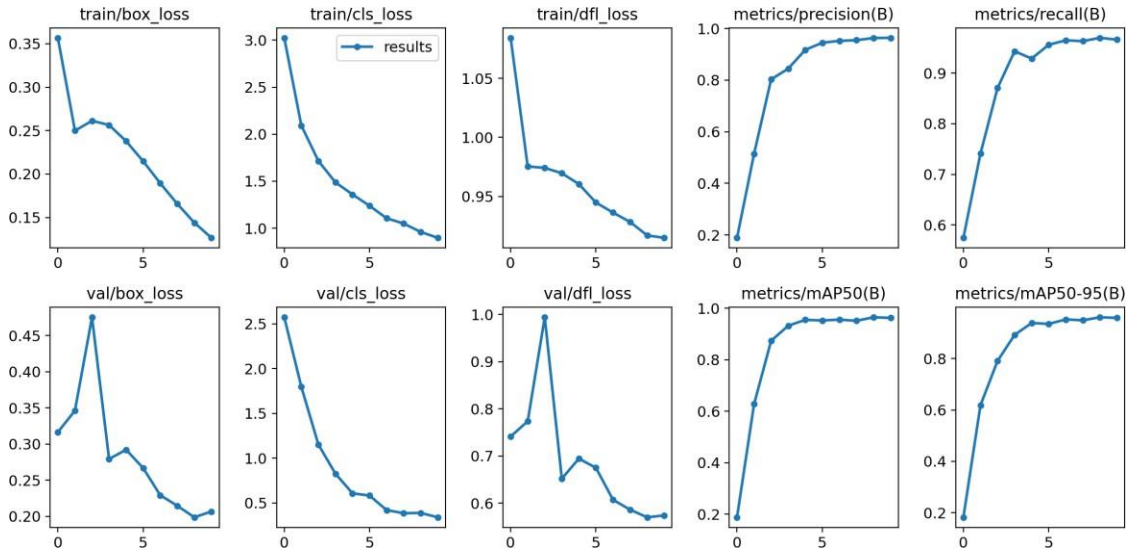

Fig 5.2 – Loss graph

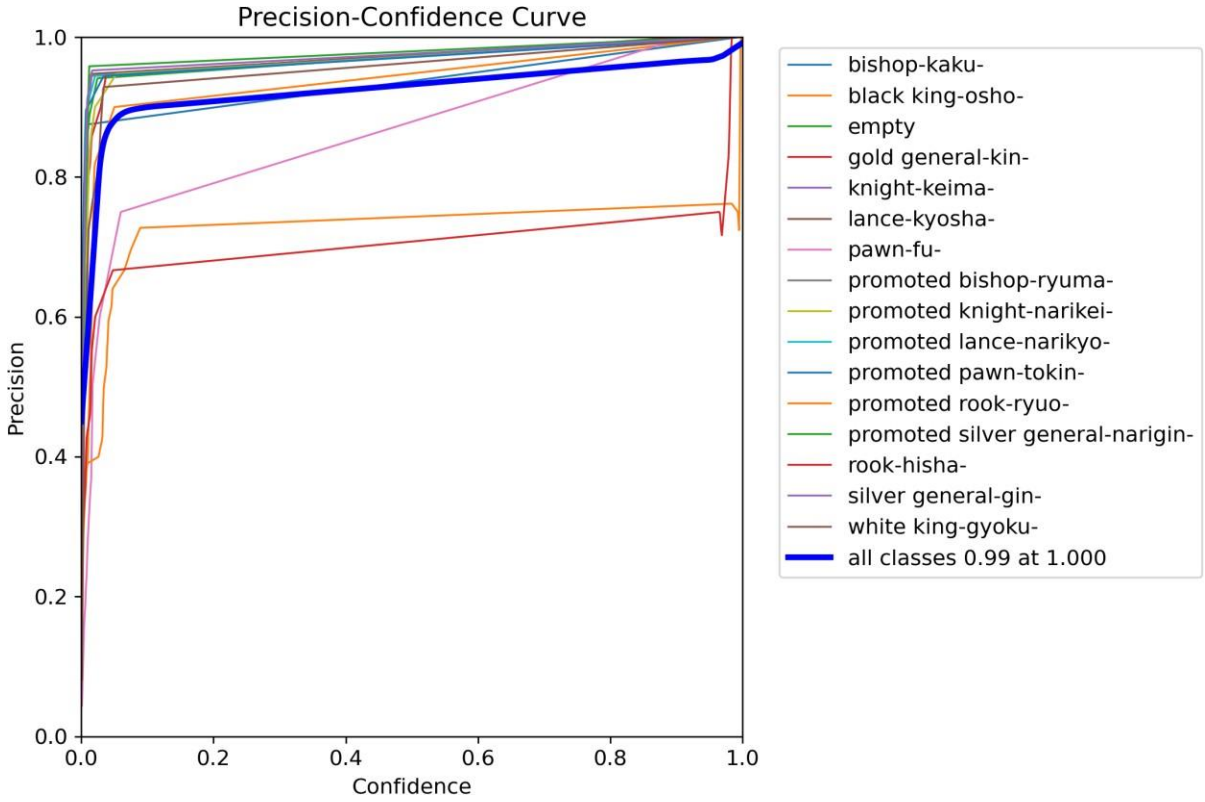

Fig 5.3 – Precision-Confidence Curve

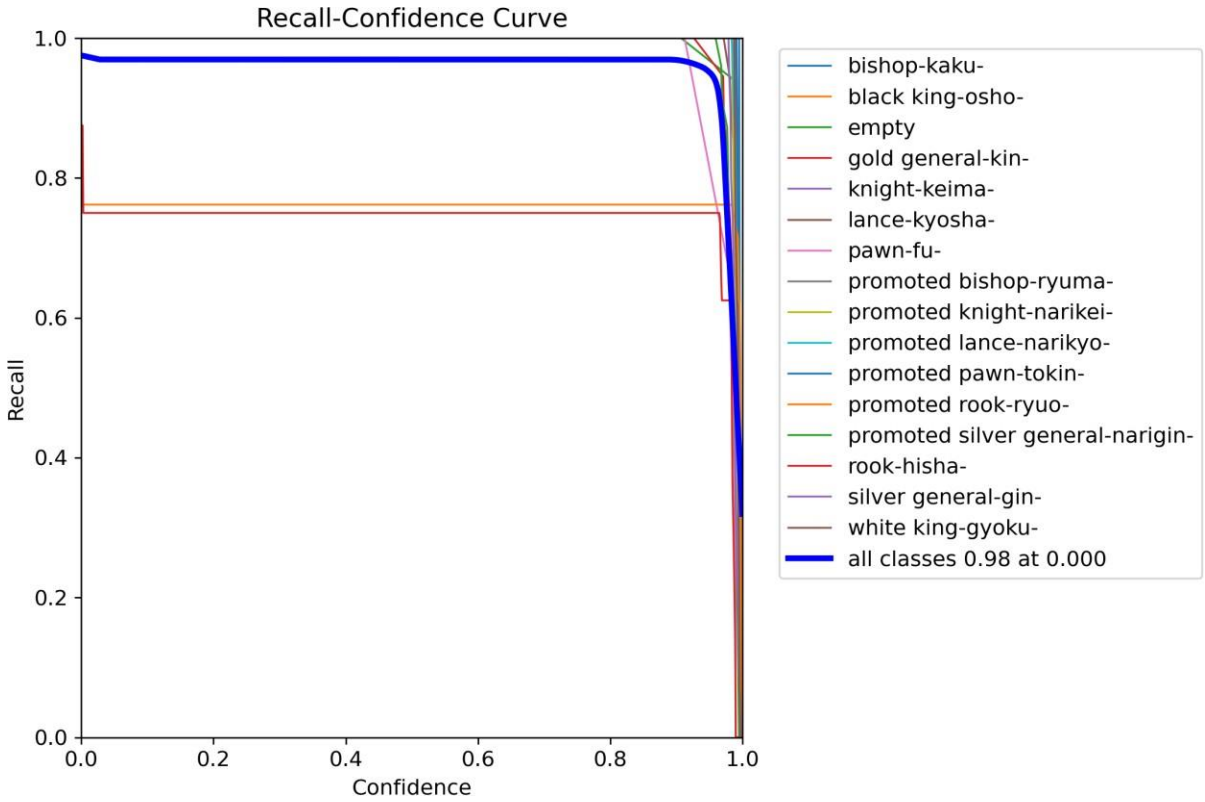

Fig 5.4 – Recall-Confidence Curve

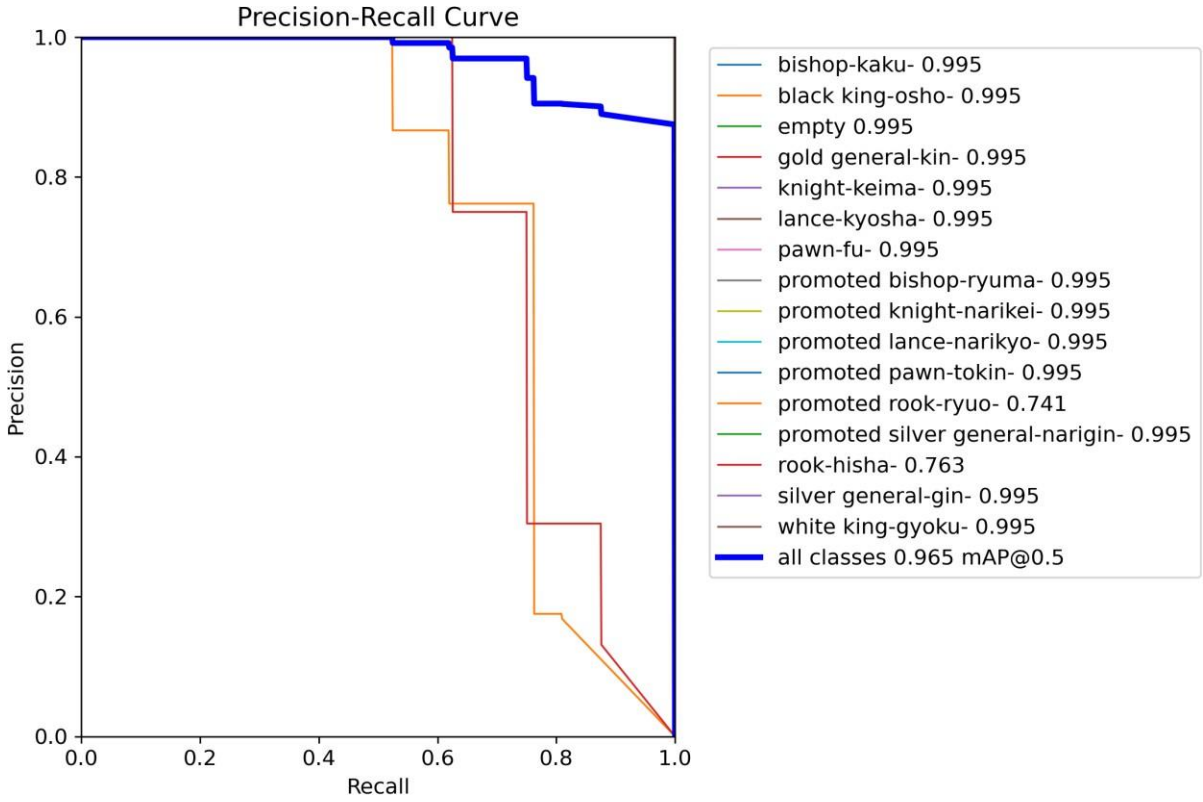

Fig 5.5 – Precision-Recall Curve

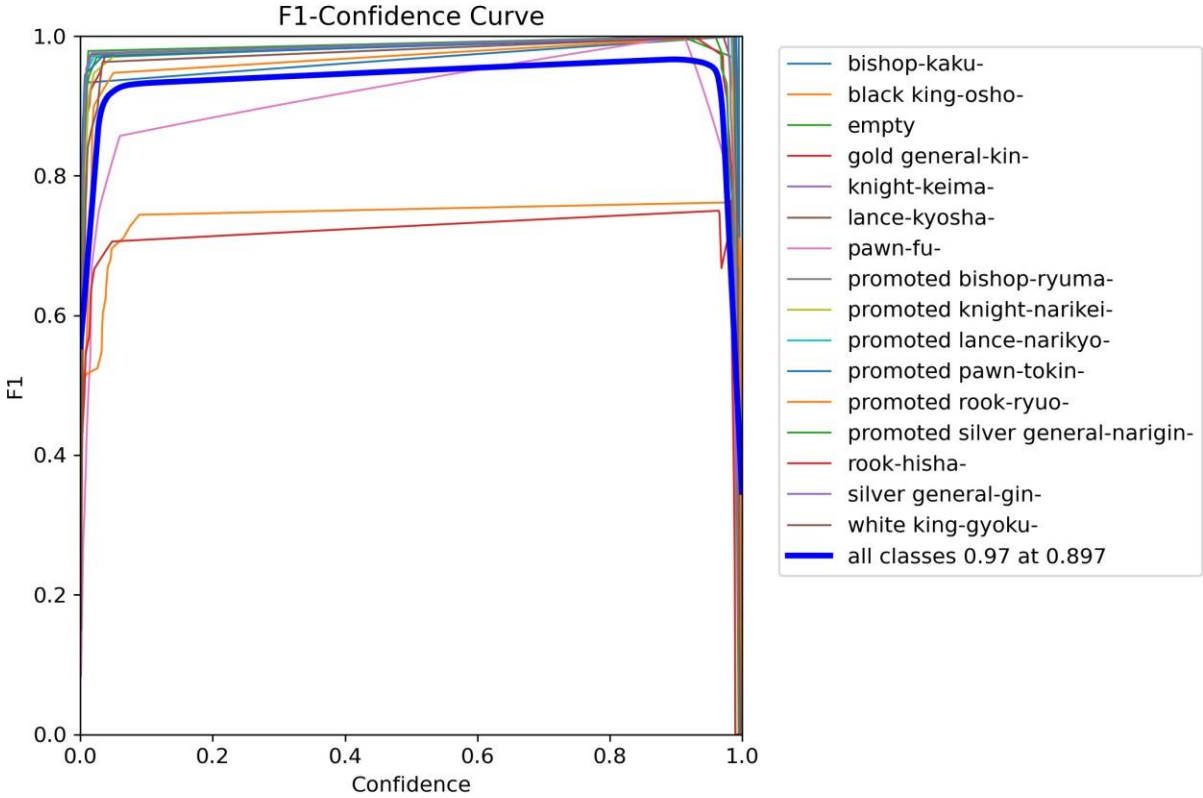

Fig 5.6 – F1-Confidence Curve

5.2 OUTPUT SCREENSHOTS

Fig 5.7 – Testing Screenshot Result - 1

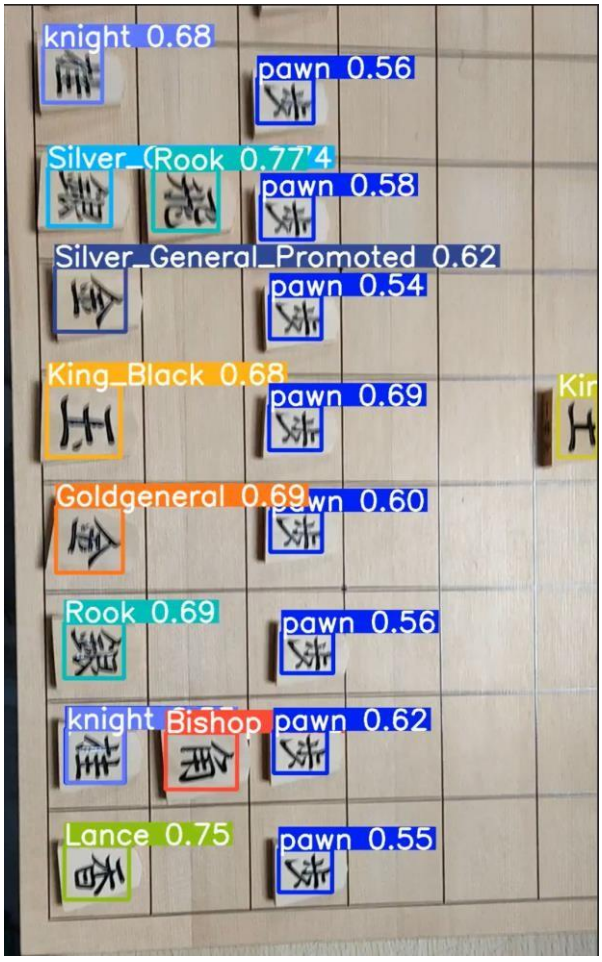

Fig 5.8 – Testing Screenshot Result - 2

Fig 5.9 – Testing Screenshot Result - 3

Fig 5.10 – Testing Screenshot Result - 4

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

In the pursuit of enhancing the traditional game of Shogi through the fusion of modern technology and ancient strategy, our project has focused on Shogi coin detection using cutting-edge object detection and recognition techniques. This endeavor has been both challenging and rewarding, offering promising solutions that contribute to the world of Shogi and computer vision. Our efforts can be summarized as follows:

We initiated our project by curating a diverse dataset of Shogi board images and videos. This dataset's significance lay in its ability to represent a wide array of board setups and lighting conditions, ensuring the robustness of our system. Using the powerful Roboflow tool, we meticulously annotated each image, specifying the class of each Shogi coin and their precise locations within the images. This process laid the groundwork for our subsequent tasks.

Our choice of the YOLOv8 (You Only Look Once) algorithm as the backbone for object detection and recognition was instrumental. YOLOv8, renowned for its real-time capabilities and accuracy, became the cornerstone of our project's success. We integrated this advanced algorithm, tailoring it to our specific use case, thus empowering our system to identify and classify Shogi coins effectively.

With our annotated dataset and YOLOv8 in place, we embarked on the critical stage of model training. We meticulously prepared the data, undertaking data augmentation, normalization, and splitting for optimal model performance. Through rigorous training and the fine-tuning of hyperparameters such as epochs and batch size, our model evolved into a powerful tool for Shogi coin detection. Extensive testing and evaluation against both a dedicated test dataset and newly supplied videos attested to the model's accuracy and real-time capabilities.

6.2 FUTURE ENHANCEMENTS

As we conclude this phase of our Shogi coin detection project, we foresee a path forward brimming with exciting possibilities and future enhancements:

1. Move Prediction

While our system proficiently detects Shogi coins, an exciting future development involves incorporating machine learning techniques to predict the next moves in a Shogi game. Such an addition would elevate our system's utility, offering valuable insights to players and enthusiasts.

2. Mobile App Integration

To ensure wider accessibility, we plan to explore the development of a user-friendly mobile application. This application would democratize Shogi coin detection and bring its benefits to a broader audience.

3. Integration with Shogi Platforms

In addition to real-time analysis and commentary during live matches, we aim to integrate our system with online Shogi platforms. This would provide players with valuable guidance and enhance their gaming experience.

4. Scalability and Adaptability

We recognize the potential for our system to evolve beyond Shogi coin detection. Its robust object recognition capabilities can be adapted to a range of applications, from other board games to inventory management and beyond. Our future works will explore these possibilities.

In conclusion, our Shogi coin detection project has brought together tradition and technology, resulting in a system that has the potential to transform the way Shogi is played and studied. With continued dedication to improvement, innovation, and scalability, we anticipate a bright future where Shogi enthusiasts and technology enthusiasts alike can reap the benefits of our endeavor. As we look ahead, we remain committed to pushing the boundaries of what's possible in the realm of computer vision and object detection.

REFERENCES

[1] D. A. Christie, T. M. Kusuma and P. Musa, "Chess piece movement detection and tracking, a vision system framework for autonomous chess playing robot," 2017 Second International Conference on Informatics and Computing (ICIC), Jayapura, Indonesia, 2017, pp. 1-6, doi: 10.1109/IAC.2017.8280621.

[2] Georg Wölflein, Ognjen Arandjelović, " Determining Chess Game State from an Image," J. Imaging 2021, 7(6), 94; https://doi.org/10.3390/jimaging7060094.

[3] Wölflein, G.; Arandjelović, O. Dataset of Rendered Chess Game State Images; OSF, 2021.

[4] Urting, D.; Berbers, Y. MarineBlue: A Low-Cost Chess Robot. In International Conference Robotics and Applications; IASTED/ACTA Press: Salzburg, Austria, 2003.

[5] Banerjee, N.; Saha, D.; Singh, A.; Sanyal, G. A Simple Autonomous Chess Playing Robot for Playing Chess against Any Opponent in Real Time. In International Conference on Computational Vision and Robotics; Institute for Project Management: Bhubaneshwar, India, 2012.

[6] Chen, A.T.Y.; Wang, K.I.K. Computer Vision Based Chess Playing Capabilities for the Baxter Humanoid Robot. In Proceedings of the International Conference on Control, Automation and Robotics, Hong Kong, China, 28–30 April 2016.

[7] Khan, R.A.M.; Kesavan, R. Design and Development of Autonomous Chess Playing Robot. Int. J. Innov. Sci. Eng. Technol. 2014, 1, 1–4.

[8] Chen, A.T.Y.; Wang, K.I.K. Robust Computer Vision Chess Analysis and Interaction with a Humanoid Robot. Computers 2019, 8, 14.

[9] Gonçalves, J.; Lima, J.; Leitão, P. Chess Robot System: A Multi-Disciplinary Experience in Automation. In Spanish Portuguese Congress on Electrical Engineering; AEDIE: Marbella, Spain, 2005.

[10] Sokic, E.; Ahic-Djokic, M. Simple Computer Vision System for Chess Playing Robot Manipulator as a Project-Based Learning Example. In Proceedings of the IEEE International Symposium on Signal Processing and Information Technology, Sarajevo, Bosnia and Herzegovina, 16–19 December 2008.

[11] Wang, V.; Green, R. Chess Move Tracking Using Overhead RGB Webcam. In Proceedings of the International Conference on Image and Vision Computing New Zealand, Wellington, New Zealand, 27–29 November 2013.

[12] Hack, J.; Ramakrishnan, P. CVChess: Computer Vision Chess Analytics. 2014. Available online: https://cvgl.stanford.edu/teaching/cs231a_winter1415/prev/projects/chess.pdf (accessed on 30 May 2021).

[13] Ding, J. ChessVision: Chess Board and Piece Recognition. 2016. Available online: https://web.stanford.edu/class/cs231a/prev_projects_2016/CS_231A_Final_Report.pdf (accessed on 30 May 2021).

[14] Danner, C.; Kafafy, M. Visual Chess Recognition. 2015. Available online: https://web.stanford.edu/class/ee368/Project_Spring_1415/Reports/Danner_Kafafy.pdf (accessed on 30 May 2021).

[15] Xie, Y.; Tang, G.; Hoff, W. Chess Piece Recognition Using Oriented Chamfer Matching with a Comparison to CNN. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision, Lake Tahoe, NV, USA, 12–15 March 2018.

[16] Czyzewski, M.A.; Laskowski, A.; Wasik, S. Chessboard and Chess Piece Recognition with the Support of Neural Networks. Found. Comput. Decis. Sci. 2020, 45, 257–280.

[17] Mehta, A.; Mehta, H. Augmented Reality Chess Analyzer (ARChessAnalyzer). J. Emerg. Investig. 2020, 2.