**Sudoku Solver Using OpenCV and DL**

**A Project Report**

Submitted in Partial fulfilment of the Requirements for the award of the Degree of

**BACHELOR OF SCIENCE (COMPUTER SCIENCE)**

**By**

**Deepak Jaygopal Gond**

**Seat No: \_\_\_\_\_\_\_\_**

**Under the esteemed guidance of**

**Prof. Javed Pathan**

**Assistant Professor**



DEPARTMENT OF COMPUTER SCIENCE

**RIZVI COLLEGE OF ARTS, SCIENCE AND COMMERCE**

***(Affiliated to University of Mumbai)***

**MUMBAI – 400053**

**MAHARASHTRA**

**2022-2023**

**RIZVI COLLEGE OF ARTS SCIENCE AND COMMERCE**

***(Affiliated to University of Mumbai)***

**MUMBAI MAHARASHTRA-400050**

**DEPARTMENT OF COMPUTER SCIENCE**

**CERTIFICATE**

This certify that the project entitled,” **Sudoku Solver Using OpenCV And DL**”**, is** bonafide work of **Deepak Jaygopal Gond** Bearing **Seat No**: \_\_\_\_\_\_\_\_\_\_ **Roll No: \_\_\_\_\_\_\_\_\_**submitted in partial fulfilment of the requirement for the award of degree of Bachelor of Science in Computer Science from University of Mumbai.

Project Guide HOD

External Examiner

Date …………. College Seal

**ACKNOWLEDGEMENT**

I owe special thanks to the Department of Computer Science of **Rizvi College of Arts, Science & Commerce** for giving me a chance to prepare this project dissertation I thank the Principal, **Professor Anjum Ara Ahmed** for her leadership and management.

I thank the Coordinator and Head of the Department **Professor Arif Patel** for providing us the required facilities and guidance throughout the course which culminated into this thesis.

Last and not the least to the project guide this semester, **Professor Javed Pathan**. Deep gratitude to the staff and the faculty of Rizvi College for their help and support.

And also, my beloved Parent and Classmate for their infinite Support and Love.

**ABSTRACT**

The Sudoku Solver project is an implementation of a computer vision system that can detect and recognize digits in a Sudoku puzzle using OpenCV and deep learning techniques. The system captures a video stream from a camera, and then performs a series of image processing steps to identify the Sudoku puzzle and extract the individual cells. These cells are then passed through a convolutional neural network (CNN) to recognize the digits in each cell. The system then solves the puzzle using a backtracking algorithm and displays the solution on the video stream.

Overall, this project demonstrates the power of computer vision and deep learning techniques to solve real-world problems. The Sudoku Solver system has potential applications in automated puzzle solving, image recognition, and optical character recognition (OCR) in general. Additionally, it showcases the capabilities of OpenCV and its libraries, as well as the usefulness of integrating different machine learning models and algorithms to achieve a complex task.

**DECLARATION**

I hereby declare that the project entitled, **“Sudoku Solver Using OpenCV And DL”** done at **Rizvi College of Arts, Science and Commerce**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfilment of the requirements for the award of degree of **BACHELOR OF SCIENCE (COMPUTER SCIENCE)** to be submitted as final year; Fifth semester project as part of our curriculum.

Deepak Jaygopal Gond

**Table Of Contents**

[**CHAPTER** **1.** **INTRODUCTION 1**](#_Toc114771330)

[1.1 Introduction to the system 1](#_Toc114771331)

[1.2 Problem Definition 1](#_Toc114771332)

[1.3 Aim 1](#_Toc114771333)

[1.4 Objective 1](#_Toc114771334)

[1.5 Goal 2](#_Toc114771335)

[1.6 Need of System 2](#_Toc114771336)

[**CHAPTER** 2. **REQUIREMENT** **SPECIFICATION 3**](#_Toc114771337)

[2.2 System Environment 3](#_Toc114771338)

[2.3 Software Requirement 3](#_Toc114771339)

[2.4 Hardware Requirement 3](#_Toc114771340)

[2.5 Methodology 3](#_Toc114771341)

[**CHAPTER 3. SYSTEM ANALYSIS 4**](#_Toc114771342)

[3.1 Introduction 4](#_Toc114771343)

[3.2 System Analysis 4](#_Toc114771344)

[3.2.1 Analysis of Existing System 4](#_Toc114771345)

[3.2.2 Analysis of Purposed System 4](#_Toc114771346)

[3.3 Gantt Chart 4](#_Toc114771347)

[**CHAPTER 4. SURVEY OF TECHNOLOGY 5**](#_Toc114771349)

[**CHAPTER 5. SYSTEM DESIGN 6**](#_Toc114771350)

[5.1 Introduction 6](#_Toc114771351)

[5.2 System Architecture 6](#_Toc114771352)

[5.3 Data Flow Diagram 6](#_Toc114771353)

[**CHAPTER 6. SYSTEM IMPLEMENTATION 7**](#_Toc114771355)

[6.1 Introduction 7](#_Toc114771356)

[6.2 Flowchart 7](#_Toc114771357)

[6.3 Code: 7](#_Toc114771358)

[6.4 Testing Approach 7](#_Toc114771359)

[6.5 Testing Case 7](#_Toc114771360)

[**CHAPTER 7. RESULTS 8**](#_Toc114771361)

[**CHAPTER** **8.** **CONCLUSION** **AND** **FUTURE** **SCOPE 9**](#_Toc114771362)

[8.1 Conclusion 9](#_Toc114771363)

[8.1.1 Advantage 9](#_Toc114771364)

[8.1.2 Limitation 9](#_Toc114771365)

[8.2 Future Improvement 9](#_Toc114771366)

[**CHAPTER 9. REFERENCE 10**](#_Toc114771367)

[9.1 Reference and Biography 10](#_Toc114771368)

**CHAPTER 1. INTRODUCTION**

* 1. **Introduction to the system**

Sudoku is a classic puzzle game that challenges players to fill a 9x9 grid with digits so that every row, column, and 3x3 sub-grid contains all the numbers from 1 to 9. Solving a Sudoku puzzle requires logical reasoning and critical thinking, making it a popular game for people of all ages.

With the advancements in computer vision and deep learning, it is now possible to automate the process of solving Sudoku puzzles using computers. In this project, we will be using OpenCV, a popular computer vision library, and deep learning techniques to create a Sudoku solver.

The Sudoku solver will take an image of a puzzle grid as input and use image preprocessing techniques to isolate the digits within the grid. Then, a convolutional neural network (CNN) will be trained to recognize the digits, allowing the solver to identify the numbers within the grid.

Once the digits have been identified, the solver will apply the rules of Sudoku to solve the puzzle using a backtracking algorithm. The solver will use logic and reasoning to fill in the missing digits, and once the puzzle has been solved, the solution will be displayed.

Overall, this project demonstrates how advanced technologies such as computer vision and deep learning can be used to automate complex tasks such as solving Sudoku puzzles. It also showcases the power of machine learning algorithms in recognizing and understanding complex patterns within images.

* 1. **Problem Definition**

The Sudoku Solver using OpenCV and Deep Learning project aims to solve the classic Sudoku puzzle game using advanced computer vision techniques and deep learning algorithms. The problem definition involves building a system that can take an image of a Sudoku puzzle as input, recognize the digits within the grid, and solve the puzzle using a backtracking algorithm.

The main challenge in this project is to develop an algorithm that can accurately identify the digits within the puzzle grid. This requires a combination of image pre-processing techniques, such as thresholding and contour detection, and machine learning techniques, such as training a CNN to recognize the digits.

Another challenge is to develop an efficient algorithm for solving the puzzle using the recognized digits. This requires applying the rules of Sudoku to fill in the missing digits while avoiding conflicts with existing digits in the grid.

The project also involves the challenge of optimizing the algorithm for real-time performance. Since Sudoku puzzles can be of varying levels of difficulty, the solver needs to be able to handle puzzles of varying complexity while maintaining fast performance.

To ensure the accuracy of the solver, the system needs to be tested on a variety of different Sudoku puzzles to ensure that it can handle a wide range of input images. Additionally, the solver needs to be able to handle input images with varying levels of noise and distortion.

Overall, the problem definition for the Sudoku Solver using OpenCV and Deep Learning project involves developing a robust, efficient, and accurate system for solving Sudoku puzzles using advanced computer vision and machine learning techniques.

* 1. **Aim**

The aim of the Sudoku Solver using OpenCV and Deep Learning project is to develop an efficient and accurate system that can solve Sudoku puzzles automatically. The project utilizes advanced computer vision and deep learning techniques to recognize the digits within the puzzle grid and apply the rules of Sudoku to solve the puzzle.

The primary aim of the project is to accurately recognize the digits within the puzzle grid. This requires developing a CNN-based digit recognition system that can identify the digits even in noisy and distorted images.

Another aim of the project is to optimize the performance of the solver to handle puzzles of varying levels of difficulty while maintaining real-time performance. The solver needs to be able to handle input images with varying levels of noise, distortion, and other image artifacts.

The project also aims to provide a user-friendly interface that allows users to input images of Sudoku puzzles and receive solutions in a user-friendly format. The interface should be simple and intuitive to use, even for users with limited technical knowledge.

Another aim of the project is to develop a system that is robust and can handle a wide variety of input images. The solver needs to be able to recognize digits accurately even in images with varying lighting conditions, background noise, and other image artifacts.

The final aim of the project is to showcase the potential of advanced computer vision and deep learning techniques in automating complex tasks such as Sudoku puzzle solving. The project aims to demonstrate how machine learning algorithms can be used to recognize complex patterns within images and use them to solve real-world problems.

* 1. **Objective**

The objective of the Sudoku Solver using OpenCV and Deep Learning project is to develop a system that can automatically solve Sudoku puzzles with high accuracy and efficiency. The project aims to achieve the following objectives:

1. Develop an efficient image processing pipeline that can detect and isolate the puzzle grid and the individual cells within the grid.
2. Train a CNN model to accurately recognize the digits within the puzzle grid, even in images with varying levels of noise and distortion.
3. Implement a backtracking algorithm that can apply the rules of Sudoku to fill in the missing digits in the grid while avoiding conflicts with existing digits.
4. Optimize the performance of the solver to handle puzzles of varying levels of difficulty while maintaining real-time performance.
5. Develop a user-friendly interface that allows users to input images of Sudoku puzzles and receive solutions in a user-friendly format.
6. Test the solver on a variety of input images to evaluate its accuracy, efficiency, and robustness.

Overall, the objective of the Sudoku Solver using OpenCV and Deep Learning project is to showcase the potential of advanced computer vision and machine learning techniques in automating complex tasks such as Sudoku puzzle solving. The project aims to develop a robust and efficient system that can accurately solve Sudoku puzzles from input images, with potential applications in various real-world scenarios.

* 1. **Goal**

The Sudoku Solver using OpenCV and Deep Learning project aims to develop a system that can solve Sudoku puzzles automatically from input images. The goals of the project are as follows:

1. Digit recognition: The first goal is to develop a CNN model that can accurately recognize the digits within the puzzle grid. The model should be trained on a large and diverse dataset of handwritten digits to ensure that it can generalize well to different images.
2. Grid detection: The second goal is to develop an efficient image processing pipeline that can detect and isolate the puzzle grid and the individual cells within the grid. The pipeline should be able to handle images with varying levels of noise and distortion.
3. Sudoku solving algorithm: The third goal is to implement a backtracking algorithm that can apply the rules of Sudoku to fill in the missing digits in the grid while avoiding conflicts with existing digits. The algorithm should be optimized for real-time performance and able to handle puzzles of varying levels of difficulty.
4. User interface: The fourth goal is to develop a user-friendly interface that allows users to input images of Sudoku puzzles and receive solutions in a user-friendly format. The interface should be simple and intuitive to use, even for users with limited technical knowledge.
5. Robustness: The fifth goal is to ensure that the system is robust and can handle a wide variety of input images. The solver should be able to recognize digits accurately even in images with varying lighting conditions, background noise, and other image artifacts.
6. Performance: The sixth goal is to optimize the performance of the solver to handle puzzles of varying levels of difficulty while maintaining real-time performance. The solver should be able to handle puzzles of varying complexity while maintaining fast performance.
7. Accuracy: The seventh goal is to ensure that the solver is accurate and can solve Sudoku puzzles correctly. The solver should be able to handle puzzles of varying levels of difficulty with high accuracy.
8. Generalization: The eighth goal is to ensure that the system can generalize well to different types of Sudoku puzzles. The solver should be able to handle puzzles of different sizes, shapes, and levels of complexity.
9. Testing and evaluation: The ninth goal is to evaluate the accuracy, efficiency, and robustness of the system through extensive testing on a variety of input images. The system should be tested on both synthetic and real-world datasets to ensure that it can handle a wide range of scenarios.

Overall, the goals of the Sudoku Solver using OpenCV and Deep Learning project are to develop a robust, efficient, and accurate system that can solve Sudoku puzzles automatically from input images. The project aims to showcase the potential of advanced computer vision and deep learning techniques in automating complex tasks such as Sudoku puzzle solving.

**1.6 Need of System**

Sudoku puzzles have been popular for many years, and solving them manually can be a challenging and time-consuming task. Therefore, there is a need for an automated Sudoku solver that can quickly and accurately solve puzzles. The use of computer vision and deep learning techniques can provide a solution to this problem. The following paragraphs will discuss the need for a Sudoku Solver using OpenCV and Deep Learning.

1. Accuracy: Manually solving a Sudoku puzzle can be prone to errors, especially if the puzzle is complicated. Therefore, an automated solver can help eliminate the chances of making mistakes while solving the puzzle.
2. Efficiency: Solving a Sudoku puzzle manually can be a time-consuming task. However, an automated solver can quickly solve puzzles within a few seconds, providing a more efficient way to solve puzzles.
3. Accessibility: Not everyone has the ability to solve Sudoku puzzles manually, which can make it challenging for them to enjoy this activity. By developing an automated solver, we can make Sudoku puzzles more accessible to everyone.
4. Learning: Solving Sudoku puzzles can improve cognitive skills such as logical thinking, problem-solving, and memory. However, not everyone has the ability to solve these puzzles. By developing an automated solver, we can provide an opportunity for people to learn from the solutions generated by the solver.
5. Real-world applications: Sudoku puzzles are not only used as a recreational activity but also have real-world applications. For example, Sudoku puzzles are used in computer science and cryptography to test algorithms and security protocols. Therefore, an automated solver can help solve these puzzles more efficiently.
6. Innovation: The development of an automated solver using computer vision and deep learning techniques can showcase the potential of these technologies in solving complex tasks. This project can encourage further research and innovation in the field of computer vision and deep learning.
7. Accessibility for the visually impaired: Individuals who are visually impaired face challenges while solving Sudoku puzzles manually. However, an automated solver that can recognize and verbalize the solutions can make these puzzles accessible to them.
8. Efficiency in game design: Game developers can use automated solvers to create and test new Sudoku puzzles more efficiently. This can lead to the development of more complex and challenging puzzles, enhancing the overall gaming experience.
9. Educational purposes: Developing an automated solver can be used as a teaching tool in computer vision and deep learning courses. It can help students understand the practical applications of these technologies in real-world problems, providing a hands-on learning experience.

Overall, the development of a Sudoku solver using OpenCV and deep learning techniques can provide an efficient and accurate way to solve Sudoku puzzles, making it more accessible to everyone, including those with visual impairments. Additionally, it can have various real-world applications and provide opportunities for learning, innovation, and game development.

**CHAPTER 2. REQUIREMENT SPECIFICATION**

**2.1** **Introduction**

To develop a Sudoku Solver using OpenCV and Deep Learning (OCR), certain requirements need to be met to ensure the accuracy and efficiency of the solver. The Sudoku puzzle is presented in a 9x9 grid format, and the solver must be able to recognize each cell's number accurately. Therefore, the Sudoku Solver requires an OCR model trained using a large dataset of handwritten digits, along with image processing techniques and integration with the Sudoku Solver to input recognized digits into the solver. Additionally, the solver should have a user-friendly interface, verbal output, integration with mobile devices, optimization techniques, and error handling mechanisms to provide a comprehensive Sudoku-solving experience for users.

**2.2 System Environment**

The Sudoku Solver using OpenCV and Deep Learning (OCR) requires a system that can handle the computational requirements of image processing, OCR, and solving algorithms. The following paragraphs outline the system requirements in detail.

1. Processor: The system should have a high-performance processor to handle the computational requirements of image processing, OCR, and solving algorithms. A multi-core processor with a clock speed of 3.0 GHz or higher is recommended.
2. Memory: The system should have sufficient memory to handle the large datasets required for OCR training and image processing. A minimum of 8 GB of RAM is recommended.
3. Graphics Card: A dedicated graphics card is not required, but it can significantly improve the system's performance when processing images and running optimization techniques.
4. Operating System: The Sudoku Solver can be developed on various operating systems such as Windows, MacOS, or Linux. However, the chosen operating system must be compatible with the development tools and libraries used in the project.
5. Development Tools: The Sudoku Solver requires several development tools and libraries, including OpenCV, TensorFlow, Keras, and Python. Therefore, the system must have these tools installed and configured correctly to develop and run the solver.

Overall, the Sudoku Solver using OpenCV and Deep Learning (OCR) requires a high-performance processor, sufficient memory, a compatible operating system, a graphics card (optional), and the required development tools and libraries to ensure the accuracy and efficiency of the solver.

**2.3 Software Requirement**

To develop the Sudoku Solver using OpenCV and Deep Learning (OCR), several software requirements must be met to ensure the accuracy and efficiency of the solver. The following paragraphs outline the software requirements in detail.

1. Python: Python is an interpreted, high-level programming language used in machine learning, data science, and image processing. The Sudoku Solver requires Python version 3.6 or higher to develop and run.
2. OpenCV: OpenCV (Open-Source Computer Vision Library) is a library of programming functions used for real-time computer vision. It is used in image processing techniques such as image filtering, edge detection, and image segmentation, which are necessary for the Sudoku Solver. The latest version of OpenCV should be installed and configured correctly.
3. TensorFlow: TensorFlow is an open-source software library used for dataflow and differentiable programming across a range of tasks. It is used to train the OCR model used in the Sudoku Solver. The latest version of TensorFlow should be installed and configured correctly.
4. Keras: Keras is an open-source software library used for developing and training neural networks. It is used in the OCR model to recognize digits from the Sudoku puzzle. The latest version of Keras should be installed and configured correctly.
5. Tesseract: Tesseract is an open-source OCR engine used to recognize text from images. It is used in the Sudoku Solver to recognize the numbers in each cell of the puzzle.
6. Streamlit: Streamlit is an open-source Python library used to build custom web applications for machine learning and data science projects. It is used to create the user interface for the Sudoku Solver. The latest version of Streamlit should be installed and configured correctly.
7. Streamlit Webrtc: Streamlit Webrtc is an open-source Python library used for real-time video and audio streaming applications. It is used in the Sudoku Solver to allow users to capture images of Sudoku puzzles from their webcam. The latest version of Streamlit Webrtc should be installed and configured correctly.
8. Streamlit-av: Streamlit-av is an open-source Python library used for audio and video playback in Streamlit applications. It is used in the Sudoku Solver to play a sound when the puzzle is solved. The latest version of Streamlit-av should be installed and configured correctly.
9. NumPy: NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions. It is used to manipulate arrays and matrices in the Sudoku Solver.
10. Git: Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. It is used in the Sudoku Solver to manage the source code, track changes, and collaborate with team members.

Overall, the Sudoku Solver using OpenCV and Deep Learning (OCR) requires Python, OpenCV, TensorFlow, Keras, Tesseract, NumPy,Streamlit,Streamlit-av,Streamlit Webrtc, and Git to develop and run the solver. All these software requirements should be installed and configured correctly to ensure the accuracy and efficiency of the Sudoku Solver.

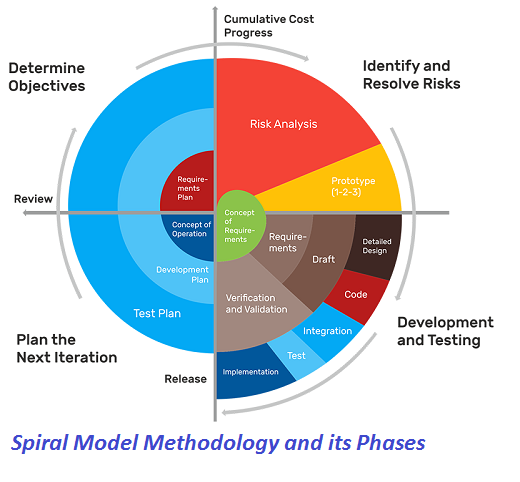
**2.4 Hardware Requirement**

The hardware requirements for the Sudoku Solver using OpenCV and Deep Learning (OCR) can be summarized as follows:

1. Processor: A modern CPU with a clock speed of at least 2.5 GHz is recommended for efficient image processing and machine learning tasks.
2. RAM: At least 8 GB of RAM is required, although a higher amount of RAM may be necessary for more complex puzzles and faster processing.
3. Storage: At least 500 GB of free storage space should be available to store the required software and data files.
4. Graphics Card: A graphics card with CUDA support can significantly improve the performance of image processing and machine learning tasks, but is not strictly necessary.
5. Webcam and Microphone: A high-quality webcam is required for capturing images of Sudoku puzzles, and a microphone is required for playing the sound when the puzzle is solved.
6. Monitor and Internet Connection: A high-resolution monitor is recommended to view the Sudoku puzzles and the web-based user interface, and an internet connection is required to install and update software packages and to use the web-based user interface.

**2.5 Methodology**

In order to achieve goals and planned results within a defined schedule and a budget, a methodology in used in a project. Regardless of which field or which trade, there are assortments of methodologies to help managers at every stage of a project from the initiation to implementation to the closure. A methodology is a model, which project managers employ for the design, planning, implementation and achievement of their project objectives. There are different project management methodologies to benefit different projects.



The spiral model, initially proposed by Boehm, is an evolutionary software process model that couples the iterative feature of prototyping with the controlled and systematic aspects of the linear sequential model. It implements the potential for rapid development of new versions of the software. Using the spiral model, the software is developed in a series of incremental releases. During the early iterations, the additional release may be a paper model or prototype. During later iterations, more and more complete versions of the engineered system are produced.

Each cycle in the spiral is divided into four parts:

1. Objective setting: Each cycle in the spiral starts with the identification of purpose for that cycle, the various alternatives that are possible for achieving the targets, and the constraints that exists.

2. Risk Assessment and reduction: The next phase in the cycle is to calculate these various alternatives based on the goals and constraints. The focus of evaluation in this stage is located on the risk perception for the project.

3. Development and validation: The next phase is to develop strategies that resolve uncertainties and risks. This process may include activities such as benchmarking, simulation, and prototyping.

4. Planning: Finally, the next step is planned. The project is reviewed, and a choice made whether to continue with a further period of the spiral. If it is determined to keep, plans are drawn up for the next step of the project. The development phase depends on the remaining risks. For example, if performance or user-interface risks are treated more essential than the program development risks, the next phase may be an evolutionary development that includes developing a more detailed prototype for solving the risks. The risk-driven feature of the spiral model allows it to accommodate any mixture of a specification-oriented, prototype-oriented, simulation-oriented, or another type of approach. An essential element of the model is that each period of the spiral is completed by a review that includes all the products developed during that cycle, including plans for the next cycle. The spiral model works for development as well as enhancement projects. task performed by every user once the software has been delivered to the customer, installed, and operational.

The steps for Spiral Model can be generalized as follows:

* The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
* A preliminary design is created for the new system.
* A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product
* A second prototype is evolved by a fourfold procedure:

1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
2. Defining the requirements of the second prototype.
3. Planning a designing the second prototype.
4. Constructing and testing the second prototype.

* At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer’s judgment, result in a less-than-satisfactory final product.
* The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
* The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
* The final system is constructed, based on the refined prototype.
* The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

**CHAPTER 3. SYSTEM ANALYSIS**

**3.1 Introduction**

System analysis for the Sudoku Solver using OpenCV and Deep Learning (OCR) involves examining the system's requirements, design, and functionality to ensure that it meets the needs of its users. This process involves identifying the system's components, evaluating their interactions, and optimizing their performance.

**3.2 System Analysis**

**3.2.1 Analysis of Existing System**

In computer programming, there are a variety of methods to solve Sudoku puzzles and some are brieﬂy described as follows:

1. Backtracking: used in many applications such as eight queen puzzles.

It visits unﬁlled cells in arbitrary order then ﬁlls in digits sequentially from

possible choices, and backtrack (i.e., discard unsuccessful choices) when dead-

locks are met. At each backtracking time, it changes the digit in the cell most

recently ﬁlled before the deadlock took place. If that speciﬁc cell is attempted

with every single possible digit, the algorithm goes back to the second prior

cell ﬁlled before the last deadlock and iterates that cell’s digit. Backtracking

algorithm is simple to implement, however, it needs a large amount of

memory space because of the recursive technique.

1. Brute Force algorithm: The basic idea is to go through all possible solutions

extensively. It does many iterations to look for all possible solutions for

Sudoku puzzles. If found solutions cannot solve the problem, the algorithm

removes them and roll-backs to the original solutions then try again. It does

not require a huge of memory space, but it requires a lot of processing time.

1. Stochastic search: this approach ﬁrstly assigns randomly digits to the

empty cells in the grid. Then, it calculates the number of errors, and rear-

ranges these ﬁlled digits around the grid until the number of errors is reduced

to zero. Finally, the solution to the Sudoku puzzle is found. The process of

stochastic search is pretty quick and consumes less memory space, but it is

diﬃcult to implement.

1. Constraint propagation: this method applies a set of criteria to the

possible candidates to ﬁnd the solution. A candidate that satisﬁes all the

criteria will be the solution to the puzzle. This typical algorithm is applied

in this study to build the application because of its fast and eﬃciency. Some

appropriate constraints are used to eliminate the candidates to reduce the

complexity whilst ﬁnding solutions.

The existing Sudoku Solvers have been developed using different techniques, ranging from traditional computer vision to machine learning-based approaches. Although these techniques can solve Sudoku puzzles with varying degrees of accuracy, there are still limitations that can affect the performance and usability of the system.

One of the main challenges of the existing systems is their sensitivity to lighting conditions and noise in the input image. The traditional computer vision techniques such as edge detection, contour detection, and Hough Transform can be affected by shadows, reflections, and other sources of noise. This can result in inaccurate extraction of the Sudoku grid, leading to incorrect recognition of the digits.

The machine learning-based approaches such as SVM and k-NN, on the other hand, require large amounts of labelled training data to achieve high accuracy. However, these algorithms can struggle with low-contrast and blurry images, resulting in poor recognition performance.

Moreover, the existing systems often lack a user-friendly interface that can simplify the process of inputting the puzzle and displaying the solution. This makes it difficult for non-technical users to use the system and can limit its adoption.

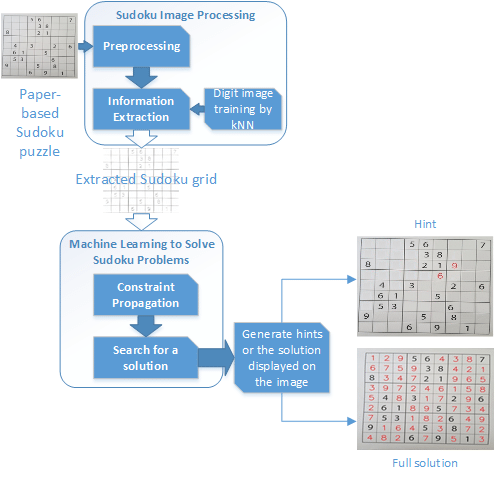
Another limitation of the existing systems is their scalability. Traditional computer vision techniques can struggle to recognize digits in large images, while machine learning-based approaches can be computationally intensive, which can limit the system's scalability.

In summary, the existing Sudoku Solvers can solve Sudoku puzzles with varying degrees of accuracy, but they can be affected by noise and lighting conditions, and often lack user-friendly interfaces. These limitations can affect the system's usability and scalability.

**3.2.2 Analysis of Purposed System**

To build the Sudoku solver application in this study, a computer vision aided framework is proposed with two main parts: (1) Sudoku image processing, and (2) machine learning to solve Sudoku problems, as shown in Fig. 2

Fig. 2. The proposed framework.

Initially providing a paper based

Sudoku puzzle, the tool automatically

captures, by an attached camera, its

image. Next, a series of image pro-

cessing and machine learning methods

are utilised to extract all necessary

information, i.e., the digits and their

correspondent cells to form a Sudoku

puzzle ready to solve. These processes

reduce considerably amount of time,

comparing to manually input digits

into the Sudoku grid. Then, the puzzle

is passed through the solver to ﬁnd an

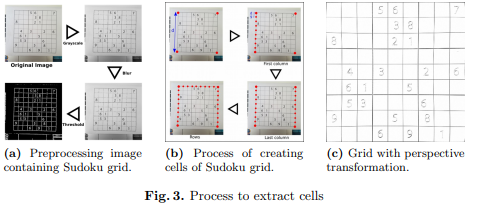
unique solution or produce hints for

users when they consult the tool. The

following sections provide each step in

details of the framework.

Sudoku Image Processing& Machine Learning: This component’s key task is to enable receiving Sudoku puzzles from the cam-era. It has two main steps: pre-processing to enhance and standardise input im-ages allowing the information extraction step obtain data more accurately. Reprocessing is a necessary and crucial step. In this step, a captured puzzle image is processed through a series of pre-processing methods including grayscale, blurring, thresholding [5] on the Sudoku grid area needed to ﬁltrate. After pre-processing (Fig. 3a), all the object contours in the image are now clearer and more accurate. Moreover, it is compulsory that the Sudoku grid is at the centre of the camera. Therefore, ﬁnding the Sudoku grid in the image can be done by looking for the biggest contour which has 4 vertices, named as top-left, top-right, bottom-right, bottom-left. The top-left point has the smallest value of x and y-coordinate, whilst the bottom-right point has the largest one. These points are also used to draw all grid cells 3. After the whole Sudoku grid is positioned, the perspective transformation will be performed. A morphological transformation, with the ellipse kernel (11x11), dilates then erodes the image. Then, it is normalized to obtain a better processing result 4. Fig. 3c describes the transformation of the extracted Sudoku grid.



The next step is to locate the digits in the Sudoku grid and to recognize

them. A cell of the corresponding digit is determined by coordinates of that

digit divided by 50 5.

Information Extraction: There are several ways to perform the optimal

digit recognition (ODR) such as [6, 7]. However, kNN is applied because it is

simple and requires only a small data to recognise digits. Training data is a step

to obtain the database for the kNN algorithm. The training samples are digits

from 0 to 9 in 10 popular fonts. During the training phase, the data of digit

contours and the learned digits are stored as a database.

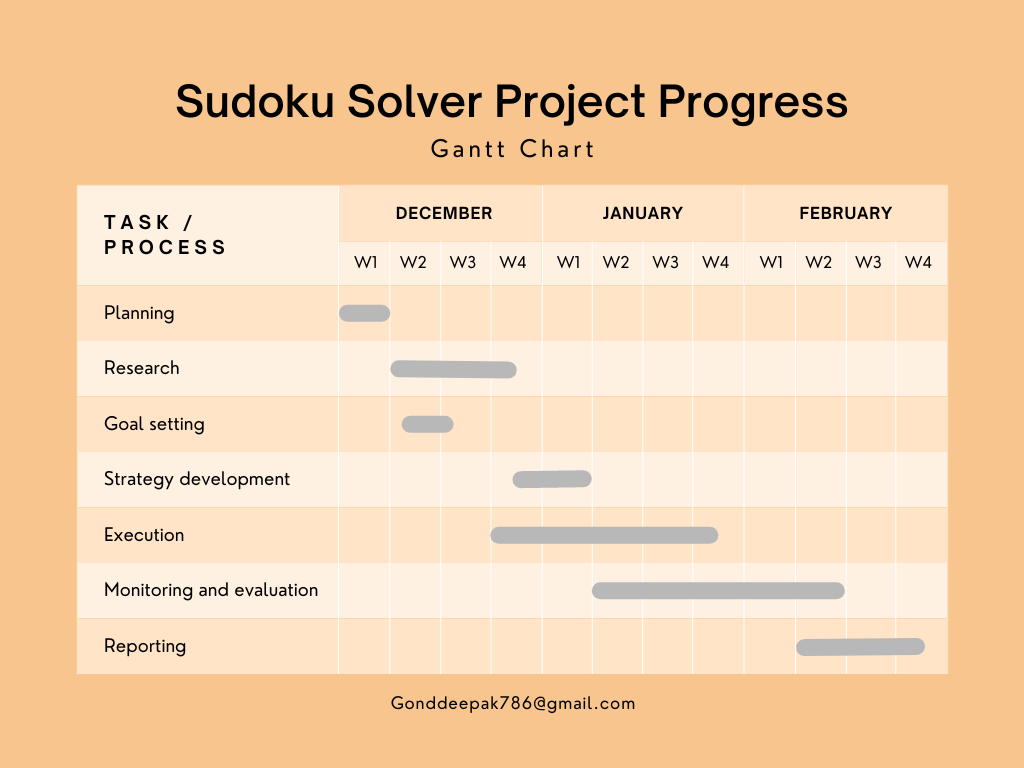
After training, extracting or reading digits from a pre-processed Sudoku im-

age can be achieved 6. Each contour will be compared with the database to

identify the correct number. Its position in which cell is also determined

**3.3 Gantt Chart**

A Gantt chart is a type of bar chart that illustrates a project schedule. This chart lists the tasks to be performed on the vertical axis, and time intervals on the horizontal axis. The width of the horizontal bars in the graph shows the duration of each activity. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements constitute the work breakdown structure of the project. Modern Gantt charts also show the dependency relationships between activities. Gantt charts can be used to show current schedule status using percent-complete shadings and a vertical "TODAY" line as shown here. Gantt charts are sometimes equated with bar charts. Gantt charts are usually created initially using an early start time approach, where each task is scheduled to start immediately when its prerequisites are complete. This method maximizes the float time available for all tasks.:



**CHAPTER 4. SURVEY OF TECHNOLOGY**

**TensorFlow:**

TensorFlow is an open-source machine learning framework developed by Google Brain Team. It provides a set of tools for building and training machine learning models, including a comprehensive set of APIs and pre-built models. The framework has become popular among developers, researchers, and businesses due to its flexibility, scalability, and ease of use.

One of the key features of TensorFlow is its ability to create and train neural networks. TensorFlow provides an extensive library of pre-built layers, activations, and loss functions, as well as support for custom layers and functions. It also supports distributed computing, allowing models to be trained across multiple devices or servers.

TensorFlow models can be developed using a variety of programming languages, including Python, C++, and Java. The framework provides high-level APIs that make it easy to create and train models, as well as low-level APIs that offer greater control and flexibility.

In addition to building and training models, TensorFlow provides tools for deploying and serving models. This includes tools for converting models to formats that can be used on different platforms, as well as serving models over the web using TensorFlow Serving.

Overall, TensorFlow is a powerful and flexible machine learning framework that provides developers with the tools they need to build and deploy machine learning models for a wide range of applications. Whether you're just getting started with machine learning or you're a seasoned expert, TensorFlow provides the tools and resources you need to create cutting-edge models and push the boundaries of what's possible.

**OpenCV:**

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. Originally developed by Intel, it is now maintained by the OpenCV community. OpenCV is written in C++, but it has bindings for many other programming languages, including Python, Java, and MATLAB.

The library provides a wide range of tools for image and video analysis, including object detection, face recognition, and optical character recognition (OCR). It can also be used for tasks such as camera calibration, stereo vision, and 3D reconstruction.

One of the key features of OpenCV is its ability to perform real-time image processing and analysis, making it a popular choice for applications such as video surveillance, robotics, and augmented reality. It provides APIs for accessing webcams and video files, and for processing the frames of video streams in real-time.

OpenCV also includes a number of machine learning algorithms, such as support vector machines (SVMs), decision trees, and deep neural networks (DNNs). These algorithms can be used for a variety of tasks, including classification, regression, and clustering.

The library is widely used in both academic and commercial settings. It has been used in a variety of applications, from detecting cancer cells in medical images to tracking cars on a highway. OpenCV is also popular in the robotics community, where it is used for tasks such as object recognition and navigation.

OpenCV is released under a BSD license, which means that it is free to use and distribute. The library has an active community of developers and users, who contribute to the development of the software and provide support to new users.

Overall, OpenCV is a powerful and versatile tool for image and video analysis, with a wide range of applications in both research and industry. Its ability to perform real-time analysis and its support for a variety of programming languages make it a popular choice for computer vision and machine learning projects.

**Streamlit:**

Streamlit is a Python-based web application framework used to build interactive data science and machine learning applications. It enables developers to easily create and share custom web applications without the need for extensive web development experience.

Streamlit is designed to be easy to use and provides a simple and intuitive API for building interactive applications. It supports a wide range of data visualizations, including plots, charts, and graphs, which can be easily customized to suit specific application requirements.

One of the key benefits of Streamlit is its ability to integrate with popular data science libraries like NumPy, Pandas, and Scikit-learn. This means developers can easily import and manipulate data, build machine learning models, and visualize the results within a Streamlit application.

Streamlit also provides a built-in caching system, which enables users to easily reuse expensive computations or data processing steps, reducing the overall time it takes to run an application.

Another benefit of Streamlit is its ability to quickly and easily share web applications with others. Streamlit applications can be shared using a simple URL, which can be accessed by anyone with an internet connection, without the need for complex deployment steps.

Streamlit is also highly extensible, with a large and growing community of developers building and sharing custom components and libraries. This enables developers to easily incorporate new features and functionality into their applications without the need to build everything from scratch.

Finally, Streamlit is an open-source project, which means it is free to use and can be customized to suit specific needs. This also means that the community of developers contributing to the project is constantly growing and improving the platform, ensuring that it remains a popular and useful tool for building interactive data science applications.

**Streamlit webrtc:**

Streamlit is a popular Python library for building interactive web applications. One of the latest additions to the Streamlit library is Streamlit WebRTC, which is a package that allows users to easily stream real-time video and audio from their webcams or other video sources directly into their Streamlit applications.

Streamlit WebRTC is built on top of the WebRTC (Web Real-Time Communications) standard, which is an open-source project supported by Google, Mozilla, and other major technology companies. WebRTC allows for peer-to-peer communication between web browsers and other devices, including cameras and microphones.

With Streamlit WebRTC, users can add real-time video and audio to their Streamlit applications with just a few lines of code. The package includes a Video Transformer class that allows users to easily manipulate the video stream in real-time, including applying filters or performing object detection and tracking.

Streamlit WebRTC also supports advanced features such as simulcast, which allows for adaptive bitrate streaming to improve performance in varying network conditions. Additionally, Streamlit WebRTC can be used in combination with other Streamlit features such as widgets and plots to create rich and interactive web applications.

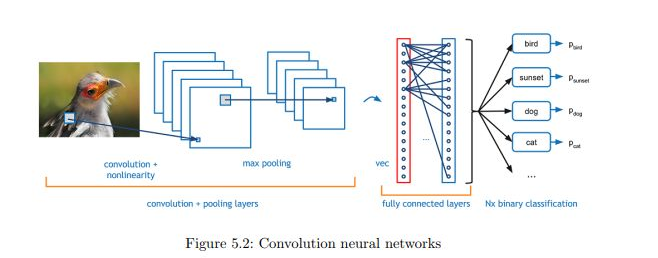
To use Streamlit WebRTC, users must first install the package using pip. They can then import the necessary modules and create a Streamlit WebRTC component by specifying the desired video and audio sources. Once the component is created, it can be added to the Streamlit application using the standard Streamlit API.

Streamlit WebRTC also supports integration with other popular Python libraries such as OpenCV and TensorFlow. This makes it easy for users to add advanced computer vision and machine learning capabilities to their real-time video streams.

Overall, Streamlit WebRTC is a powerful tool for adding real-time video and audio to Streamlit applications. Its easy-to-use API and support for advanced features make it an attractive option for developers looking to build interactive and engaging web applications.

**Convolutional Neural Network:**

Unlike regular Neural Networks, in the layers of CNN, the neurons are arranged in 3 dimensions: width, height, depth. The neurons in a layer will only be connected to a small region of the layer (window size) before it, instead of all of the neurons in a fully-connected manner. Moreover, the final output layer would have dimensions (number of classes), because by the end of the CNN architecture we will reduce the full image into a single vector of class scores

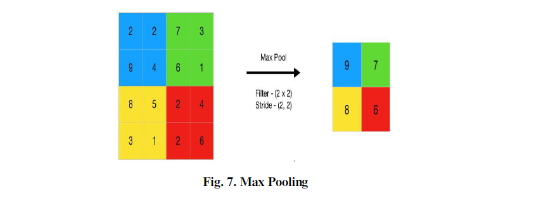


**1.Convolution Layer:** In convolution layer we take a small window size [typically of length 5\*5] that extends to the depth of the input matrix. The layer consists of learnable filters of window size. During every iteration we slid the window by stride size [typically 1], and compute the dot product of filter entries and input values at a given position. As we continue this process well create a 2-Dimensional activation matrix that gives the response of that matrix at every spatial position. That is, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color.

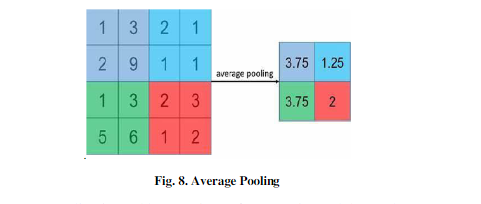
**2. Pooling Layer:** We use pooling layer to decrease the size of activation matrix and ultimately reduce the learnable parameters. There are two types of pooling:

a) **Max Pooling:** In max pooling we take a window size [for example window of size 2\*2], and only take the maximum of 4 values. Well lid this window and continue this process, so well finally get a activation matrix half of its

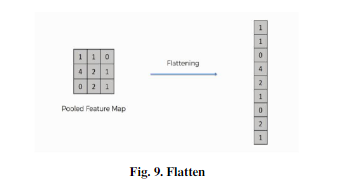
original Size.



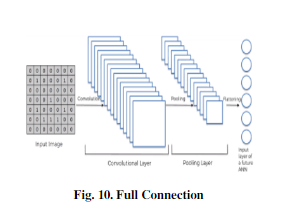
b) **Average Pooling:** In average pooling we take average of all values in a window.



**3.Flatten:** The obtained resultant matrix will be in muti-dimension. Flattening is converting the data into a 1-dimensional array for inputting the layer to the next layer. We flatten the convolution layers to create a single feature vector.



**4. Fully Connected Layer:** In convolution layer neurons are connected only to a local region, while in a fully connected region, well connect the all the inputs to neurons.



1. **Final Output Layer:** After getting values from fully connected layer, well connect them to final layer of neurons [having count equal to total number of classes], that will predict the probability of each image to be in different classes

**CHAPTER 5. SYSTEM DESIGN**

**5.1 Introduction**

**5.2 System Architecture**

**5.3 Data Flow Diagram**

**CHAPTER 6. SYSTEM IMPLEMENTATION**

**6.1 Introduction**

**6.2 Flowchart**

**6.3 Code:**

**6.4 Testing Approach**

**6.5 Testing Case**

**CHAPTER 7. RESULTS**

**CHAPTER 8. CONCLUSION AND FUTURE SCOPE**

**8.1 Conclusion**

**8.1.1 Advantage**

**8.1.2 Limitation**

**8.2 Future Improvement**

**CHAPTER 9. REFERENCE**

**9.1 Reference and Biography**

* 1. https://norvig.com/sudoku.html
  2. https://medium.com/@neshpatel/solving-sudoku-part-ii-9a7019d196a2
  3. https://medium.com/@neshpatel/solving-sudoku-part-i-7c4bb3097aa7
  4. https://aakashjhawar.medium.com/sudoku-solver-using-opencv-and-dl-part-2-bbe0e6ac87c5