Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Informatics

Master's Thesis

An interactive system for teaching school going children.

MD SAHIN ALAM

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

DIPLOMA THESIS ASSIGNMENT

Md Sahin Alam

Informatics

Thesis title

An interactive system for teaching school going children.

Objectives of thesis

The objective of the work is to develop the web-based system for teaching kids alphabet.

The result of the work will be online educational platform focused on teaching school-going children about alphabets, and number systems as well as drawing different shapes. In this day and age, children are always hooked up to digital devices and so the goal is to improve learning in children using a digital platform. Using a digital device, it is easier to capture children's attention and so having a platform like this will significantly improve their learning ability.

Methodology

I developed a system to collect multiple sets of real-time data from people. However, due to the limited number of available participants, I collected raw data from online sources such as Bangla consonants, vowels, numerical as well as English characters and everyday common drawings of shapes. Due to the difference in sources of data, I had to preprocess the data; categorised the data based on the individual character collected and kept those in their separate folders. Different images have different backgrounds so to correct those, I inverted the images, converted the images from RGB profiles to grayscale profiles, cropped the images to remove extraneous information. This resulted in having images of different sizes so I had to resize the image.

At this stage, the data is ready to test and train the system. I will use the data to train and test the system based on 6 categories: Bangla consonants, Bangla vowels, Bangla numbers, English alphabets, English numbers and drawings of shapes. I will be using the Scikit-Learn library to train all machine learning algorithms, test the data and measure which algorithm's accuracy is good for my data finally I will be using the best algorithm to build my web-based systems.

The proposed extent of the thesis
60

Keywords
Convolution Neural Networks, discrimination of alphabet characters, Machine Learning, Scikit-Learn, Flask, TensorFlow, Panda, OpenCV

Recommended information sources

GÉRON, A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems.* Beijing ; Boston ; Farnham ; Sevastopol ; Tokyo: O'Reilly, 2019. ISBN 978-1-4920-3264-9.

Expected date of thesis defence
2022/23 WS – FEM

The Diploma Thesis Supervisor
doc. Ing. Arnošt Veselý, CSc.

Supervising department
Department of Information Engineering

Electronic approval: 9. 3. 2023
Ing. Martin Pelikán, Ph.D.
Head of department

Electronic approval: 13. 3. 2023
doc. Ing. Tomáš Šubrt, Ph.D.
Dean

Prague on 21. 03. 2023

Declaration

I declare that I have worked on my master's thesis titled "**An interactive system for teaching school-going children.**" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the master's thesis, I declare that the thesis does not break any copyrights.

In Prague on 20/03/2023 ____________________________

Acknowledgement

The completion of this thesis would not have been possible without the assistance and support of several individuals who have contributed in various ways. I would like to express my sincere gratitude to **doc. Ing. Arnošt Veselý, CSc.**, for his invaluable guidance and mentorship throughout this project. His insightful suggestions and feedback were instrumental in shaping my research and helped me to attain a deeper understanding of the subject matter.

I am also deeply grateful to my family and friends for their unwavering support and encouragement throughout this journey. Their love, patience, and understanding were a constant source of motivation and inspiration, and I am truly blessed to have them in my life.

I would like to extend my heartfelt thanks to all those who assisted me in collecting data and feedback for my end system, particularly those who volunteered their time and effort to participate in this study. The valuable insights and feedback provided by the students were critical in shaping my research and helped me to achieve a more comprehensive understanding of the topic.

Finally, I would like to express my gratitude to **Ikram Khan** for his invaluable review of my thesis paper, and **Rubaiat Jahan Suva** for her immense support throughout this project. Their feedback and assistance were invaluable, and I am deeply indebted to them both.

An interactive system for teaching school going children.

Abstract

Web-based interactive systems are becoming more common in educational systems for school-going children's. Using AI, machine learning, and deep learning have huge potential in building such systems to improve personalised and adaptive learning experiences for these children. In this thesis paper I discuss and compare machine learning and deep learning algorithms to demonstrate that the Convoluted Neural Network is the most effective at recognising hand-written Bengali and English alphabets, digits and common shapes for improving accuracy of such interactive learning systems. From the results of this comparison I developed an interactive educational platform to improve student engagement, learning outcomes, and enjoyment of the learning process by adding a drawing tool as well as visual and auditory responses. The goal of this platform is to demonstrate the possibility of replacing traditional learning systems with digital interactive ones.

Keywords: Convolution Neural Networks, discrimination of alphabet characters, Machine Learning, Scikit-Learn, Flask, TensorFlow, Panda, OpenCV

Interaktivní systém pro výuku dětí v základní škole.

Abstrakt

Webové interaktivní systémy se stávají častějšími v edukačních systémech pro školou povinné děti. Využití umělé inteligence, strojového učení a hlubokého učení má obrovský potenciál při budování takových systémů, které zlepšují personalizované a adaptivní učební zkušenosti pro tyto děti. V této diplomové práci diskutuji a srovnávám algoritmy strojového a hlubokého učení, abych prokázal, že konvoluční neuronová síť je nejúčinnější při rozpoznávání ručně psaných bengálských a anglických abeced, číslic a běžných tvarů pro zlepšení přesnosti těchto interaktivních vzdělávacích systémů. Na základě výsledků tohoto srovnání jsem vyvinul interaktivní vzdělávací platformu pro zlepšení zapojení studentů, výsledků učení a potěšení z učebního procesu pomocí nástroje pro kreslení, vizuálních a zvukových odezev. Cílem této platformy je ukázat možnost nahrazení tradičních učebních systémů digitálními interaktivními.

Klíčová slova: Konvoluční neuronové sítě, rozpoznání písmen abecedy, strojové učení, Scikit-Learn, Flask, TensorFlow, Panda, OpenCV.

Table of content

1 Introduction

"The only way to do great work is to love what you do" - Steve Jobs.

The year 2020 will be remembered as a year of great turbulence and uncertainty, with the emergence of a pandemic that shook the world to its core. COVID-19 has disrupted our lives in many ways, leading to countless losses and hardships. During this difficult time, I witnessed first-hand the devastating effects of the pandemic on society, including the education sector. Schools were closed, children were forced to stay at home, and the traditional methods of learning were disrupted.

As an avid lover of playing with children, I could not help but think about the potential impact of the pandemic on their education. It was in this moment that the idea of developing an educational platform that could be accessed through technology devices such as smartphones, laptops, and tablets came to mind. As I love AI, machine learning, and deep learning technology, I realized that I could use these tools to make the educational platform more engaging and effective.

Artificial intelligence (AI) has revolutionized the way we interact with technology, and its impact on education is no exception. Machine learning and deep learning, which are subsets of AI, have the potential to transform the way children learn by enabling personalized and adaptive learning experiences.

Machine learning involves the use of algorithms and statistical models to enable computers to learn from data and improve their performance without being explicitly programmed. Deep learning, on the other hand, involves the use of artificial neural networks to simulate human intelligence and solve complex problems.

According to a report by McKinsey (1), AI and machine learning have the potential to transform education by enabling personalized learning experiences, improving student outcomes, and reducing costs. The report suggests that AI can be used to create adaptive learning environments that adjust to the needs and learning styles of individual students.

As I embark on this journey, I am reminded of the wise words of Steve Jobs, who believed that the key to achieving great work is to love what you do. My fascination with AI, machine learning, and deep learning technology has fueled my determination to develop an interactive educational platform that will make a positive impact on the lives of school-going children.

In conclusion, this thesis aims to explore the potential of AI, machine learning, and deep learning technology in facilitating personalized and adaptive learning experiences for school-going children during and beyond the pandemic. The primary motivation behind this research is to develop an interactive educational platform that can be accessed through technology devices, enabling children to learn with fun while making the best use of AI, machine learning, and deep learning technology. The next sections will discuss the literature review, methodology, and results of this research, culminating in recommendations for future work in this field.

2 Objectives and Methodology

2.1 Objectives

The objective of the work is to develop the web-based system for teaching kids alphabet. The result of the work will be online educational platform focused on teaching school-going children about alphabets, and number systems as well as drawing different shapes. In this day and age, children are always hooked up to digital devices and so the goal is to improve learning in children using a digital platform. Using a digital device, it is easier to capture children's attention and so having a platform like this will significantly improve their learning ability.

2.2 Methodology

I developed a system to collect multiple sets of real-time data from people. However, due to the limited number of available participants, I collected raw data from online sources such as Bangla consonants, vowels, numerical as well as English characters and everyday common drawings of shapes. Due to the difference in sources of data, I had to pre-process the data; categorised the data based on the individual character collected and kept those in their separate folders. Different images have different backgrounds so to correct those, I inverted the images, converted the images from RGB profiles to grayscale profiles, cropped the images to remove extraneous information. This resulted in having images of different sizes so I had to resize the image.

At this stage, the data is ready to test and train the system. I will use the data to train and test the system based on 6 categories: Bangla consonants, Bangla vowels, Bangla numbers, English alphabets, English numbers and drawings of shapes. I will be using the Scikit-Learn library to train all machine learning algorithms, test the data and measure which algorithm's accuracy is good for my data finally I will be using the best algorithm to build my web-based systems.

3 Literature Review

3.1 Current Study System

The education system in Bangladesh has undergone significant changes in recent years, with a focus on improving access to education and enhancing the quality of education at all levels. However, the pre-school education system in Bangladesh has not received the same level of attention, despite being an essential component of a child's early education. This study aims to explore the current pre-school education systems in Bangladesh and identify potential areas for improvement.

3.1.1 Current Pre-School Education Systems in Bangladesh

In Bangladesh, pre-school education is primarily provided by private institutions, with limited government involvement. The pre-school education system is divided into two categories: informal and formal. Informal pre-school education is provided by day-care centers and other similar institutions, while formal pre-school education is provided by pre-primary schools.

The informal pre-school education system in Bangladesh is largely unregulated, with limited standards and guidelines for curriculum development and teacher training. Many of these institutions operate in poor conditions, with inadequate facilities and untrained teachers.

The formal pre-school education system in Bangladesh is slightly more regulated, with guidelines and standards developed by the Ministry of Primary and Mass Education. However, the quality of education provided by pre-primary schools varies widely, with some schools offering high-quality education, while others offer poor quality education.

Furthermore, there are significant disparities in access to pre-school education in Bangladesh, with children from low-income families and rural areas being less likely to attend pre-primary schools. According to a survey by the Bangladesh Bureau of

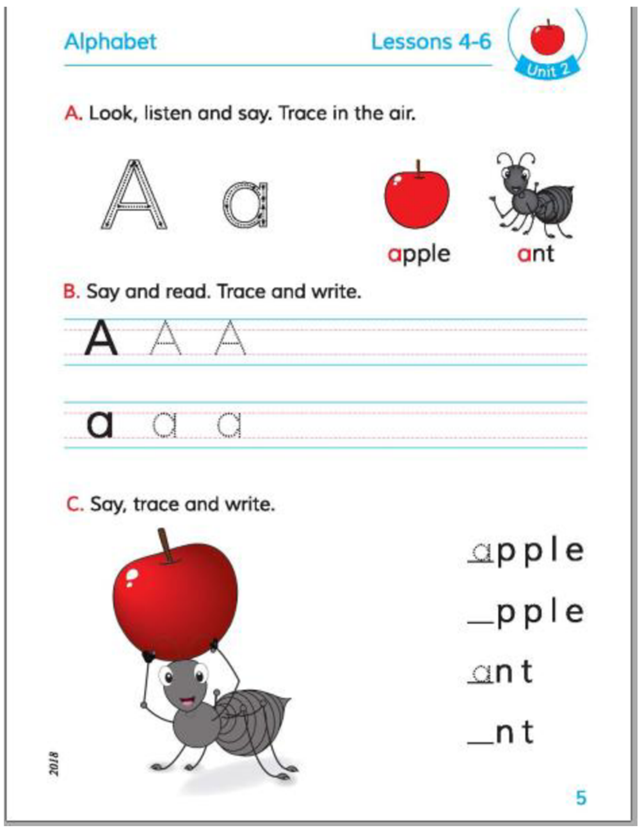
Statistics (2), only 24 percent of children aged 3-5 years in rural areas attend pre-primary schools, compared to 60 percent in urban areas.

English For Today-Class One Book

In this educational book (3), children embark on a linguistic journey that begins with learning greetings, pre-writing skills, and engaging in alphabet songs. They gradually progress to learning the English alphabet, recognizing individual letters within words, and mastering the skill of counting numbers.

To enrich their language development, the book incorporates short rhymes that are both entertaining and educational. The children are also encouraged to identify the names of various objects or animals in provided images, fostering their vocabulary expansion.

Furthermore, students learn the names of different body parts, promoting their understanding of human anatomy. Engaging in drawing exercises and coloring activities not only strengthens their fine motor skills but also encourages creativity and self-expression. Through this comprehensive and interactive book, children develop a solid foundation in language and artistic skills that will serve them well in their future educational endeavors.



Alphabet Lessons 4-6

Unit 2

A. Look, listen and say. Trace in the air.

A a

apple ant

B. Say and read. Trace and write.

A A A

a a a

C. Say, trace and write.

apple

_pple

ant

_nt

2018

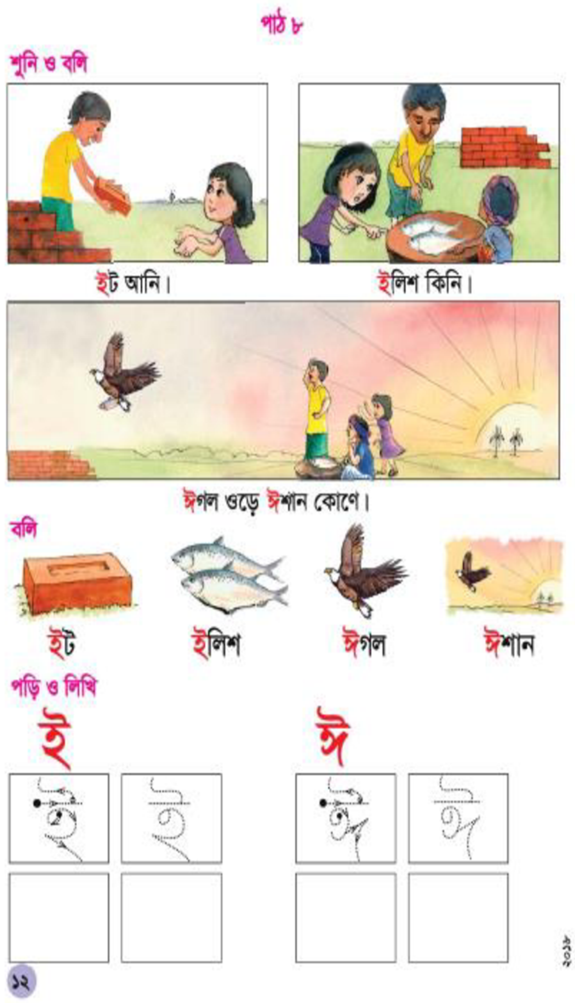
5

Figure 1: English Book-Class One

My Bengali book (আমার বাংলা বই)- Class One Book

In this educational book (3), children are introduced to the process of self-expression by learning how to present themselves. This serves as the starting point for their linguistic journey. They begin by acquiring the basics of the Bengali alphabet, starting with individual vowels and gradually moving on to consonants. This systematic approach ensures that they develop a strong foundation in the language.

As they progress, students are introduced to various diacritical marks, enabling them to read and write more complex words and sentences. In addition to mastering the alphabet, the book nurtures a love for literature by exposing children to poetry and stories. Through this comprehensive and engaging approach, young learners are equipped with essential language skills that will support their continued education and personal growth.



পাঠ ৮

শুনি ও বলি

ইট আনি।

ইলিশ কিনি।

ঈগল ওড়ে ঈশান কোণে।

বলি

ইট ইলিশ ঈগল ঈশান

পড়ি ও লিখি

ই ঈ

২০১৮

১২

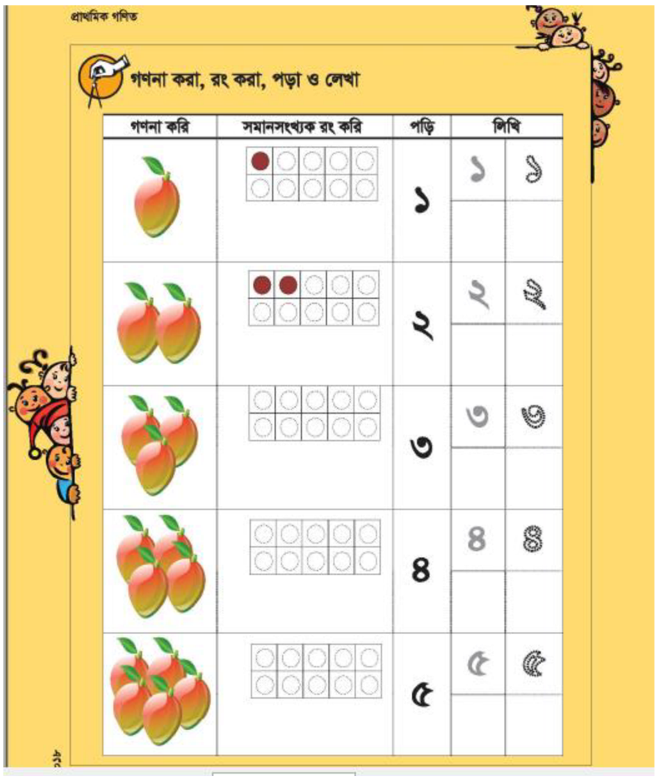
Figure 2: Bangla Book-Class One

Elementary Mathematics (প্রাথমিক গণিত)-Class One Book

This educational book (3) introduces children to the fundamentals of mathematics, beginning with counting numbers from 0 to 50 and teaching them to compare these numbers effectively. Students learn basic arithmetic operations, such as addition and subtraction, focusing on calculations involving numbers between 0 and 10.

The book also covers essential geometric shapes, familiarizing children with the names and properties of squares, triangles, circles, and lines. Engaging in counting exercises using images, such as identifying the number of apples in a picture, helps students develop their numeracy skills and enhances their observational abilities.

Additionally, the book includes matching exercises where children pair images with corresponding numbers, further reinforcing their understanding of numerical concepts. Problem-solving activities, such as filling boxes with appropriate numbers, encourage students to apply their newfound skills in practical situations. Through this comprehensive approach, children build a strong foundation in mathematics that will support their academic growth.



প্রাথমিক গণিত

গণনা করা, রং করা, পড়া ও লেখা

গণনা করি	সমানসংখ্যক রং করি	পড়ি	লিখি	
	●○○○○ ○○○○○	১	১	১
	●●○○○ ○○○○○	২	২	২
	○○○○○ ○○○○○	৩	৩	৩
	○○○○○ ○○○○○	৪	৪	৪
	○○○○○ ○○○○○	৫	৫	৫

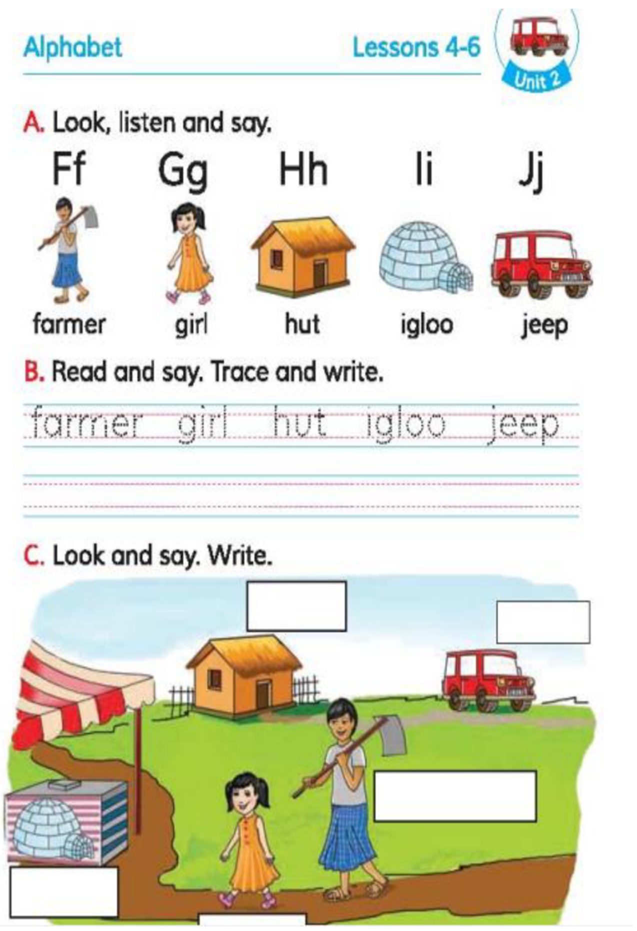
Figure 3: Math Book-Class One

English For Today-Class Two Book

Initially, students begin by memorizing the entire alphabet and numbers (3). They practice counting and writing numerals while actively listening to their teacher's instructions. Students are then encouraged to describe various images, articulating their thoughts and writing down their observations. They also focus on identifying the final sound of each word depicted in the images and practice writing capital letters for each corresponding word.

Furthermore, students learn to express numbers from one to ten in written form, enhancing their number literacy. Engaging in activities such as connecting dots to complete images and adding color to their creations helps develop their fine motor skills and creativity. They also participate in reciting and enacting rhymes, promoting their linguistic and cognitive abilities.

Lastly, students are introduced to different shapes through visual and auditory methods, allowing them to recognize and name each shape with confidence. By reading and coloring within the lines of a butterfly illustration, students refine their precision and focus, fostering essential developmental skills.



Alphabet Lessons 4-6 Unit 2

A. Look, listen and say.

Ff Gg Hh Ii Jj

farmer girl hut igloo jeep

B. Read and say. Trace and write.

farmer girl hut igloo jeep

C. Look and say. Write.

Figure 4: English Book-Class Two

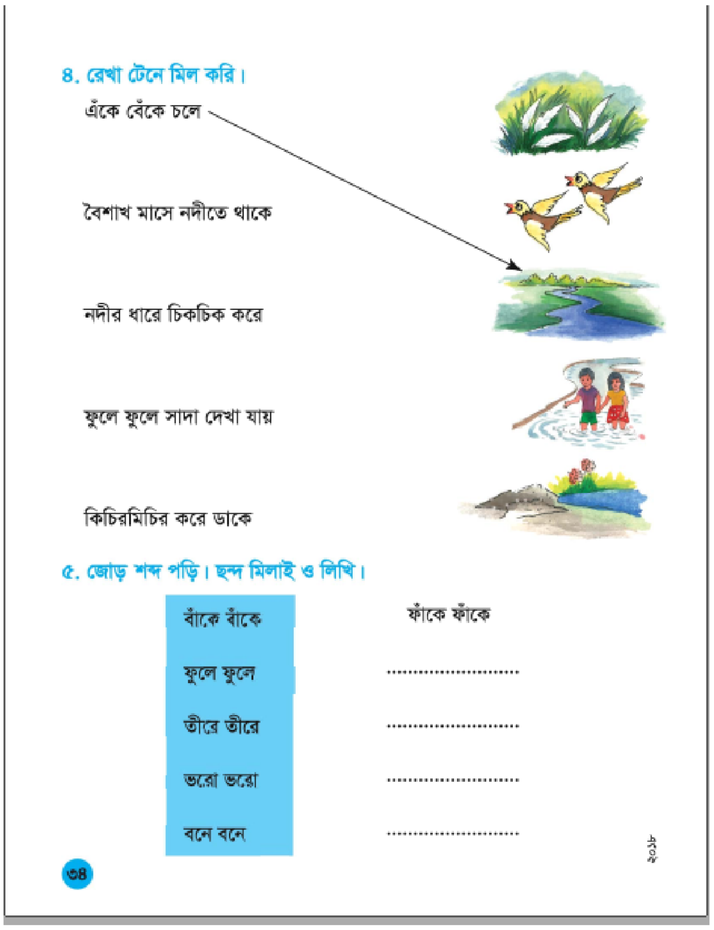
My Bengali book (আমার বাংলা বই)- Class Two Book

In this book (3), children begin their learning journey by introducing themselves through a self-introduction form. They are then guided to answer questions based on provided images, encouraging critical thinking and comprehension. Various exercises, such as filling in gaps by analyzing pictures, help students develop their problem-solving abilities.

To enhance their vocabulary, children are tasked with constructing words from random letters and formulating sentences based on visual cues. They also engage in storytelling exercises using a series of pictures, which fosters their creativity and narrative skills. Counting and identifying animals in images, as well as writing the numbers in both digits and words, further strengthens their numeracy abilities.

Moreover, students are exposed to poetry, learning to appreciate, read, and write poems. Interactive conversations with their peers promote social skills and fluency in the language. As their abilities progress, children are introduced to joint letters, expanding their understanding of the writing system.

Matching exercises, where students draw lines connecting sentences to corresponding images, help reinforce their comprehension and retention. Lastly, they learn the names of all the months in the Bengali calendar, enriching their cultural knowledge and awareness.



৪. রেখা টেনে মিল করি।

এঁকে বেঁকে চলে

বৈশাখ মাসে নদীতে থাকে

নদীর ধারে চিকচিক করে

ফুলে ফুলে সাদা দেখা যায়

কিচিরমিচির করে ডাকে

৫. জোড় শব্দ পড়ি। ছন্দ মিলাই ও লিখি।

বাঁকে বাঁকে	ফাঁকে ফাঁকে
ফুলে ফুলে	....
তীরে তীরে	....
ভরো ভরো	....
বনে বনে	....

৩৪

২০১৮

Figure 5: Bangla Book-Class Two

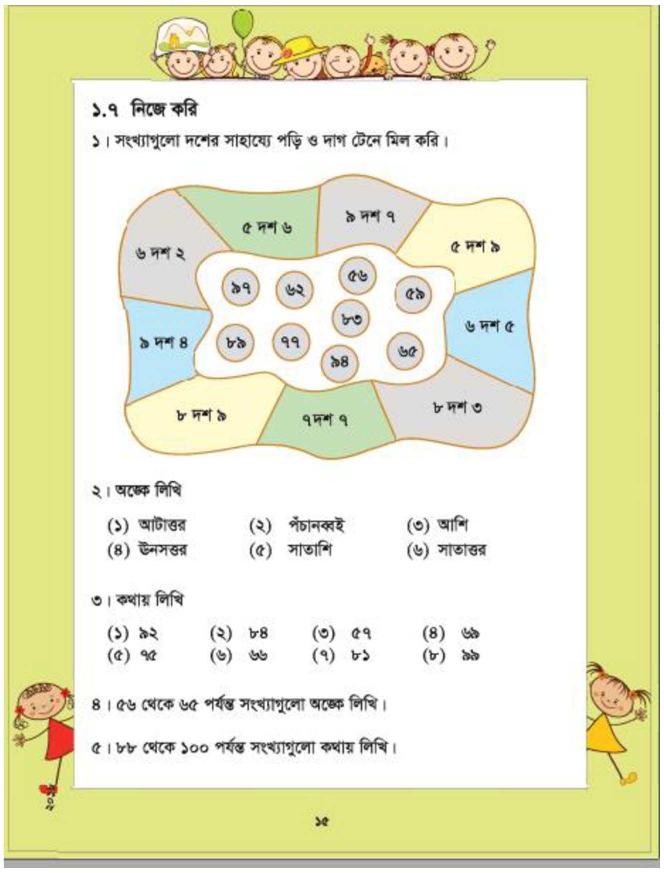
Elementary Mathematics (প্রাথমিক গণিত)-Class Two Book

Students begin by counting the quantity of objects depicted in images, practicing their numeracy skills by writing the results both in numerals and words (3). They are then introduced to the concept of comparing numbers, learning to identify which of the two is larger. To reinforce this understanding, they are asked to circle the larger number among a given set.

By arranging random numbers in ascending order, students develop their organizational and analytical skills. They continue to expand their number literacy by reading and writing numbers from fifty-one to one hundred. Additionally, they are taught the concepts of even and odd numbers, enhancing their foundational mathematical knowledge.

Students participate in exercises that involve circling numbers with odd summations and practice writing numbers from eighty to one hundred in words. They also learn subtraction, focusing on identifying even results. These exercises are followed by analytical math tasks that challenge their problem-solving abilities.

Lastly, students are introduced to the fundamental operations of multiplication and division, specifically focusing on calculations involving numbers between 0 and 10. Through these activities, they build a strong foundation in mathematics that will support their future academic endeavors.



১.৭ নিজে করি

১। সংখ্যাগুলো দশের সাহায্যে পড়ি ও দাগ টেনে মিল করি।

৫ দশ ৬
৯ দশ ৭
৬ দশ ২
৫ দশ ৯
৯৭
৬২
৫৬
৫৯
৮৩
৯ দশ ৪
৮৯
৭৭
৯৪
৬৫
৬ দশ ৫
৮ দশ ৯
৭দশ ৭
৮ দশ ৩

২। অঙ্কে লিখি

(১) আটাত্তর	(২) পঁচানব্বই	(৩) আশি
(৪) উনসত্তর	(৫) সাতাশি	(৬) সাতাত্তর

৩। কথায় লিখি

(১) ৯২	(২) ৮৪	(৩) ৫৭	(৪) ৬৯
(৫) ৭৫	(৬) ৬৬	(৭) ৮১	(৮) ৯৯

৪। ৫৬ থেকে ৬৫ পর্যন্ত সংখ্যাগুলো অঙ্কে লিখি।

৫। ৮৮ থেকে ১০০ পর্যন্ত সংখ্যাগুলো কথায় লিখি।

১৫

Figure 6: Math Book-Class Two

3.1.2 Exploring how children learn from their everyday life

Understanding how children learn from their everyday life is crucial to developing effective learning strategies. This study aimed to explore the various ways in which children in Bangladesh learn and gain knowledge. The study involved observing and interviewing children and their parents to gain insights into their learning experiences. The findings of the study can help develop effective learning strategies for school-going children in Bangladesh.

Child Learning from Everyday Life:

Through the study, it was found that children in Bangladesh learn in various ways. At school, in the classroom, teachers teach various courses, which helps children learn how to read and write. The formal education system in Bangladesh plays a crucial role in providing children with the necessary foundational knowledge and skills.

Children also learn through play. While playing with toys or other children, different pictures are shown to them for learning, such as animals, shapes, and colors. This helps them learn the names of things and improves their basic knowledge.
In addition to traditional forms of learning, technology has also become a useful tool for children's education. Various educational programs are shown on television, such as Sisimpur, which provide children with a fun and interactive way of learning. Similarly, cartoons like Mina Cartoon offer basic knowledge, and YouTube channels like TuTiTuTV provide knowledge about alphabets, letters, and numbers.

Learning Activities for School-Going Children in Bangladesh:
Based on the findings of this study, there are several learning activities that can be implemented for school-going children in Bangladesh. The formal education system should continue to provide foundational knowledge and skills to children. Additionally, there should be more emphasis on play-based learning, which can make learning more fun and engaging for children.
Moreover, there is a need to develop and promote educational programs and cartoons that offer more advanced and diverse knowledge for children. Educational channels on YouTube can also be used to provide additional learning opportunities for children.

3.2 Artificial Intelligence

Artificial Intelligence (AI) refers to the development of machines that can perform tasks that typically require human intelligence, such as reasoning, learning, and problem-solving. The field of AI has grown significantly in recent years, with applications in a wide range of industries, including healthcare, finance, and transportation. In this article, we will explore the term AI, its scope, and the current work being done in the field.

The term AI was first coined by John McCarthy in 1956, who defined it as "the science and engineering of making intelligent machines". Since then, the definition of AI has evolved to encompass a range of techniques and applications, including machine learning, natural language processing, computer vision, and robotics.

The scope of AI is vast and continues to expand as new applications are developed. Machine learning is a subfield of AI that involves training algorithms on large datasets in order to make predictions or decisions. This has led to the development of applications such as voice assistants, recommendation systems, and fraud detection systems. Natural language processing (NLP) involves teaching machines to understand and interpret human language. This has led to the development of applications such as chatbots, sentiment analysis, and language translation systems. Computer vision involves teaching machines to interpret visual data, such as images and videos. This has led to the development of applications such as facial recognition systems, object detection systems, and autonomous vehicles. Robotics involves the design and development of robots that can perform tasks autonomously. This has led to the development of applications such as industrial robots, surgical robots, and service robots.

The current work being done in AI is focused on developing new techniques and applications that can improve efficiency, accuracy, and performance. One area of focus is deep learning, which involves the use of neural networks with multiple layers to learn complex representations of data. This has led to breakthroughs in applications such as image recognition and natural language processing. Another area of focus is reinforcement learning, which involves training agents to make decisions based on feedback from their environment. This has led to breakthroughs in applications such as game playing and robotics. Additionally, researchers are exploring new techniques for explainable AI, which aim to provide insights into how AI systems arrive at their decisions.

Despite the rapid progress being made in AI, there are still significant challenges that need to be addressed. One of the biggest challenges is the development of ethical and responsible AI systems. This includes addressing issues such as bias, privacy, and transparency. Another challenge is the development of AI systems that can adapt to changing environments and learn from experience. This requires the development of new techniques for lifelong learning and transfer learning.

3.2.1 Four Categories Of Artificial Intelligence Definitions

The four categories of Artificial Intelligence (AI) that are based on their level of intelligence and complexity, AI can also be categorized based on their approach to thinking and acting. These categories include thinking humanly, thinking rationally, acting humanly, and acting rationally.

Thinking Humanly:

Thinking humanly (4) AI aims to replicate human thought processes and the way humans learn, perceive, and reason. This approach involves studying human cognition, psychology, and neuroscience to develop AI systems that can think and learn in a similar way to humans. This approach aims to develop AI that can solve problems in the same way that humans do.

Thinking Rationally:

Thinking rationally (4) AI aims to replicate human reasoning and decision-making processes using logical and mathematical rules. This approach involves developing AI systems that can reason deductively, based on a set of pre-defined rules and assumptions. This approach aims to develop AI that can make decisions based on logical reasoning, rather than simply relying on past experiences or intuition.

Acting Humanly:

Acting humanly (4) AI aims to replicate human behavior and actions, rather than focusing on thinking and reasoning. This approach involves developing AI systems that can interact with humans in a way that is similar to human-human interaction, such as through natural language processing or gesture recognition. This approach aims to develop AI that can act and communicate like humans.

Acting Rationally:

Acting rationally (4) AI aims to develop systems that can act rationally to achieve specific goals, regardless of whether their actions mimic human behaviour or not. This approach involves developing AI systems that can identify the best course of action to achieve a specific goal, based on available information and knowledge. This approach aims to develop AI that can act rationally, even if their actions do not necessarily resemble human behaviour.

3.3 Machine Learning

Machine learning is a subfield of artificial intelligence that focuses on developing algorithms and models that enable computers to learn from data and make predictions or decisions. The applications of machine learning are diverse, including image and speech recognition, natural language processing, fraud detection, and recommendation systems.

In this review, I will discuss some of the key parts of machine learning, including data pre-processing, feature selection and engineering, model selection and training, model evaluation, and hyperparameter tuning.

Data Pre-processing

Data pre-processing is a crucial step in any machine learning project, as the quality of the data used for training and testing can significantly impact the accuracy and performance of the resulting model. The pre-processing steps may involve cleaning, transforming, or normalizing the data to make it suitable for analysis. In some cases, missing values or outliers may need to be addressed.

In a recent study by Wang (5), the authors proposed a novel data pre-processing method called Multi-Task Learning Based Data Pre-processing (MTL-DP) for predicting the performance of computer systems. The method involves using a multi-task learning approach to simultaneously predict multiple performance metrics, which allows for better feature representation and improved performance compared to traditional pre-processing methods.

Feature Selection and Engineering

Feature selection and engineering involves identifying the most relevant features in the data and creating new features that may improve the accuracy and performance of the model. This can be a challenging task, as the number of potential features can be very large, and selecting the wrong features can lead to overfitting or underfitting.

In a study by Nguyen (6), the authors proposed a feature selection method based on mutual information and fuzzy clustering for predicting the sentiment of social media posts. The method involves clustering the data based on the mutual information between the features and the target variable, and selecting the most representative features from each cluster. The results showed that the proposed method outperformed several other feature selection methods in terms of accuracy and computational efficiency.

Model Selection and Training

Model selection and training involves choosing the appropriate machine learning algorithm for the problem at hand, considering factors such as data type, size, and complexity, and training the model on the available data. The choice of algorithm can have a significant impact on the performance of the model, and different algorithms may be better suited for different types of data and tasks.

In a recent study by Shahin (7), the authors compared the performance of several machine learning algorithms for predicting the onset of type 2 diabetes. The algorithms included logistic regression, decision trees, random forests, and neural networks. The results showed that the neural network algorithm achieved the highest accuracy, but the decision tree algorithm was the most interpretable and could provide insights into the underlying factors contributing to the onset of diabetes.

Model Evaluation

Model evaluation involves assessing the accuracy and performance of the trained model, using metrics such as precision, recall, and F1 score. It is important to evaluate the model on both the training and testing data to ensure that it is not overfitting or underfitting the data.

In a study by Xie (8), the authors proposed a novel evaluation metric called Normalized Pairwise Margin (NPM) for evaluating binary classification models. The metric is based on the margin between the predicted probabilities of the positive and negative classes, and is normalized to account for class imbalance and model complexity. The results showed that NPM outperformed several other evaluation metrics in terms of discrimination and calibration.

Hyperparameter Tuning

Hyperparameter tuning involves adjusting the settings of the machine learning algorithm to optimize its performance on the given dataset. This is typically done using a validation set or cross-validation to evaluate the performance of different hyperparameter settings.

In a recent study by Zhang (9), the authors proposed a Bayesian optimization approach for hyperparameter tuning in deep neural networks. The method involves constructing a probabilistic model of the objective function and using it to guide the search for optimal hyperparameters. The results showed that the proposed method outperformed several other hyperparameter tuning methods in terms of efficiency and effectiveness.

3.3.1 Categories of Machine Learning

Machine learning can be categorized into three main categories: supervised learning, unsupervised learning, and reinforcement learning. Deep learning is a subfield of machine learning that focuses on building neural networks with multiple layers to learn complex representations of data. Deep learning has been particularly successful in computer vision, natural language processing, and speech recognition.

Supervised Learning:

Supervised learning involves training a machine learning model on labeled data, where each data point is associated with a label or target variable. The goal is to learn a mapping between the input features and the output labels, so that the model can make accurate predictions on new, unseen data. Supervised learning can be further categorized into two types of tasks: regression and classification.

Regression tasks involve predicting a continuous output variable, such as the price of a house or the age of a person. Linear regression is a common algorithm used for regression tasks.

Classification tasks involve predicting a categorical output variable, such as the type of animal in an image or the sentiment of a tweet. Common algorithms used for classification tasks include logistic regression, decision trees, and support vector machines.

Deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have been used to achieve state-of-the-art performance on supervised learning tasks, particularly in computer vision and natural language processing, LeCun (10).

Unsupervised Learning:

Unsupervised learning involves training a machine learning model on unlabeled data, where there is no target variable or labels provided. The goal is to discover patterns or structure in the data, such as clusters or groups of similar data points. Unsupervised learning can be further categorized into two types of tasks: clustering and dimensionality reduction.

Clustering tasks involve grouping similar data points together, based on their features or characteristics. K-means clustering and hierarchical clustering are common algorithms used for clustering tasks, Alpaydin (11).

Dimensionality reduction tasks involve reducing the number of features or variables in the data, while preserving as much of the original information as possible. Principal component analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE) are common algorithms used for dimensionality reduction tasks, Murphy (12).

Deep learning models, such as autoencoders and generative adversarial networks (GANs), have been used to learn representations of data in an unsupervised manner (13).

Reinforcement Learning:

Reinforcement learning involves training a machine learning model to make decisions based on feedback from the environment, such as rewards or penalties. The goal is to learn a policy or set of actions that maximize the cumulative reward over time. Reinforcement learning can be applied to a wide range of tasks, including game playing, robotics, and autonomous driving.

Deep reinforcement learning has been particularly successful in game playing, where models such as AlphaGo and AlphaZero have achieved superhuman performance in games such as Go and chess (14).

3.3.2 Machine Learning Algorithms

There are many machine learning algorithms, each with its own strengths and weaknesses. Here are some common machine learning algorithms that I will use to test my dataset and it is outcomes.

Linear Regression:

Linear regression (15) is a commonly used algorithm in machine learning for regression tasks, where the goal is to predict a continuous output variable. The algorithm assumes that there is a linear relationship between the input features and the output variable, and seeks to learn a linear function that best fits the training data.

The basic idea behind linear regression is to find the line that best fits the data points, based on the principle of minimizing the sum of squared errors between the predicted and actual values. The line is defined by the equation:

$$y = b_0 + b_1x$$

where y is the predicted output variable, x is the input feature, b0 is the y-intercept or bias term, and b1 is the slope or weight of the input feature.

In practice, there may be multiple input features, and the linear function can be expressed as:

$$y = b_0 + b_1x_1 + b_2x_2 + \ldots + b_nx_n$$

where xi represents the ith input feature, and bn represents the weight or coefficient associated with that feature.

The goal of linear regression is to estimate the values of the coefficients that minimize the sum of squared errors between the predicted and actual values. This is typically done using an optimization algorithm such as gradient descent, which iteratively adjusts the values of the coefficients to minimize the cost function.

Once the coefficients have been estimated, the model can be used to make predictions on new, unseen data. The prediction for a given input feature is simply the value of the linear function for that input:

$$y = b_0 + b_1x_1 + b_2x_2 + \ldots + b_nx_n$$

Linear regression is a simple and interpretable algorithm that can be applied to a wide range of regression tasks. However, it may not be appropriate for data that does not exhibit a linear relationship between the input features and output variable. In such cases, other regression algorithms, such as decision trees or support vector machines, may be more appropriate.

Logistic Regression:

Logistic regression (15) is a widely used algorithm in machine learning for binary classification tasks, where the goal is to predict a binary output variable (e.g. 0 or 1, true or false, yes or no) based on one or more input features. The algorithm estimates the probability of the positive class (i.e. 1) given the input features, and uses a threshold to make a binary prediction.

The logistic regression model is based on the logistic function, which is defined as:

$$p(x) = \frac{1}{1 + e^{-z}}$$

where p(x) is the probability of the positive class, x is the input feature vector, and z is the linear function of the input features:

$$z = b_0 + b_1x_1 + b_2x_2 + \ldots + b_nx_n$$

where bi represents the weight or coefficient associated with the ith input feature.

The logistic function has an S-shaped curve, which maps any input value to a probability between 0 and 1. The logistic regression model estimates the values of the coefficients that best fit the training data, by minimizing the negative log-likelihood of the observed labels given the predicted probabilities.

Once the coefficients have been estimated, the model can be used to make predictions on new, unseen data. The predicted probability of the positive class is calculated using the logistic function:

$$p(x) = \frac{1}{1 + e^{-z}}$$

and a threshold is used to make a binary prediction. Common thresholds include 0.5, which corresponds to the point where the predicted probability is equal to the threshold, and 0.7 or 0.8, which can be used to increase the precision of the predictions at the expense of recall.

Logistic regression is a simple and interpretable algorithm that can be applied to a wide range of binary classification tasks. However, it may not be appropriate for data that does not exhibit a linear relationship between the input features and output variable, or for multi-class classification tasks. In such cases, other classification algorithms, such as decision trees, random forests, or support vector machines, may be more appropriate.

Decision Trees:
Decision trees (15) are a widely used algorithm in machine learning for both regression and classification tasks. The basic idea behind decision trees is to recursively split the input space into regions based on the input features, in order to make predictions on new, unseen data.

The decision tree consists of nodes that represent the input features, edges that represent the possible values of the input features, and leaves that represent the predicted output variable. The tree is constructed by recursively splitting the input space into regions, based on the input features that best discriminate the training data.

The splitting criterion depends on the task and can be based on various metrics such as information gain, entropy, or Gini impurity. The goal is to find the splits that result in the greatest reduction in the impurity or uncertainty of the output variable, while minimizing the complexity of the tree.

The decision tree can be represented by a binary tree structure, where each internal node represents a decision on a specific feature, and each leaf node represents a predicted output value. The decision rules for each node can be represented by a

Boolean function of the input features, which determines which branch to follow based on the value of the input feature.

More specifically, the decision tree algorithm can be summarized as follows:

1. Define a root node for the tree.
2. Select the feature that best discriminates the training data, based on the splitting criterion.
3. Create a new internal node for the selected feature.
4. Create a branch for each possible value of the selected feature.
5. Recursively apply steps 2-4 to each branch, using the remaining features and training data.
6. Stop splitting when a stopping criterion is met, such as a maximum depth or a minimum number of samples per leaf.
7. Assign a predicted output value to each leaf node, based on the majority class or the mean value of the training data in that leaf.

Once the tree has been constructed, it can be used to make predictions on new, unseen data. The prediction is made by traversing the tree from the root to the leaf node that corresponds to the input features, based on the decision rules represented by the edges.

Support Vector Machines (SVM):

Support Vector Machines (SVM) (15) are a widely used algorithm in machine learning for classification and regression tasks. The basic idea behind SVM is to find the hyperplane that maximally separates the data into classes or predicts the output variable with the smallest error.

In binary classification tasks, the hyperplane can be represented by the equation:

$$w^T x + b = 0$$

where w is the weight or coefficient vector, x is the input feature vector, and b is the bias term. The hyperplane divides the input space into two regions, one for each class.

The distance between the hyperplane and the closest data points from each class is called the margin.

The goal of SVM is to find the hyperplane that maximizes the margin, subject to the constraint that all data points are correctly classified. This is typically done using an optimization algorithm such as quadratic programming.

In cases where the data is not linearly separable, SVM can be extended to use a kernel function that maps the input features to a higher-dimensional space, where the data may become linearly separable. Common kernel functions include the linear kernel, polynomial kernel, and radial basis function (RBF) kernel.

Once the hyperplane has been found, it can be used to make predictions on new, unseen data. The predicted class for a given input feature is determined by the sign of the hyperplane equation:

$$f(x) = sign(w^T x + b)$$

where $sign()$ is the sign function that returns $+1$ or -1 depending on the sign of its argument.

SVM is a powerful algorithm that can be applied to a wide range of classification and regression tasks, and is particularly useful in cases where the data is not linearly separable. However, SVM can be sensitive to the choice of kernel function and hyperparameters, and may suffer from scalability issues when dealing with large datasets.

Random Forest:

Random forest (15) is a widely used ensemble learning algorithm in machine learning for classification and regression tasks. The basic idea behind random forest is to build multiple decision trees on randomly sampled subsets of the input data and features, and then combine their predictions to improve the accuracy and robustness of the model.

Each decision tree in the random forest is constructed using a random subset of the input data, sampled with replacement (i.e. with bootstrapping). Additionally, at each node of the tree, only a random subset of the input features is considered for splitting, in order to introduce diversity and reduce overfitting.

The final prediction of the random forest is made by aggregating the predictions of all the decision trees, either by taking the majority vote for classification tasks, or the average or median for regression tasks.

The random forest algorithm can be summarized as follows:

- Select the number of decision trees to build (n_estimators) and the size of the random feature subset (max_features).
- For each decision tree, randomly sample a subset of the input data with replacement.
- For each node of the tree, randomly select a subset of the input features of size max_features.
- Split the data at each node based on the selected feature that best discriminates the data, using a splitting criterion such as information gain, entropy, or Gini impurity.
- Grow the tree until a stopping criterion is met, such as a maximum depth or a minimum number of samples per leaf.
- Repeat steps 2-5 for all decision trees.
- Aggregate the predictions of all decision trees to obtain the final prediction.

Random forest is a powerful algorithm that can be applied to a wide range of classification and regression tasks, and is particularly useful in cases where the data is noisy, high-dimensional, or has complex nonlinear relationships. However, random forest may require tuning of hyperparameters such as the number of decision trees, the size of the random feature subset, and the stopping criteria, in order to achieve optimal performance.

Convolutional Neural Networks (CNNs):

Convolutional Neural Networks (CNNs) (13) are a powerful class of neural networks commonly used in image and video recognition tasks. CNNs leverage a mathematical operation called convolution, which allows the network to learn and extract features from images.

The basic idea behind CNNs is to use a series of convolutional layers to extract progressively more complex features from the input image, followed by one or more fully connected layers to classify the image based on the extracted features.

The convolutional layer works by sliding a small window called a filter or kernel over the input image and computing a dot product between the filter weights and the pixel values in the window. This produces a feature map that highlights certain patterns or features in the image, such as edges, corners, or textures.

The output of the convolutional layer can be further processed using non-linear activation functions such as ReLU (Rectified Linear Unit) to introduce non-linearity and improve the model's ability to learn complex features.

CNNs also commonly use pooling layers, which downsample the feature maps by taking the maximum or average value of a small region of the map. This reduces the spatial resolution of the feature maps, while preserving the most salient features.

The final layers of the CNN typically consist of one or more fully connected layers, which use the extracted features to make a prediction about the class of the input image.

CNNs can be trained using backpropagation and gradient descent, with the objective of minimizing a loss function such as cross-entropy between the predicted and actual class labels.

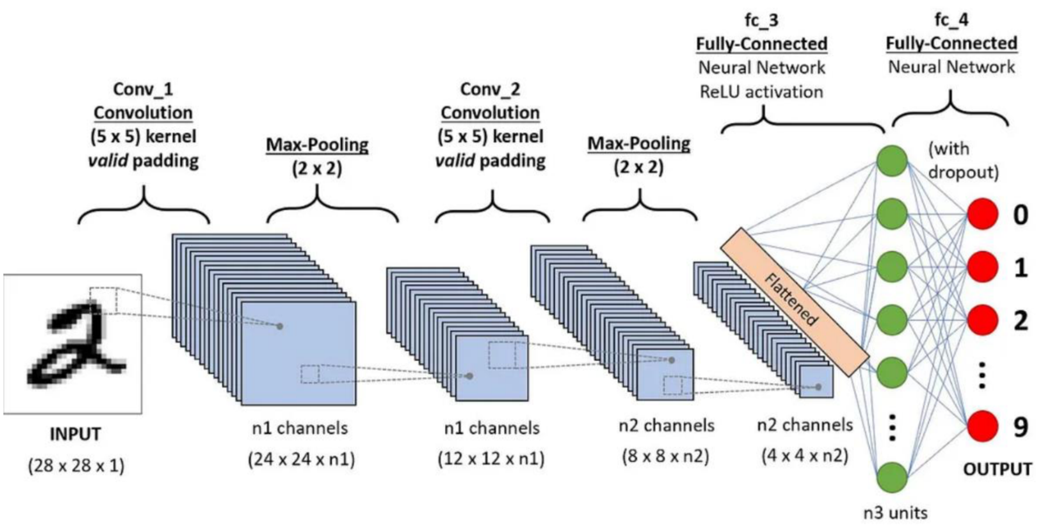


Figure 7: CNN Architecture (16)

The equations for CNNs are complex and depend on the specific architecture and parameters of the network. However, the basic idea behind the convolutional layer can be represented by the equation:

$$h(i,j,k) = ReLU(\sum\left(x(p,q,r) * w(i-p+1, j-q+1, r, k) + b(k)\right))$$

where $h(i,j,k)$ is the output feature map at position (i,j) and channel k, $x(p,q,r)$ is the input image pixel at position (p,q) and channel r, $w(i,j,r,k)$ is the weight or filter at position (i,j) and channel r and k, $b(k)$ is the bias term for channel k, and $ReLU()$ is the rectified linear unit activation function.

3.4 Other Technologies for Building Applications

Building user-friendly applications is a complex task that demands a comprehensive understanding of various technologies and tools. To create an intuitive user experience, it is crucial to have a deep understanding of the principles of user interface design. Additionally, advanced tools such as Flask, OpenCV, Scikit-Learn, Pandas, NumPy, and TensorFlow can be utilized to develop powerful applications with cutting-edge features.

Furthermore, the development of functional web pages and user interfaces requires proficiency in front-end development tools such as HTML, CSS, Bootstrap, and JavaScript. Keeping up-to-date with the latest trends and best practices in user interface design and front-end development is crucial for delivering a user-friendly experience. This necessitates continuous learning and staying informed about emerging technologies and tools in the field.

Flask Framework:

Flask is a lightweight web framework (17) for Python that is widely used for building web applications and APIs. Flask provides a simple and flexible architecture for handling HTTP requests and responses, and supports a wide range of extensions and plugins for database access, authentication, and other functionality.

OpenCV:

OpenCV (18) is a popular computer vision library that provides a wide range of tools and functions for image and video processing, including feature detection, object recognition, and tracking. OpenCV can be used with a wide range of programming languages, including Python, and is widely used in applications such as robotics, autonomous vehicles, and medical imaging.

Pandas:

Pandas (19) is a data manipulation library for Python that provides powerful tools for working with structured data, including data frames and time series. Pandas can be used to load, clean, and transform data from a wide range of sources, and provides a wide range of functions for statistical analysis and visualization.

NumPy:

NumPy (20) is a numerical computing library for Python that provides support for arrays, matrices, and other numerical data structures. NumPy provides a wide range of functions for mathematical and scientific computing, including linear algebra, Fourier analysis, and random number generation.

TensorFlow:

TensorFlow (21) is an open-source machine learning library developed by Google that provides a wide range of tools and functions for building and training machine learning models. TensorFlow supports a wide range of neural network architectures, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and deep belief networks (DBNs).

These libraries can be used together to build powerful and flexible machine learning applications. For example, OpenCV can be used to preprocess images and extract features, Pandas and NumPy can be used to load and manipulate data, and TensorFlow can be used to build and train machine learning models. Flask can be used to create a web application that exposes the machine learning model as an API, allowing users to submit data and receive predictions in real time.

Scikit-Learn:

Scikit-Learn (also known as sklearn) (22)is a popular machine learning library for Python that provides a wide range of tools and functions for building and evaluating machine learning models. Scikit-Learn includes a wide range of algorithms for classification, regression, clustering, and dimensionality reduction, as well as tools for data preprocessing, feature selection, and model evaluation.

One of the key features of Scikit-Learn is its consistency and ease of use, which makes it easy for developers and data scientists to build and evaluate machine learning models. Scikit-Learn provides a consistent interface for all of its algorithms, with a similar set of functions and parameters for each model. This makes it easy to switch between different models and compare their performance.

Scikit-Learn includes a wide range of algorithms for classification, including logistic regression, support vector machines, decision trees, random forests, and neural networks. Scikit-Learn also includes algorithms for regression, clustering, and dimensionality reduction, as well as tools for model selection and hyperparameter tuning.

Scikit-Learn is built on top of other popular Python libraries, including NumPy, SciPy, and Matplotlib, and provides integration with other tools such as Pandas and Jupyter notebooks. Scikit-Learn also includes a wide range of tools for data preprocessing and feature engineering, such as scaling, normalization, and imputation.

4 Practical Part

4.1 Data Collection

After completing my background study, I realized the need for collecting a large amount of data. To achieve this, I developed a system that could collect data from multiple individuals. The system was designed using PHP, HTML, CSS, and JavaScript, and it consisted of an online HTML scratch form hosted on a server. I distributed the form to around 50 people, and 30 of them responded by providing us with data.

The system was used to collect data on Bangla and English alphabets as well as Bangla and English digits. All the data collected was downloaded from the server and separated into different folders for ease of use.

Overall, the system proved to be an effective method for collecting the required data. Its online-based interface made it easy to distribute the form to multiple individuals and gather data from them. The collected data would prove to be valuable for my research, and I was grateful to have developed such a functional and efficient system.

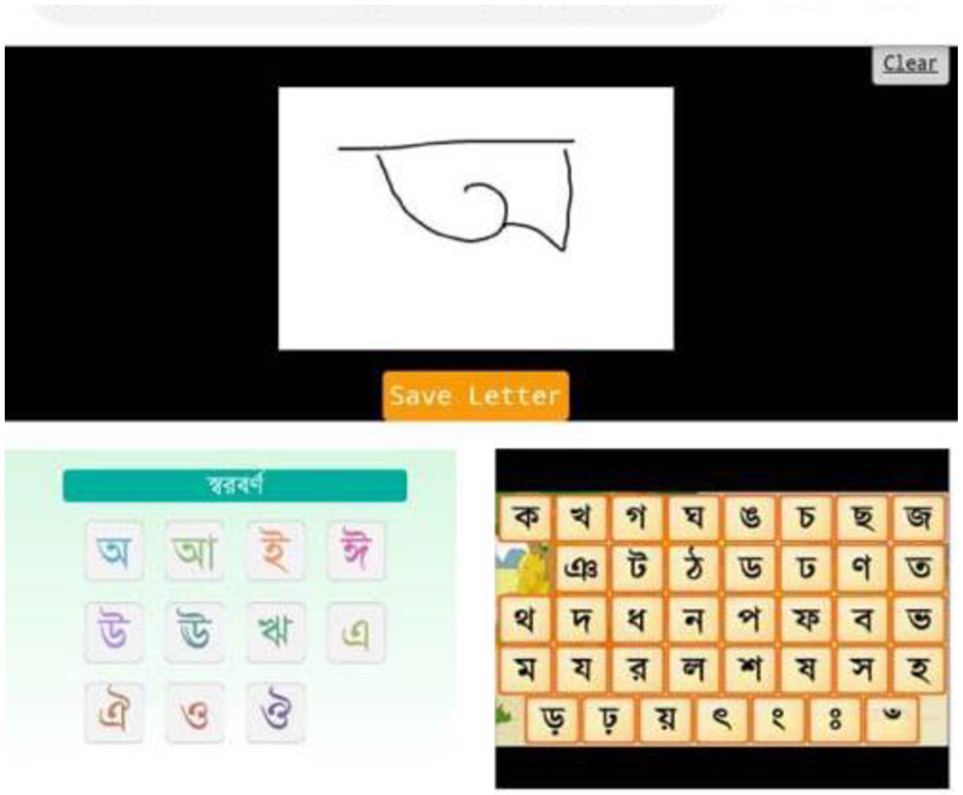


Figure 8: Data Collections System

I was able to collect a total of 4265 data entries through the system that I had developed. This data was classified into different categories based on the type of data collected.

Out of the total data collected, 740 data entries belonged to the Sorborno class, 1648 data entries belonged to the Benjonborno class, 1063 data entries belonged to the English Alphabet class, 367 data entries belonged to the Bangla Digit class, and 448 data entries belonged to the English Digit class. Unfortunately, there were some data entries that I was unable to classify, and they were deemed garbage data that I did not use in my analysis.

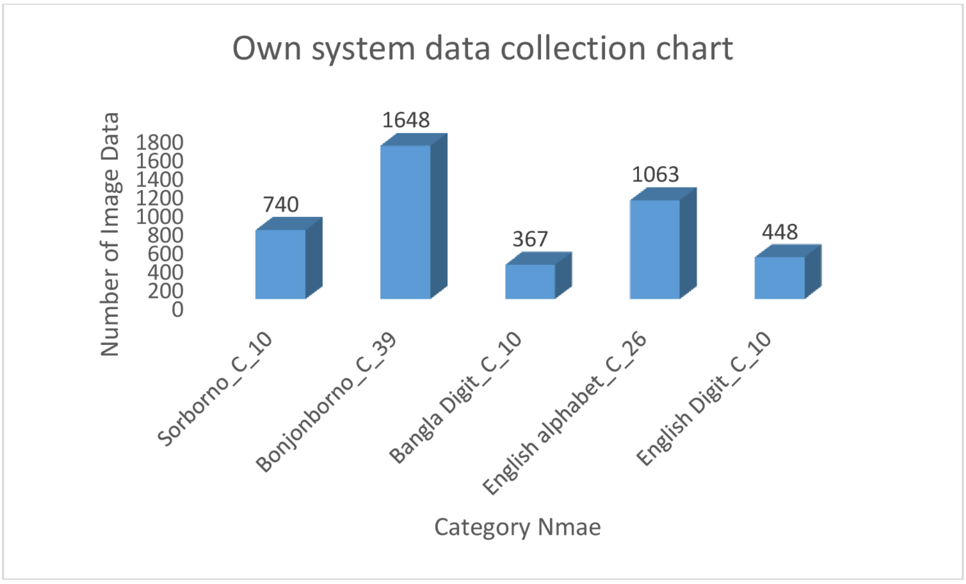


Figure 9: Own System Data Collection Chart

Here I attached the collected data view.

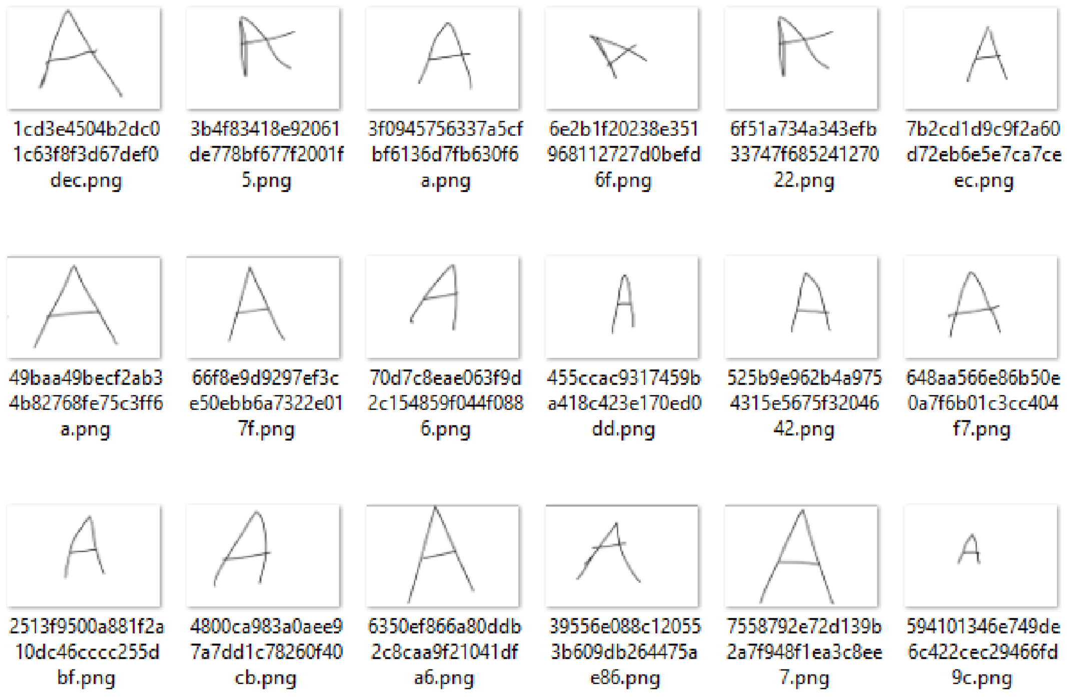

1cd3e4504b2dc01c63f8f3d67def0dec.png
3b4f83418e92061de778bf677f2001f5.png
3f0945756337a5cfbf6136d7fb630f6a.png
6e2b1f20238e351968112727d0befd6f.png
6f51a734a343efb33747f68524127022.png
7b2cd1d9c9f2a60d72eb6e5e7ca7ceec.png
49baa49becf2ab34b82768fe75c3ff6a.png
66f8e9d9297ef3ce50ebb6a7322e017f.png
70d7c8eae063f9d2c154859f044f0886.png
455ccac9317459ba418c423e170ed0dd.png
525b9e962b4a9754315e5675f3204642.png
648aa566e86b50e0a7f6b01c3cc404f7.png
2513f9500a881f2a10dc46cccc255dbf.png
4800ca983a0aee97a7dd1c78260f40cb.png
6350ef866a80ddb2c8caa9f21041dfa6.png
39556e088c120553b609db264475ae86.png
7558792e72d139b2a7f948f1ea3c8ee7.png
594101346e749de6c422cec29466fd9c.png

Figure 10: Collected Data English Alphabets

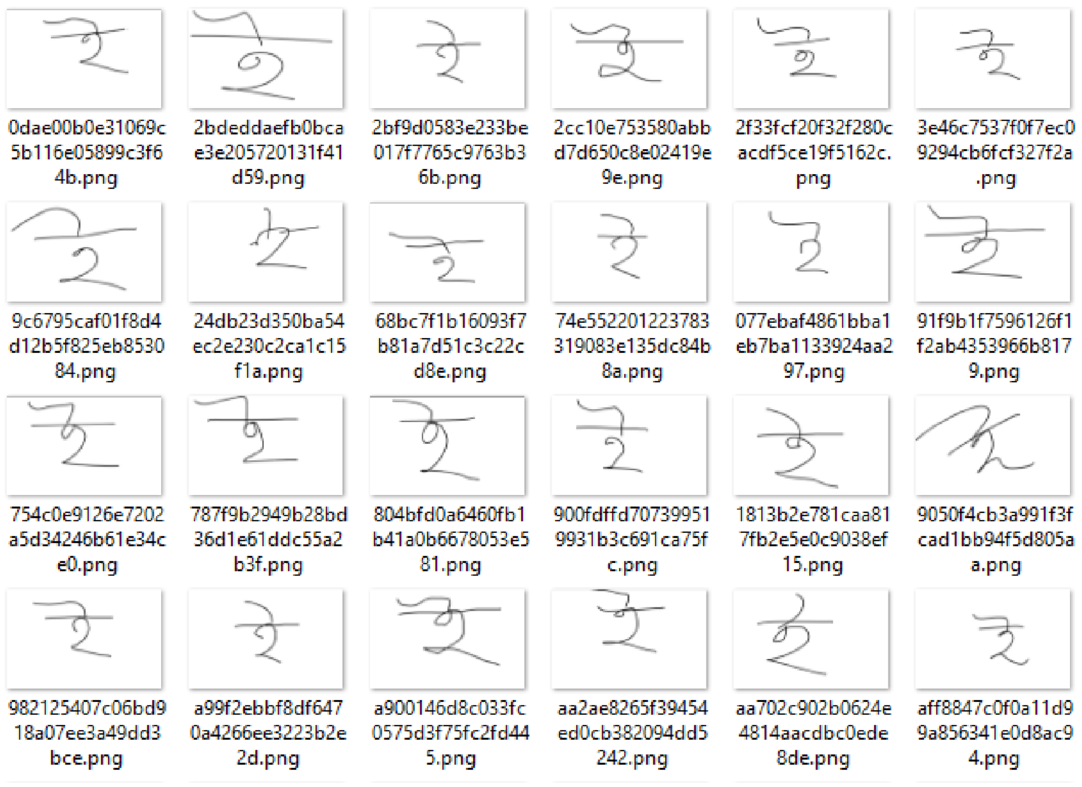

0dae00b0e31069c5b116e05899c3f64b.png
2bdeddaefb0bcae3e205720131f41d59.png
2bf9d0583e233be017f7765c9763b36b.png
2cc10e753580abbd7d650c8e02419e9e.png
2f33fcf20f32f280cacdf5ce19f5162c.png
3e46c7537f0f7ec09294cb6fcf327f2a.png
9c6795caf01f8d4d12b5f825eb853084.png
24db23d350ba54ec2e230c2ca1c15f1a.png
68bc7f1b16093f7b81a7d51c3c22cd8e.png
74e552201223783319083e135dc84b8a.png
077ebaf4861bba1eb7ba1133924aa297.png
91f9b1f7596126f1f2ab4353966b8179.png
754c0e9126e7202a5d34246b61e34ce0.png
787f9b2949b28bd36d1e61ddc55a2b3f.png
804bfd0a6460fb1b41a0b6678053e581.png
900fdffd707399519931b3c691ca75fc.png
1813b2e781caa817fb2e5e0c9038ef15.png
9050f4cb3a991f3fcad1bb94f5d805aa.png
982125407c06bd918a07ee3a49dd3bce.png
a99f2ebbf8df6470a4266ee3223b2e2d.png
a900146d8c033fc0575d3f75fc2fd445.png
aa2ae8265f39454ed0cb382094dd5242.png
aa702c902b0624e4814aacdbc0ede8de.png
aff8847c0f0a11d99a856341e0d8ac94.png

Figure 11: Collected Data Bangla Sorborno

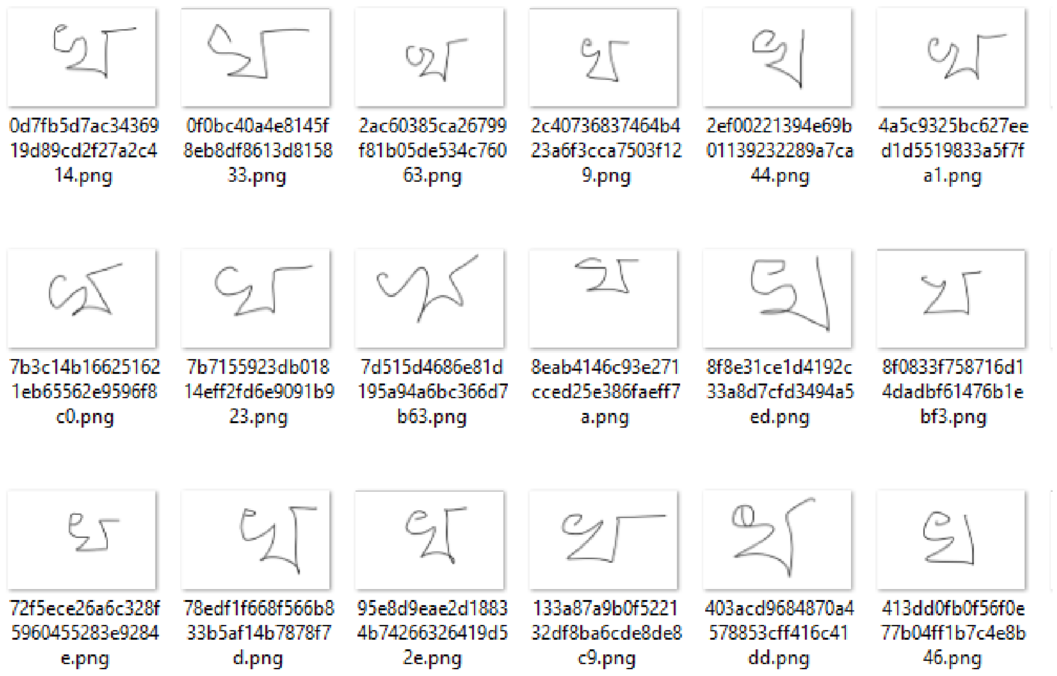

0d7fb5d7ac3436919d89cd2f27a2c414.png
0f0bc40a4e8145f8eb8df8613d815833.png
2ac60385ca26799f81b05de534c76063.png
2c40736837464b423a6f3cca7503f129.png
2ef00221394e69b01139232289a7ca44.png
4a5c9325bc627eed1d5519833a5f7fa1.png
7b3c14b166251621eb65562e9596f8c0.png
7b7155923db01814eff2fd6e9091b923.png
7d515d4686e81d195a94a6bc366d7b63.png
8eab4146c93e271cced25e386faeff7a.png
8f8e31ce1d4192c33a8d7cfd3494a5ed.png
8f0833f758716d14dadbf61476b1ebf3.png
72f5ece26a6c328f5960455283e9284e.png
78edf1f668f566b833b5af14b7878f7d.png
95e8d9eae2d18834b74266326419d52e.png
133a87a9b0f522132df8ba6cde8de8c9.png
403acd9684870a4578853cff416c41dd.png
413dd0fb0f56f0e77b04ff1b7c4e8b46.png

Figure 12: Collected Data Bangla Benjonborno

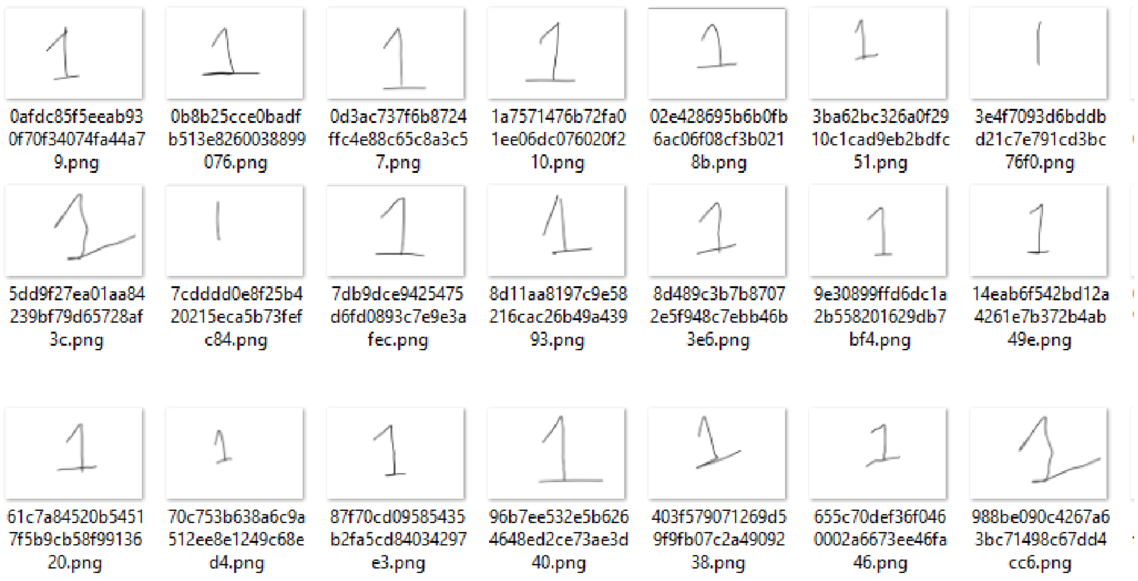

0afdc85f5eeab930f70f34074fa44a79.png
0b8b25cce0badfb513e8260038899076.png
0d3ac737f6b8724ffc4e88c65c8a3c57.png
1a7571476b72fa01ee06dc076020f210.png
02e428695b6b0fb6ac06f08cf3b0218b.png
3ba62bc326a0f2910c1cad9eb2bdfc51.png
3e4f7093d6bddbd21c7e791cd3bc76f0.png
5dd9f27ea01aa84239bf79d65728af3c.png
7cdddd0e8f25b420215eca5b73fefc84.png
7db9dce9425475d6fd0893c7e9e3afec.png
8d11aa8197c9e58216cac26b49a43993.png
8d489c3b7b87072e5f948c7ebb46b3e6.png
9e30899ffd6dc1a2b558201629db7bf4.png
14eab6f542bd12a4261e7b372b4ab49e.png
61c7a84520b54517f5b9cb58f9913620.png
70c753b638a6c9a512ee8e1249c68ed4.png
87f70cd09585435b2fa5cd84034297e3.png
96b7ee532e5b6264648ed2ce73ae3d40.png
403f579071269d59f9fb07c2a4909238.png
655c70def36f0460002a6673ee46fa46.png
988be090c4267a63bc71498c67dd4cc6.png

Figure 13: Collected Data English Digit

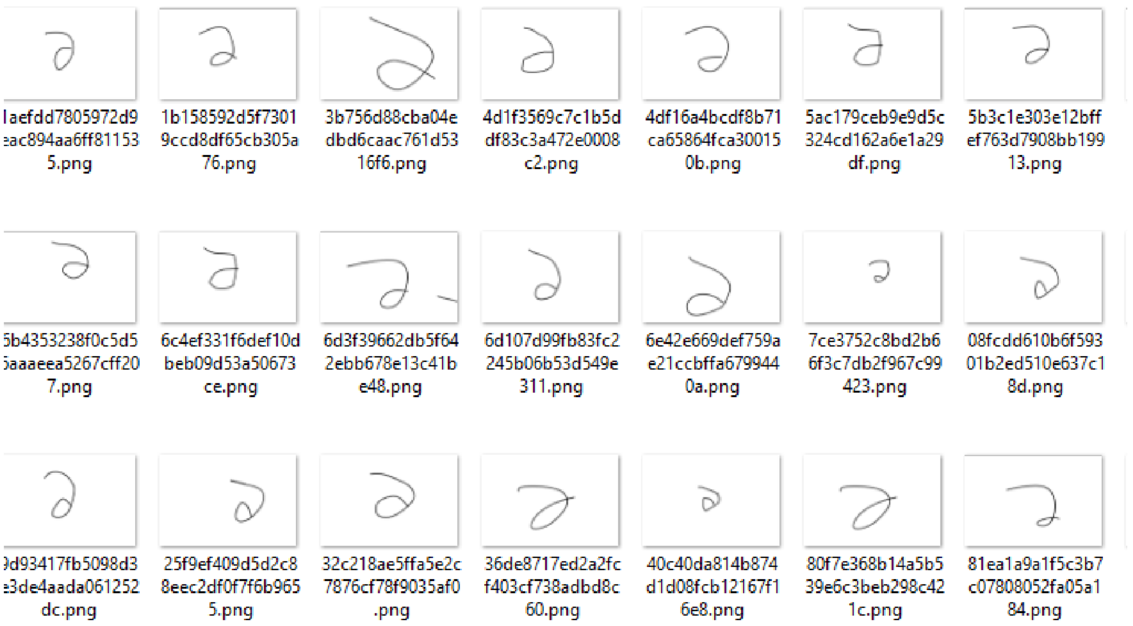

laefdd7805972d9
ac894aa6ff81153
5.png
1b158592d5f73019ccd8df65cb305a76.png
3b756d88cba04edbd6caac761d5316f6.png
4d1f3569c7c1b5ddf83c3a472e0008c2.png
4df16a4bcdf8b71ca65864fca300150b.png
5ac179ceb9e9d5c324cd162a6e1a29df.png
5b3c1e303e12bffef763d7908bb19913.png
5b4353238f0c5d5
aaaeea5267cff20
7.png
6c4ef331f6def10dbeb09d53a50673ce.png
6d3f39662db5f642ebb678e13c41be48.png
6d107d99fb83fc2245b06b53d549e311.png
6e42e669def759ae21ccbffa6799440a.png
7ce3752c8bd2b66f3c7db2f967c99423.png
08fcdd610b6f59301b2ed510e637c18d.png
d93417fb5098d3
3de4aada061252
dc.png
25f9ef409d5d2c88eec2df0f7f6b9655.png
32c218ae5ffa5e2c7876cf78f9035af0.png
36de8717ed2a2fcf403cf738adbd8c60.png
40c40da814b874d1d08fcb12167f16e8.png
80f7e368b14a5b539e6c3beb298c421c.png
81ea1a9a1f5c3b7c07808052fa05a184.png

Figure 14: Collected Data Bangla Digit

After collecting the data, I pre-processed it and run it through various algorithms in the Scikit-Learn library. Unfortunately, all of these algorithms provided very poor accuracy when applied to my data. While Support Vector Classification (SVC) provided slightly better accuracy than other algorithms I used, it still fell short of acceptable levels.

I went back to studying and learned about the Convolutional Neural Network (CNN) algorithm. I discovered that this algorithm is best for image and video, suited for large datasets and realized that I needed more data to apply it effectively. As a result, I collected additional image data from various sources to enhance my dataset.

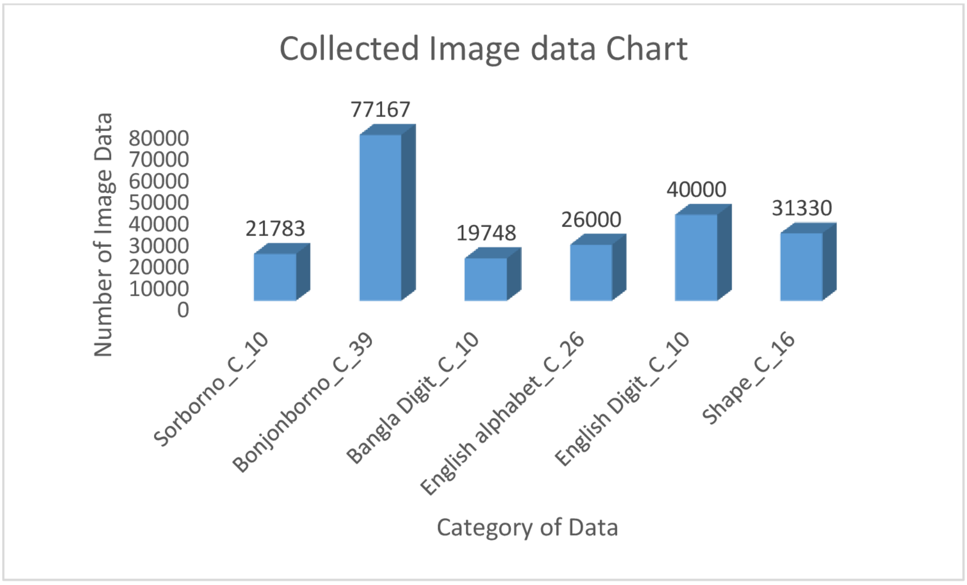


Figure 15:Collected Data From Various Sources (23) (24) (25) (26)

The Sorborno_C_10 category encompasses 10 subcategories of Bengali vowels, while the Bonjonborno_C_39 category consists of 39 subcategories of Bengali consonants. Bengali digits include numbers from 0 to 9. In addition, English Alphabets_C_26 represents 26 subcategories of characters, ranging from A to Z. English Digit_10 comprises 10 subcategories of numbers, spanning from 0 to 9. Lastly, Shape_C_16 includes 16 distinct subcategories of shapes, such as apple, bird, book, butterfly, candle, chair, circle, cup, fish, flower, house, line, square, star, tree, and triangle. Ultimately, the comprehensive dataset encompasses a total of 216,028 handwritten images, providing a rich and diverse collection for analysis.

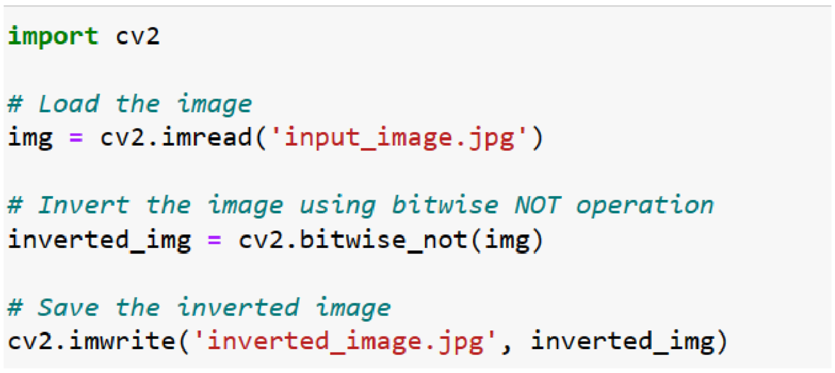
4.2 Data Pre-processing

Data pre-processing is a crucial step in preparing data for analysis. Prior to running the algorithms, I needed to pre-process the images to ensure they were in a format that the algorithms could handle. Pre-processing can involve various techniques such as normalization, resizing, and image augmentation to improve the quality and consistency of the data.

To pre-process my data, I employed a range of methods that were customized for my system. These steps were necessary to prepare the images for analysis by the algorithms. A detailed description of the pre-processing steps, along with a figure, is provided below.

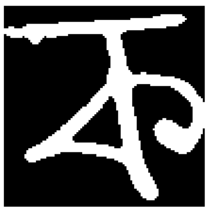
4.2.1 Inverted Image

To invert an image in Python, I used the OpenCV library. Here is the code that demonstrates how to invert an image:

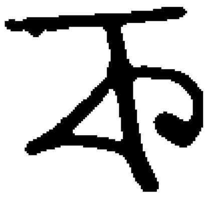


```
import cv2

# Load the image
img = cv2.imread('input_image.jpg')

# Invert the image using bitwise NOT operation
inverted_img = cv2.bitwise_not(img)

# Save the inverted image
cv2.imwrite('inverted_image.jpg', inverted_img)
```

Figure 16: Snapshot of Inverted Image code



Original Image

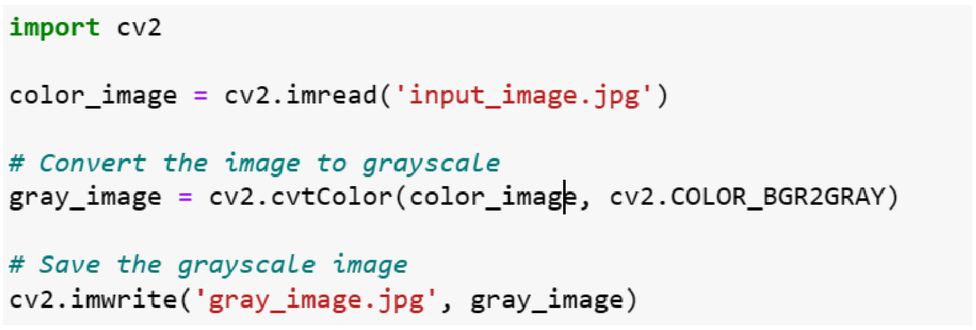


Inverted Image

In the code provided, I first loaded the input image using the **'*cv2.imread()*'** function. Then I inverted the image using the ***'cv2.bitwise_not()'*** function, which performed a bitwise NOT operation on each pixel of the image. Finally, I saved the inverted image using the ***'cv2.imwrite()'*** function.

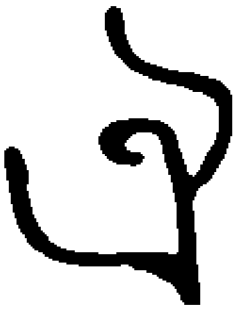
4.2.2 RGB to grayscale

To convert an RGB image to grayscale in Python, I used the OpenCV library. Here is the code that demonstrates how to convert an image from RGB to grayscale:

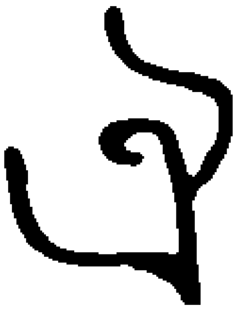


```
import cv2

color_image = cv2.imread('input_image.jpg')

# Convert the image to grayscale
gray_image = cv2.cvtColor(color_image, cv2.COLOR_BGR2GRAY)

# Save the grayscale image
cv2.imwrite('gray_image.jpg', gray_image)
```

Figure 17: Snapshot RGB to Grayscale code



Input Image RGB (23, 240, 144)

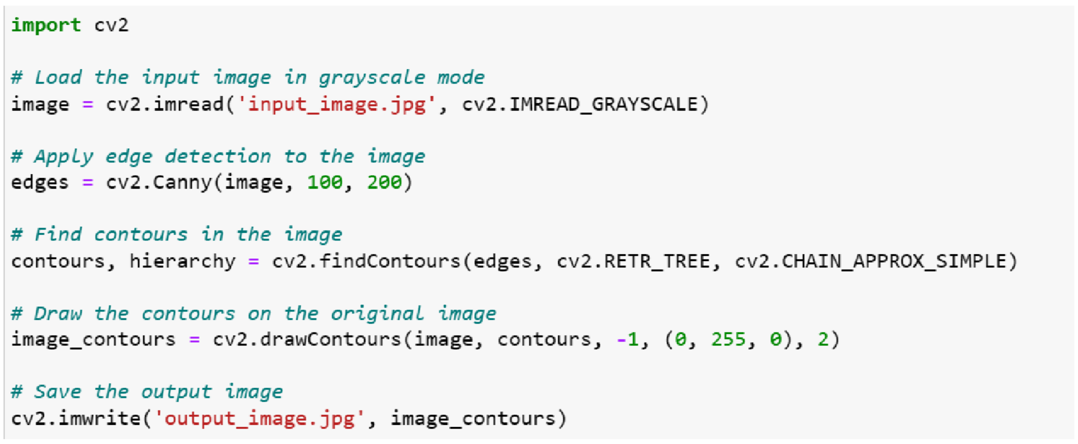


Output Image grayscale (45, 245)

In the code provided, I first loaded the input image in color mode using the ***cv2.imread()*** function. Then I converted the image to grayscale using the ***cv2.cvtColor()*** function, which took the input image and the color conversion code as arguments. The color conversion code ***cv2.COLOR_BGR2GRAY*** specified that the input image was in BGR format and should be converted to grayscale. Finally, I saved the grayscale image using the ***cv2.imwrite()*** function. In the above example, the two images are not a good example. However to clarify, the basic difference between them is that RGB is 3 dimensional and grayscale is 2 dimensional image.

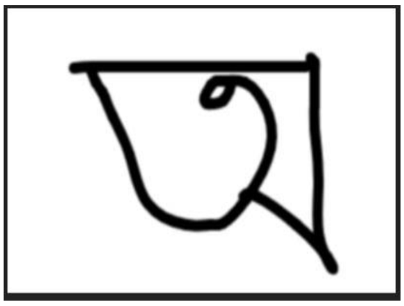
4.2.3 Image Contour Detection

To detect contours in an image using Python and OpenCV, I used the ***cv2.findContours()*** function. Here is the code that demonstrates how to find and draw contours in an image:

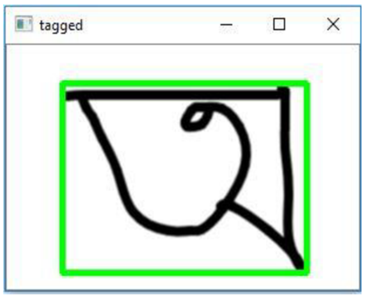


```
import cv2

# Load the input image in grayscale mode
image = cv2.imread('input_image.jpg', cv2.IMREAD_GRAYSCALE)

# Apply edge detection to the image
edges = cv2.Canny(image, 100, 200)

# Find contours in the image
contours, hierarchy = cv2.findContours(edges, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

# Draw the contours on the original image
image_contours = cv2.drawContours(image, contours, -1, (0, 255, 0), 2)

# Save the output image
cv2.imwrite('output_image.jpg', image_contours)
```

Figure 18: Snapshot of Image Contour Detection Code



Input Image



Output Image

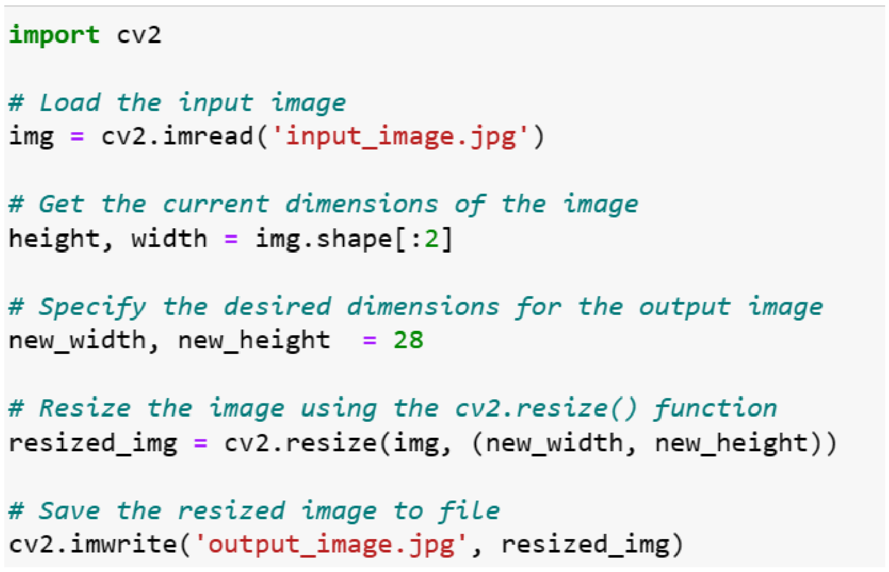
In the above code, I first loaded the input image in grayscale mode using the ***cv2.imread()*** function with the ***cv2.IMREAD_GRAYSCALE*** flag. Then I applied edge detection to the image using the ***cv2.Canny()*** function, which detects the edges in the image based on the gradient of the pixel intensities.

Next, I found the contours in the image using the ***cv2.findContours()*** function, which takes the edge-detected image, retrieval mode, and contour approximation method as arguments. The contours were returned as a list of points, along with a hierarchy of nested contours if applicable.

Finally, I drew the contours on the original image using the cv2.drawContours() function, which takes the input image, list of contours, contour index (set to -1 to draw all contours), color, and thickness as arguments. I saved the output image using the ***cv2.imwrite()***.

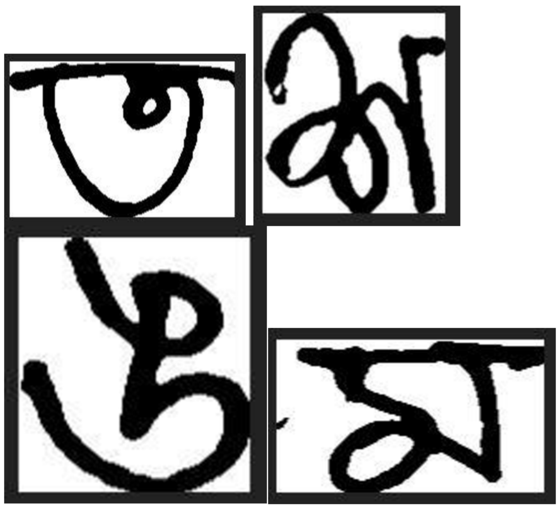
4.2.4 Image Resize

Here is a Python code snippet to resize an image using the OpenCV library:

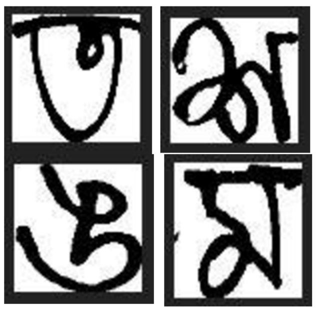


```python
import cv2

# Load the input image
img = cv2.imread('input_image.jpg')

# Get the current dimensions of the image
height, width = img.shape[:2]

# Specify the desired dimensions for the output image
new_width, new_height  = 28

# Resize the image using the cv2.resize() function
resized_img = cv2.resize(img, (new_width, new_height))

# Save the resized image to file
cv2.imwrite('output_image.jpg', resized_img)
```

Figure 19: Snapshot of Image Resize code



Input Image size are different



Output Image are specific size(28x28)

In the above code, I first loaded the input image using the ***cv2.imread()*** function. Then, I used the shape attribute to get the current height and width of the image.

Next, I specified the desired dimensions for the output image using the **new_width** and **new_height** variables. I used the dimension 28×28 to reduce memory usage. This dimension returned a **numpy** array of size 784. Then I resized the image using the ***cv2.resize()*** function and saved the output image using the **cv2.imwrite()** function.

4.3 Training the Algorithms and Comparison

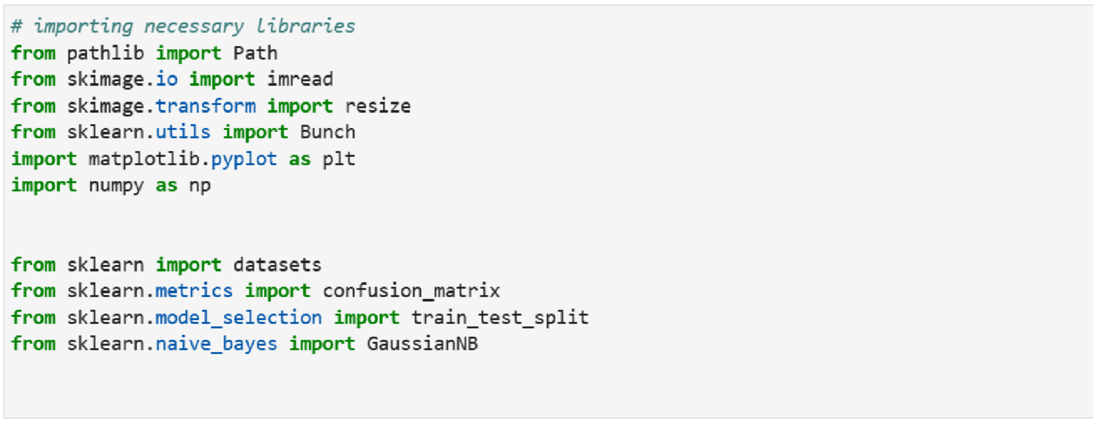
In this section, I will delve into a thorough comparison between various algorithms, ultimately selecting the best one based on a comprehensive evaluation. To substantiate my choice, I will provide detailed explanations and reasoning behind the decision. Furthermore, I will present statistics gathered from running each algorithm on my dataset, which will demonstrate their respective strengths and weaknesses.

Additionally, I will share code snippets and visualizations to offer readers a clear understanding of the implementation and performance of the chosen algorithm. This in-depth analysis will serve as a valuable guide for those seeking to implement the most effective solution for their specific problem.

4.3.1 Training Gaussian Naive Bayes

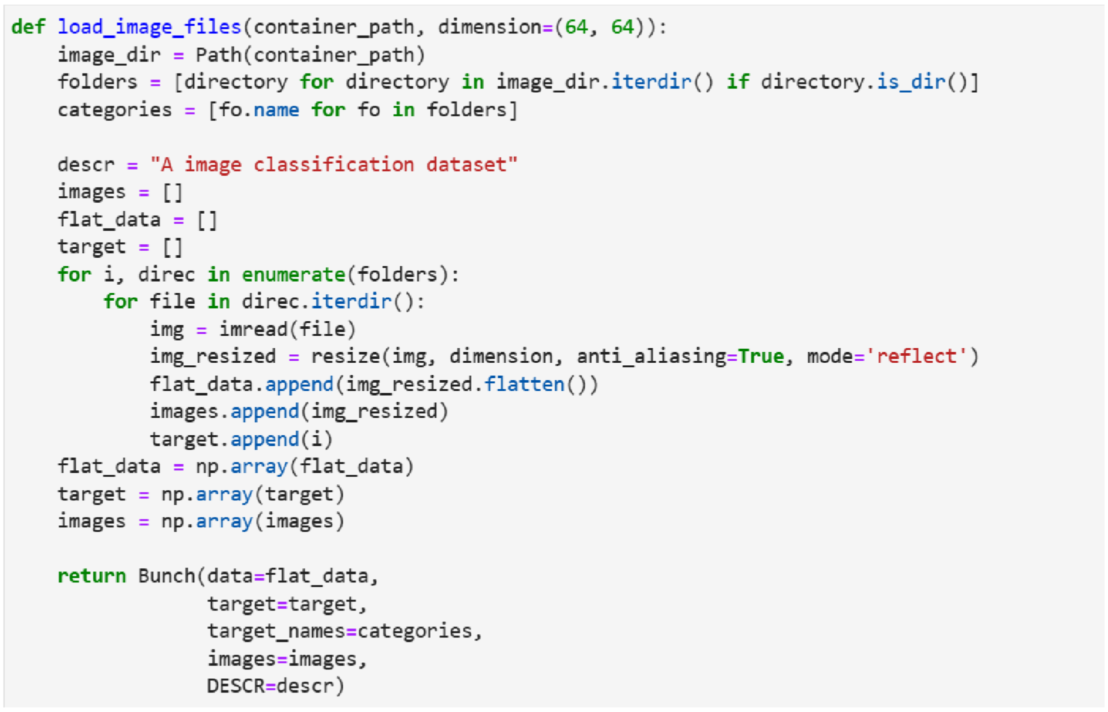
I began by implementing the Gaussian Naive Bayes (GaussianNB) algorithm on my custom dataset, which focused on the Sorborno category. The dataset was partitioned into training and testing subsets:

- The dataset consisted of a total of 740 data points.
- 593 of these data points were allocated for training the model.
- The remaining 147 data points were used to test the model's performance.



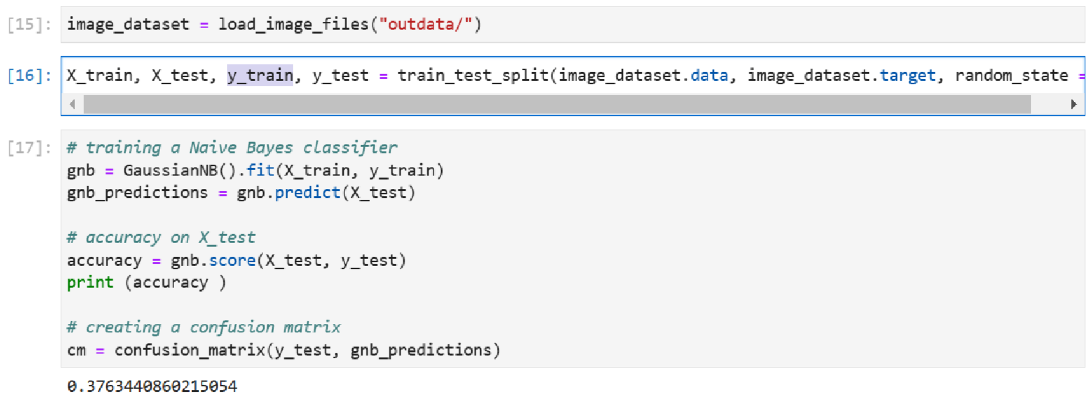
```python
# importing necessary libraries
from pathlib import Path
from skimage.io import imread
from skimage.transform import resize
from sklearn.utils import Bunch
import matplotlib.pyplot as plt
import numpy as np


from sklearn import datasets
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
```

Figure 20: Importing all the required library for the Gaussian Naive Bayes



```python
def load_image_files(container_path, dimension=(64, 64)):
    image_dir = Path(container_path)
    folders = [directory for directory in image_dir.iterdir() if directory.is_dir()]
    categories = [fo.name for fo in folders]

    descr = "A image classification dataset"
    images = []
    flat_data = []
    target = []
    for i, direc in enumerate(folders):
        for file in direc.iterdir():
            img = imread(file)
            img_resized = resize(img, dimension, anti_aliasing=True, mode='reflect')
            flat_data.append(img_resized.flatten())
            images.append(img_resized)
            target.append(i)
    flat_data = np.array(flat_data)
    target = np.array(target)
    images = np.array(images)

    return Bunch(data=flat_data,
                 target=target,
                 target_names=categories,
                 images=images,
                 DESCR=descr)
```

Figure 21: Defining the function process my data into the algorithm



```
[15]: image_dataset = load_image_files("outdata/")

[16]: X_train, X_test, y_train, y_test = train_test_split(image_dataset.data, image_dataset.target, random_state

[17]: # training a Naive Bayes classifier
      gnb = GaussianNB().fit(X_train, y_train)
      gnb_predictions = gnb.predict(X_test)

      # accuracy on X_test
      accuracy = gnb.score(X_test, y_test)
      print (accuracy )

      # creating a confusion matrix
      cm = confusion_matrix(y_test, gnb_predictions)

      0.3763440860215054
```

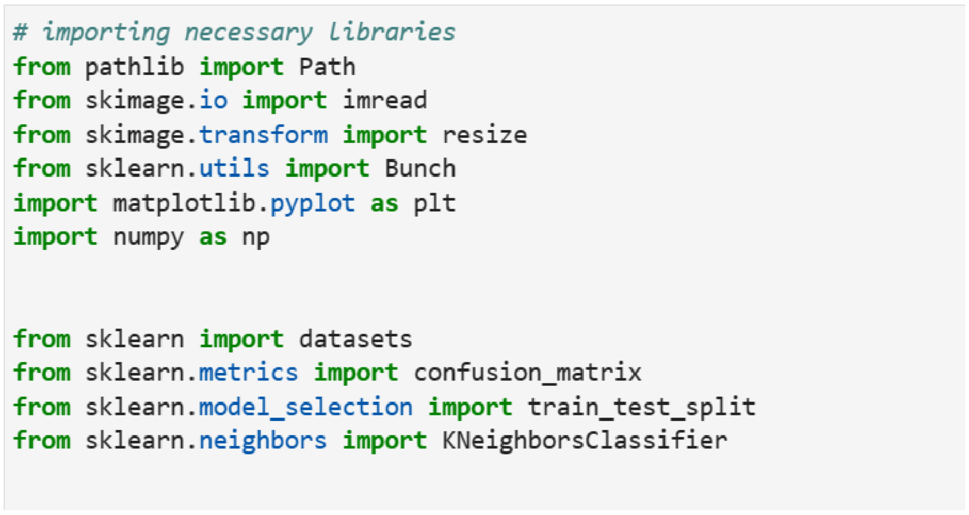
Figure 22: GaussianNB Algorithm Using scikit-learn

Here loading dataset and splitting the training and testing set. Upon applying the GaussianNB algorithm and evaluating its performance based on the test and train datasets, the model achieved an accuracy of 37%.

4.3.2 Training K-Nearest Neighbors Classifier

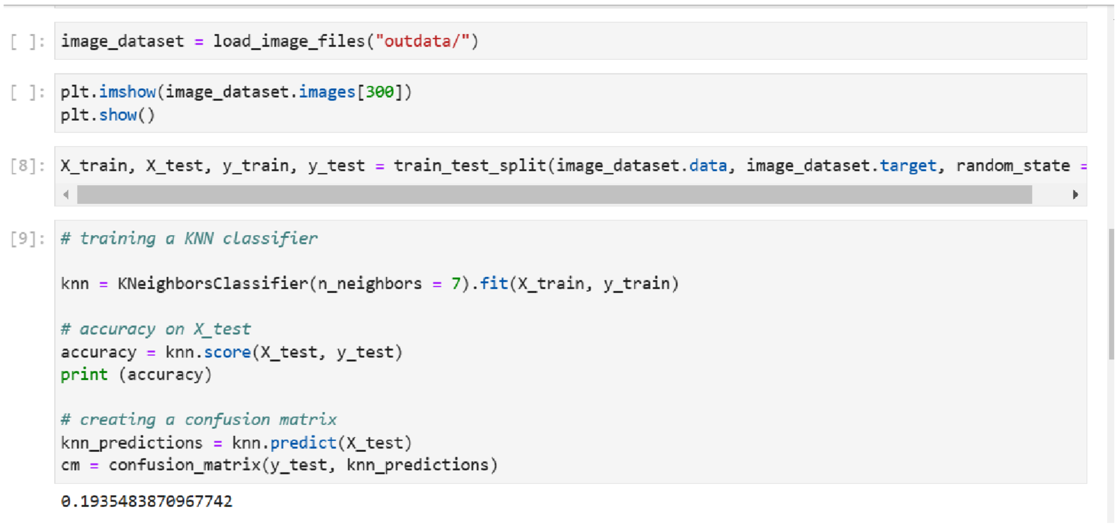
Next, I implemented the K-Nearest Neighbors Classifier (KNeighborsClassifier) algorithm on the same custom dataset focused on the Sorborno category. The dataset was again divided into training and testing subsets:

- The dataset comprised a total of 740 data points.
- 593 of these data points were allocated for training the model.
- The remaining 147 data points were utilized to test the model's performance.



```python
# importing necessary libraries
from pathlib import Path
from skimage.io import imread
from skimage.transform import resize
from sklearn.utils import Bunch
import matplotlib.pyplot as plt
import numpy as np


from sklearn import datasets
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

Figure 23: Importing all the required library for the KNeighborsClassifier



```
[ ]: image_dataset = load_image_files("outdata/")

[ ]: plt.imshow(image_dataset.images[300])
     plt.show()

[8]: X_train, X_test, y_train, y_test = train_test_split(image_dataset.data, image_dataset.target, random_state =

[9]: # training a KNN classifier

     knn = KNeighborsClassifier(n_neighbors = 7).fit(X_train, y_train)

     # accuracy on X_test
     accuracy = knn.score(X_test, y_test)
     print (accuracy)

     # creating a confusion matrix
     knn_predictions = knn.predict(X_test)
     cm = confusion_matrix(y_test, knn_predictions)

     0.1935483870967742
```

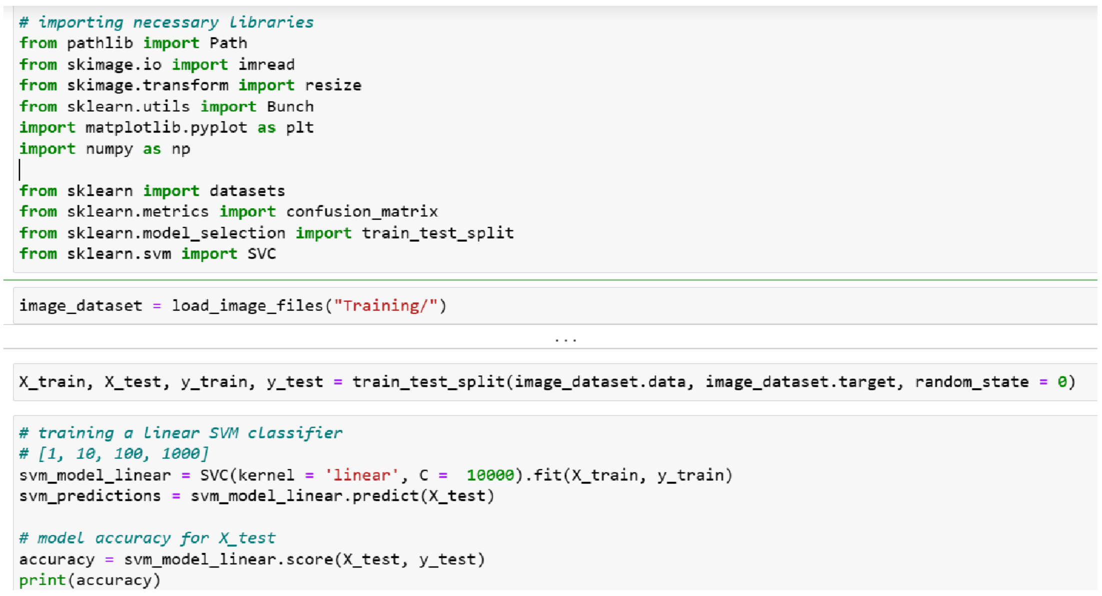
Figure 24:KNeighborsClassifier Algorithm using scikit-learn

Here I did same process loading dataset and splitting the training and testing set Upon applying the **KNeighborsClassifier** algorithm and evaluating its performance based on the test and train datasets, the model achieved an accuracy of 19%.

4.3.3 Training Support Vector Classifier

Subsequently, I implemented the Support Vector Classifier (SVC) algorithm on the same custom dataset focused on the Sorborno category. The dataset was divided into training and testing subsets, just as before:

- The dataset consisted of a total of 740 data points.
- 593 of these data points were allocated for training the model.
- The remaining 147 data points were used to test the model's performance.



```
# importing necessary libraries
from pathlib import Path
from skimage.io import imread
from skimage.transform import resize
from sklearn.utils import Bunch
import matplotlib.pyplot as plt
import numpy as np

from sklearn import datasets
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
```

```
image_dataset = load_image_files("Training/")
```

```
X_train, X_test, y_train, y_test = train_test_split(image_dataset.data, image_dataset.target, random_state = 0)
```

```
# training a linear SVM classifier
# [1, 10, 100, 1000]
svm_model_linear = SVC(kernel = 'linear', C =  10000).fit(X_train, y_train)
svm_predictions = svm_model_linear.predict(X_test)

# model accuracy for X_test
accuracy = svm_model_linear.score(X_test, y_test)
print(accuracy)
```

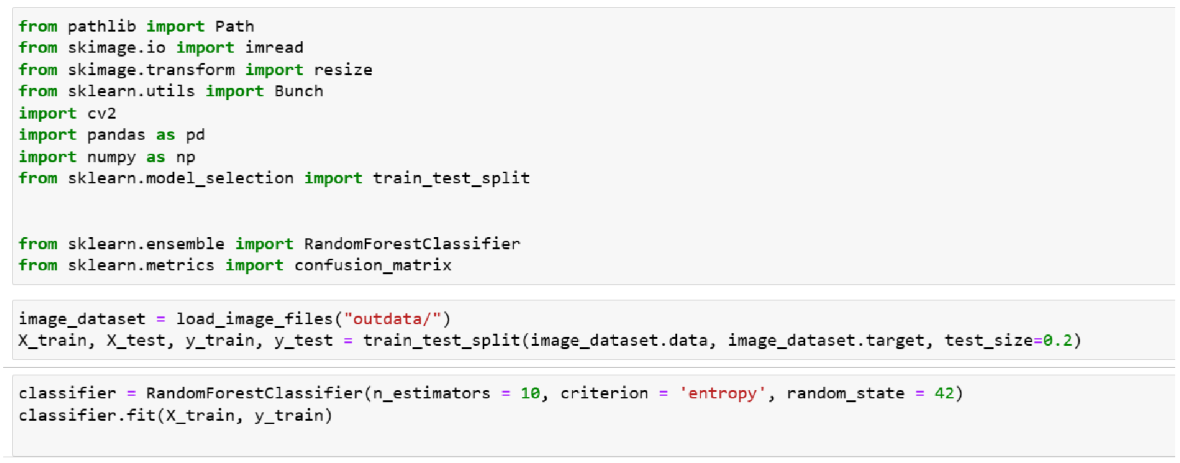
Figure 25: Training SVC code snapshot using scikit-learn

Upon applying the SVC algorithm and evaluating its performance based on the test and train datasets, the model achieved an accuracy of 66%.

4.3.4 Training Random Forest Classifier

Subsequently, I implemented the Random Forest Classifier algorithm on the same custom dataset focused on the Sorborno category. As with the previous algorithms, the dataset was divided into training and testing subsets:

- The dataset comprised a total of 740 data points.
- 593 of these data points were allocated for training the model.
- The remaining 147 data points were utilized to test the model's performance.



```python
from pathlib import Path
from skimage.io import imread
from skimage.transform import resize
from sklearn.utils import Bunch
import cv2
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split


from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
```

```python
image_dataset = load_image_files("outdata/")
X_train, X_test, y_train, y_test = train_test_split(image_dataset.data, image_dataset.target, test_size=0.2)
```

```python
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 42)
classifier.fit(X_train, y_train)
```

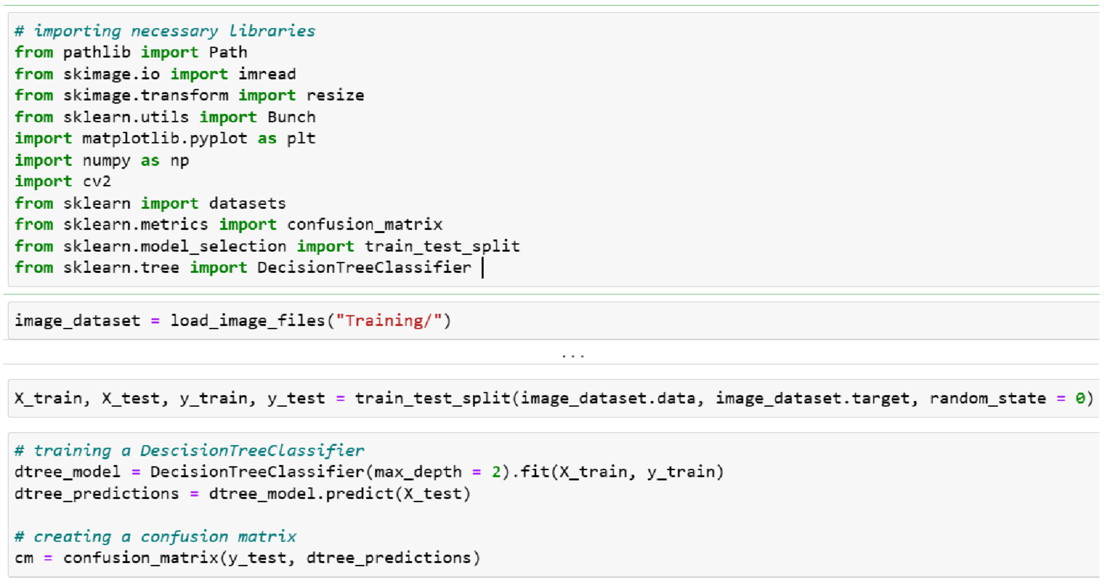
Figure 26: Training Random Forest Classifier code snapshot using scikit-learn

Upon applying the **Random Forest Classifier** algorithm and evaluating its performance based on the test and train datasets, the model achieved an accuracy of 44%.

4.3.5 Training Decision Tree Classifier

Lastly, I implemented the Decision Tree Classifier algorithm on the same custom dataset focused on the Sorborno category. As with the other algorithms, the dataset was divided into training and testing subsets:

- The dataset consisted of a total of 740 data points.
- 593 of these data points were allocated for training the model.
- The remaining 147 data points were used to test the model's performance.



```
# importing necessary libraries
from pathlib import Path
from skimage.io import imread
from skimage.transform import resize
from sklearn.utils import Bunch
import matplotlib.pyplot as plt
import numpy as np
import cv2
from sklearn import datasets
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

```
image_dataset = load_image_files("Training/")
```

```
...
```

```
X_train, X_test, y_train, y_test = train_test_split(image_dataset.data, image_dataset.target, random_state = 0)
```

```
# training a DescisionTreeClassifier
dtree_model = DecisionTreeClassifier(max_depth = 2).fit(X_train, y_train)
dtree_predictions = dtree_model.predict(X_test)

# creating a confusion matrix
cm = confusion_matrix(y_test, dtree_predictions)
```

Figure 27: Training Decision Tree Classifier code snapshot Using scikit-learn

Upon applying the Decision Tree Classifier algorithm and evaluating its performance based on the test and train datasets, the model achieved an accuracy of 52%.

4.3.6 Comparison

After applying various classification algorithms to the custom dataset focused on the Sorborno category, the following accuracies were obtained:

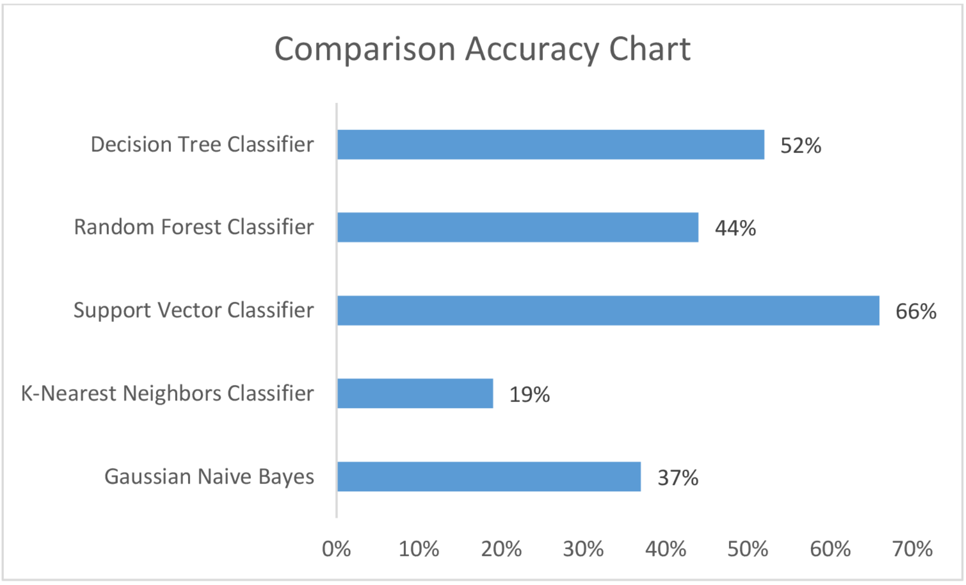


Figure 28: Comparison Chart between above algorithms

Based on these results, the Support Vector Classifier (SVC) outperforms the other algorithms, achieving the highest accuracy at 66%. On the other hand, the K-Nearest Neighbors Classifier yields the lowest accuracy at 19%, making it the least effective option among the tested algorithms. Although the Support Vector Classifier achieved the highest accuracy among the tested algorithms, the performance is still not entirely satisfactory for the problem at hand. Consequently, I decided to investigate Convolutional Neural Network (CNN) algorithms further, as they may offer improved results. In the subsequent sections of my literature review chapter, I will present a detailed analysis of CNN algorithms and their outcomes, providing additional insights to guide the selection of the most appropriate algorithm for this particular dataset.

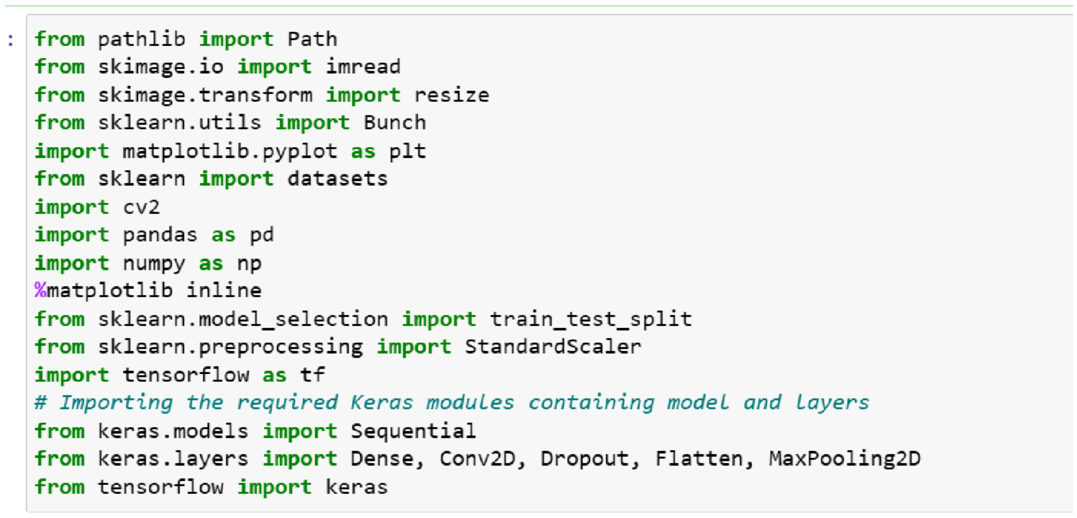
4.4 Training The Convolutional Neural Network (CNN)

Leveraging the Convolutional Neural Network (CNN) algorithm, I plan to process my collected dataset, which comprises multiple categories: Sorborno, Bonjonborno, Bangla Digit, English Alphabet, English Digit, and 16 types of common shapes. For each category, I will independently apply the algorithm, enabling a tailored approach to capture the unique characteristics of each group.

In this section, I will focus on one specific category, Bonjonborno, which contains the largest number of subcategories, totalling 39. End of the section I will show you accuracy chart for rest of categories data. By concentrating on this category, I will demonstrate the effectiveness of the chosen algorithm on a complex dataset.

Following the training process, I will save the resulting models for future utilization within my applications. This approach ensures the development of specialized models designed to address the specific needs of each category, thereby maximizing performance and enhancing the overall effectiveness of the system.

4.4.1 Implementations of CNN on Bonjonborno Category



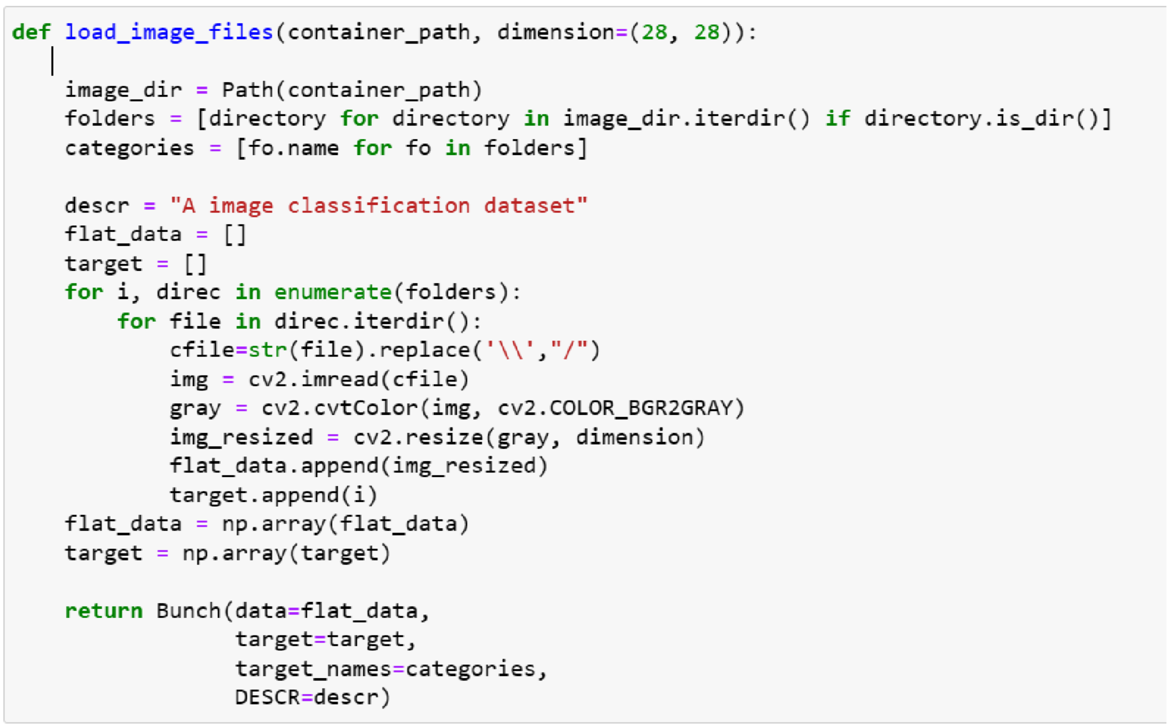
```
from pathlib import Path
from skimage.io import imread
from skimage.transform import resize
from sklearn.utils import Bunch
import matplotlib.pyplot as plt
from sklearn import datasets
import cv2
import pandas as pd
import numpy as np
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
# Importing the required Keras modules containing model and layers
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D
from tensorflow import keras
```

Figure 29: Imported all the required library for final outcome

In the provided code snippet, essential libraries and modules have been imported to facilitate the implementation of the Convolutional Neural Network (CNN) algorithm. These imports encompass the following functionalities:

- File and path manipulation using Path from the ***pathlib*** library.
- Image processing operations, such as reading and resizing images, through ***imread*** and resize from the skimage.io and ***skimage.transform*** libraries.
- Dataset manipulation and visualization tools, including ***Bunch***, ***matplotlib.pyplot***, and datasets from the ***sklearn.utils*** and ***sklearn*** libraries.
- Computer vision functionality provided by the ***cv2*** library.
- Data manipulation and numerical computing using ***pandas*** and ***numpy*** libraries.
- Machine learning utilities, such as ***train_test_split*** and ***StandardScaler***, from the ***sklearn.model_selection*** and ***sklearn.preprocessing*** libraries.
- ***Tensorflow*** framework, imported as ***tf***.
- ***Keras*** modules containing essential model and layers for implementing CNN, including ***Sequential***, ***Dense***, ***Conv2D***, ***Dropout***, ***Flatten***, and ***MaxPooling2D*** from the ***keras.models*** and ***keras.layers*** libraries.
- ***Keras*** integration with ***TensorFlow***, imported from the ***tensorflow*** library.

These imports provide a comprehensive toolkit for developing, training, and evaluating a CNN-based model for the targeted dataset, ensuring an efficient and streamlined implementation process.



```python
def load_image_files(container_path, dimension=(28, 28)):

    image_dir = Path(container_path)
    folders = [directory for directory in image_dir.iterdir() if directory.is_dir()]
    categories = [fo.name for fo in folders]

    descr = "A image classification dataset"
    flat_data = []
    target = []
    for i, direc in enumerate(folders):
        for file in direc.iterdir():
            cfile=str(file).replace('\\',"/")
            img = cv2.imread(cfile)
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            img_resized = cv2.resize(gray, dimension)
            flat_data.append(img_resized)
            target.append(i)
    flat_data = np.array(flat_data)
    target = np.array(target)

    return Bunch(data=flat_data,
                 target=target,
                 target_names=categories,
                 DESCR=descr)
```

Figure 30: This function pre-process the data and load data into model

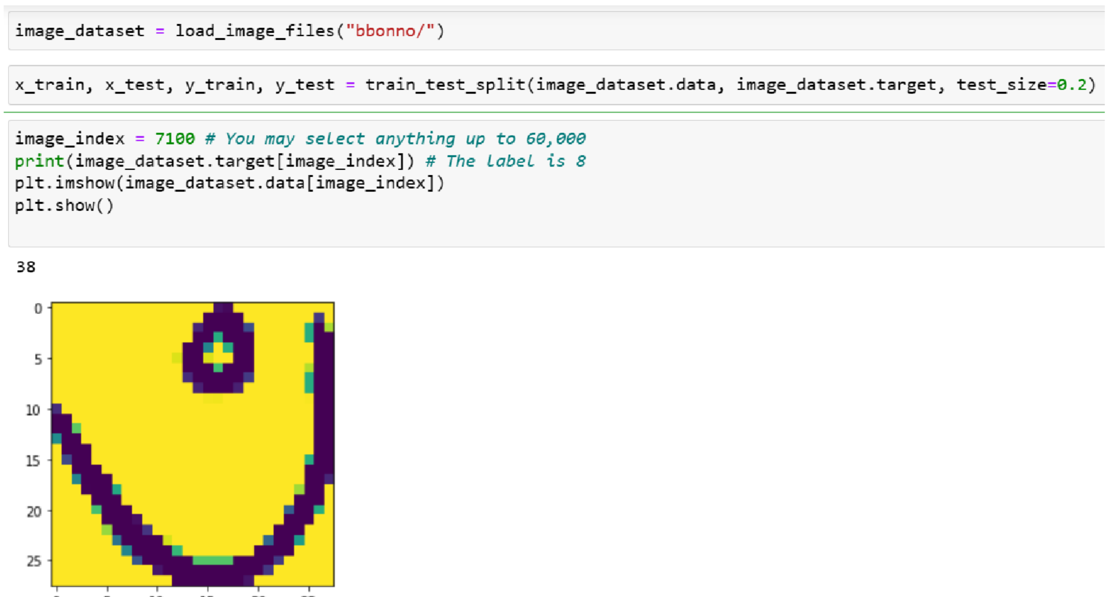
The ***load_image_files*** function serves as a utility for pre-processing and loading image data into a structured format, specifically a ***numpy*** array. It takes two parameters as input: ***container_path***, which represents the path to the folder containing the image data, and dimension, which is a tuple indicating the desired dimensions for resizing the images (default is set to 28x28 pixels).

Within the function, the following steps are carried out:

- The ***container_path*** is converted into a Path object, and the subfolders are identified as individual categories.
- The description variable ***descr*** is initialized with the string "A image classification dataset".

- Three lists, ***flat_data***, target, and categories, are initialized to store the pre-processed image data, corresponding category indices, and category names, respectively.
- The function iterates through each category folder, reads and processes each image file, and appends the pre-processed image data to ***flat_data*** and the corresponding category index to target.
- Image processing steps include converting the image to grayscale using ***cv2.cvtColor()*** and resizing it to the specified dimensions using ***cv2.resize()***.
- The lists ***flat_data*** and target are then converted to ***numpy*** arrays.
- Finally, a ***Bunch*** object is returned, containing the pre-processed image data, target indices, category names, and dataset description.

By utilizing this function, the image data is effectively pre-processed and transformed into a format suitable for further analysis, such as training a CNN-based model.



```
image_dataset = load_image_files("bbonno/")
```

```
x_train, x_test, y_train, y_test = train_test_split(image_dataset.data, image_dataset.target, test_size=0.2)
```

```
image_index = 7100 # You may select anything up to 60,000
print(image_dataset.target[image_index]) # The label is 8
plt.imshow(image_dataset.data[image_index])
plt.show()
```

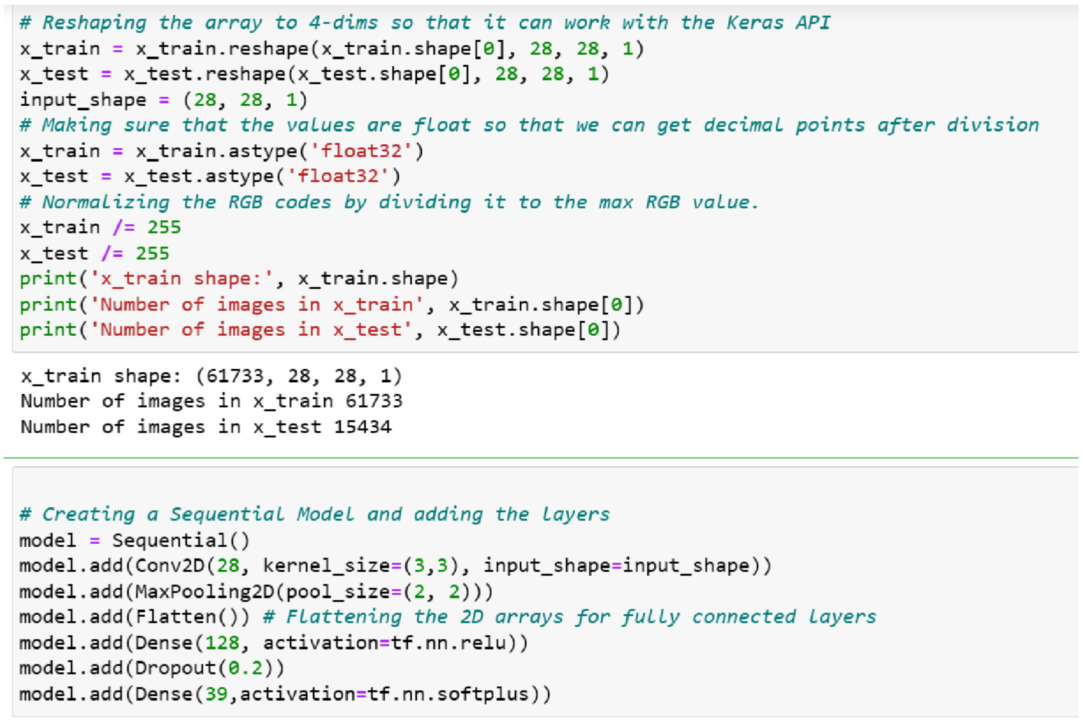
```
38
```

Figure 31: After Pre-processing the data plotting one image data

The code snippet provided executes the following steps:

- The ***load_image_files()*** function is called with the "***bbonno/***" path as its argument, which loads and pre-processes the image data from the specified directory, returning the ***image_dataset*** object.
- The dataset is then split into training and testing sets using the ***train_test_split()*** function from ***sklearn.model_selection***. It takes the image data, target labels, and a ***test_size*** parameter as input (set to 0.2, representing a 20% split for the test set). The function returns ***x_train, x_test, y_train***, and ***y_test*** variables containing the respective training and testing data and labels.
- The ***image_index*** variable is initialized with the value ***7100***, which is used to select a specific image from the dataset for display.
- The corresponding target label for the selected image is printed using ***print(image_dataset.target[image_index]).***
- The selected image is displayed using ***plt.imshow(image_dataset.data[image_index])***, followed by a call to ***plt.show()*** to render the image.

This code snippet demonstrates loading, pre-processing, and splitting the image dataset, as well as displaying a specific image and its corresponding label from the dataset.



```
# Reshaping the array to 4-dims so that it can work with the Keras API
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)
# Making sure that the values are float so that we can get decimal points after division
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
# Normalizing the RGB codes by dividing it to the max RGB value.
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print('Number of images in x_train', x_train.shape[0])
print('Number of images in x_test', x_test.shape[0])
```

```
x_train shape: (61733, 28, 28, 1)
Number of images in x_train 61733
Number of images in x_test 15434
```

```
# Creating a Sequential Model and adding the Layers
model = Sequential()
model.add(Conv2D(28, kernel_size=(3,3), input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten()) # Flattening the 2D arrays for fully connected layers
model.add(Dense(128, activation=tf.nn.relu))
model.add(Dropout(0.2))
model.add(Dense(39,activation=tf.nn.softplus))
```

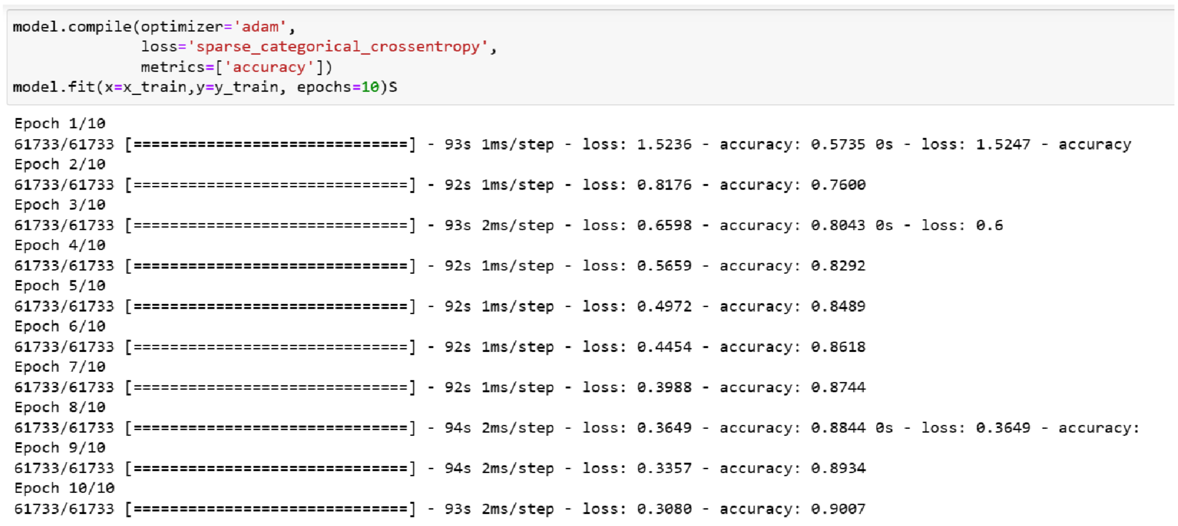
Figure 32: Preparing the image data and creating a CNN model using Keras

The provided code snippet outlines the steps involved in preparing the image data and creating a CNN model using Keras. Here is a summary of each step:

1. Reshape the input data (***x_train*** and ***x_test***) to 4-dimensional arrays to make them compatible with the ***Keras*** API. Each sample is reshaped to a 28x28x1 array, representing a 28x28 pixel grayscale image.
2. Convert the input data's datatype to ***'float32'*** to ensure decimal values are preserved after division.
3. Normalize the input data by dividing each pixel value by the maximum possible value (255). This scales the pixel values to the range of 0 to 1, which generally improves model performance.
4. Print the shapes of ***x_train*** and ***x_test*** along with the number of images in each set.
5. Define the CNN model using a ***Keras*** Sequential model, which allows stacking layers on top of each other. The model includes the following layers:

- A ***Conv2D*** layer with 28 filters, a kernel size of 3x3, and input shape set to 28x28x1 (corresponding to the input images). This layer performs the convolution operation.
- A ***MaxPooling2D*** layer with a pool size of 2x2, which reduces the spatial dimensions of the feature maps.
- A ***Flatten*** layer, which flattens the ***2D*** feature maps into a ***1D*** array to be fed into the fully connected layers.
- A ***Dense*** (fully connected) layer with ***128*** units and a ***ReLU*** activation function.
- A ***Dropout*** layer with a rate of ***0.2***, which randomly drops a fraction of the input units during training to prevent overfitting.
- A final ***Dense*** layer with ***39*** units (corresponding to the number of classes) and a ***softplus*** activation function to produce the class probabilities.

This code snippet prepares the image data and constructs a CNN model suitable for classifying the images into one of the 39 categories.



```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x=x_train,y=y_train, epochs=10)S
```

```
Epoch 1/10
61733/61733 [==============================] - 93s 1ms/step - loss: 1.5236 - accuracy: 0.5735 0s - loss: 1.5247 - accuracy
Epoch 2/10
61733/61733 [==============================] - 92s 1ms/step - loss: 0.8176 - accuracy: 0.7600
Epoch 3/10
61733/61733 [==============================] - 93s 2ms/step - loss: 0.6598 - accuracy: 0.8043 0s - loss: 0.6
Epoch 4/10
61733/61733 [==============================] - 92s 1ms/step - loss: 0.5659 - accuracy: 0.8292
Epoch 5/10
61733/61733 [==============================] - 92s 1ms/step - loss: 0.4972 - accuracy: 0.8489
Epoch 6/10
61733/61733 [==============================] - 92s 1ms/step - loss: 0.4454 - accuracy: 0.8618
Epoch 7/10
61733/61733 [==============================] - 92s 1ms/step - loss: 0.3988 - accuracy: 0.8744
Epoch 8/10
61733/61733 [==============================] - 94s 2ms/step - loss: 0.3649 - accuracy: 0.8844 0s - loss: 0.3649 - accuracy:
Epoch 9/10
61733/61733 [==============================] - 94s 2ms/step - loss: 0.3357 - accuracy: 0.8934
Epoch 10/10
61733/61733 [==============================] - 93s 2ms/step - loss: 0.3080 - accuracy: 0.9007
```

Figure 33: CNN model is compiled and trained

In the provided code snippet, the CNN model is compiled and trained using the following steps:

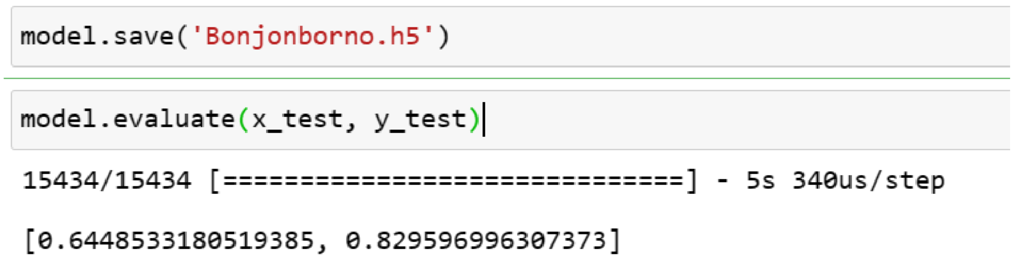
1. Compile the model with the specified parameters:

   - The optimizer is set to ***'adam'***, a popular optimization algorithm for deep learning models. It adjusts the model's weights based on the calculated gradients to minimize the loss function.
   - The loss function is set to ***'sparse_categorical_crossentropy'***, which is appropriate for multi-class classification problems with integer labels. It measures the dissimilarity between the predicted class probabilities and the true class labels, and the optimizer aims to minimize this value.
   - The performance metric is set to ***'accuracy'***, which calculates the proportion of correctly classified samples during training.

2. Train the model using the ***model.fit()*** method with the following inputs:

   - ***x_train*** and ***y_train*** are the training dataset and corresponding labels, respectively.
   - ***epochs*** is set to ***10***, which represents the number of complete passes through the entire training dataset. The model weights are updated incrementally with each pass.

This code snippet compiles the CNN model with the specified optimizer, loss function, and performance metric, and then trains the model on the provided training data for 10 epochs.



```
model.save('Bonjonborno.h5')
```

```
model.evaluate(x_test, y_test)
```

```
15434/15434 [==============================] - 5s 340us/step

[0.6448533180519385, 0.829596996307373]
```

Figure 34: saving the trained model and evaluating its performance

The provided code snippet demonstrates two actions: saving the trained model and evaluating its performance on the test dataset.

1. Save the trained model: The ***model.save()*** method is called with the filename ***'Bonjonborno.h5'***. This saves the model's architecture, optimizer, and learned weights to an HDF5 file. This allows you to reuse the model later without retraining it, making it convenient for deployment in applications.

2. Evaluate the model: The ***model.evaluate()*** method is called with the test dataset (***x_test*** and ***y_test***). This function calculates the model's performance on the test dataset in terms of the loss function and the specified performance metric (accuracy, in this case).

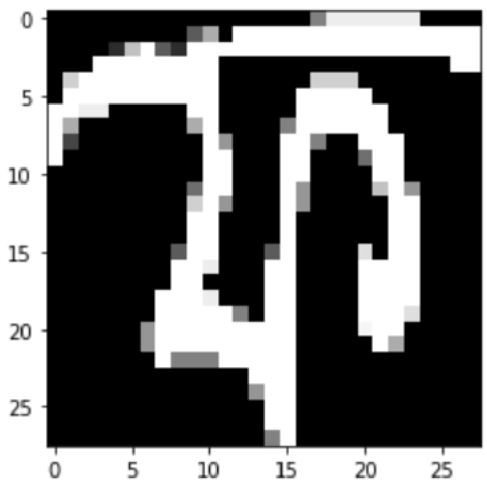
The output provided indicates that the evaluation process has completed, with the following results:

Loss***: 0.6448533180519385***
Accuracy: ***0.829596996307373*** (approximately **82.96%)**

This shows that the model has achieved an accuracy of about 82.96% on the test dataset, which is a relatively good performance.

```
image_index = 0
print(y_test[image_index])

plt.imshow(x_test[image_index].reshape(28, 28),cmap='Greys')
plt.show()
pred = model.predict(x_test[image_index].reshape(1, 28, 28, 1))
print(pred.argmax())
```

```
21
```



```
21
```

Figure 35: Inspect a single image and compare it to the model's predicted label

This code allows you to visually inspect a single image from the test dataset, display its true label, and compare it to the model's predicted label.

4.4.2 The architecture of the Convolutional Neural Network (CNN) model

The flowchart provided above outlines the architecture of the Convolutional Neural Network (CNN) model used for image classification. Here is a brief description of each layer in the model:

1. **Input**: The input layer accepts grayscale images of size 28x28x1. This is the starting point for the network, where raw image data is fed into the model.

2. **Conv2D**: The first convolutional layer uses 28 filters and a kernel size of 3x3 to extract features from the input image. This layer helps the model identify patterns such as edges, corners, and textures present in the images.

3. **MaxPooling2D**: The max pooling layer with a pool size of 2x2 is used to reduce the spatial dimensions of the feature maps, which helps to minimize computational complexity and control overfitting.

4. **Flatten**: The flatten layer is responsible for converting the 2D feature maps into a 1D vector. This transformation is necessary for connecting the convolutional layers to the fully connected layers.

5. **Dense**: The first fully connected layer has 128 neurons and utilizes a ReLU (Rectified Linear Unit) activation function. This layer enables the model to learn non-linear relationships and complex patterns in the data.

6. **Dropout**: A dropout layer with a rate of 0.2 is included to prevent overfitting. During training, this layer randomly drops out (i.e., sets to zero) a fraction of the neurons, making the model more robust and less reliant on any single neuron.

7. **Dense**: The final dense layer acts as the output layer, consisting of 39 neurons and a Softplus activation function. It produces the predicted class probabilities for each of the categories in the dataset.

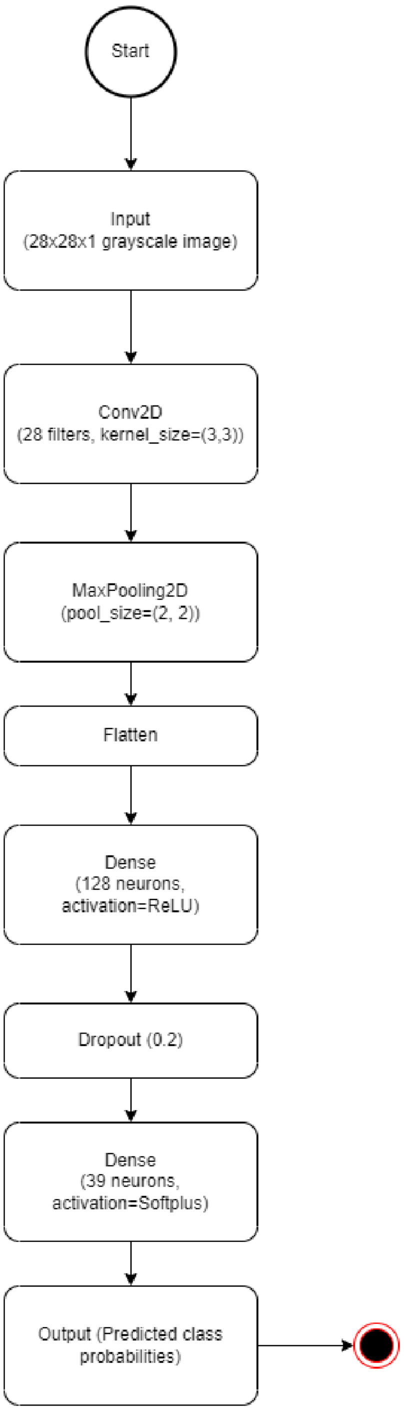


Figure 36: Architecture of CNN

The model is compiled with the Adam optimizer, sparse categorical crossentropy loss function, and accuracy metric. The training process consists of 10 epochs, during which the model learns to recognize and classify the different image categories.

4.4.3 Classification accuracy achieved by the CNN model

The chart presented here displays the classification accuracy achieved by the CNN model for each category of data:

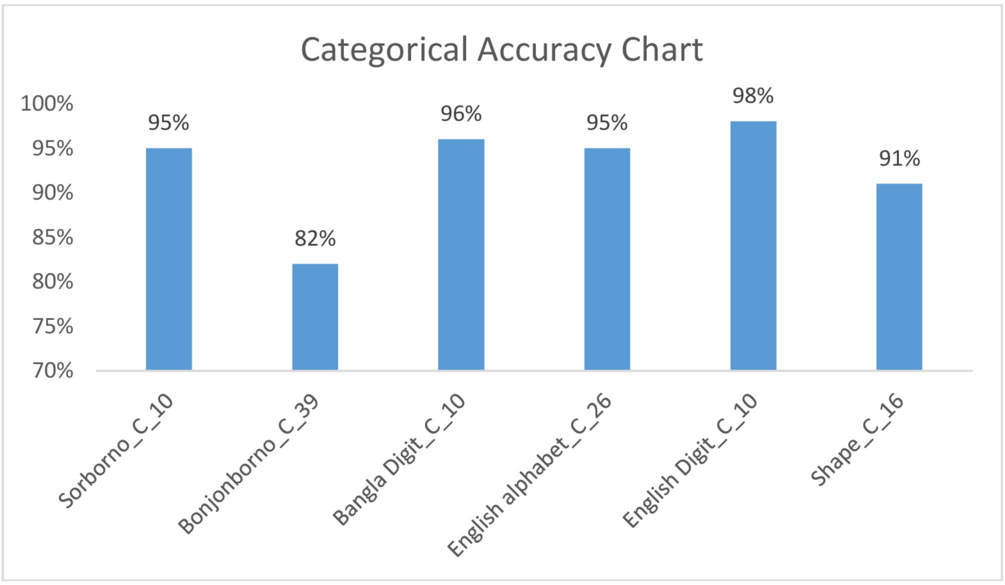


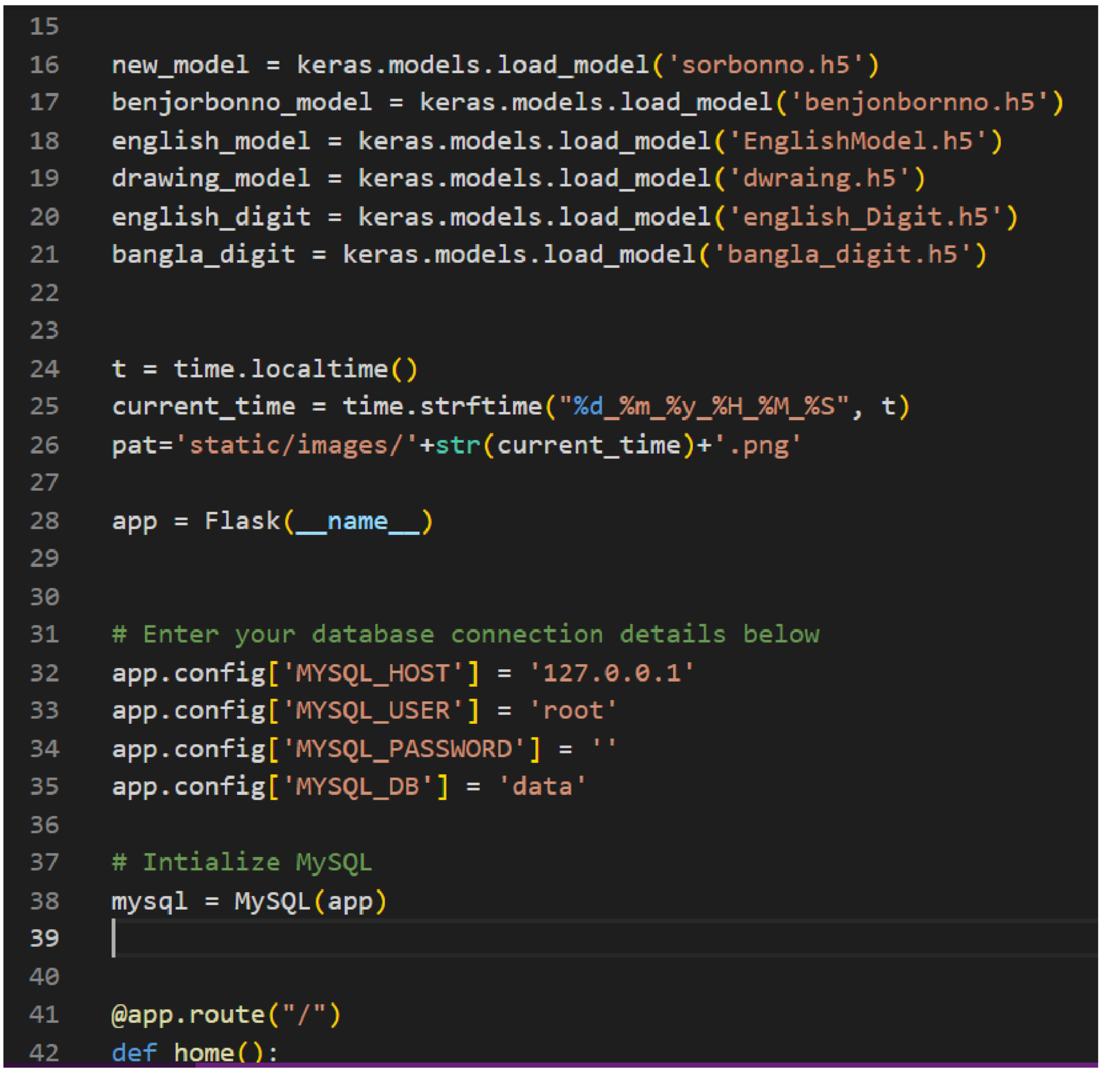
Figure 37: Categorical Accuracy Chart

The model has shown impressive performance across all categories, with the majority of them achieving over 90% accuracy. The Bonjonborno_C_39 category has the lowest accuracy at 82%, but this is still considered a reasonably good performance. Overall, the CNN model has demonstrated its effectiveness in classifying the different categories of data.

4.5 Implementing user interface.

4.5.1 Using Flask

This code is for a Flask web application that provides an interface for users to interact with various image classification models. The application has several routes for different classification tasks such as Bangla Sorbonno, Bangla Benjonbonno, English alphabets, English numbers, Bangla numbers, and drawings.



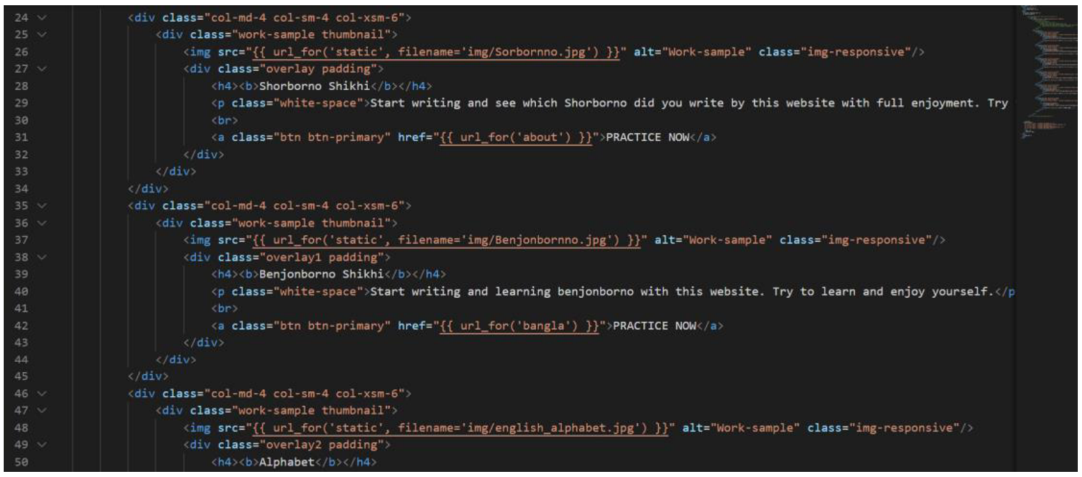
```python

new_model = keras.models.load_model('sorbonno.h5')
benjorbonno_model = keras.models.load_model('benjonbornno.h5')
english_model = keras.models.load_model('EnglishModel.h5')
drawing_model = keras.models.load_model('dwraing.h5')
english_digit = keras.models.load_model('english_Digit.h5')
bangla_digit = keras.models.load_model('bangla_digit.h5')


t = time.localtime()
current_time = time.strftime("%d_%m_%y_%H_%M_%S", t)
pat='static/images/'+str(current_time)+'.png'

app = Flask(__name__)


# Enter your database connection details below
app.config['MYSQL_HOST'] = '127.0.0.1'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'data'

# Intialize MySQL
mysql = MySQL(app)


@app.route("/")
def home():
```

Figure 38: Flask Code Snapshot

The code imports the necessary libraries, loads the pre-trained models, and defines the Flask application routes. The application has different pages for each of the classification tasks, where users can draw images that will be classified by the corresponding model. The application saves the drawn images and makes predictions using the appropriate model.

4.5.2 Using HTML, CSS, Bootstrap and JavaScript to designing the templates

This is an HTML template file for the home page of a website called "Hello Kid's". The page is designed to display a list of practice categories with images, titles, and descriptions. Users can click on the "PRACTICE NOW" button under each category to start practicing.



```html
<div class="col-md-4 col-sm-4 col-xsm-6">
    <div class="work-sample thumbnail">
        <img src="{{ url_for('static', filename='img/Sorbornno.jpg') }}" alt="Work-sample" class="img-responsive"/>
        <div class="overlay padding">
            <h4><b>Shorborno Shikhi</b></h4>
            <p class="white-space">Start writing and see which Shorborno did you write by this website with full enjoyment. Try
            <br>
            <a class="btn btn-primary" href="{{ url_for('about') }}">PRACTICE NOW</a>
        </div>
    </div>
</div>
<div class="col-md-4 col-sm-4 col-xsm-6">
    <div class="work-sample thumbnail">
        <img src="{{ url_for('static', filename='img/Benjonbornno.jpg') }}" alt="Work-sample" class="img-responsive"/>
        <div class="overlay1 padding">
            <h4><b>Benjonborno Shikhi</b></h4>
            <p class="white-space">Start writing and learning benjonborno with this website. Try to learn and enjoy yourself.</p
            <br>
            <a class="btn btn-primary" href="{{ url_for('bangla') }}">PRACTICE NOW</a>
        </div>
    </div>
</div>
<div class="col-md-4 col-sm-4 col-xsm-6">
    <div class="work-sample thumbnail">
        <img src="{{ url_for('static', filename='img/english_alphabet.jpg') }}" alt="Work-sample" class="img-responsive"/>
        <div class="overlay2 padding">
            <h4><b>Alphabet</b></h4>
```

Figure 39: The snapshot of template design code

The file uses the Jinja2 templating engine for rendering dynamic content in the Flask web application. It extends a "template.html" file, which should include the common structure and layout for all pages on the website, such as the header and footer.

The main content of the page is organized into a grid of six categories: Shorborno Shikhi, Benjonborno Shikhi, Alphabet, Drawing, English Digit, and Bangla Digit. Each category is represented by a thumbnail image with an overlay containing the title, description, and a "PRACTICE NOW" button.

5 Results and Discussion

After investing countless hours and tireless effort, I have successfully developed a system that has exceeded all expectations. Upon visiting children both at their homes and schools, I observed their immense attraction to the website. It captivated them as if they were playing a video game. The platform's drawing feature and instantaneous results piqued their curiosity and fueled their enthusiasm.

The students from various schools clamored for the opportunity to try this innovative platform. They discovered that practicing on the website was far more enjoyable than writing in their traditional exercise books. The auditory feedback, in which they could hear the sounds corresponding to their written work, further heightened their engagement. Eagerly, they delved into each topic available in the menu bar, one after another, propelled by their newfound excitement for learning.

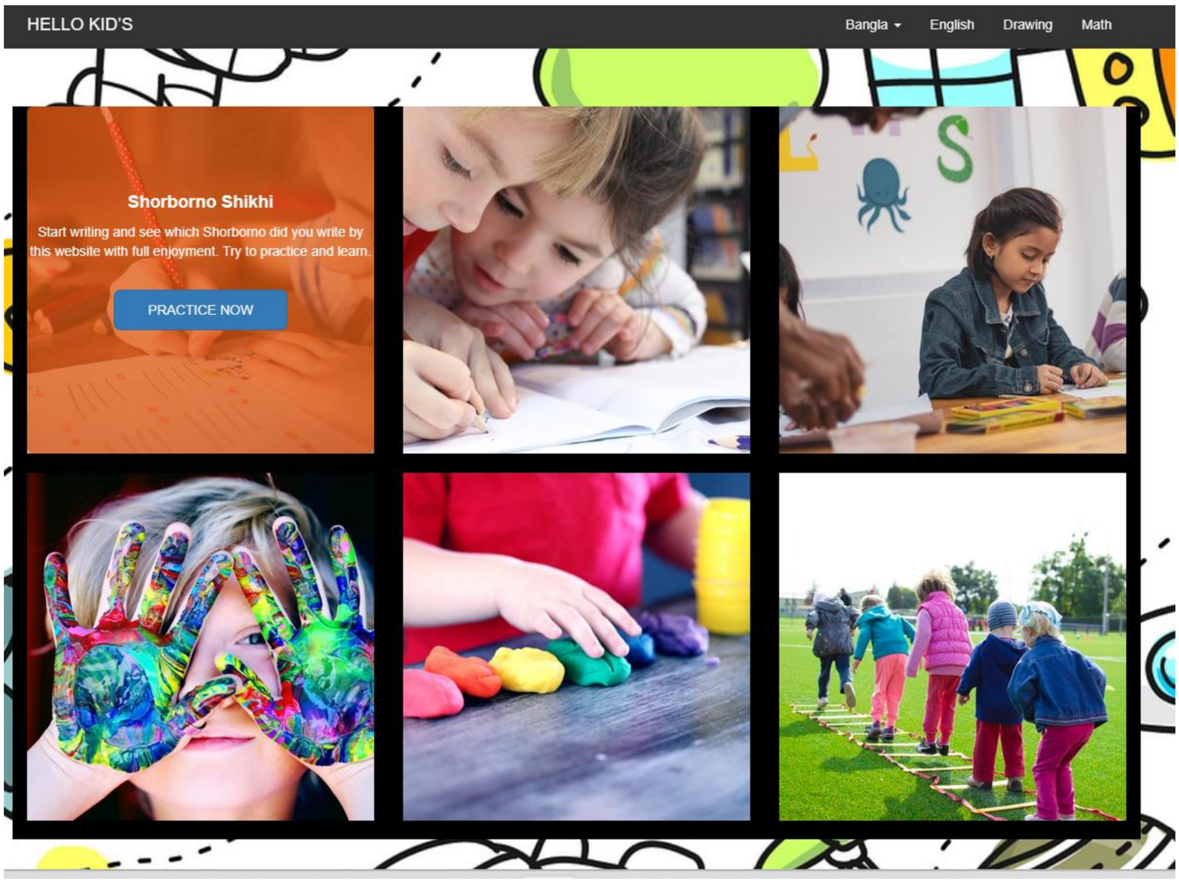
As the children continued to explore and engage with the platform, it became apparent that the website was fostering a genuine love for learning. The innovative approach to education seemed to tap into their innate curiosity, encouraging them to practice and master new skills with a sense of joy and wonder. Teachers and parents alike noticed a positive shift in the children's attitude towards learning, as well as significant improvement in their academic performance.

Furthermore, the website's interactive and immersive nature inspired the children to learn collaboratively. They shared their discoveries, challenges, and achievements with each other, creating a supportive and dynamic learning community. This collaborative environment not only fostered a strong sense of camaraderie among the students, but also helped them develop essential communication and teamwork skills.

Ultimately, the success of this ground-breaking platform lies in its ability to create a captivating and enriching educational experience for children. By seamlessly blending learning with entertainment, it has transformed traditional education into a delightful adventure, unlocking the unlimited potential of young minds and inspiring them to reach for the stars.

5.1 Home Page

Welcome to the vibrant and intuitive homepage of our innovative learning system. Here, children are presented with an array of engaging topics, each accompanied by a visually appealing and descriptive image. This user-friendly design allows young learners to effortlessly navigate through the platform and select the subject matter that sparks their interest, inviting them to embark on a captivating educational journey.



Start writing and see which Shorborno did you write by this website with full enjoyment. Try to practice and learn.

Figure 40: User Interface Home Page

5.2 Page Selection

In this illustrative example, a single topic is carefully chosen from the diverse range of subjects available. Upon selection, learners are seamlessly guided to the dedicated practice page for their chosen topic. In this particular case, "English Digits" has been thoughtfully picked, providing an opportunity for young minds to delve into the fascinating world of numbers and hone their skills.

English Digit
Start learning English digit and count anything as you want. Try to learn this and have fun with practice.
PRACTICE NOW

Figure 41: Selecting Page "English Digit"

5.3 Sorborno (স্বরবর্ণ) Page

This dedicated practice page is specifically designed for mastering Bangla Sorborno. Learners can freely write any Sorborno they wish, exploring and experimenting with the language. Should they desire to erase their work and start anew, the conveniently placed "Clear" button provides an effortless way to remove any written content, enabling a fresh and unblemished canvas for continuous learning.

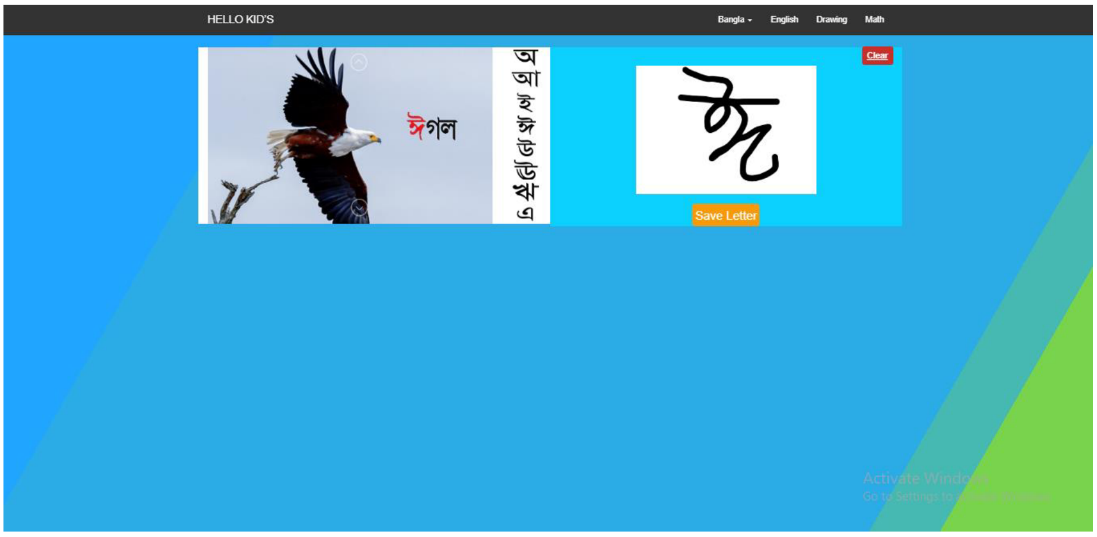


Figure 42: Sorborno Page

Upon submitting their work, learners will be presented with the results, allowing them to evaluate the accuracy of their intended letter. This immediate feedback empowers them to refine their skills and knowledge, fostering continuous improvement and growth in their learning journey.

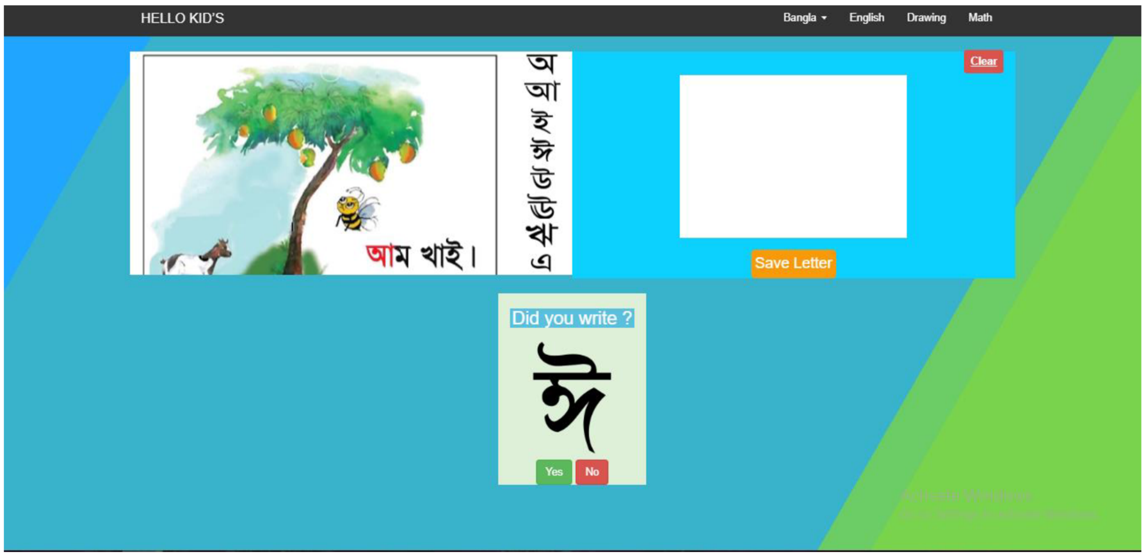


Figure 43: Result View

5.4 Bonjonborno (ব্যঞ্জনবর্ণ) Page

Welcome to the Bonjonborno page, where children have the opportunity to explore, learn, and practice a wide range of Bonjonborno characters. This engaging and interactive platform encourages curiosity and mastery, offering a comprehensive learning experience for young minds.

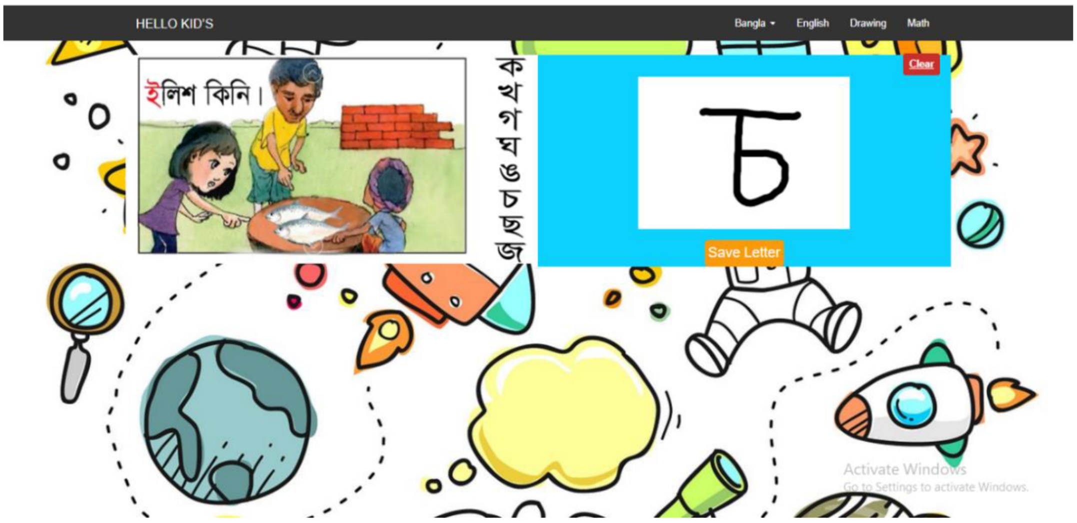


Figure 44: Bonjonborno Page

5.5 English Alphabet Page

Discover the English alphabet page, an all-encompassing platform designed to facilitate the learning and practice of each letter. Children can immerse themselves in this interactive environment, gaining immediate feedback on their progress and honing their skills with confidence and clarity.

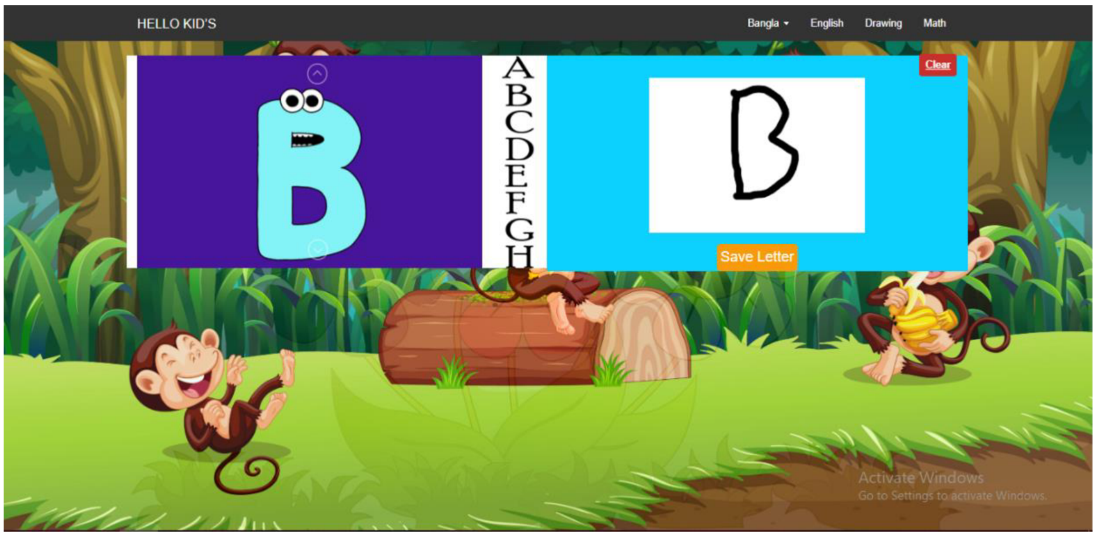


Figure 45: English Alphabet Page

5.6 English Digit Page

Children can engage in writing and practicing English digits, covering numbers from 0 to 9. Should any mistakes occur, the user-friendly platform allows for effortless erasure and amendment, promoting a smooth and enjoyable learning journey.

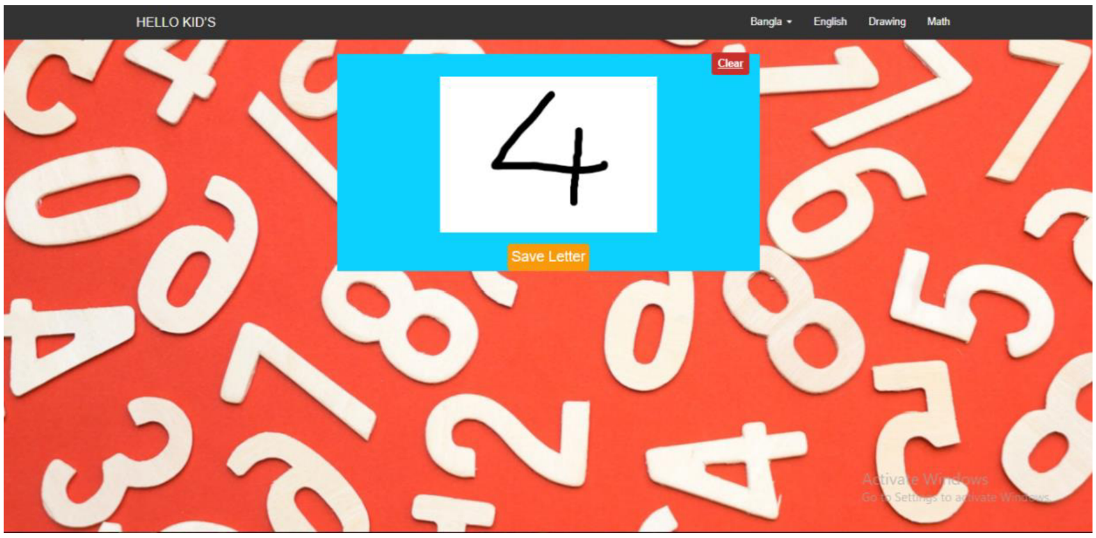


Figure 46: English Digit

5.7 Drawing Page

This engaging page is particularly captivating for young minds, allowing them to unleash their creativity through drawing and immediately witnessing the results. Additionally, they can delight in the auditory feedback provided by the platform, as it verbalizes their written work.

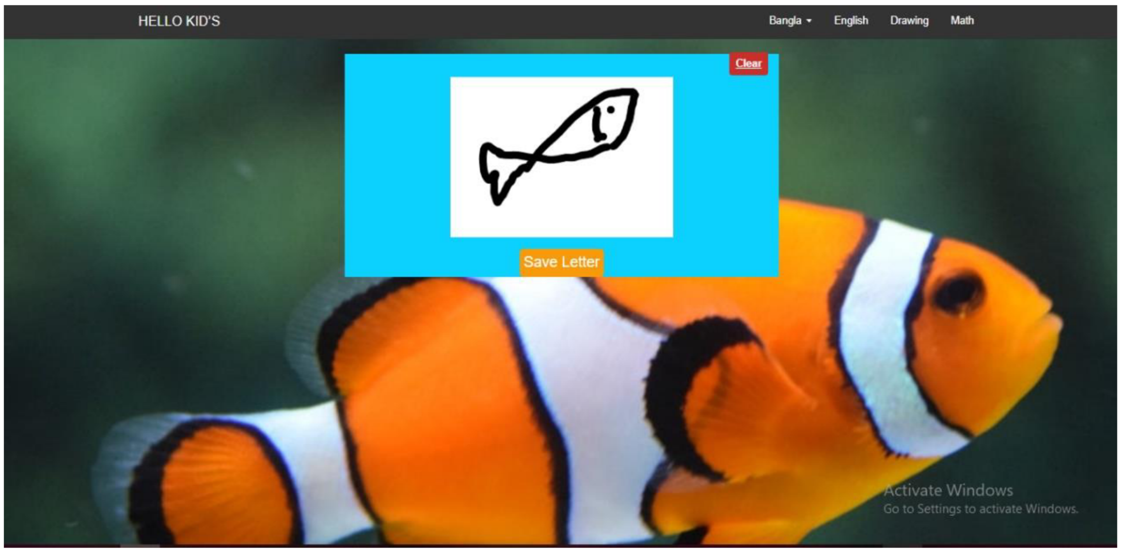


Figure 47: Drawing Page

6 Conclusion

In conclusion, this innovative educational platform has been meticulously designed to cater to the diverse learning needs of children. By integrating interactive and visually stimulating elements, it fosters an engaging environment that nurtures curiosity and excitement for learning. Furthermore, the system encourages creative expression and provides auditory feedback, both of which contribute to a comprehensive and immersive learning experience.

As I continue to refine and develop the platform, my focus will be on improving the accuracy of the model for Bonjonborno, ensuring that children receive the most accurate and helpful feedback possible. In addition to the current web-based platform, I am planning to expand my reach by developing applications for various platforms, such as Android, Apple, and Microsoft. These apps will not only be more accessible to children, but also offer a more attractive and user-friendly experience.

By making education feel like play, this ground-breaking website and its forthcoming applications are poised to revolutionize the way children learn and develop essential skills, fostering a lifelong love for learning and setting them on a path to success. Through constant innovation and improvement, I am committed to creating the most engaging and effective educational tools for the children of today and tomorrow.

References

1. **McKinsey & Company.** How artificial intelligence will impact K-12 teachers. *McKinsey & Company.* [Online] 01 2020. [Cited: 11 04, 2022.] https://www.mckinsey.com/industries/education/our-insights/how-artificial-intelligence-will-impact-k-12-teachers.
2. **Bangladesh Bureau of Statistics.** Multiple Indicator Cluster Survey 2012-2013, Final Report. [Online] 2015. https://www.bbs.gov.bd/site/page/4222a409-4d7c-4b5d-a8d8-9ae90aefb14d/MICS%20Final%20Report%202012-13%20English.
3. **Ministry of Primary and Mass Education.** National Curriculum and Textbook Board Pre-Primary Curriculum. [Online] 2013. [Cited: 05 10, 2022.] http://www.nctb.gov.bd/site/page/56324be4-3c4b-46fe-aab9-d552a5b9eb8b/-.
4. **Russell, S. J., & Norvig, P.** *Artificial intelligence: A modern approach.* s.l. : Pearson Education., 2010.
5. *Multi-task learning based data preprocessing for performance prediction of computer systems.* **Wang, Y., Wu, S., Wu, S., & Ye, Y.** s.l. : Journal of Supercomputing, 2001, Vols. 77(5), 4719-4738.
6. *A feature selection method based on mutual information and fuzzy clustering for sentiment analysis on social media.* **Nguyen, T. H., Duong, A. D., Nguyen, V. P., Le, T. T., & Huynh, V. N.** s.l. : Journal of Ambient Intelligence and Humanized Computing., 2021, Vols. 12(2), 1867-1876.
7. *Comparison of different machine learning algorithms for predicting the onset of type 2 diabetes.* **Shahin, A., Jafari, A., & Khan, S. A.** s.l. : Journal of Medical Systems,, 2021, Vols. 45(6), 1-8.
8. *Normalized pairwise margin: A novel evaluation metric for binary classification models.* **Xie, Y., Zhang, J., Chen, Q., & Xie, Z.** s.l. : IEEE Transactions on Neural Networks and Learning Systems,, 2021, Vols. 32(3), 1127-1139.
9. *Bayesian optimization for hyperparameter tuning in deep neural networks.* **Zhang, Y., Li, J., Li, Y., Li, D., & Li, B.** s.l. : Neurocomputing, 2021, Vols. 441, 87-101.
10. *Deep learning.* **LeCun, Y., Bengio, Y., & Hinton, G.** s.l. : Nature, 2015, Vols. 521(7553), 436-444.
11. **Alpaydin.** *Introduction to machine learning.* s.l. : MIT Press, 2010.
12. **Murphy, K. P.** *Machine learning: A probabilistic perspective.* s.l. : MIT Press, 2012.
13. **Goodfellow, I., Bengio, Y., & Courville, A.** *Deep learning.* 2016 : MIT Press.
14. **Sutton, R. S., & Barto, A. G.** *Reinforcement learning: An introduction.* 2018 : MIT Press.
15. **Géron, Aurélien.** *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2n Edition.* s.l. : O'Reilly Media, 2019.
16. **Saha, Sumit.** A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. *Towards Data Science.* [Online] 12 15, 2018. [Cited: 12 05, 2022.] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.
17. **Flask Pallets Projects.** Tutorial. [Online] [Cited: 02 12, 2022.] https://flask.palletsprojects.com/en/2.2.x/tutorial/.
18. **OpenCV.** OpenCV: OpenCV-Python Tutorials. [Online] [Cited: 04 01, 2022.] https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html.
19. **Pandas.** Pandas: User Guide. [Online] [Cited: 04 22, 2022.] https://pandas.pydata.org/docs/user_guide/index.html.

20. **NumPy.** NumPy: User Guide. [Online] [Cited: 04 11, 2022.] https://numpy.org/doc/stable/user/index.html.
21. **Tensor Flow.** Deep Convolutional Generative Adversarial Network. [Online] [Cited: 05 04, 2022.] https://www.tensorflow.org/tutorials/generative/dcgan.
22. **Scikit-learn.** User Guide. [Online] [Cited: 06 22, 2022.] https://scikit-learn.org/stable/user_guide.html.
23. **PATEL, SACHIN.** A-Z Handwritten Alphabets. [Online] [Cited: 03 02, 2022.] https://www.kaggle.com/datasets/sachinpatel21/az-handwritten-alphabets-in-csv-format.
24. **COLIANNI, STUART.** MNIST as .jpg. *Kaggle.* [Online] [Cited: 03 03, 2022.] https://www.kaggle.com/datasets/scolianni/mnistasjpg.
25. **Google Cloud .** quickdraw_dataset. [Online] [Cited: 03 12, 2022.] https://console.cloud.google.com/storage/browser/quickdraw_dataset/full/numpy_bitmap;tab=objects?pli=1&prefix=&forceOnObjectsSortingFiltering=false.
26. **Mendeley Data.** BanglaLekha-Isolated. [Online] [Cited: 03 20, 2022.] https://data.mendeley.com/datasets/hf6sf8zrkc/2#file-8a68156d-8a76-44d3-93e5-d14b61880526.

7 List of pictures and graphs