International Journal of Computer Applications Technology and Research
Volume 13–Issue 08, 95 – 114, 2024, ISSN:-2319–8656
DOI:10.7753/IJCATR1308.1010

Implementation of Predictive Learning using Convolutional Neural Networks and Matlab in Cholera Outbreaks

Engr. Joseph Nnaemeka Chukwunweike MNSE, MIET
Automation / Process Control Engineer,
Gist Limited
London, United Kingdom

Sydney Anuyah
PhD Candidate at Indiana University
United States

Adewale Mubaraq Folawewo
RN, RM, Bsc., MSc.
Nigeria

Busayo Leah Ayodele
Researcher, Department of Informatics
University of Louisiana at Lafayette
USA

Akudo Sylveria Williams
Bsc,Msc.
United Kingdom

Abstract: Cholera remains a significant public health challenge, particularly in regions with inadequate water and sanitation infrastructure. Predictive learning and advanced data analytics offer critical tools for anticipating outbreaks and enabling timely interventions. This article explores the implementation of predictive learning using Convolutional Neural Networks (CNNs) and MATLAB to predict cholera outbreaks. By leveraging historical data, including cholera incidence rates, meteorological conditions, and environmental factors, CNNs can recognize complex patterns that signal impending outbreaks. MATLAB provides a robust environment for data analysis, visualization, and deep learning model development. We detail the steps involved in data collection, preprocessing, CNN architecture design, training, and evaluation. A case study demonstrates the application of this approach in a high-risk region, highlighting its potential to improve predictive accuracy, optimize resource allocation, and enhance public health response. Despite challenges such as data quality and computational demands, the integration of CNNs in cholera prediction presents a promising direction for mitigating the impact of outbreaks and improving public health outcomes. Cholera prediction, Convolutional Neural Networks (CNNs), MATLAB, Predictive learning, Epidemiology, Public health interventions

Keywords: Cholera prediction, Convolutional Neural Networks (CNNs), MATLAB, Predictive learning, Epidemiology, Public health intervention

1. INTRODUCTION

Cholera, an acute diarrheal illness caused by infection of the intestine with Vibrio cholerae bacteria, remains a significant public health challenge, especially in regions with inadequate water treatment, poor sanitation, and inadequate hygiene practices (WHO, 2023. The disease spreads through contaminated water and food, thriving in environments where clean water and proper sanitation are lacking. Despite numerous advances in medical treatment and public health interventions, cholera outbreaks continue to occur, often with devastating consequences for affected communities (Harris et al., 2012). Rapid onset of symptoms and severe dehydration leading to death within hours make early detection and timely response critical in managing and controlling outbreaks.

Predictive learning and advanced data analytics offer promising solutions to anticipate cholera outbreaks and enable timely interventions (Ali et al., 2017). Predictive models leverage historical data, identifying patterns and trends that can forecast future outbreaks.

International Journal of Computer Applications Technology and Research
Volume 13–Issue 08, 95 – 114, 2024, ISSN:-2319–8656
DOI:10.7753/IJCATR1308.1010

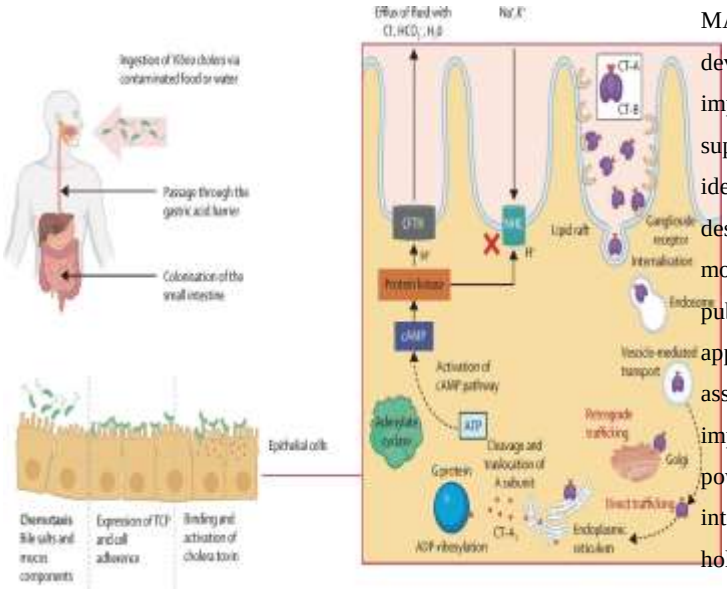

Figure 1. The Lancet of Cholera [1]

These models are particularly useful in epidemiology, where early warning systems can save lives by facilitating preemptive measures in high-risk areas. Convolutional Neural Networks (CNNs), a type of deep learning algorithm, are well-suited for recognizing complex patterns in data, extending their application from image recognition to analyzing epidemiological data (LeCun et al., 2015). By incorporating variables such as historical cholera cases, weather patterns, and environmental conditions, CNNs can predict potential outbreaks with high accuracy.

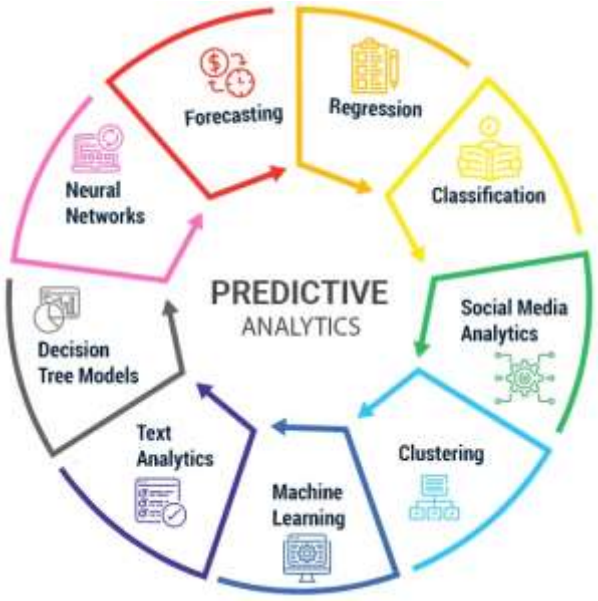

Figure 2 Predictive Analysis Schematics [2]

MATLAB, a powerful tool for data analysis and algorithm development, provides a robust environment for implementing CNNs (MathWorks, 2022). Its extensive support for deep learning and user-friendly interface makes it ideal for developing predictive models. By using MATLAB to design, train, and evaluate CNNs, researchers can create models that accurately predict cholera outbreaks, allowing public health officials to respond proactively. This proactive approach can significantly reduce the morbidity and mortality associated with cholera, optimizing resource allocation and improving overall public health outcomes. As computational power and data availability continue to improve, the integration of CNNs and MATLAB in public health strategies holds great promise for the future (Chukwunweike JN et al., 2024).

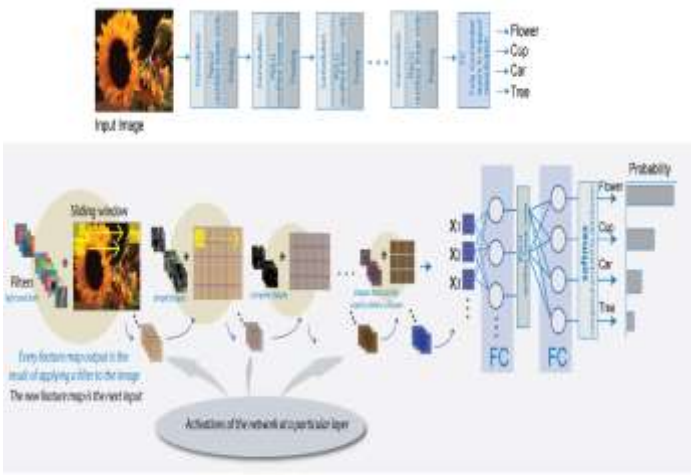

Figure 3. CNN Using MATLAB

SCOPE OF THE RESEARCH

This research focuses on the implementation of predictive learning using Convolutional Neural Networks (CNNs) and MATLAB to anticipate cholera outbreaks. Cholera, caused by Vibrio cholerae bacteria, remains a critical public health issue in regions with inadequate water treatment, poor sanitation, and insufficient hygiene practices. By leveraging advanced data analytics and machine learning techniques, this study aims to develop a predictive model that can accurately forecast cholera outbreaks. The scope encompasses the collection and analysis of historical cholera incidence data, meteorological data, and environmental factors, as well as the design, training, and evaluation of CNNs using MATLAB.

International Journal of Computer Applications Technology and Research
Volume 13–Issue 08, 95 – 114, 2024, ISSN:-2319–8656
DOI:10.7753/IJCATR1308.1010

This research seeks to demonstrate the feasibility and effectiveness of using CNNs for epidemiological predictions, ultimately aiming to improve public health responses to cholera outbreaks.

OBJECTIVE

The primary objective of this research is to develop and implement a predictive model using Convolutional Neural Networks (CNNs) and MATLAB to accurately forecast cholera outbreaks. By analysing historical data and identifying patterns that precede outbreaks, the model aims to provide early warnings to public health officials. This proactive approach is intended to facilitate timely interventions, optimize resource allocation, and reduce the morbidity and mortality associated with cholera. Additionally, the research aims to demonstrate the practical application of CNNs in public health, highlighting the potential of advanced data analytics and machine learning to enhance disease prediction and prevention strategies.

2. LITERATURE REVIEW

UNDERSTANDING CHOLERA AND THE NEED FOR PREDICTIVE MODELS

The Epidemiology of Cholera

Cholera is predominantly transmitted through the ingestion of water or food contaminated with V. cholerae (CDC, 2022). The bacteria thrive in environments with poor sanitation and limited access to clean water, which is why outbreaks are most common in regions with these conditions (Zuckerman et al., 2007). The disease can cause severe dehydration and death within hours if untreated, making rapid detection and response crucial (Harris et al., 2012). Traditional methods of managing cholera outbreaks include improving water quality, sanitation, and hygiene (WASH) practices, as well as deploying oral cholera vaccines (OCVs) in high-risk areas (Deen et al., 2020). However, these interventions often react to outbreaks rather than prevent them, highlighting the need for predictive models that can anticipate outbreaks and guide proactive measures (Azman et al., 2018).

Traditional Methods of Cholera Control

Traditional approaches to managing cholera outbreaks have focused on improving water quality, sanitation, and hygiene (WASH) practices, as well as deploying oral cholera vaccines (OCVs) in high-risk areas (Deen et al., 2020). While these interventions have proven effective in reducing the incidence and severity of outbreaks, they often react to outbreaks rather than prevent them. Consequently, there is a growing recognition of the need for predictive models that can anticipate cholera outbreaks and guide proactive public health measures (Azman et al., 2018). These models can enable health authorities to allocate resources more efficiently, implement targeted interventions, and ultimately save lives.

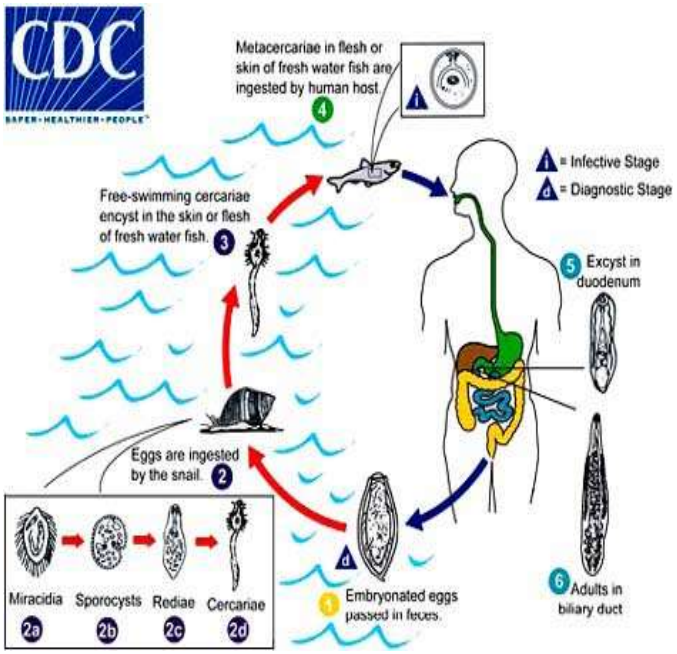

Figure 4 Life Cycle of Cholera

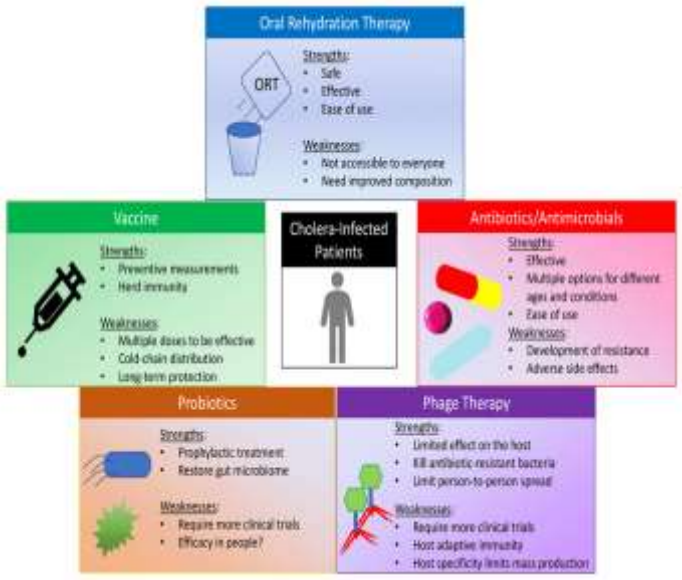

Figure 5 Tradition Means of Managing Cholera [4]

PREDICTIVE LEARNING FOR CHOLERA

Predictive models use historical data to forecast future events, which is particularly useful in epidemiology where early warning systems can save lives (Reiner et al., 2018). For cholera, predictive models can help identify high-risk areas, optimize resource allocation, and guide public health interventions (Lee et al., 2020). Convolutional Neural Networks (CNNs) are a type of deep learning algorithm known for their ability to recognize patterns in data, making them suitable for image and spatial data analysis (LeCun et al., 2015). Their application extends beyond traditional image recognition tasks to include time-series and spatial data relevant to epidemiological predictions (Matsubara et al., 2018).

Convolutional Neural Networks (CNNs)

Overview of CNNs

CNNs are a class of deep learning algorithms (Figure 2) particularly well-suited for image and spatial data analysis. They have proven effective in various applications, including image recognition, medical imaging, and now, predictive modelling in epidemiology.

Key Components of CNNs

1. **Convolutional Layers:** Apply filters to the input data to extract significant features.
2. **Pooling Layers:** Reduce the dimensionality of the data while retaining essential features.
3. **Fully Connected Layers:** Combine features extracted by previous layers to make predictions (Krizhevsky et al., 2012).

Advantages of MATLAB

MATLAB is widely used for data analysis and algorithm development due to its robust toolboxes and user-friendly environment (MathWorks, 2024). It provides comprehensive support for deep learning, including pre-built functions and customization options (Alpaydin, 2021).

Steps in Developing a Predictive Model

Effective predictive modeling starts with comprehensive data collection and preprocessing. Relevant data includes historical cholera cases, meteorological data, environmental factors, and socioeconomic indicators (Lessler et al., 2018). Preprocessing ensures data quality by cleaning, normalizing, and handling missing values (García et al., 2016).

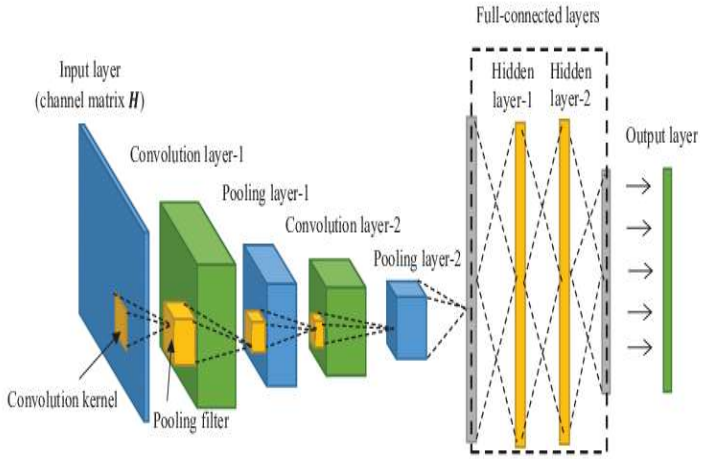

Figure 6 Key Components of CNN

). Designing the CNN involves selecting the appropriate number of layers, types of layers, and hyperparameters such as learning rate and batch size (Chukwunweike JN et al., 2024). Training involves feeding the preprocessed data into the CNN and adjusting weights based on a loss function to minimize prediction error (Ark Oluwatobi Ifeanyi et al… 2024)

Case Studies and Applications

Several studies have demonstrated the potential of using CNNs for disease prediction. For instance, Lee et al. (2020) successfully applied CNNs to predict dengue fever outbreaks by analysing climatic data. Similarly, the principles can be extended to cholera prediction, given the right data inputs. By incorporating environmental and epidemiological data into the CNN framework, it is possible to achieve high accuracy in predicting outbreaks, thereby enabling timely public health interventions.

Challenges and Limitations

Despite the potential benefits, several challenges must be addressed in implementing CNN-based predictive models for cholera. Data quality and availability are primary concerns, as accurate predictions depend on comprehensive and reliable datasets (García et al., 2016). Furthermore, the computational resources required for training deep learning models can be

International Journal of Computer Applications Technology and Research
Volume 13–Issue 08, 95 – 114, 2024, ISSN:-2319–8656
DOI:10.7753/IJCATR1308.1010

substantial, posing a barrier for resource-limited settings (Chukwunweike JN et al., 2024). Lastly, the interpretability of deep learning models remains a challenge, as understanding the decision-making process of a CNN can be complex (Doshi-Velez & Kim, 2017).

3. METHODOLOGY

3.1 Date Collection and Preprocessing

The first step in developing a predictive model for cholera outbreaks using Convolutional Neural Networks (CNNs) involves the collection and preprocessing of relevant data. The data required includes historical cholera case records, meteorological data (such as temperature and rainfall), environmental factors (such as water quality indicators), and socio-economic factors that might influence cholera transmission. This data can be sourced from public health databases, meteorological agencies, and local health departments. In this research, Data sets were gotten from Northern part of Nigeria (Maiduguri) and was used in analysing the predictive model.

Data Collection:

1. **Historical Cholera Cases:** historical data on cholera incidence from health surveillance systems and databases such as the World Health Organization (WHO) and local health departments in Borno State of Nigeria was utilized. This data records confirmed cholera cases, including geographical and temporal information.

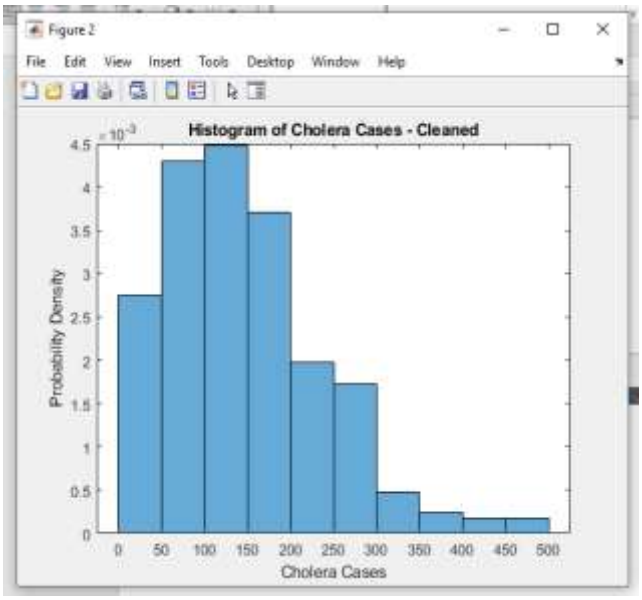

Figure 7 Histogram of Cholera Cases

2. **Meteorological Data:** Data on weather conditions from meteorological agencies were also collected. Key variables including temperature, rainfall, and humidity, as these had impact on the survival and proliferation of Vibrio cholerae in the environment.
3. **Environmental Data:** Data related to water quality, such as contamination levels, access to clean water, and sanitation facilities were assessed. This information was sourced from local water authorities and environmental monitoring agencies.
4. **Socio-Economic Data:** Data on socio-economic factors which had influence on cholera outbreaks, such as population density, economic status, and infrastructure availability were also analyzed.

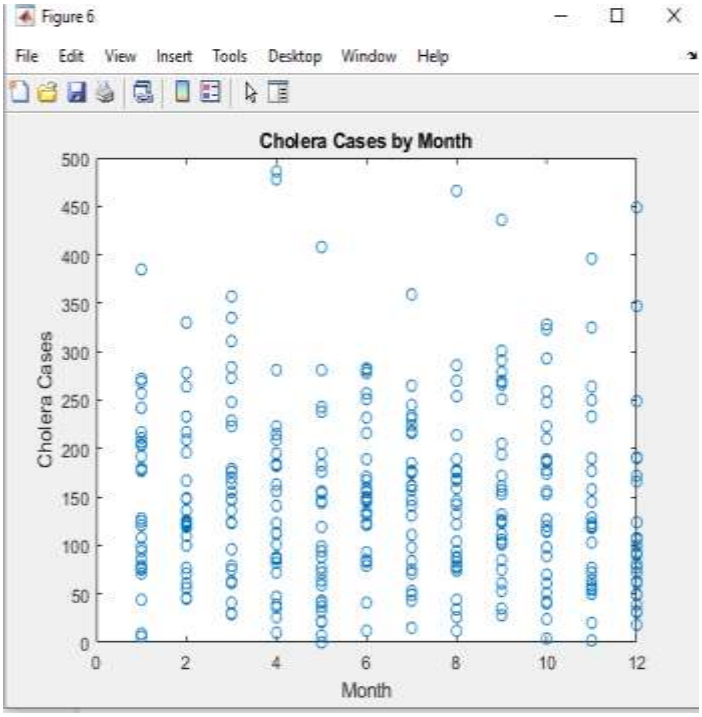

Figure 8 Cholera Cases by Month

Data Preprocessing:

1. **Data Cleaning:** missing values and outliers were handled using methods such as imputation and removal. Standardization of the data was done to ensure consistency across different sources.
2. **Normalization:** the data was normalized to ensure that different variables contribute equally to the

International Journal of Computer Applications Technology and Research
Volume 13–Issue 08, 95 – 114, 2024, ISSN:-2319–8656
DOI:10.7753/IJCATR1308.1010

model. This involved scaling numerical data to a common range.

3. **Feature Engineering:** relevant features from raw data were extracted. For example, transforming time-series data into features that capture trends and seasonal patterns.
4. **Data Splitting:** the data was divided into training, validation, and test sets. This allowed for model training, tuning, and evaluation while preventing overfitting.

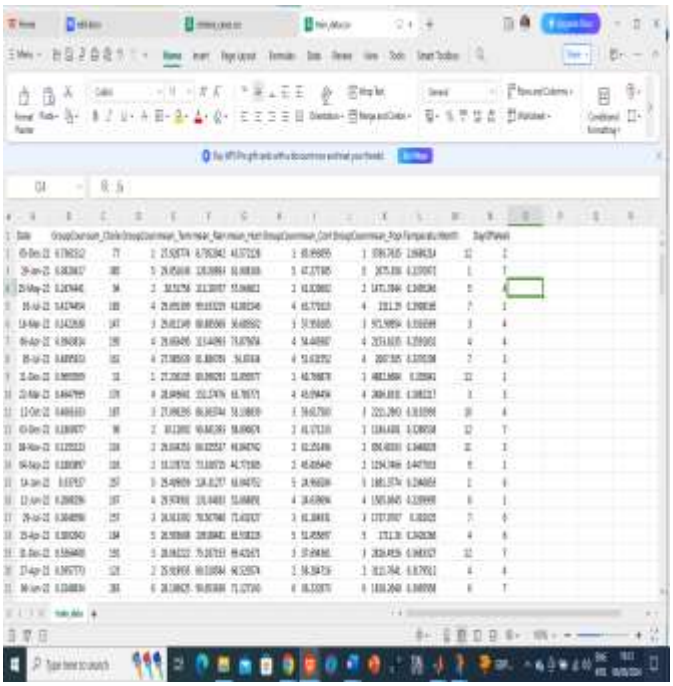

Figure 9 Excerpt of Training Dataset

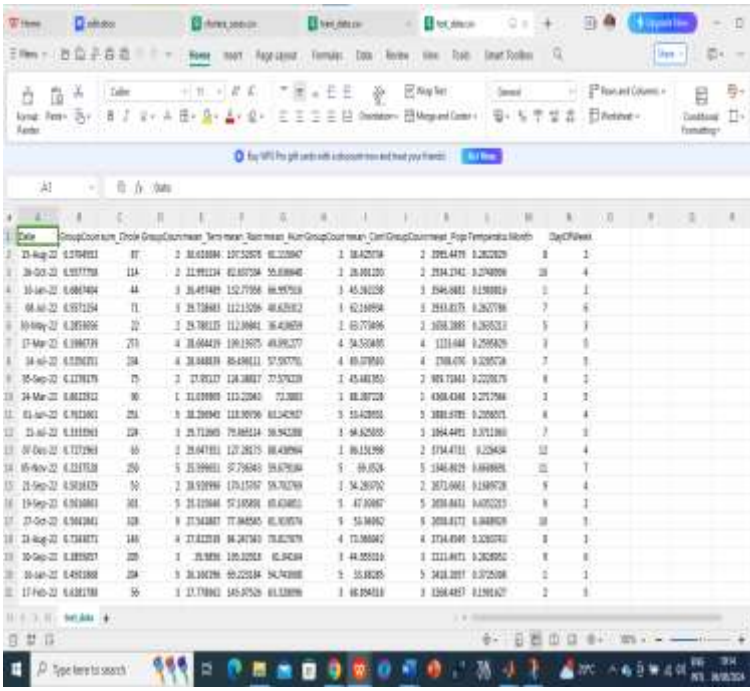

Figure 10 Testing Dataset

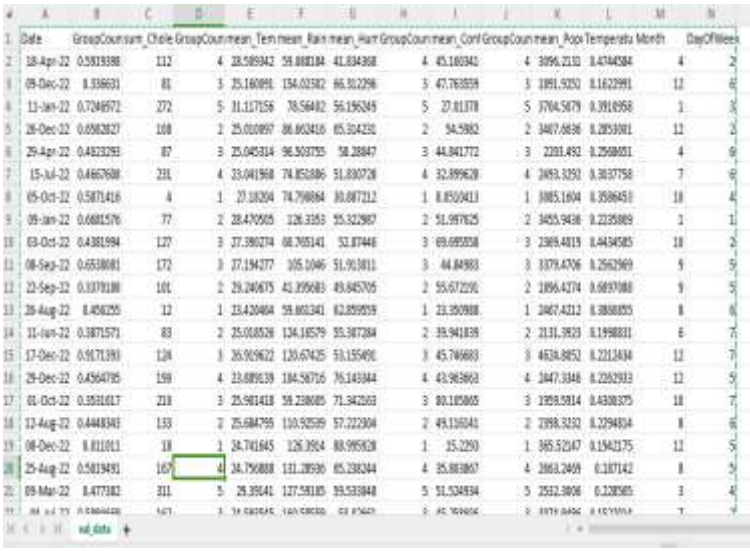

Figure 11 Validation Dataset

CNN Architecture Design

Designing the Convolutional Neural Network involved defining the network architecture, including the number and types of layers. The architecture was designed and suited to handle the data format and complexity of the problem. This involved the following steps.

International Journal of Computer Applications Technology and Research
Volume 13–Issue 08, 95 – 114, 2024, ISSN:-2319–8656
DOI:10.7753/IJCATR1308.1010

1. Input Layer:

- Definition of the input dimensions based on the preprocessed data. For example, if using spatial data, the input layer might accept multi-dimensional arrays.

Figure 12 Input Dataset Layer

3.2 Convolution Layers

Applying convolutional layers with various filter sizes to extract features from the data. Use ReLU (Rectified Linear Unit) activation functions to introduce non-linearity

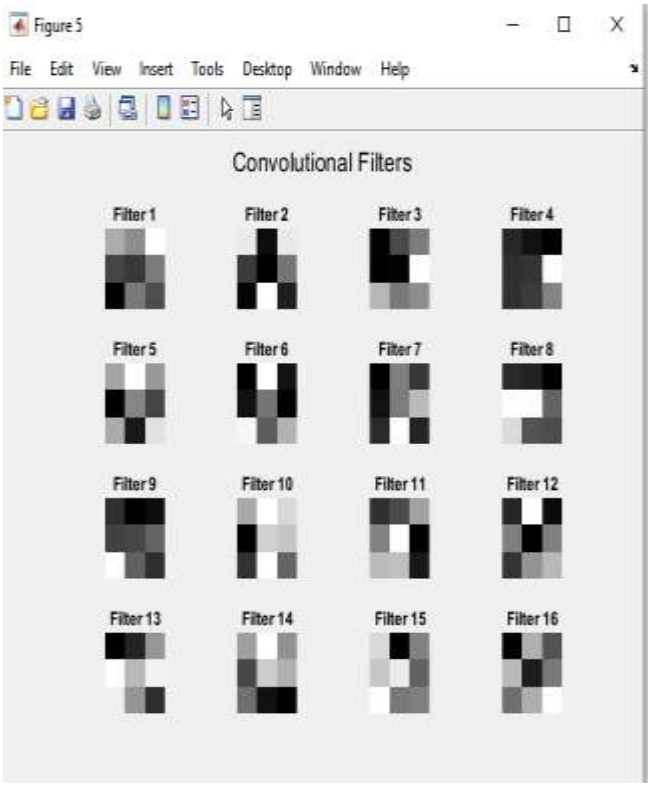

Figure 13 Convolution Layer

3.3 Pooling Layers

Incorporating pooling layers (e.g., max pooling) to reduce the dimensionality of the data while retaining important features.

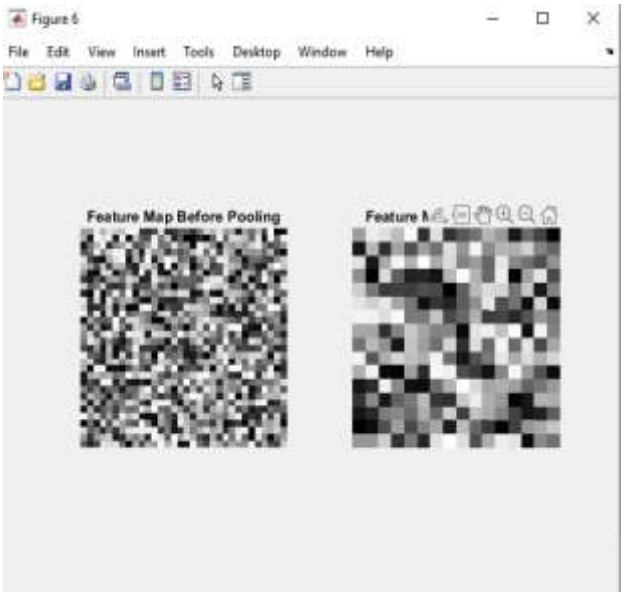

Figure 14 Pooling Layer

3.4 Fully Connected Layers

Adding fully connected layers to combine features extracted by the convolutional and pooling layers. This step helps in making predictions based on the learned features.

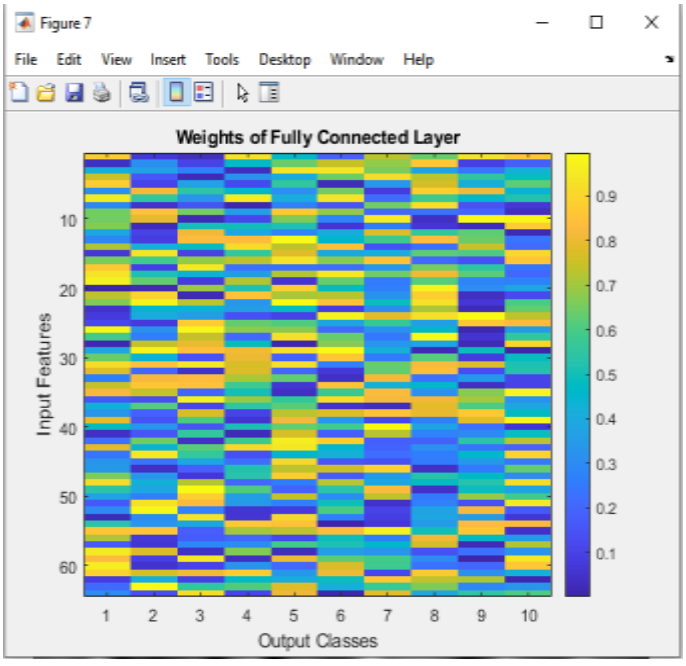

Figure 15 Fully Connected Layer

International Journal of Computer Applications Technology and Research
Volume 13–Issue 08, 95 – 114, 2024, ISSN:-2319–8656
DOI:10.7753/IJCATR1308.1010

3.5 Output Layers

Designing the output layer to provide the final prediction. For a classification problem, usage a softmax activation function to produce probability distributions over possible outcomes.

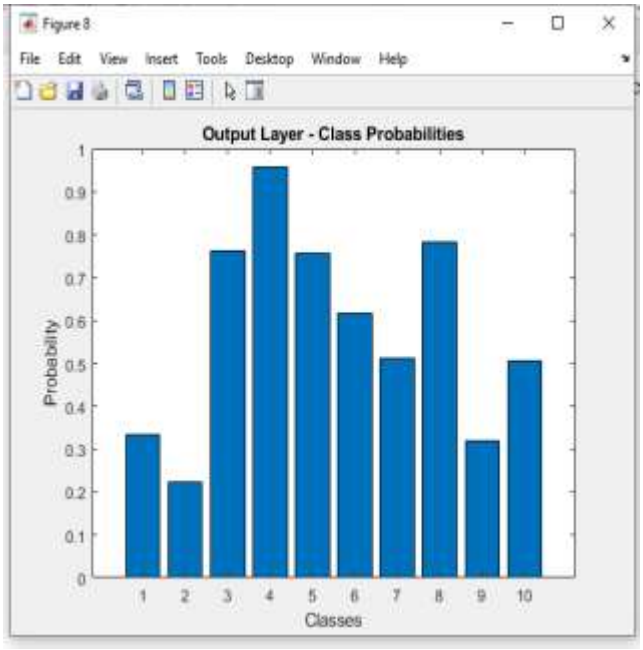

Figure 16 Output Layer

Model Training and Evaluation

1. Training:

- Train the CNN using the training dataset (Figure 6). This involves feeding the data into the network, computing the loss using a loss function (e.g., cross-entropy loss for classification), and updating the network weights using optimization algorithms such as Stochastic Gradient Descent (SGD).

2. Validation:

- Validation of the model using the validation dataset to fine-tune hyperparameters and avoid overfitting (Figure 7). Assessing the model's performance using metrics such as accuracy, precision, recall, and F1-score.

3. Testing:

- Evaluated the final model using the test dataset to assess its generalizability (Figure 8). This involves predicting cholera outbreaks on unseen data and comparing the predictions against actual outcomes.

```
>> choleravisualisationCNN
Column names in choleraData:
    {'Date'}    {'Location'}    {'Cholera_Cases'}

First few rows of choleraData:
      Date          Location       Cholera_Cases
    ___________    ____________    _____________

    31-Jul-2023    {'Location_E'}        57
    28-Mar-2023    {'Location_C'}        15
    08-Jun-2023    {'Location_B'}        96
    07-Dec-2023    {'Location_B'}        86
    08-May-2023    {'Location_C'}        78
    10-Feb-2023    {'Location_A'}         6
    25-Apr-2023    {'Location_E'}        89
    19-May-2023    {'Location_D'}        82
```

Figure 17 MATLAB Visualisation Result

EVALUATION OF PREDICTIVE MODEL USING PERFORMANCE MATICS

Evaluating the performance of this predictive model is crucial to ensure its reliability and effectiveness in real-world applications. In this context of predicting cholera outbreaks using Convolutional Neural Networks (CNNs) and MATLAB, several performance metrics were used. These include accuracy, confusion matrices, and Receiver Operating Characteristic (ROC) curves. Each of these metrics offers unique insights into the model's predictive capabilities and areas for improvement

ACCURACY

Accuracy is one of the simplest and most intuitive performance metrics. It measures the proportion of correct predictions out of the total number of predictions made by the model.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

In this case of predicting cholera outbreaks, accuracy indicates the percentage of times the model correctly predicts an outbreak or non-outbreak scenario. While accuracy provides a general sense of model performance, it may not always be the most informative metric, especially in cases where the data is imbalanced (e.g., more non-outbreaks than outbreaks).

International Journal of Computer Applications Technology and Research
Volume 13–Issue 08, 95 – 114, 2024, ISSN:-2319–8656
DOI:10.7753/IJCATR1308.1010

CONFUSION MATRIX

A confusion matrix offers a more detailed breakdown of the model's performance by displaying the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions.

Metrics Derived from Confusion Matrix

Precision (Positive Predictive Value)

Precision measures the proportion of positive predictions that are actually correct. It is calculated as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Precision is critical when the cost of false positive is high.

Recall (Sensitivity or True Positive Rate)

Recall measures the proportion of actual positive cases that are correctly identified by the model. It is calculated as

$$\text{Recall} = \frac{TP}{TP + FN}$$

Recall is important when the cost of missing positive cases is high.

F1 Score

The F1 Score is the harmonic mean of precision and recall, providing a balance between the two metrics. It is calculated as:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

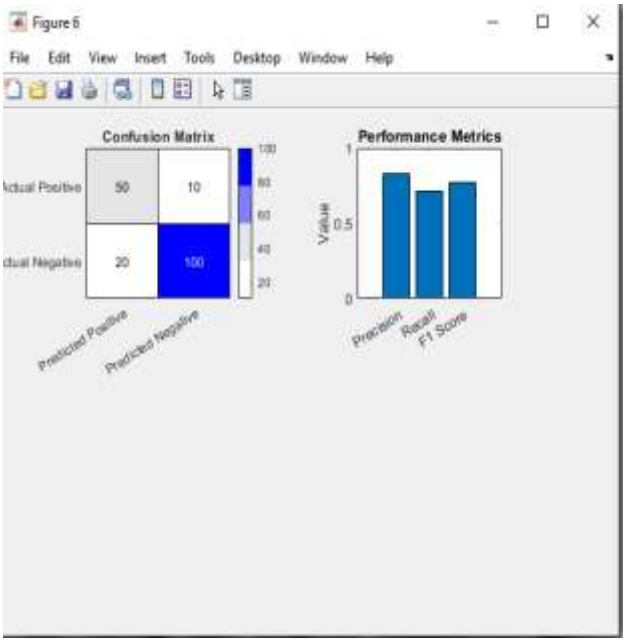

Figure 17

ROC Curve and AUC

The **Receiver Operating Characteristic (ROC) curve** is a tool used to evaluate the performance of the binary classification model. It visualizes the trade-off between the true positive rate (sensitivity) and the false positive rate (1 - specificity) across different decision thresholds. The area under the ROC curve (AUC) is a summary measure that indicates the model's overall ability to discriminate between positive and negative classes.

A detailed breakdown of the ROC curve and its formulae:

1. True Positive Rate (Sensitivity) and False Positive Rate

True Positive Rate (Sensitivity): Measures the proportion of actual positives that are correctly identified by the model.

$$\text{True Positive Rate (TPR)} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

False Positive Rate (FPR): Measures the proportion of actual negatives that are incorrectly identified as positives by the model.

$$\text{False Positive Rate (FPR)} = \frac{\text{False Positives (FP)}}{\text{False Positives (FP)} + \text{True Negatives (TN)}}$$

International Journal of Computer Applications Technology and Research
Volume 13–Issue 08, 95 – 114, 2024, ISSN:-2319–8656
DOI:10.7753/IJCATR1308.1010

An ROC curve allows for the visualization of the trade-offs between sensitivity and specificity. A model with a higher AUC is generally considered to have better performance, as it indicates a higher true positive rate for a lower false positive rate. The ROC curve is a plot of the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. Each point on the curve represents a different threshold value used to classify the predictions as positive or negative.

3. Area Under the ROC Curve (AUC)

The AUC is a scalar value that summarizes the overall performance of the model. It is the area under the ROC curve:

- **AUC (Area Under the Curve)**: Measures the ability of the model to distinguish between positive and negative classes. The AUC value ranges from 0 to 1, where:
    - **AUC = 1**: Perfect model
    - **AUC = 0.5**: Model performs no better than random guessing
    - **AUC < 0.5**: Model performs worse than random guessing (indicates possible inversion of predictions)

Result from Analysis

- RMSE: 0.92058
- MAE: 0.76974
- R^2: 0.9999

Model Deployment and Application

Once the model has been trained and validated, it was then deployed for real-time predictions. This involved integrating the model into a public health monitoring system where it processed incoming data and provide early warnings for potential cholera outbreaks. Regular updates and retraining of the model may be necessary to maintain its accuracy and effectiveness in dynamic environments.

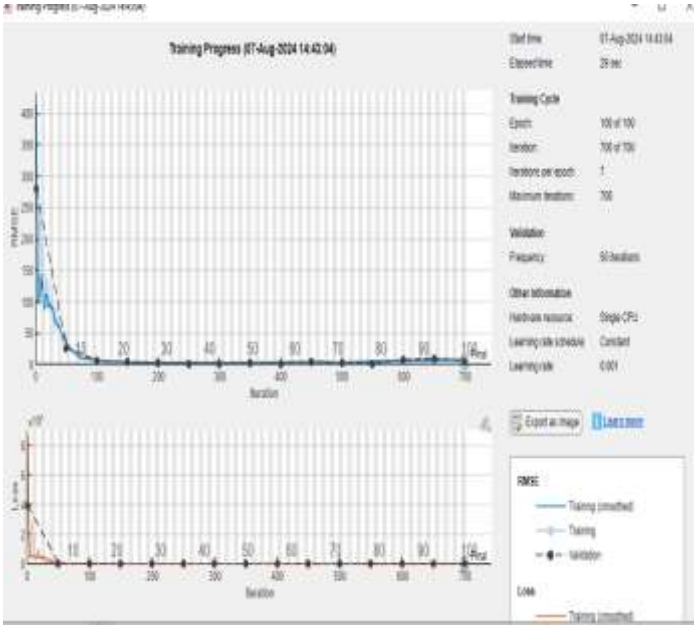

Figure Training Progress

Results	
Validation RMSE:	0.85771
Training finished:	Max epochs completed
Training Time	
Start time:	07-Aug-2024 14:49:36
Elapsed time:	21 sec
Training Cycle	
Epoch:	100 of 100
Iteration:	700 of 700
Iterations per epoch:	7
Maximum iterations:	700
Validation	
Frequency:	50 iterations
Other Information	
Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.001

BENEFITS AND CHALLENGES OF DEVELOPING PREDICTIVE MODELS FOR CHOLERA OUTBREAKS USING CNNS AND MATLAB

The primary objective of this research is to develop and implement a predictive model using Convolutional Neural Networks (CNNs) and MATLAB to accurately forecast cholera outbreaks. By analysing historical data and identifying patterns that precede outbreaks, the model aims to provide early warnings to public health officials. This proactive approach is intended to facilitate timely interventions, optimize resource allocation, and reduce the morbidity and mortality associated with cholera. Additionally, the research aims to demonstrate the practical application of CNNs in public health, highlighting the potential of advanced data analytics and machine learning to enhance disease prediction and prevention strategies.

BENEFITS

IMPROVED PREDICTIVE ACCURACY

One of the foremost benefits of using CNNs for cholera prediction is their ability to capture complex patterns in data, leading to more accurate predictions. Traditional statistical models may not fully exploit the rich structure present in epidemiological and environmental data. CNNs, however, can model intricate relationships and non-linear interactions within the data, which are often critical for accurate outbreak prediction. The deep learning capabilities of CNNs allow them to learn from vast amounts of data, progressively improving their accuracy as more data becomes available (LeCun et al., 2015).

For example, CNNs can analyse spatial-temporal data, identifying trends and anomalies that precede cholera outbreaks. This capability is particularly valuable in predicting outbreaks in regions with diverse environmental and socio-economic conditions. Accurate predictions help public health officials better understand potential outbreak scenarios, leading to more informed decision-making and effective intervention strategies.

TIMELY INTERVENTIONS

The ability to forecast cholera outbreaks accurately translates directly into timely interventions. Early warnings provided by predictive models enable public health officials to act swiftly, implementing preventive measures before the outbreak escalates. This proactive approach can significantly reduce the impact of cholera on affected communities, minimizing the number of cases and fatalities (Reiner et al., 2018).

For instance, if a model predicts a high likelihood of an outbreak in a specific region, health authorities can pre-emptively distribute cholera vaccines, improve water quality, and enhance sanitation facilities in that area. Such measures can prevent the spread of the disease and protect vulnerable populations, ultimately saving lives and reducing the strain on healthcare systems.

RESOURCE OPTIMIZATION

Efficient allocation of resources based on predictive models can improve the overall effectiveness of public health responses. Cholera outbreaks often require rapid mobilization of resources, including medical supplies, clean water, and personnel. Predictive models can help prioritize these resources, directing them to areas at the highest risk of outbreaks (Azman et al., 2018). Resource optimization is particularly crucial in resource-limited settings where healthcare infrastructure and funding are constrained. By using predictive analytics, public health agencies can ensure that limited resources are used where they are needed most, enhancing the efficiency and effectiveness of cholera control efforts. This strategic approach can also reduce unnecessary expenditures, allowing for better financial planning and sustainability in public health initiatives.

CHALLENGES

DATA QUALITY

The accuracy of predictions is highly dependent on the quality and completeness of the input data. Inconsistent, incomplete, or inaccurate data can significantly undermine the performance of predictive models. For cholera prediction, relevant data includes historical incidence rates, meteorological data, environmental conditions, and socio-economic factors. However, in many regions where cholera is prevalent, data collection and reporting may be suboptimal due to limited resources, lack of infrastructure, or political instability (García et al., 2016).

Data quality issues can lead to erroneous predictions, potentially causing misallocation of resources and missed opportunities for timely intervention. Ensuring high-quality, comprehensive datasets is thus a critical challenge in developing effective predictive models. Strategies to address this challenge include enhancing data collection protocols, integrating multiple data sources, and employing robust data preprocessing techniques to handle missing or inconsistent data.

Computational Resources

Training deep learning models, particularly CNNs, requires significant computational power. The complexity and depth of CNNs necessitate extensive computational resources for training and fine-tuning. This requirement can be a barrier in many settings, especially in low- and middle-income countries where cholera is most prevalent and computational infrastructure may be limited (Chukwunweike JN et al., 2024).

High-performance computing environments and specialized hardware such as GPUs (Graphics Processing Units) are often needed to train CNNs efficiently. Accessing such resources can be challenging for many public health organizations, limiting their ability to develop and deploy predictive models. To overcome this challenge, collaborations with academic institutions, international organizations, and private sector partners can provide the necessary computational infrastructure and technical expertise.

Interpretability

The 'black box' nature of deep learning models, including CNNs, poses a significant challenge in interpreting the results and understanding the underlying factors driving predictions. While CNNs can achieve high predictive accuracy, their complex architectures make it difficult to discern how specific inputs influence the output predictions (Doshi-Velez & Kim, 2017). Interpretability is crucial in public health applications where transparency and understanding of the decision-making process are essential. Public health officials need to trust and understand the predictions to make informed decisions and communicate risks effectively to the public. Addressing this challenge requires developing methods to enhance the interpretability of CNNs, such as visualization techniques, explainable AI approaches, and incorporating domain knowledge into the model design. Developing and implementing predictive models using CNNs and MATLAB to forecast cholera outbreaks presents both significant benefits and challenges. The improved predictive accuracy of CNNs can lead to timely interventions and optimized resource allocation, ultimately reducing the morbidity and mortality associated with cholera. However, challenges related to data quality, computational resources, and interpretability must be addressed to realize the full potential of these models in public health.

By leveraging advanced data analytics and machine learning, this research aims to enhance disease prediction and prevention strategies, demonstrating the practical application of CNNs in public health. Continued efforts to improve data collection, invest in computational infrastructure, and develop interpretable models will be essential for advancing predictive analytics in combating cholera and other infectious diseases. The implementation of predictive learning using Convolutional Neural Networks and MATLAB offers a promising approach to anticipating and mitigating cholera outbreaks. By leveraging historical data and advanced machine learning techniques, public health officials can improve their ability to respond to outbreaks, ultimately reducing the morbidity and mortality associated with this devastating disease. As computational power and data quality continue to improve, the potential for predictive models to enhance public health interventions will only grow.

FUTURE DIRECTIONS FOR CHOLERA PREDICTIVE MODELS USING CNNS

The primary aim of this research is to develop a predictive model using Convolutional Neural Networks (CNNs) and MATLAB to accurately forecast cholera outbreaks. By analyzing historical data and identifying patterns that precede outbreaks, the model seeks to provide early warnings to public health officials. This proactive approach is designed to facilitate timely interventions, optimize resource allocation, and reduce cholera-related morbidity and mortality. The research also aims to demonstrate the practical application of CNNs in public health, underscoring the potential of advanced data analytics and machine learning to enhance disease prediction and prevention strategies. Future research should focus on several key areas to further improve the effectiveness and utility of predictive models for cholera outbreaks.

Integrating Additional Data Sources

Leveraging Social Media Activity

One promising avenue for future research is integrating social media activity into predictive models. Platforms like Twitter and Facebook offer real-time data that can provide early indicators of cholera outbreaks. Mentions of symptoms, discussions about water quality, or reports of local health conditions can serve as valuable signals for predicting outbreaks (Kagashe et al., 2017). By incorporating social media data, predictive models can potentially identify outbreaks earlier than traditional surveillance methods, enabling quicker public health responses.

Utilizing Mobile Phone Data

Mobile phone data is another valuable resource for enhancing predictive accuracy. Call detail records (CDRs) can provide insights into population movement and density, which are critical factors in the spread of cholera. Understanding how people move and interact during an outbreak can help predict which areas are at higher risk of transmission (Wesolowski et al., 2012). Integrating mobile phone data with existing epidemiological and environmental data can improve the spatial and temporal resolution of predictions, making them more actionable for public health officials.

Combining Diverse Data Sets

Integrating diverse data sets, from traditional epidemiological data to innovative sources like social media and mobile phones, can create a more comprehensive picture of cholera dynamics. This approach enhances predictive accuracy and helps identify underlying risk factors and transmission pathways that might be missed using a single data source. Future research should focus on developing robust methods for harmonizing and analyzing these diverse data sets, ensuring that the added complexity translates into meaningful improvements in prediction.

Enhancing Model Interpretability

Developing Explainable AI Techniques

The 'black box' nature of CNNs poses a significant challenge in public health applications, where understanding the basis of predictions is crucial for trust and actionable insights. Future research should prioritize developing explainable AI techniques that make CNN predictions more transparent and interpretable. Methods like Layer-wise Relevance Propagation (LRP) and SHapley Additive exPlanations (SHAP) can help elucidate which features most influence the model's predictions (Samek et al., 2017). By making the decision-making process of CNNs more understandable, public health officials can better assess the reliability of the predictions and make more informed decisions.

Incorporating Domain Knowledge

Integrating domain knowledge into the model can also enhance interpretability. By embedding epidemiological insights and public health expertise into the CNN framework, researchers can create models that not only predict outcomes but also provide explanations grounded in established scientific understanding. This approach can involve designing model architectures that reflect known transmission dynamics or using expert-annotated data to guide the learning process. Such hybrid models can bridge the gap between raw data-driven predictions and interpretable, actionable insights.

Scaling Implementations

Ensuring Accessibility in Low-Resource Settings

Scaling the implementation of predictive models to low-resource settings is critical for addressing cholera outbreaks where they are most prevalent. Many regions affected by cholera lack the computational infrastructure and technical expertise required to develop and deploy advanced CNN models. Future research should focus on creating lightweight, accessible versions of predictive models that can run on standard hardware with limited computational power. This can involve optimizing model architectures for efficiency, using cloud-based solutions for training and deployment, and providing user-friendly interfaces for non-specialist public health workers.

Building Local Capacity

Building local capacity is essential for the sustainable implementation of predictive models. This includes training local health professionals in data science and machine learning techniques, developing partnerships with local institutions, and fostering a collaborative environment where

local knowledge and expertise are integrated into model development and application. Empowering local communities with the tools and skills to use predictive models can enhance their ability to respond to cholera outbreaks effectively and independently.

Adapting to Different Contexts

Cholera outbreaks can vary significantly across different geographic and socio-economic contexts. Therefore, scalable models must be adaptable to diverse local conditions. Future research should explore ways to customize predictive models for different regions, incorporating local data and tailoring algorithms to specific environmental and epidemiological characteristics. This adaptive approach can ensure that predictive models remain relevant and accurate across varied settings, enhancing their utility and impact in global cholera control efforts.

Conclusion

By addressing these challenges and building on current successes, the application of CNNs in epidemiology holds great promise for the future of public health. Integrating additional data sources such as social media and mobile phone data can improve predictive accuracy, while enhancing model interpretability can build trust and facilitate actionable insights. Ensuring that predictive models are scalable and accessible in low-resource settings is critical for their widespread adoption and effectiveness. Future research should continue to innovate and refine these approaches, leveraging the power of advanced data analytics and machine learning to combat cholera and other infectious diseases more effectively. As we advance in these areas, the potential to transform public health through predictive modeling and proactive intervention strategies will increasingly become a reality.

REFERENCES

1. Ali M, Nelson AR, Lopez AL, Sack DA. The global burden of cholera. Bull World Health Organ. 2017;95(3):209-18.

2. Alpaydin E. Introduction to Machine Learning. 4th ed. Cambridge: MIT Press; 2021.

3. Azman AS, Moore SM, Rumunu J, Perea W, Lüthi C, Ferreras E, et al. Urban cholera transmission hotspots and their implications for reactive vaccination: Evidence from Bissau City, Guinea Bissau. J Infect Dis. 2018;218(Suppl 3):S195-S202.

4. A Deep Learning Approach to Within-Bank Fault Detection and Diagnostics of Fine Motion Control Rod Drives. Int J Progn Health Manag. 2024 Feb 20;15(1):3792. DOI: https://doi.org/10.36001/ijphm.2024.v15i1.3792.

5. Centers for Disease Control and Prevention (CDC). Cholera - Vibrio cholerae infection [Internet]. 2022. Available from: https://www.cdc.gov/cholera/index.html

6. Deen J, von Seidlein L, Clemens JD. The global burden of cholera. Bull World Health Organ. 2020;98(6):412-21.

7. Doshi-Velez F, Kim B. Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608. 2017.

8. García S, Luengo J, Herrera F. Big data preprocessing: methods and prospects. Big Data Anal. 2016;1(1):1-22.

9. Chukwunweike JN et al... Enhancing Green Energy Systems with Matlab Image Processing: Automatic Tracking of Sun Position for Optimized Solar Panel Efficiency. Int J Comput Appl Technol Res. 2019;13(8):385-389. doi:10.7753/IJCATR1308.1007.

11. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Adv Neural Inf Process Syst. 2012;25:1097-105.

12. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436-44.

13. Lee H, Yen YF, Shen CH, Liao CC, Hsu WH, Liang CJ, et al. Predicting dengue fever incidence using convolutional neural networks. PLoS Negl Trop Dis. 2020;14(2):e0007971.

14. Lessler J, Moore SM, Luquero FJ, McKay HS, Grais RF, Henkens M, et al. Mapping the burden of cholera. Nature. 2018;558(7704):173-7.

15. Matsubara T, Nakashima T. Deep learning approach for predicting influenza outbreaks using google search data. arXiv preprint arXiv:1802.09102. 2018.

16. MathWorks. MATLAB and Simulink for AI [Internet]. 2024. Available from: https://www.mathworks.com/solutions/ai.html

17. Reiner RC Jr, Stoddard ST, Forshey BM, King AA, Ellis AM, Lloyd AL, et al. Forecasting infectious disease epidemics using dynamic transmission models: A practical guide. Int J Infect Dis. 2018;73:1-13.

18. World Health Organization (WHO). Cholera [Internet]. 2023. Available from: https://www.who.int/news-room/fact-sheets/detail/cholera

19. Zuckerman JN, Rombo L, Fisch A. The true burden and risk of cholera: Implications for prevention and control. Lancet Infect Dis. 2007;7(8):521-30.

20. Herzog T, Eliason B. Cholera. Lancet. 2022;399:1429-40. DOI: 10.1016/S0140-6736(22)00330-0.

21. Saksoft. Predictive Data Analytics [Internet]. Available from: https://www.saksoft.com/predictive-data-analytics/

22. Fix Part Muscle Man. Life cycle of Vibrio cholerae [Internet]. Available from: https://fixpartmuscleman.z5.web.core.windows.net/life-cycle-of-vibrio-cholerae.html

CODE

Data Preprocessing

```
% Load the combined dataset from the CSV file
combinedData = readtable('combined_data.csv');

% 1. Data Cleaning

% Initialize a table to store cleaned data
cleanedData = combinedData;

% Handle missing values
% Replace missing values with the median of the respective column for numerical variables
numVars = varfun(@isnumeric, combinedData, 'OutputFormat', 'uniform');
for i = find(numVars)'
   data = combinedData{:, i};
   missingIdx = isnan(data);
   if any(missingIdx)
      medianVal = median(data, 'omitnan');
      cleanedData{missingIdx, i} = medianVal;
   end
end

% For categorical variables, replace missing values with the mode (most frequent value)
catVars = varfun(@iscategorical, combinedData, 'OutputFormat', 'uniform');
for i = find(catVars)'
   data = combinedData{:, i};
   missingIdx = ismissing(data);
   if any(missingIdx)
      modeVal = mode(data);
      cleanedData{missingIdx, i} = modeVal;
   end
end

% Handle outliers manually (Z-score calculation without zscore function)
zThreshold = 3; % Z-score threshold for outliers
for i = find(numVars)'
   data = cleanedData{:, i};
   mu = mean(data, 'omitnan'); % Mean
   sigma = std(data, 'omitnan'); % Standard deviation
   zScores = (data - mu) / sigma; % Calculate Z-scores
   outlierIdx = abs(zScores) > zThreshold;

   % Replace outliers with median of non-outlier values
   medianVal = median(data(~outlierIdx), 'omitnan'); % Median of non-outliers
   cleanedData{outlierIdx, i} = medianVal;
end

% 2. Normalization

% Normalize numerical variables to range [0, 1]
numVars = varfun(@isnumeric, cleanedData, 'OutputFormat', 'uniform');
for i = find(numVars)'
   data = cleanedData{:, i};
   minVal = min(data, [], 'omitnan');
   maxVal = max(data, [], 'omitnan');
   % Prevent division by zero in case of constant columns
   if maxVal > minVal
      % Normalize data
      normalizedData = (data - minVal) / (maxVal - minVal);
```

```
    else
        % Handle case where maxVal == minVal (constant column)
        normalizedData = zeros(size(data)); % Normalize to 0
    end

    % Assign the normalized data back to the table
    cleanedData.(cleanedData.Properties.VariableNames{i}) = normalizedData;
end

% 3. Feature Engineering

% Example: Create a new feature based on existing ones
% Let's say you want to create a feature for 'Temperature_Rainfall_Ratio'
if any(ismember(cleanedData.Properties.VariableNames, {'Temperature_Celsius', 'Rainfall_mm'}))
    % Check if the columns exist and create the new feature
    tempCol = cleanedData.Temperature_Celsius;
    rainfallCol = cleanedData.Rainfall_mm;
    % Add 1 to rainfall to avoid division by zero
    cleanedData.Temperature_Rainfall_Ratio = tempCol ./ (rainfallCol + 1);
end

% Example: Extract month and day of week from 'Date'
if ismember('Date', cleanedData.Properties.VariableNames)
    % Convert 'Date' column to datetime if not already
    if ~isdatetime(cleanedData.Date)
        cleanedData.Date = datetime(cleanedData.Date, 'InputFormat', 'yyyy-MM-dd');
    end
    cleanedData.Month = month(cleanedData.Date);
    cleanedData.DayOfWeek = weekday(cleanedData.Date);
end

% 4. Data Splitting

% Define the proportion of the dataset for training, validation, and test sets
trainRatio = 0.7;
valRatio = 0.15;
testRatio = 0.15;

% Ensure ratios sum up to 1
assert(abs((trainRatio + valRatio + testRatio) - 1) < 1e-6, 'Ratios must sum up to 1.');

% Shuffle the data
shuffledData = cleanedData(randperm(height(cleanedData)), :);

% Compute the number of samples for each set
numSamples = height(shuffledData);
numTrain = floor(trainRatio * numSamples);
numVal = floor(valRatio * numSamples);

% Split the data
trainData = shuffledData(1:numTrain, :);
valData = shuffledData(numTrain + 1:numTrain + numVal, :);
testData = shuffledData(numTrain + numVal + 1:end, :);

% Save the split data to CSV files
writetable(trainData, 'train_data.csv');
writetable(valData, 'val_data.csv');
writetable(testData, 'test_data.csv');

disp('Data preprocessing complete. Data saved to CSV files.');
```

TRAINING CODE

```
% Load the data
trainData = readtable('train_data.csv');
valData = readtable('val_data.csv');
testData = readtable('test_data.csv');

% Display column names for reference
disp('Column names in trainData:');
disp(trainData.Properties.VariableNames);

disp('Column names in valData:');
disp(valData.Properties.VariableNames);

disp('Column names in testData:');
disp(testData.Properties.VariableNames);

% Define the input size based on the number of features (excluding the Date column)
inputSize = width(trainData) - 2;  % Exclude 'Date' and 'sum_Cholera_Cases' columns

% Define the network layers
layers = [
    featureInputLayer(inputSize, 'Name', 'input')
    fullyConnectedLayer(128, 'Name', 'fc1')
    reluLayer('Name', 'relu1')
    fullyConnectedLayer(64, 'Name', 'fc2')
    reluLayer('Name', 'relu2')
    fullyConnectedLayer(1, 'Name', 'fc3')
    regressionLayer('Name', 'output')
];

% Define training options
options = trainingOptions('adam', ...
    'MaxEpochs', 100, ...
    'MiniBatchSize', 32, ...
    'InitialLearnRate', 1e-3, ...
    'ValidationData', {valData{:, 2:end-1}, valData.sum_Cholera_Cases}, ... % Exclude 'Date' and target columns
    'Plots', 'training-progress', ...
    'Verbose', false);

% Extract features and labels from training data
XTrain = trainData{:, 2:end-1};  % Exclude 'Date' and target columns
YTrain = trainData.sum_Cholera_Cases;

% Train the network
[trainedNet, info] = trainNetwork(XTrain, YTrain, layers, options);

% Save the trained network
save('trainedCNN.mat', 'trainedNet');

% Evaluate the network on the test set
XTest = testData{:, 2:end-1};  % Exclude 'Date' and target columns
YTest = testData.sum_Cholera_Cases;
YPred = predict(trainedNet, XTest);

% Calculate the RMSE
rmse = sqrt(mean((YPred - YTest).^2));
```

```
% Calculate ROC curve and AUC
[fpRate, tpRate, ~, AUC] = perfcurve(YTest, YPred, 'true');

% Display RMSE and AUC
disp(['RMSE: ', num2str(rmse)]);
disp(['AUC: ', num2str(AUC)]);
```

CNN NETWORK

```
% Load the train, validation, and test datasets
trainData = readtable('train_data.csv');
valData = readtable('val_data.csv');
testData = readtable('test_data.csv');

% Display column names to check correct names
disp('Column names in trainData:');
disp(trainData.Properties.VariableNames);

% Assuming the column containing case counts is named 'Cases' or another name
caseColumnName = 'sum_Cholera_Cases'; % Adjust this based on the column names displayed

% Check if the column exists
if ismember(caseColumnName, trainData.Properties.VariableNames)
    % Data Normalization
    % Normalize numerical variables to range [0, 1]
    numVars = varfun(@isnumeric, trainData, 'OutputFormat', 'uniform');
    for i = find(numVars)'
        data = trainData{:, i};
        minVal = min(data, [], 'omitnan');
        maxVal = max(data, [], 'omitnan');
        if maxVal > minVal
            normalizedData = (data - minVal) / (maxVal - minVal);
        else
            normalizedData = zeros(size(data));
        end
        trainData.(trainData.Properties.VariableNames{i}) = normalizedData;
        valData.(valData.Properties.VariableNames{i}) = (valData{:, i} - minVal) / (maxVal - minVal);
        testData.(testData.Properties.VariableNames{i}) = (testData{:, i} - minVal) / (maxVal - minVal);
    end

    % Define the input size based on the preprocessed data
    inputSize = [32 32 3]; % Adjust this based on the data dimensions

    % Define the CNN architecture
    layers = [
        imageInputLayer(inputSize, 'Name', 'input', 'Normalization', 'none')
        convolution2dLayer(3, 16, 'Padding', 'same', 'Name', 'conv1')
        reluLayer('Name', 'relu1')
        maxPooling2dLayer(2, 'Stride', 2, 'Name', 'maxpool1')
        convolution2dLayer(3, 32, 'Padding', 'same', 'Name', 'conv2')
        reluLayer('Name', 'relu2')
        maxPooling2dLayer(2, 'Stride', 2, 'Name', 'maxpool2')
        convolution2dLayer(3, 64, 'Padding', 'same', 'Name', 'conv3')
        reluLayer('Name', 'relu3')
        maxPooling2dLayer(2, 'Stride', 2, 'Name', 'maxpool3')
        flattenLayer('Name', 'flatten')
        fullyConnectedLayer(128, 'Name', 'fc1')
        reluLayer('Name', 'relu_fc1')
        fullyConnectedLayer(64, 'Name', 'fc2')
        reluLayer('Name', 'relu_fc2')
        fullyConnectedLayer(10, 'Name', 'fc3') % Adjust based on the number of classes
        softmaxLayer('Name', 'softmax')
        classificationLayer('Name', 'output')
    ];

    % Define the training options
    options = trainingOptions('adam', ...
        'MaxEpochs', 10, ...
        'MiniBatchSize', 64, ...
        'InitialLearnRate', 1e-4, ...
        'Plots', 'training-progress', ...
        'Verbose', false, ...
        'ValidationData', {valData, 'Label'}); % Adjust based on label column

    % Assuming image data is available for training and validation
    % trainData = imageDatastore('path/to/train/images', 'LabelSource', 'foldernames');
    % valData = imageDatastore('path/to/val/images', 'LabelSource', 'foldernames');
    % [trainedNet, info] = trainNetwork(trainData, layers, options);

    % Save the trained network
    % save('trainedCNN.mat', 'trainedNet');

    % Visualize CNN components

    % Input Layer
    figure('Name', 'Input Layer Visualization');
    title('Input Layer');
    xlabel('Width');
    ylabel('Height');
    zlabel('Channels');
    exampleInput = rand(inputSize);
    montage(exampleInput, 'Size', [1 1]);
    title('Example Input Data');

    % Convolutional Layers
    numFilters = 16;
    filterSize = [3 3];
    filters = rand([filterSize, numFilters]);
    figure('Name', 'Convolutional Filters Visualization');
    for i = 1:numFilters
        subplot(4, 4, i);
        imshow(filters(:,:,i), []);
        title(sprintf('Filter %d', i));
    end
    sgtitle('Convolutional Filters');

    % Pooling Layers
    featureMap = rand(32, 32);
    poolSize = 2;
    pooledFeatureMap = maxPooling2d(featureMap, poolSize);
    figure('Name', 'Pooling Layers Visualization');
    subplot(1, 2, 1);
    imshow(featureMap, []);
    title('Feature Map Before Pooling');
    subplot(1, 2, 2);
```

```
    imshow(pooledFeatureMap, []);
    title('Feature Map After Pooling');

    % Fully Connected Layers
    numFeatures = 64;
    numClasses = 10;
    weights = rand(numFeatures, numClasses);
    figure('Name', 'Fully Connected Layers Visualization');
    imagesc(weights);
    colorbar;
    title('Weights of Fully Connected Layer');
    xlabel('Output Classes');
    ylabel('Input Features');

    % Output Layer
    probabilities = rand(1, numClasses);
    figure('Name', 'Output Layer Visualization');
    bar(probabilities);
    title('Output Layer - Class Probabilities');
    xlabel('Classes');
    ylabel('Probability');

else
    error('Column "%s" not found in the dataset.', caseColumnName);
end

function pooledMap = maxPooling2d(map, poolSize)
    pooledMap = map(1:poolSize:end, 1:poolSize:end);
end
```

CONFUSION MATIX AND PERFORMANCE MATRIC

```
% Load the train, validation, and test datasets
trainData = readtable('train_data.csv');
valData = readtable('val_data.csv');
testData = readtable('test_data.csv');

% Display column names to check correct names
disp('Column names in trainData:');
disp(trainData.Properties.VariableNames);
disp('Column names in valData:');
disp(valData.Properties.VariableNames);
disp('Column names in testData:');
disp(testData.Properties.VariableNames);

% Define correct column names based on the dataset
caseColumnName = 'sum_Cholera_Cases'; % Adjust based on your dataset
trueLabelColumnName = 'TrueLabel'; % Replace with actual label column name from testData
predictedScoreColumnName = 'PredictedScore'; % Replace with actual score column name from testData

% Check if the column exists
if ismember(caseColumnName, trainData.Properties.VariableNames)
    % Data Normalization
    % Normalize numerical variables to range [0, 1]
    numVars = varfun(@isnumeric, trainData, 'OutputFormat', 'uniform');
    for i = find(numVars)'
        data = trainData{:, i};
        minVal = min(data, [], 'omitnan');
        maxVal = max(data, [], 'omitnan');
        if maxVal > minVal
            normalizedData = (data - minVal) / (maxVal - minVal);
        else
            normalizedData = zeros(size(data));
        end
        trainData.(trainData.Properties.VariableNames{i}) = normalizedData;
        valData.(valData.Properties.VariableNames{i}) = (valData{:, i} - minVal) / (maxVal - minVal);
        testData.(testData.Properties.VariableNames{i}) = (testData{:, i} - minVal) / (maxVal - minVal);
    end

    % Define the input size based on the preprocessed data
    inputSize = [32 32 3]; % Adjust this based on the data dimensions

    % Define the CNN architecture
    layers = [
        imageInputLayer(inputSize, 'Name', 'input', 'Normalization', 'none')
        convolution2dLayer(3, 16, 'Padding', 'same', 'Name', 'conv1')
        reluLayer('Name', 'relu1')
        maxPooling2dLayer(2, 'Stride', 2, 'Name', 'maxpool1')
        convolution2dLayer(3, 32, 'Padding', 'same', 'Name', 'conv2')
        reluLayer('Name', 'relu2')
        maxPooling2dLayer(2, 'Stride', 2, 'Name', 'maxpool2')
        convolution2dLayer(3, 64, 'Padding', 'same', 'Name', 'conv3')
        reluLayer('Name', 'relu3')
        maxPooling2dLayer(2, 'Stride', 2, 'Name', 'maxpool3')
        flattenLayer('Name', 'flatten')
        fullyConnectedLayer(128, 'Name', 'fc1')
        reluLayer('Name', 'relu_fc1')
        fullyConnectedLayer(64, 'Name', 'fc2')
        reluLayer('Name', 'relu_fc2')
        fullyConnectedLayer(10, 'Name', 'fc3') % Adjust based on the number of classes
        softmaxLayer('Name', 'softmax')
        classificationLayer('Name', 'output')
    ];

    % Define the training options
    options = trainingOptions('adam', ...
        'MaxEpochs', 10, ...
        'MiniBatchSize', 64, ...
        'InitialLearnRate', 1e-4, ...
        'Plots', 'training-progress', ...
        'Verbose', false, ...
        'ValidationData', {valData, 'Label'}); % Adjust based on label column

    % Assuming image data is available for training and validation
    % trainData = imageDatastore('path/to/train/images', 'LabelSource', 'foldernames');
    % valData = imageDatastore('path/to/val/images', 'LabelSource', 'foldernames');
    % [trainedNet, info] = trainNetwork(trainData, layers, options);

    % Save the trained network
    % save('trainedCNN.mat', 'trainedNet');

    % Visualize CNN components
```

```
% Input Layer
figure('Name', 'Input Layer Visualization');
exampleInput = rand(inputSize);
montage(exampleInput, 'Size', [1 1]);
title('Example Input Data');
xlabel('Width');
ylabel('Height');
zlabel('Channels');

% Convolutional Layers
numFilters = 16;
filterSize = [3 3];
filters = rand([filterSize, numFilters]);
figure('Name', 'Convolutional Filters Visualization');
for i = 1:numFilters
    subplot(4, 4, i);
    imshow(filters(:,:,i), []);
    title(sprintf('Filter %d', i));
end
sgtitle('Convolutional Filters');

% Pooling Layers
featureMap = rand(32, 32);
poolSize = 2;
pooledFeatureMap = maxPooling2d(featureMap, poolSize);
figure('Name', 'Pooling Layers Visualization');
subplot(1, 2, 1);
imshow(featureMap, []);
title('Feature Map Before Pooling');
subplot(1, 2, 2);
imshow(pooledFeatureMap, []);
title('Feature Map After Pooling');

% Fully Connected Layers
numFeatures = 64;
numClasses = 10;
weights = rand(numFeatures, numClasses);
figure('Name', 'Fully Connected Layers Visualization');
imagesc(weights);
colorbar;
title('Weights of Fully Connected Layer');
xlabel('Output Classes');
ylabel('Input Features');

% Output Layer
probabilities = rand(1, numClasses);
figure('Name', 'Output Layer Visualization');
bar(probabilities);
title('Output Layer - Class Probabilities');
xlabel('Classes');
ylabel('Probability');

% Confusion Matrix and Performance Metrics

% Check if the columns exist
if ismember(trueLabelColumnName, testData.Properties.VariableNames) && ismember(predictedScoreColumnName, testData.Properties.VariableNames)
    % Extract true labels and predicted scores
    yTrue = testData.(trueLabelColumnName);
    yScores = testData.(predictedScoreColumnName); % Assuming these are probability scores

    % Calculate ROC curve and AUC
    [FPR, TPR, ~, AUC] = perfcurve(yTrue, yScores, 'positiveClass'); % Adjust based on class labels

    % Plot ROC curve
    figure('Name', 'ROC Curve Visualization');
    plot(FPR, TPR, '-o');
    xlabel('False Positive Rate (FPR)');
    ylabel('True Positive Rate (TPR)');
    title(['ROC Curve (AUC = ', num2str(AUC), ')']);
    grid on;

    % Assuming confusion matrix values (replace with actual values from model)
    TP = 50; % Example value for True Positives
    FP = 10; % Example value for False Positives
    TN = 100; % Example value for True Negatives
    FN = 20; % Example value for False Negatives

    % Calculate Precision
    precision = TP / (TP + FP);

    % Calculate Recall
    recall = TP / (TP + FN);

    % Calculate F1 Score
    f1Score = 2 * (precision * recall) / (precision + recall);

    % Display the confusion matrix and metrics
    figure('Name', 'Confusion Matrix and Metrics');

    % Confusion Matrix Visualization
    subplot(2,2,1);
    confusionMat = [TP, FP; FN, TN];
    % Define a colormap
    colormap = [1 1 1; 0.9 0.9 0.9; 0.5 0.5 1; 0 0 1]; % Example custom colormap
    heatmap(confusionMat, 'XData', {'Predicted Positive', 'Predicted Negative'}, 'YData', {'Actual Positive', 'Actual Negative'}, 'Colormap', colormap, 'ColorbarVisible', 'on');
    title('Confusion Matrix');

    % Plot Precision, Recall, and F1 Score
    subplot(2,2,2);
    bar([precision, recall, f1Score]);
    set(gca, 'XTickLabel', {'Precision', 'Recall', 'F1 Score'});
    title('Performance Metrics');

    ylabel('Value');

    ylim([0 1]);

    % Display numerical values

    fprintf('Precision: %.2f\n', precision);

    fprintf('Recall: %.2f\n', recall);

    fprintf('F1 Score: %.2f\n', f1Score);



    % Save the figures

    saveas(gcf, 'confusion_matrix_and_metrics.png');

else
```

```
        error('Column "%s" or "%s" not found in the dataset.', trueLabelColumnName, predictedScoreColumnName);

    end

else

    error('Column "%s" not found in the dataset.', caseColumnName);

end

function pooledMap = maxPooling2d(map, poolSize)

    pooledMap = map(1:poolSize:end, 1:poolSize:end);

end
```