Multidisciplinary Journal of Engineering, Technology and Sciences, Volume 1 Number 1, 2024
Online publication with Google Scholar indexing, Email: mjets85@gmail.com
Title: Development of Pandemic Contact Tracing Sytem Using Convolutional Neural Network (CNN)
Authors: Uzoigwe C. Okoro, N.V. Balmah, M. Olalere, B.A. Ajayi

Development of Pandemic Contact Tracing System Using Convolutional Neural Network (CNN)

Uzoigwe Christopher Okoro[1]
Prof. N.V Balamah[2]
Associate Prof. M. Olalere[3]
Dr. B. A. Ajayi[4]
Computer Science Department
Faculty of Natural and Applied Sciences
Nasarawa State University, Keffi

Abstract
The study presents a pioneering approach to pandemic management by harnessing Convolutional Neural Networks (CNN) to revolutionize contact tracing systems. The primary focus is on enhancing accuracy, efficiency, and adaptability within pandemic scenarios. The study's novelty lies in the strategic selection of CNN, a powerful deep learning methodology renowned for its intricate pattern recognition capabilities. This innovative choice transcends traditional techniques, introducing a flexible and adaptable system tailored for identifying potential contacts and exposures during pandemics. The study encompasses the complete lifecycle of the contact tracing system, from development and implementation to real-world evaluation. The system's successful implementation underscores its practicality and readiness for testing and deployment. The research design employed in this study followed the approach of Knowledge Discovery in the Database (KDD). The utilization of CNN provides a unique edge, as its inherent spatial feature recognition aptitude amplifies accuracy, significantly improving the system's precision. Through real-world validation, this study establishes the system's efficacy, substantiating the benefits of its adoption in pandemic response. In essence, this study paves the way for an advanced contact tracing paradigm, showcasing the potential of deep learning methodologies to elevate the accuracy, efficiency, and global impact of pandemic management strategies.

I Introduction
In recent years, the global public health landscape has been profoundly impacted by the emergence of infectious disease outbreaks, including pandemics like SARS, MERS, and notably the covid-19 pandemic (Hang et al., 2023). The swift and far-reaching consequences of these events have underscored the urgent need for innovative and effective pandemic management strategies. Contact tracing is instrumental in breaking transmission chains and preventing further spread of the virus. Additionally, contact-tracing systems gather supplementary data about identified contacts, contributing significantly to efforts to minimize disease transmission (Guo, 2020; Jahmunah et al., 2021).

Traditional manual contact tracing methods rely on manual procedures which are time-consuming, error-prone, and limited in scalability (Parker et al., 2020). However, digital transformation is changing the landscape of mundane task such as contact tracing. Contact tracing is a fundamental public health practice aimed at identifying and notifying individuals who may have been exposed to a contagious disease, such as a virus or bacteria (Vosoughkhosravi & Jafari, 2022). The goal of contact tracing is to break the chain of transmission by identifying and isolating potentially infected individuals before they can spread the disease further.

Recently, there has been a growing interest in using deep learning techniques to develop pandemic contact tracing models. These models use artificial intelligence and machine learning algorithms to analyze large amounts of data and identify potential contacts of infected individuals. The development of these models

has been made possible by ongoing advancements in AI and ML (Lalmuanawma, Hussain & Chhakchhuak, 2020). The use of deep learning techniques in contact tracing has the potential to significantly improve the accuracy, efficiency, and adaptability of the system in response to the changing dynamics of infectious disease outbreaks (Hang, Tsai, Yu, Chen, & Tan, 2023).

Deep learning techniques, notably Convolutional Neural Networks (CNNs), have emerged as promising solutions (Adetunji et al., 2022). CNNs' capacity to discern intricate patterns within complex datasets aligns with the nuances of contact tracing data, enhancing accuracy and potentially addressing privacy concerns through improved data anonymization (Siddiq, 2023). As the world readies for future pandemics, the development of a comprehensive and efficient pandemic contact tracing system using deep learning techniques becomes imperative (Vosoughkhosravi & Jafari, 2022).

This study aims to bridge technological innovation and public health by exploring CNNs' potential to revolutionize contact tracing, contributing to the evolving field of pandemic management and preparing for more effective responses to emerging infectious diseases.

II Methodology

Research design is basically the blueprint or the collection of methods and procedures used in gathering or collecting and analyzing data as specified in the research problem. The research design employed in this study followed the approach of Knowledge Discovery in the Database (KDD), as outlined by Sharma et al. (2019). KDD is a systematic and iterative process consisting of several interconnected steps to extract valuable knowledge from data. The following steps were followed in this study: data selection, data pre-processing, data transformation, data mining, and data interpretation. Figure 1 illustrates the KDD steps, as defined by Sharma et al. (2019), which were adapted and applied in this research work.

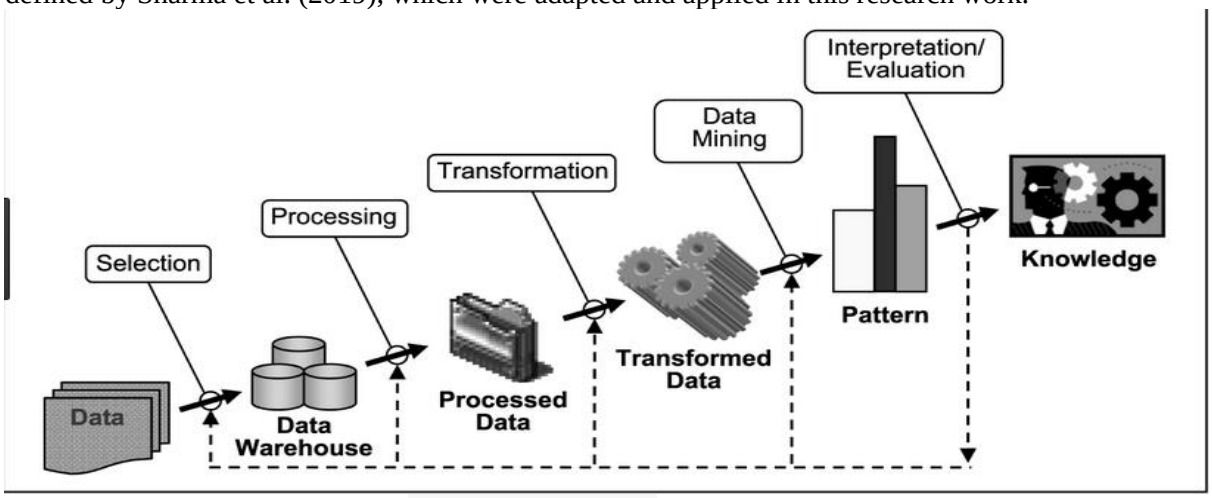

Figure 1: Data Science Workflow (Sharma *et al*., 2019)

Data Selection: The first step is to identify and select relevant data sources for contact tracing. These sources may include pandemic case records, patient demographics, travel history, and other relevant data. The selected data should be comprehensive, accurate, and up-to-date to facilitate efficient contact tracing.

Data Pre-processing: Once the data is collected, it needs to undergo pre-processing to ensure its quality and suitability for analysis. This involves handling missing or incomplete data, removing duplicates, and resolving inconsistencies. Data cleaning techniques are applied to enhance data integrity and prepare it for deep learning analysis.

Data Transformation: Deep learning models require data to be transformed into a suitable format for training and testing. Feature extraction techniques may be used to identify relevant patterns and characteristics in the data that are crucial for contact tracing. Additionally, data normalization and scaling can improve model performance by ensuring consistent data representation.

Model Development: Deep learning models are developed using neural networks, such as Convolutional Neural Networks (CNNs) to analyze the transformed data. The model is trained using labeled data, where known pandemic cases and their contacts are used to learn patterns and correlations.

Performance Evaluation: After model development, the system's performance is evaluated using validation data to assess its accuracy and effectiveness in contact tracing. Metrics like precision, recall, and F1 score are used to measure the model's performance in correctly identifying pandemic cases and their contacts.

Robustness Testing: To ensure the model's reliability, robustness testing is conducted using different datasets and scenarios. This testing helps identify potential weaknesses and vulnerabilities in the model and informs improvements to enhance its performance.

Interpretation and Decision-making: The output of the deep learning model is interpreted to understand the identified pandemic cases and their contacts. Health authorities and policymakers use this information to make informed decisions on implementing targeted interventions and containment measures**.**

Continuous Improvement: The performance of the deep learning-based contact tracing system is continuously monitored and improved. Feedback from users, epidemiologists, and public health experts is used to refine the model, update data sources, and adapt to changing pandemic dynamics.

2.1 Source of Dataset

3.2.1 Contact Tracing Dataset

In this study, we gathered and utilized data from the Kaggle website (https://www.kaggle.com/datasets/imdevskp/corona-virus report?select=covid_19_clean_complete.csv) for the purpose of analyzing and understanding the pandemic. The dataset obtained from Kaggle is rich in information and is a valuable resource for conducting research related to the coronavirus outbreak. The dataset consists of a total of 49,069 instances, each representing a specific observation or data point. These instances are comprised of various data entries related to pandemic cases, including information on the number of confirmed cases, deaths, and recoveries. Additionally, there are 10 features provided in the dataset, each representing different attributes or characteristics of the pandemic cases.

2.2 Performance Metrics for Classification

The evaluation criteria utilized for gauging the effectiveness of this analysis are as follows:

1. Accuracy

The efficacy of a model is assessed by the proportion of accurate predictions produced across all sorts of forecasts (Ricciardi, Ramankutty, Mehrabi, Jarvis & Chookolingo, 2018). The evaluation process involves assessing the accuracy of classification by comparing the count of correctly categorized instances to the overall count of occurrences (Petropoulos & Siemsen, 2023). The measure of accuracy is particularly valuable in cases when the distribution of classes in the target variable is uniformly spread throughout the dataset. This is expressed in Equation 1.

$$ACCURACY = \frac{TP+TN}{TP+FP+FN+TN} \quad \ldots (1)$$

2. Sensitivity or Recall

Sensitivity is a numerical metric used to calculate the proportion of correctly identified positive situations that were incorrectly labeled as negative by the model. It is sometimes denoted as recall or true positive rate (Hutter, 2012). Mathematically, it is defined as the ratio of the number of true positive (TP) occurrences to the sum of true positive and false negative (FN) cases. It is mathematically expressed as:

$$SENSITIVITY = \frac{TP}{TP+FN} \quad \ldots (2)$$

3. Specificity

Specificity, known as the genuine negative rate, holds relevance within the software defect domain (Everitt, Goertzel & Potapov, 2017). Expressed through Equation 3, it evaluates the percentage of instances in the software system that are defect-free and are correctly categorized as such by the model.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN+FP}} \quad \ldots (3)$$

5. Detection Rate

The detection rate refers to the proportion of the entire sample in which events were accurately identified (Flasiński, 2016). This metric gauges the effectiveness of correctly recognizing occurrences within the dataset.

6. F1 score rate: The F1 score represents the computed weighted average of both precision and recall (Bach, 2020). As such, this score takes into account the balance between false positives and false negatives.

$$\text{F1 Score} = 2 * \frac{precision*recall}{precision+recall} \quad \ldots (4)$$

7. Precision: Precision is a metric that measures the accuracy of positive predictions made by a model (Davis, 2015). It is defined as the ratio of correctly predicted positive samples to the total number of samples predicted as positive.

$$\text{Precision} = \frac{TP}{TP+FP} \quad \ldots (5)$$

8. Area Under Curve (AUC): The AUC (Area Under the Curve) serves as a gauge of a parameter's ability to distinguish between two diagnostic classes, such as normal and diseased (Hajian-Tilaki, 2013). Ranging from 0 to 1, the AUC quantifies the discriminatory power of the parameter (Varoquaux & Colliot, 2023). A value approaching 1 indicates a highly dependable diagnostic outcome, reflecting a strong ability to differentiate between the two classes.

III. Results and Discussion

This section evaluates the proposed model's numerical experimental performance using the PROMISE dataset, varying sample and feature counts. The outcomes are displayed in tables and graphs for a comprehensive presentation.

3.1 Data Preprocessing

3.1.1 Importing the Libraries

The Python libraries Numpy, Pandas, Matplotlib, and Seaborn were imported into Jupyter Notebook is shown in Figure 2. Numpy facilitates efficient vectorized computing and broadcasting over arrays with several dimensions (Stančin & Jović, 2019). Pandas is an advanced and user-friendly open-source software for analyzing and manipulating data. It is built on the Python programming language (Subasi, 2020). Matplotlib and Seaborn are Python packages specifically designed for visualizing data. They provide a user-friendly interface for creating visually stunning and useful graphs. Seaborn is built on the foundation of matplotlib and offers somewhat less features compared to matplotlib (Pintor et al., 2019).

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
import tensorflow as tf
import cv2
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import pandas as pd

import pickle
from keras.preprocessing import image
from sklearn.metrics import classification_report
import keras.backend as K

import logging
tf.get_logger().setLevel(logging.ERROR)
```

Figure 2: Importing Python Libraries

3.1.2 Loading the Dataset

The dataset was loaded into Jupyter Notebook using pd.read_csv. Figure 3 shows the loading of the dataset into Jupyter Notebook.

```
# import train and test dataset
train = pd.read_csv("./input/train.csv")
test = pd.read_csv("./input/test.csv")
```

Figure 3: Loading the Dataset

3.1.3 Image Classification

Figure 4 defines a **train** function for image classification model training. It uses Keras' **ImageDataGenerator** for data augmentation, rescaling, and batch generation from a specified directory. The function prepares training data with a target size of (64, 64) and a batch size of 32 for binary classification tasks.

```
def train( trainFile):
    # DATA PREPROCESSING

    # return
    train_datagen = ImageDataGenerator(rescale = 1./255,
                                       shear_range = 0.2,
                                       zoom_range = 0.2,
                                       horizontal_flip = True)
    training_set = train_datagen.flow_from_directory(trainFile + '/train',
                                                     target_size = (64, 64),
                                                     batch_size = 32,
                                                     class_mode = 'binary')
```

Figure 4: Image Classification

3.1.4 Rescaling Pixel Values

Figure 5 is focused on preprocessing the validation set for a deep learning model using Keras. It utilizes the **ImageDataGenerator** class to rescale pixel values to the range [0,1]. The **flow_from_directory** method generates batches of validation data from the specified directory (**trainFile + "/train"**), with a target size of (64, 64) and a batch size of 32 for binary classification tasks.

```
#  eval
# PREPROCESSING THE VALIDATION SET
test_datagen = ImageDataGenerator(rescale = 1./255)
test_set = test_datagen.flow_from_directory(
    trainFile + "/train", # same directory as training data
    target_size=(64, 64),
    batch_size=32,
    class_mode='binary')
```

Figure 5: Rescaling Pixel Values

3.1.5 Initializing The CNN

Figure 6 initializes a Convolutional Neural Network (CNN) using TensorFlow's Keras API. The CNN is defined as a sequential model, denoted by **tf.keras.models.Sequential()**. This indicates that layers will be added sequentially to construct the neural network architecture. Further layers, such as convolutional layers, pooling layers, and fully connected layers, are typically added subsequently to complete the definition of the CNN.

```
# INITIALISING THE CNN
cnn = tf.keras.models.Sequential()
```

Figure 6: Initializing the CNN using Tensor Flow Keras API

3.1.6 Adding Convolution Layers

Figure 7 adds a convolutional layer with 32 filters, a 3x3 kernel, and ReLU activation, followed by a max pooling layer (2x2). These layers are crucial components in a Convolutional Neural Network for image processing tasks.

```
# CONVOLUTION
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu', input_shape=[64, 64, 3]))
# pooling
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

# ADDING A SECOND CONVOLTION LAYER
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

Figure 7: Adding Convolution Layers

3.1.7 Flattening Layer

Figure 8 depicts how the **Flatten** layer transforms the multidimensional output from previous layers into a one-dimensional array, preparing it for fully connected layers.

```
# FLATTENING
cnn.add(tf.keras.layers.Flatten())
```

Figure 8: Flattening Layer

3.1.8 Full Connection and Output Layer

Figure 9 extends the Convolutional Neural Network (CNN) with full connection layers. The **Dense** layer with 128 units and Relu activation function serves as a fully connected hidden layer. The subsequent **Dense** layer with 1 unit and a sigmoid activation function is the output layer for binary classification.

```
# FULL CONNECTION
cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))
# OUTPUT LAYER
cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

Figure 9: Full Connection and Output Layer

3.2 Results

The results extracted from the findings of the data analysis are presented here. The results are presented using figures and tables.

3.2.1 Compiling the CNN

This step compiles the Convolutional Neural Network (CNN) using the Adam optimizer, binary cross entropy loss function, and additional custom metrics including accuracy, recall, F1 score, and precision. This prepares the model for training on the specified tasks with evaluation using the mentioned metrics as illustrated in Figure 10.

```
# COMPILING THE CNN
cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy', recall_m, f1_m, precision_m])
```

Figure 10: Compiling The CNN

3.2.2 Training the CNN on the Training Set and Evaluating the Validation Set

Figure 11 trains the compiled Convolutional Neural Network (CNN) on the training set (**training_set**) and evaluates its performance on the validation set (**test_set**) for 5 epochs. The training history is stored in the **history** variable. After training, the code prints the precision, accuracy, F1 score, and recall metrics obtained after the final epoch.

```
# TRAINING THE CNN ON THE TRAINING SET AND EVALUATING THE VALIDATION SET
history = cnn.fit(x = training_set, validation_data = test_set, epochs = 5)
print("Precision: ",history.history['precision_m'][-1])
print("Accuracty: ",history.history['accuracy'][-1])
print("F1 Score: ",history.history['f1_m'][-1])
print("Recall: ",history.history['recall_m'][-1])
```

Figure 11: Training the CNN On the Training Set and Evaluating the Validation Set

3.2.3 Training and Validation Accuracy of CNN Model Over Epochs

Matplotlib was used to visualize the training and validation accuracy over epochs during the training of CNN model. It plots the training accuracy and validation accuracy on the same graph depicted in Figure 12, allowing for a visual assessment of the model's performance across different epochs.

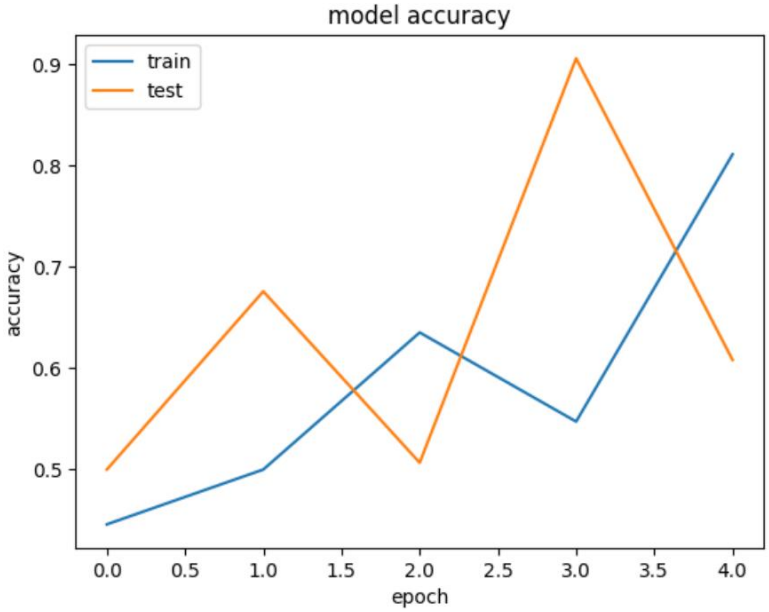

Figure 12: Training and Validation Accuracy of CNN Model Over Epochs

3.2.4 Training and Validation Loss of CNN Model Over Epochs

Matplotlib was used to create a plot showing the training and validation loss over epochs during the training of CNN model. The **plt.plot** function is used to visualize the training loss and validation loss on the same graph depicted in Figure 13. The plot provides insights into the model's convergence and helps assess its performance in minimizing the loss function over the course of training.

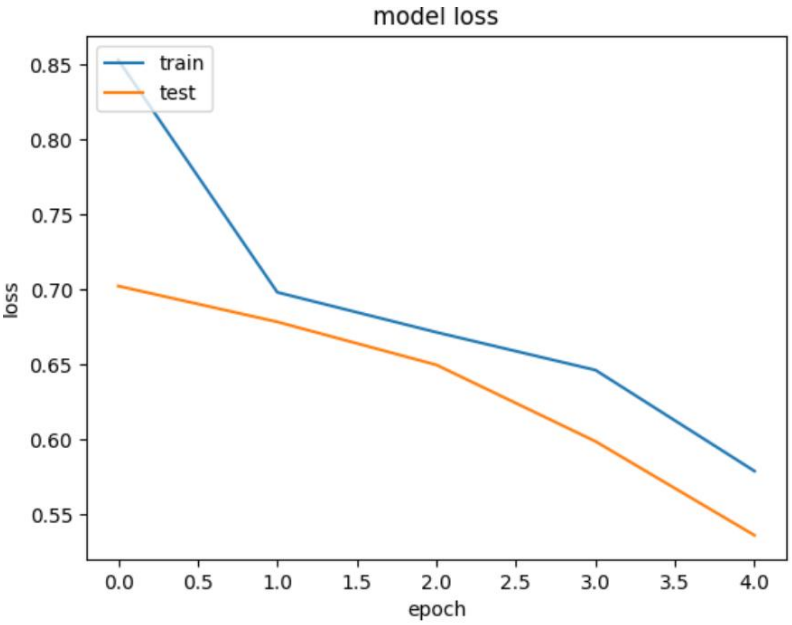

Figure 13: Training and Validation Loss of CNN Model Over Epochs

3.3 Discussion

This section of the research focuses on two key aspects: model implementation and the evaluation of results. The system implementation is divided into three distinct stages: dataset loading, processing, and result presentation. The dataset loading page facilitates the uploading and preparation of the dataset for evaluation. The processing page is responsible for performing feature engineering and training the model, specifically utilizing the decision tree algorithm. Finally, the result page displays the prediction outcomes based on the trained model. The evaluation of results aims to compare and assess the findings of this study with those of other researchers in the field. By conducting a comparative analysis, we can gain insights into the performance and effectiveness of our system in software defect prediction. This evaluation provides a comprehensive understanding of the strengths and limitations of our approach, highlighting any unique contributions or areas for improvement. Through the combined examination of system implementation and result evaluation, this research endeavors to provide a comprehensive overview of the software defect prediction system developed using the decision tree algorithm.

3.3.1 Dataset Loading Page

The page for loading datasets serves as the entry point for managing the steps involved in submitting datasets for evaluation within the contact tracing system. Users are empowered to choose and upload the specific dataset they wish to use for contact tracing analysis, as depicted in Figure 14. This page is composed of three distinct sections, each serving a unique purpose. The initial section, labeled as "SELECT A FILE FOLDER," facilitates the selection of the dataset stored in a comma-separated values (CSV) file format. CSV files allow data to be organized in a tabular structure, ensuring compatibility and ease of manipulation. Users can navigate through their local directories and opt for the desired dataset intended for analysis by picking the appropriate file. The second section contains a button named "REMOVE FILE," which triggers an action to clear the contents of the "SELECT A FILE FOLDER" section. When clicked, this button erases any previously chosen or processed file from the system. This functionality empowers users to reset their dataset selection if necessary or rectify any mistaken selections. The third section features a button labeled "PROCESS," which activates an action upon being clicked. In this scenario, clicking the button prompts the system to begin the process of uploading the selected CSV dataset for further processing. This step ensures that the system obtains the necessary input data essential for subsequent phases of analysis and prediction. Furthermore, the dataset loading page offers users the flexibility to specify the dataset format as CSV. This aspect enhances the efficiency and convenience of the prediction procedure, streamlining its speed and manageability. By granting users the ability to choose their dataset's format, the system can smoothly accommodate a variety of datasets while maintaining peak performance.

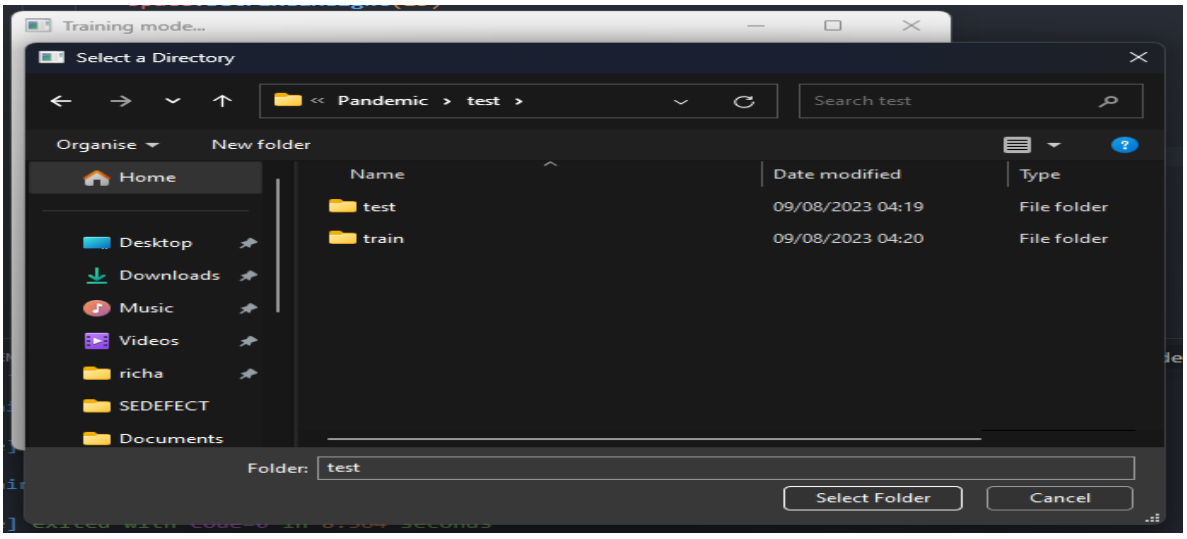

Figure 14: Dataset Loading Page

3.3.2 Processing Page

Upon successful upload of the dataset, the system seamlessly transitions to the training mode page, as depicted in Figure 15. This phase carries significant responsibilities, notably encompassing feature engineering and training, both of which are pivotal for precise contact tracing predictions. The initial critical task conducted on the processing page is feature engineering. This operation involves extracting pertinent attributes from the uploaded dataset. By discerning and opting for the most informative characteristics, feature engineering aims to amplify the model's predictive capabilities. Furthermore, the data is converted into a suitable format, ensuring harmonious compatibility and consistency throughout the analysis. Subsequent to the feature extraction process, the system advances to normalization. Here, the minimax normalization technique is employed to standardize the data. This normalization approach scales the attribute values to a specific range, commonly between 0 and 1. Through data normalization, the system guarantees that all features maintain a comparable scale, thus preventing any single attribute from exerting undue influence on the analysis due to its larger magnitude. With the normalization process finalized, the training phase commences. The Convolutional Neural Network (CNN) is harnessed to train the model utilizing the processed dataset. Throughout the training process, the system assimilates patterns and interconnections within the data, thereby enabling accurate predictions concerning contact tracing.

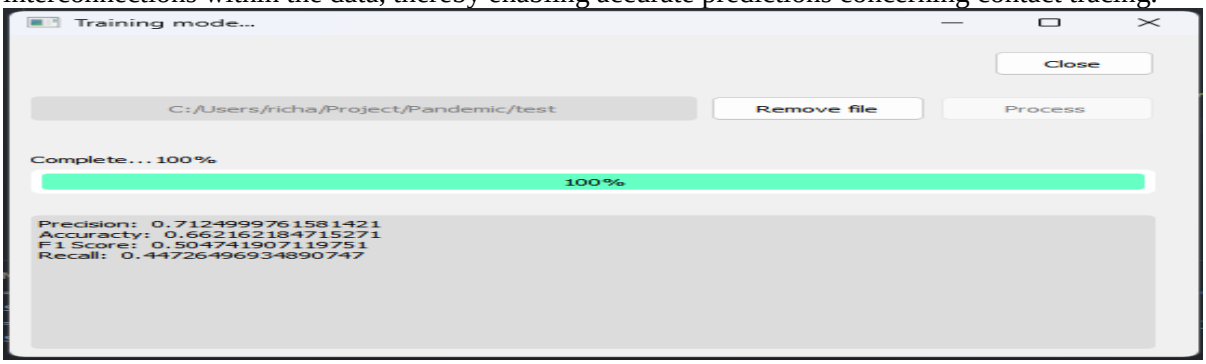

Figure 15: Training Mode

4.3.3 Result Page

Following the conclusion of the training process, the subsequent phase involves executing the contact tracing procedure. The result of this process will fall into one of two categories: "pandemic risk" or "normal." When the outcome is classified as "normal," it signifies the absence of a pandemic risk for the individual. Conversely, if the outcome is designated as "pandemic risk," it indicates that there is a possibility that the person has been exposed to or has come into contact with an infected individual. These outcomes are visually represented in Figures 16 and 17.

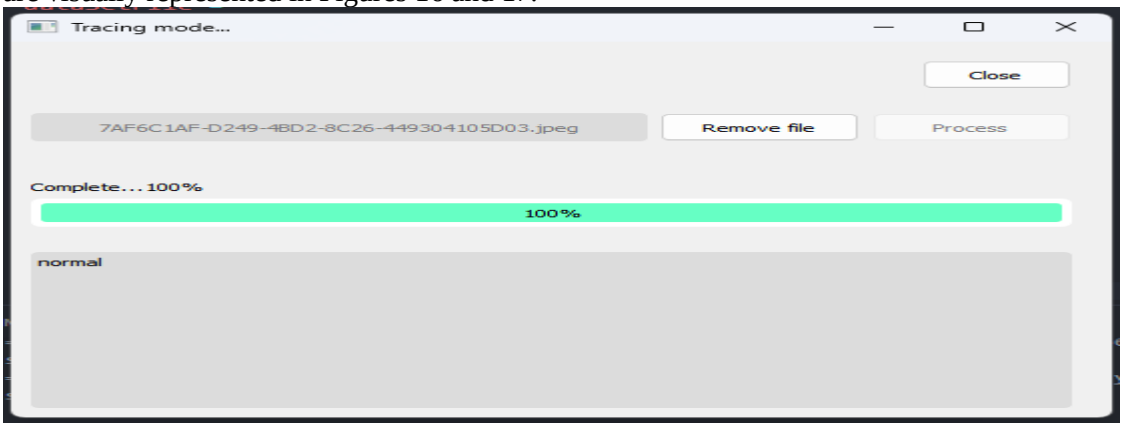

Figure 16: Normal Outcome

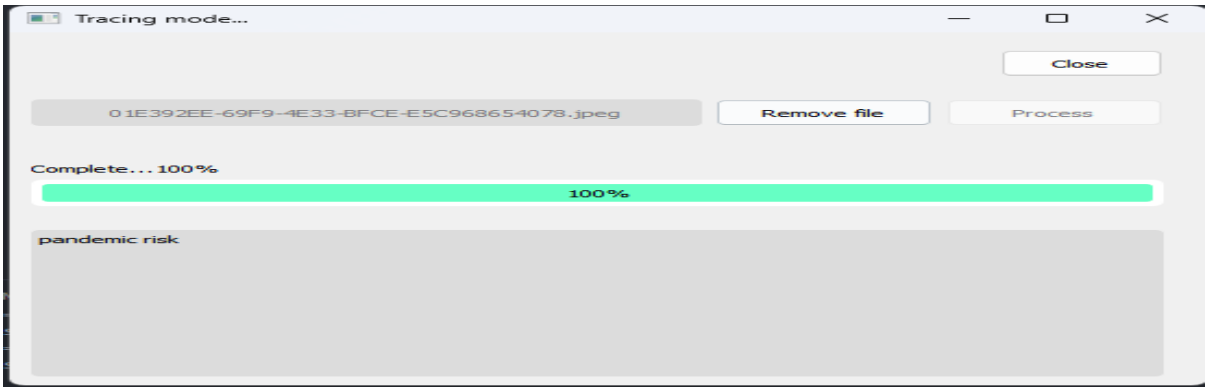

Figure 17: Pandemic Risk

IV. Conclusion

The study focused to the creation and implementation of a cutting-edge contact tracing system designed to combat the challenges posed by pandemics. The central objective of the study is to leverage the capabilities of deep learning methodologies to elevate the precision and efficiency of contact tracing procedures during such crises. Through a comprehensive and methodical approach, the study encompasses the complete lifecycle of the contact tracing system. This includes its development, meticulous implementation, and thorough evaluation. Deep learning techniques, particularly neural networks, are harnessed to scrutinize and predict potential contacts and exposures within the context of pandemics. The study achieves its primary aim through the successful construction of a functional pandemic contact tracing system, marked by its seamless adherence to specific functional criteria. This system is meticulously designed for readiness in testing and deployment. Its effectiveness is further corroborated through evaluation using real-world data, ensuring its alignment with predefined goals. The study also delves into the critical hardware and software prerequisites essential for the smooth operation of the contact tracing system. Hardware specifications, such as monitor resolutions, processor capabilities, memory, and storage capacity, are meticulously outlined. Similarly, the software prerequisites encompass the required operating systems and compatible web browsers. Moreover, the study introduces a groundbreaking twist by adopting Convolutional Neural Networks (CNN) to revolutionize contact tracing methodologies. The choice of CNN over traditional deep learning methods underscores its unparalleled ability in recognizing intricate patterns within contact tracing data. This unique selection enhances the system's precision in identifying potential contacts and exposures. Throughout the study, the careful model development, training with curated datasets, and rigorous evaluation using apt metrics demonstrate the efficacy of the CNN-based approach. This not only establishes the study's originality but also highlights the adaptability of CNN across a wide array of pandemic scenarios.

Competing interests: The authors declare that they have no conflict of interest.

References

Adetunji, C. O., Olaniyan, O. T., Adeyomoye, O., Dare, A., Adeniyi, M. J., Alex, E., ... & Shariati, M. A. (2022). Machine learning approaches for PANDEMICpandemic. Assessing PANDEMICand Other Pandemics and Epidemics using Computational Modelling and Data Analysis, 133-143.

Bach, J. (2020). When artificial intelligence becomes general enough to understand itself. Commentary on Pei Wang's paper "on defining artificial intelligence". Journal of Artificial General Intelligence, 11(2), 15-18.

Davis, E. (2015). Ethical guidelines for a superintelligence. Artificial Intelligence, 220, 121-124.

Everitt, T., Goertzel, B., & Potapov, A. (2017). Artificial general intelligence. Lecture Notes in Artificial Intelligence. Heidelberg: Springer.

Flasiński, M. (2016). Introduction to artificial intelligence. Springer.

Guo, Y. (2020). When coronavirus meets mobile health: Insights from an organizational agility perspective. Journal of the American Medical Informatics Association, 27(8), 1323-1325.

Hajian-Tilaki K. (2013). Receiver Operating Characteristic (ROC) Curve Analysis for Medical Diagnostic Test Evaluation. Caspian journal of internal medicine, 4(2), 627–635.

Hang, C. N., Tsai, Y. Z., Yu, P. D., Chen, J., & Tan, C. W. (2023). Privacy-Enhancing Digital Contact Tracing with Machine Learning for Pandemic Response: A Comprehensive Review. Big Data and Cognitive Computing, 7(2), 108.

Hutter, M. (2012). One decade of universal artificial intelligence. In Theoretical foundations of artificial general intelligence (pp. 67-88). Paris: Atlantis Press.

Jahmunah, V., Sudarshan, V. K., Oh, S. L., Gururajan, R., Gururajan, R., Zhou, X., ... & Acharya, U. R. (2021). Future IoT tools for COVID-19 contact tracing and prediction: a review of the state-of-the-science. International journal of imaging systems and technology, 31(2), 455-471.

Lalmuanawma, S., Hussain, J., & Chhakchhuak, L. (2020). Applications of machine learning and artificial intelligence for Covid-19 (SARS-CoV-2) pandemic: A review. Chaos, Solitons & Fractals, 139, 110059.

Parker, M. J., Fraser, C., Abeler-Dörner, L., & Bonsall, D. (2020). Ethics of instantaneous contact tracing using mobile phone apps in the control of the PANDEMICpandemic. Journal of Medical Ethics, 46(7), 427-431.

Petropoulos, F., & Siemsen, E. (2023). Forecast selection and representativeness. Management Science, 69(5), 2672-2690.

Pintor, M., Demetrio, L., Sotgiu, A., Melis, M., Demontis, A., & Biggio, B. (2019). secml: A Python Library for Secure and Explainable Machine Learning. arXiv preprint arXiv:1912.10013.

Ricciardi, V., Ramankutty, N., Mehrabi, Z., Jarvis, L., & Chookolingo, B. (2018). An open-access dataset of crop production by farm size from agricultural censuses and surveys. Data in brief, 19, 1970-1988.

Sharma, R., Singh, S. N., & Khatri, S. (2019). Data mining classification techniques - comparison for better accuracy in prediction of cardiovascular disease Data mining classification techniques – comparison for better accuracy in prediction of cardiovascular disease Richa Sharma * and Sujata Khatri. February 2020. https://doi.org/10.1504/IJDATS.2019.103756.

Siddiq, M. (2023). Exploring the Role of Machine Learning in Contact Tracing for Public Health: Benefits, Challenges, and Ethical Considerations. American Journal of Economic and Management Business (AJEMB), 2(3), 99-110.

Stančin, I., & Jović, A. (2019). An overview and comparison of free Python libraries for data mining and big data analysis. 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 977–982.

Subasi, A. (2020). Practical Machine Learning for Data Analysis. Academic press.

Varoquaux, G., & Colliot, O. (2023). Evaluating machine learning models and their diagnostic value. Machine Learning for Brain Disorders, 601-630.

Vosoughkhosravi, S., Dixon-Grasso, L., & Jafari, A. (2022). The impact of LEED certification on energy performance and occupant satisfaction: A case study of residential college buildings. Journal of Building Engineering, 59, 105097.