# **Methods for Making AI Systems More Transparent and Interpretable to Facilitate Accountability**

**A PROJECT REPORT**

**Submitted by**

A Shree Vyshnavi

Vedashri Marichetty

*in partial fulfillment for the award of the degree*

*of*

**Btech(Hons)**

*in*

**Computer Science**



**School of Computer Science and Engineering**

**RV University**

**RV Vidyaniketan,8th Mile, Mysuru Road, Bengaluru, Karnataka, India - 562112**

**December & 2024**

i

**DECLARATION**

I, A Shree Vyshnavi(1RVU22CSE001), student fifth semester B.Tech and Vedashri Marichetty(1RVU23CSE533), student third semester B.Tech**,** in **Computer Science & Engineering,** at School of Computer Science and Engineering, **RV University,** hereby declare that the project work titled “**Methods for Making AI Systems More Transparent and Interpretable to Facilitate Accountability**” has been carried out by us and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Computer Science & Engineering** during the academic year **2023-2024**. Further, the matter presented in the project has not been submitted previously by anybody for the award of any degree or any diploma to any other University, to the best of our knowledge and faith.

Name: A Shree Vyshnavi

Vedashri Marichetty Signature

USN: 1RVU22CSE001

1RVU23CSE533

Place: Bangalore

Date: December 2024



**School of Computer Science and Engineering**

RV University

RV Vidyaniketan,8th Mile, Mysuru Road, Bengaluru, Karnataka, India – 562112

ii

**CERTIFICATE**

This is to certify that the project work titled **“Methods for Making AI Systems More Transparent and Interpretable to Facilitate Accountability''** is performed by Shree Vyshnavi(1RVU22CSE)Vedashri Marichetty(1RVU23CSE533)**,** a bonafide students of Bachelor of Technology at the School of Computer Science and Engineering, RV university, Bangaluru in partial fulfillment for the award of degree Bachelor of Technology in Computer Science & Engineering , during the Academic year **2020-2021**.

**Prof. Santhosh S Nair Dr. Sudhakar KN**  **Dr. G Shobha**

**Guide**

Assistant Professor Head of the Department Dean

SOCSE SOCSE SOCSE

RV University RV University RV University

Date: Date: Date:

Name of the Examiner Signature of Examiner

1.

2.

iii

**ACKNOWLEDGEMENT**

It is a great pleasure for us to acknowledge the assistance and support of a large number of individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to the School of Computer Science and Engineering, RV University, for providing us with a great opportunity to pursue our Bachelor’s Degree in this institution.

In particular we would like to thank Dr. G Shobha, Dean, School of Computer Science and Engineering, RV University, for her constant encouragement and expert advice.

It is a matter of immense pleasure to express our sincere thanks to Dr. Sudhakar KN, Head of the department, Computer Science & Engineering University, for providing right academic guidance that made our task possible.

We would like to thank our guide Santhosh S. Head of the Department Dept. of Computer Science & Engineering, RV University, for sparing his valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.

We are also grateful to our family and friends who provided us with every requirement throughout the course.

We would like to thank one and all who directly or indirectly helped us in completing the Project work successfully.

|  |  |
| --- | --- |
| Date: December 2024 | Shree Vyshnavi  Vedashri Marichetty |
| Place: Bangalore | 1RVU22CSE001  1RVU23CSE533 |
|  | 5th sem - BTech(Hons)  3rd sem - BTech(Hons) |

iv

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
|  | **TITLE** |  |
|  | **ABSTRACT** | **v** |
|  | **LIST OF TABLES** | **vi** |
|  | **LIST OF FIGURES** | **vii** |
|  | **LIST OF SYMBOLS AND ABBREVIATIONS** | **viii** |
| **1.0** | **INTRODUCTION** | **1** |
|  | 1.1 Sub section | 1 |
|  | 1.2 Sub section | 1 |
| **2.0** | **RELATED WORK** | **2** |
|  | 2.1 Sub section | 2 |
|  | 2.2 Sub section | 2 |
| **3.0** | **METHODOLOGY** | **3** |
|  | 3.1 Sub section | 3 |
|  | 3.2 Sub section | 3 |
| **4.0** | **IMPLEMENTATION** | **4** |
|  | 4.1 Sub section | 4 |
|  | 4.2 Sub section | 4 |
| **5.0** | **RESULT AND DISCUSSION** | **5** |
|  | 5.1 Sub section | 5 |
|  | 5.2 Sub section | 5 |
| **6.0** | **CONCLUSION** | **6** |
| **7.0** | **FUTURE SCOPE** | **7** |
|  | **REFERENCES** | **8** |
|  | **APPENDIX** | **9** |

**ABSTRACT**

This project aims to enhance the **transparency** and **interpretability** of AI systems in medical imaging, focusing on **COVID-19 detection** using **Chest X-ray images**. The methodology involves preprocessing the dataset by handling missing values, normalizing features, and splitting it into training and testing sets. A **Convolutional Neural Network (CNN)** model is developed to classify chest X-ray images, achieving reliable accuracy in identifying COVID-19 cases. To ensure interpretability, advanced techniques such as **SHAP (SHapley Additive exPlanations)** and **LIME (Local Interpretable Model-agnostic Explanations)** are applied. SHAP provides global insights into feature importance, while LIME generates localized explanations, highlighting the regions influencing individual predictions. The visualizations from SHAP and LIME improve understanding of the model’s decision-making process, increasing **accountability** and **trustworthiness**. The project demonstrates the integration of deep learning with interpretability tools to address critical healthcare challenges, ensuring that AI-driven predictions are both accurate and explainable. This approach paves the way for more ethical and transparent deployment of AI systems in sensitive domains.

v

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Title** | **Page No.** |
| Table 5.1 | Colour indication for Lime analysis | 20 |

vi

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No**. |
| Figure 3.1 | CNN Model Architecture | 14 |
| Figure 3.2 | Methodology | 14 |
| Figure 5.1 | Model Accuracy and loss | 18 |
| Figure 5.2 | Model interpretation using Shap | 19 |
| Figure 5.3 | Model interpretation using Lime | 19 |
|  |  |  |
|  |  |  |

vii

**LIST OF SYMBOLS AND ABBREVIATIONS**

|  |  |
| --- | --- |
| **Symbol** | **Explanation** |
| $ | Dollar |
|  |  |

viii

1. **INTRODUCTION**

Artificial intelligence has become a transformative force in many sectors, with healthcare being one of the most impacted. In recent years, AI-powered systems have shown great promise in assisting clinicians by automating tasks such as disease diagnosis, risk prediction, and treatment planning. Despite these advancements, the lack of transparency and interpretability in AI models has raised concerns among practitioners and policymakers. Trust in AI systems is particularly critical in healthcare, where decisions can have life-altering consequences.

The objective of this project was to address the interpretability challenges of AI systems, specifically deep learning models used for medical image analysis. By enhancing transparency and accountability, we aim to make these systems more reliable and acceptable for real-world deployment. The focus was on using a CNN for classifying chest X-ray images as normal or COVID-19 positive, leveraging advanced interpretability tools like SHAP and LIME. The results were visualized and monitored using TensorBoard to ensure clarity and traceability throughout the model development process.

1. **Related work**

The use of AI in healthcare, particularly deep learning models, has grown exponentially. However, the interpretability of these models remains a challenge. Traditional deep learning approaches treat the model as a black box, making it difficult to understand the reasoning behind predictions. Several studies have attempted to address these limitations through attention mechanisms and feature visualization techniques, but these methods often lack generalizability and scalability.

Recent advancements in interpretability tools such as SHAP and LIME have provided a new pathway to address these challenges. SHAP, based on Shapley values, offers a global and local explanation of feature contributions, while LIME provides localized insights by approximating the model’s behavior around a specific prediction.

In the context of medical image analysis, many existing implementations focus solely on improving accuracy without addressing the interpretability gap. This project builds upon prior efforts by integrating these tools with a CNN to ensure the system is both accurate and transparent, making it more suitable for real-world applications in healthcare.

1. **METHODOLOGY**

#### **Methodological Approach**

The project followed a structured approach involving dataset preparation, model development, interpretability integration, and performance monitoring. Each step was designed to balance the goals of high accuracy and transparency.

#### **Data Collection**

The CoronaHack Chest X-Ray dataset, sourced from Kaggle, was chosen for this project. The dataset contains labeled chest X-ray images categorized as normal or COVID-19 positive. This dataset was selected because of its relevance and accessibility for the project’s goals.

#### **Data Preprocessing**

1. **Handling Missing Values:** Missing or incomplete data entries were identified and addressed to ensure consistency.
2. **Normalization:** All images were resized and normalized to ensure they met the input requirements of the CNN model.
3. **Splitting Data:** The dataset was divided into training (80%) and testing (20%) subsets to validate the model’s generalizability.

#### **Methods of Analysis**

1. **Model Architecture:** A CNN was selected due to its proven efficacy in image classification tasks. Layers were carefully designed to capture the intricate patterns in X-ray images, such as lung opacities.
2. **Interpretability Tools:**
   * SHAP was applied to understand the contributions of each feature to the predictions, offering both global and local explanations.
   * LIME was used to explain individual predictions, enabling a better understanding of localized model behavior.
3. **Visualization:** TensorBoard was employed to monitor and visualize model training, loss, accuracy, and other performance metrics in real-time.

#### **Evaluation of Methodological Choices**

The chosen combination of CNN, SHAP, and LIME ensured a balance between predictive performance and interpretability, making the model suitable for sensitive applications like healthcare.



Figure 3.1: CNN Model Architecture

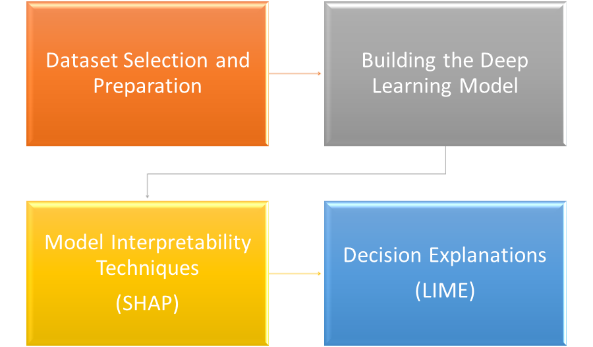


Figure 3.2: Methodology

1. **IMPLEMENTATION**

#### Step 1: Dataset Preparation

* Imported the Chest X-ray Images dataset from Kaggle.
* Handled missing values in the dataset.
* Encoded categorical variables and normalized numerical features to ensure uniform data distribution.
* Split the dataset into training and testing sets to evaluate model performance effectively.

Step 2: Model Development

* Selected Convolutional Neural Network (CNN) for its proven effectiveness in medical image classification.
* Designed the network architecture, including layers such as convolution, pooling, and fully connected layers.

model = Sequential([

    Conv2D(32, (3, 3), activation='relu', input\_shape=(150, 150, 3)),

    MaxPooling2D((2, 2)),

    Conv2D(64, (3, 3), activation='relu'),

    MaxPooling2D((2, 2)),

    Conv2D(128, (3, 3), activation='relu'),

    MaxPooling2D((2, 2)),

    Flatten(),

    Dense(512, activation='relu'),

    Dropout(0.5),

    Dense(1, activation='sigmoid')

])

* Configured appropriate optimizers (Adam) and loss function (binary\_crossentropy) for binary classification tasks.
* Trained the model using the preprocessed training data and validated its performance on the test set.

#### Step 3: Model Evaluation

* Monitored model performance using accuracy and loss metrics for both training and validation datasets.
* Plotted the accuracy and loss curves to analyze learning behavior and check for signs of overfitting or underfitting.

#### Step 4: Interpretability Techniques

* Applied SHAP (SHapley Additive exPlanations) to provide global feature importance and visualize how specific image regions influenced the predictions.
* Utilized LIME (Local Interpretable Model-agnostic Explanations) to generate localized explanations, highlighting regions of the X-ray images that contributed positively or negatively to individual predictions.
* Visualized SHAP and LIME outputs to enhance the interpretability of the CNN model’s decision-making process.

#### Step 5: Results Analysis

* Analyzed SHAP and LIME visualizations to identify the key features and regions of chest X-ray images influencing predictions.

import lime

from lime import lime\_image

from skimage.segmentation import mark\_boundaries

# Initialize the LIME explainer

explainer = lime\_image.LimeImageExplainer()

# Select a sample image from the validation set

sample\_image, \_ = next(validation\_generator)

sample\_image = sample\_image[0]

# Generate explanation

explanation = explainer.explain\_instance(sample\_image, model.predict, top\_labels=1, hide\_color=0, num\_samples=1000)

# Display the explanation

temp, mask = explanation.get\_image\_and\_mask(label=0, positive\_only=False, num\_features=10, hide\_rest=False)

plt.imshow(mark\_boundaries(temp, mask))

plt.title("LIME Explanation")

plt.show()

import shap

import matplotlib.pyplot as plt

# Define the background dataset for SHAP

background, \_ = next(train\_generator)

background = background[:100]

# Create an explainer

explainer = shap.DeepExplainer(model, background)

# Select a sample image from the validation set

sample\_images, \_ = next(validation\_generator)

sample\_images = sample\_images[:10]

# Compute SHAP values for a sample from the validation set

shap\_values = explainer.shap\_values(sample\_images)

# Plot SHAP values for the first 10 samples in the validation set

for i in range(10):

    plt.figure()

    # Display the original image, not the stacked one

    shap.image\_plot([shap\_values[j][i] for j in range(len(shap\_values))],

                    sample\_images[i], # Pass a single image here

                    show=False)

    plt.title(f"SHAP values for sample {i}")

    plt.show()

* Evaluated the model's accuracy and reliability through performance metrics and interpretability insights.

#### Step 6: Conclusion

* Ensured the AI system was both accurate and interpretable, enabling stakeholders to understand and trust the predictions.
* Discussed how the integration of interpretability tools enhances the accountability and transparency of AI systems in healthcare applications.

1. **RESULT AND DISCUSSION**

Our study demonstrated the usefulness of transparency and interpretability criteria in enhancing the reliability of our deep learning model for COVID-19 detection from chest X-ray images. The convolutional neural network (CNN) functioned consistently and with a respectable degree of accuracy when it was first trained using a test set. To comprehend the model's predictions, we employed SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations). SHAP values provided a comprehensive understanding of feature importance and demonstrated how particular regions of the chest X-rays significantly influenced the model's predictions. LIME explanations, which showed the areas within certain images that influenced specific projections, offered specific insights. The model's decision-making process was clarified and made more clear and understandable with the use of the SHAP value and LIME explanation visualisations.

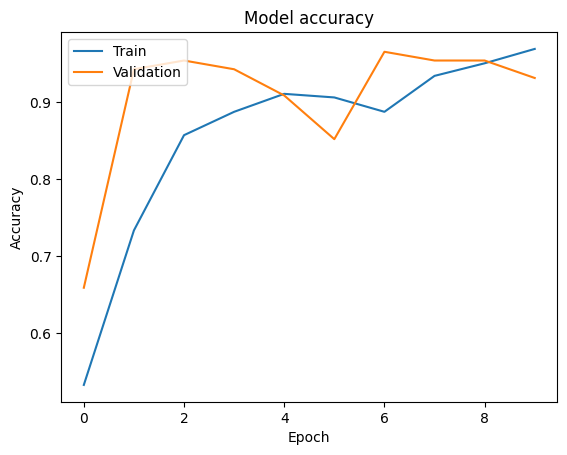


Figure 5.1: Model Accuracy and loss

Figure 5.1 depicts the Accuracy and Loss of the model on both the training and validation datasets over a series of 10 epochs.

Training Accuracy/Loss (blue line): The accuracy of the model on the training data improves steadily over the epochs, starting from around 0.6 and reaching close to 0.95. This indicates that the model is learning well from the training data. The loss on the training data starts high, around 1.0, and decreases steadily over the epochs, reaching a low of around 0.1. This indicates that the model is learning and improving its performance on the training data.

Validation Accuracy/Loss (orange line): The accuracy on the validation data also starts high, rapidly increasing and fluctuating between epochs 2 and 9, before stabilizing slightly below 0.95. The validation loss also starts high and decreases rapidly, with some fluctuations, stabilizing around 0.1. The fluctuations are normal and indicate variability in the validation dataset.

The graph suggests that the model is performing well on both the training and validation datasets, with high accuracy and no significant signs of overfitting or underfitting. The fluctuations in the validation accuracy are normal and can be attributed to variations in the validation data. Overall, both training and validation losses decrease over time, showing that the model is learning effectively. The final low loss values for both training and validation sets suggest that the model has achieved good generalization and is not overfitting.

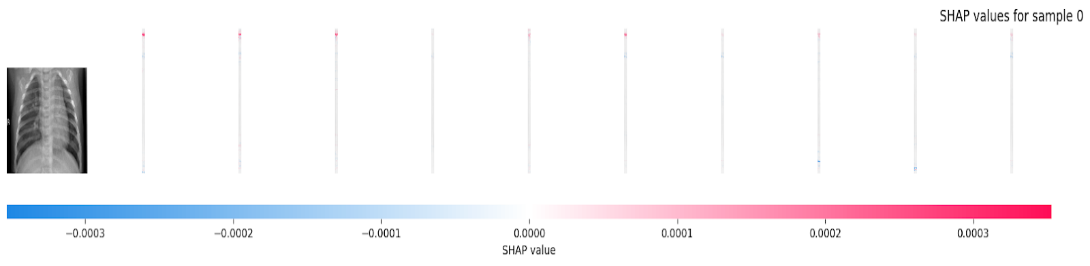


Figure 5.2: Model interpretation using Shap

Figure 5.2 shows SHAP values for a sample image, ranging from -0.003 to 0.003, indicating the contribution of each feature to the model's prediction. Features with negative SHAP values (up to -0.003) slightly decrease the model's confidence in its prediction, while those with positive SHAP values (up to 0.003) slightly increase it. The small magnitude of these values suggests a subtle impact by each feature. This visualization helps in understanding which features influence the model's decision and in what direction, thereby enhancing the transparency and interpretability of the AI system.

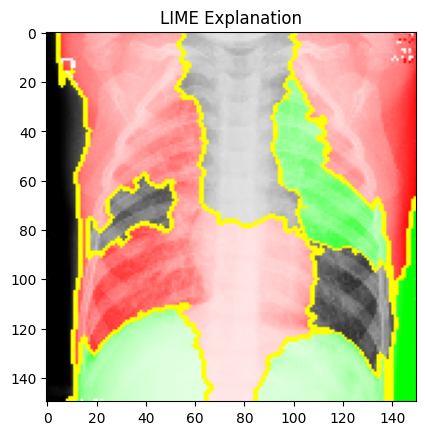


Figure 5.3: Decision explanation using Lime

Figure 5.3 and Table 5.1 presents the LIME explanation visualization, highlighting the regions of a chest X-ray image most influential in the model's prediction. Red areas indicate regions that contribute positively to the prediction, strongly supporting the model's decision, while green areas represent regions that negatively impact the prediction, opposing the model's decision. Yellow outlines show the superpixel segmentation used by LIME to divide the image into interpretable chunks. This visualization helps to understand which parts of the image the model is focusing on to make its predictions, thereby enhancing the interpretability and transparency of the model.

|  |  |
| --- | --- |
| **Colour** | **Indication** |
| Red Areas | Indicate regions that contribute positively to the prediction, meaning these areas strongly support the model's current decision. |
| Green Areas | Represent regions that negatively impact the prediction, meaning these areas oppose the model's decision. |
| Yellow Outlines | These boundaries show the superpixel segmentation used by LIME to divide the image into interpretable chunks. |

Table 5.1 : Colour indication for Lime analysis

1. **CONCLUSION**

With a focus on COVID-19 detection utilising chest X-ray images, the study provided a comprehensive approach to enhance the openness and interpretability of AI systems. We showed how deep learning approaches can enhance the accountability and reliability of AI models when combined with SHAP and LIME for interpretability. The visual assistance provided by SHAP and LIME helped to explain key elements and clarify some forecasts, facilitating a better understanding of the model's decision-making process. This helped identify any potential biases in the model's behaviour and validate the model's predictions. Future research should prioritise refining these interpretability methodologies and exploring their use in other important domains. Additionally, it is essential to develop standardized protocols for integrating transparency and accountability measures in AI systems to ensure their responsible and ethical deployment in real-world applications.

1. **FUTURE SCOPE**

1. **Integration with Real-World Healthcare Systems**

* Deploy the model in hospitals and diagnostic centers to assist radiologists in detecting COVID-19 and other respiratory diseases from chest X-ray images.
* Develop a user-friendly interface for seamless integration into existing **healthcare management systems**.

2. **Expansion to Other Medical Conditions**

* Extend the model to detect a broader range of diseases such as **pneumonia**, **tuberculosis**, and **lung cancer** using chest X-ray or CT images.
* Incorporate multi-modal data (e.g., clinical reports and patient history) to improve diagnostic accuracy.

3. **Improved Interpretability Methods**

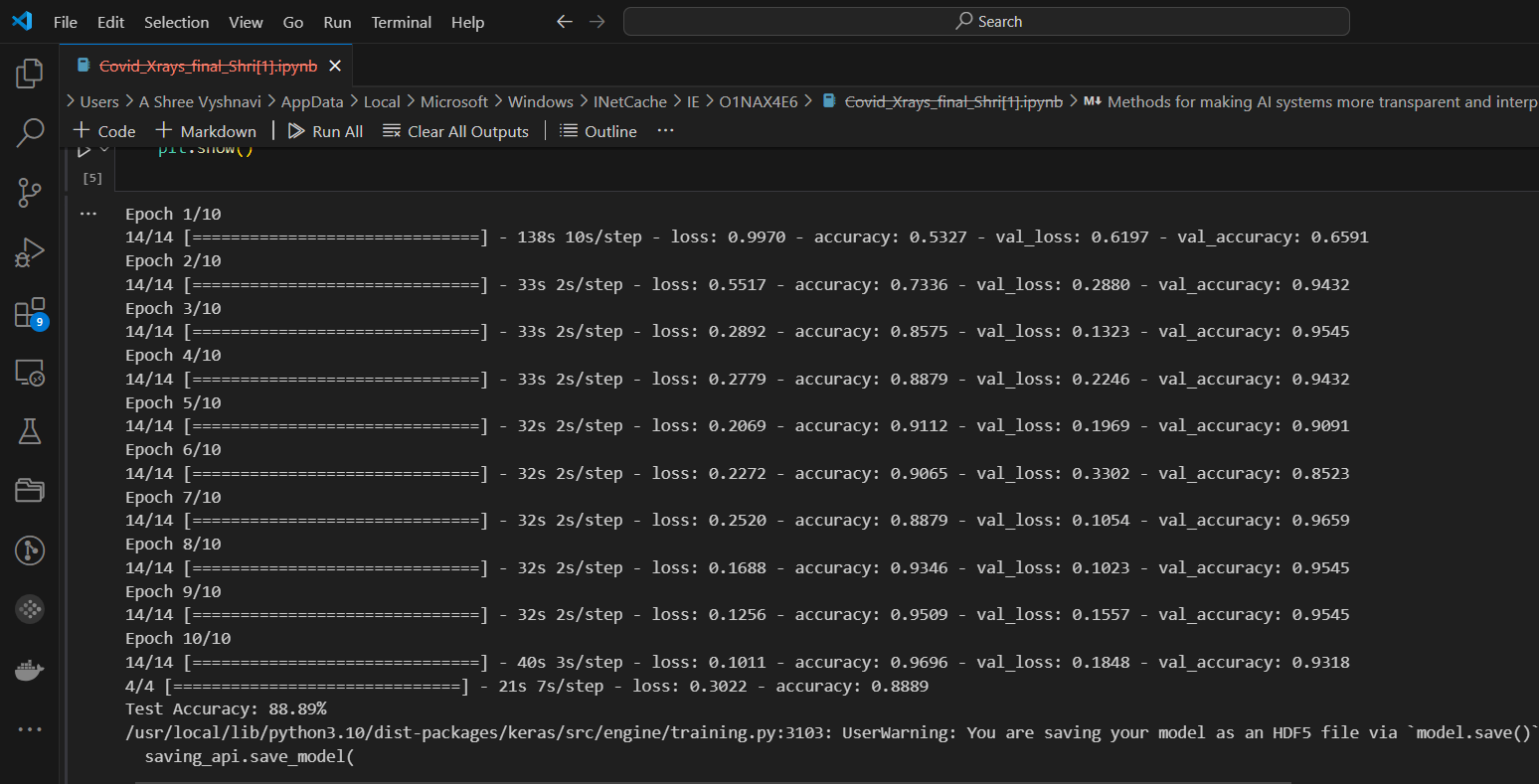
* Explore advanced interpretability techniques such as **Grad-CAM** or **Integrated Gradients** to provide more precise and intuitive explanations of model predictions.
* Combine SHAP and LIME with real-time visualization tools to enhance usability for medical professionals.

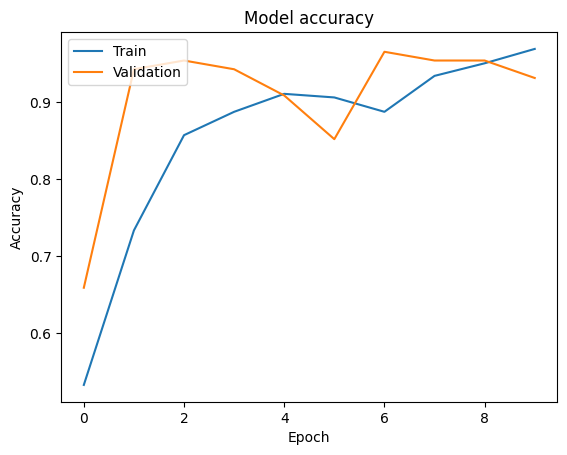
**REFERENCES**

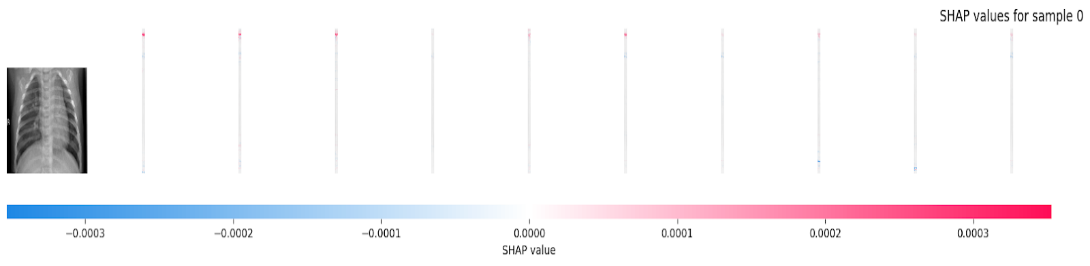
1. Lipton, Z. C. (2016). The Mythos of Model Interpretability. ICML Workshop on Human Interpretability in Machine Learning (WHI).
2. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
3. Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. arXiv preprint arXiv:1409.0473.
4. Lundberg, S. M., & Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. Advances in Neural Information Processing Systems.
5. Mittelstadt, B. D., Russell, C., & Wachter, S. (2019). Explaining Explanations in AI. Proceedings of the Conference on Fairness, Accountability, and Transparency.
6. Shah, V., & Konda, S. R. (2021). Neural Networks and Explainable AI: Bridging the Gap between Models and Interpretability. International Journal of Computer Science and Technology, 5(2), 163-166. DOI: 10.5281/zenodo.10779335.

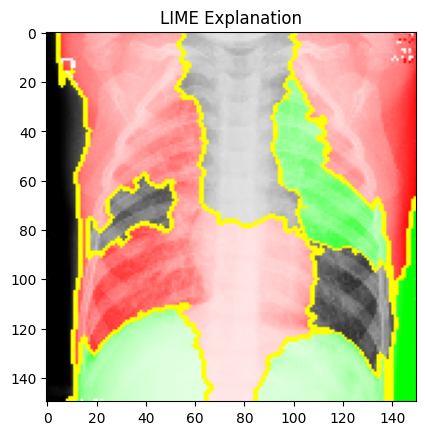
**APPENDIX**

Screen shot:









Source code :

#install required libraries

pip install numpy pandas matplotlib tensorflow keras shap lime

from google.colab import drive

drive.mount('/content/drive')

import pandas as pd

import os

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

import matplotlib.pyplot as plt

# Load the metadata

metadata\_path = '/content/drive/My Drive/summer\_internship\_ai/Chest\_xray\_Corona\_Metadata.csv'

metadata\_df = pd.read\_csv(metadata\_path)

# Assuming the metadata contains columns 'X\_ray\_image\_name' and 'Label'

metadata\_df['filename'] = metadata\_df['X\_ray\_image\_name']

metadata\_df['class'] = metadata\_df['Label'].apply(lambda x: 'normal' if x == 'Normal' else 'infected')

# Paths to your image directories

base\_dir = '/content/drive/My Drive/summer\_internship\_ai/Coronahack-Chest-XRay-Dataset/test'  # Adjust this to the actual base directory of your images

# Add the full path to the filename

metadata\_df['filename'] = metadata\_df['filename'].apply(lambda x: os.path.join(base\_dir, x))

train\_df = metadata\_df.sample(frac=0.7, random\_state=1)

temp\_df = metadata\_df.drop(train\_df.index)

val\_df = temp\_df.sample(frac=0.5, random\_state=1)

test\_df = temp\_df.drop(val\_df.index)

# Check the DataFrames

print(train\_df.head())

print("\n\n",val\_df.head())

print("\n\n",test\_df.head())

train\_datagen = ImageDataGenerator(rescale=1./255)

val\_datagen = ImageDataGenerator(rescale=1./255)

test\_datagen = ImageDataGenerator(rescale=1./255)

train\_generator = train\_datagen.flow\_from\_dataframe(

    train\_df,

    x\_col='filename',

    y\_col='class',

    target\_size=(150, 150),

    batch\_size=32,

    class\_mode='binary'

)

validation\_generator = val\_datagen.flow\_from\_dataframe(

    val\_df,

    x\_col='filename',

    y\_col='class',

    target\_size=(150, 150),

    batch\_size=32,

    class\_mode='binary'

)

test\_generator = test\_datagen.flow\_from\_dataframe(

    test\_df,

    x\_col='filename',

    y\_col='class',

    target\_size=(150, 150),

    batch\_size=32,

    class\_mode='binary',

    shuffle=False

)

# Verify data generators

print(f"Number of training samples: {train\_generator.samples}")

print(f"Number of validation samples: {validation\_generator.samples}")

print(f"Number of test samples: {test\_generator.samples}")

#define the model

model = Sequential([

    Conv2D(32, (3, 3), activation='relu', input\_shape=(150, 150, 3)),

    MaxPooling2D((2, 2)),

    Conv2D(64, (3, 3), activation='relu'),

    MaxPooling2D((2, 2)),

    Conv2D(128, (3, 3), activation='relu'),

    MaxPooling2D((2, 2)),

    Flatten(),

    Dense(512, activation='relu'),

    Dropout(0.5),

    Dense(1, activation='sigmoid')

])

#compile the model

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Train the model

history = model.fit(

    train\_generator,

    validation\_data=validation\_generator,

    epochs=10

)

# Evaluate the model on the test set

if test\_generator.samples > 0:

    evaluation = model.evaluate(test\_generator)

    print(f"Test Accuracy: {evaluation[1]\*100:.2f}%")

else:

    print("No test samples found.")

# Save the model

model.save('covid19\_xray\_model.h5')

# Plot training & validation accuracy values

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('Model accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Train', 'Validation'], loc='upper left')

plt.show()

# Plot training & validation loss values

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('Model loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'Validation'], loc='upper left')

plt.show()

import lime

from lime import lime\_image

from skimage.segmentation import mark\_boundaries

# Initialize the LIME explainer

explainer = lime\_image.LimeImageExplainer()

# Select a sample image from the validation set

sample\_image, \_ = next(validation\_generator)

sample\_image = sample\_image[0]

# Generate explanation

explanation = explainer.explain\_instance(sample\_image, model.predict, top\_labels=1, hide\_color=0, num\_samples=1000)

# Display the explanation

temp, mask = explanation.get\_image\_and\_mask(label=0, positive\_only=False, num\_features=10, hide\_rest=False)

plt.imshow(mark\_boundaries(temp, mask))

plt.title("LIME Explanation")

plt.show()

import shap

import matplotlib.pyplot as plt

# Define the background dataset for SHAP

background, \_ = next(train\_generator)

background = background[:100]

# Create an explainer

explainer = shap.DeepExplainer(model, background)

# Select a sample image from the validation set

sample\_images, \_ = next(validation\_generator)

sample\_images = sample\_images[:10]

# Compute SHAP values for a sample from the validation set

shap\_values = explainer.shap\_values(sample\_images)

# Plot SHAP values for the first 10 samples in the validation set

for i in range(10):

    plt.figure()

    # Display the original image, not the stacked one

    shap.image\_plot([shap\_values[j][i] for j in range(len(shap\_values))],

                    sample\_images[i], # Pass a single image here

                    show=False)

    plt.title(f"SHAP values for sample {i}")

    plt.show()

GitHub Link: [ashreevyshnavi/Covid-detection-transparency-and-Interpretable](https://github.com/ashreevyshnavi/Covid-detection-transparency-and-Interpretable)