A project report on

EXPLORING EXPLAINABLE AI

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology

In

Computer Science Engineering

By

PRATYUSH ROKADE (18BCD7091)

L. SATYAJIT (18BCD7143)

VIT-AP UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

VIT-AP UNIVERSITY

AMARAVATI – 522237

June, 2022

DECLARATION

I hereby declare that the thesis entitled “EXPLORING EXPLAINABLE AI” submitted by me, for the award of the degree Bachelor of Technology in Computer Science Engineering VIT is a record of bonafide work carried out by me under the supervision of Guide Name

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Amaravati

Date: 16 June, 2022 Signature of the Candidate Signature of the Candidate

CERTIFICATE

This is to certify that the Senior Design Project titled "**Exploring Explainable AI**" that is being submitted by **Pratyush Rokade (18BCD7091)** and **L. Satyajit (18BCD7143)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Dr. BKSP Kumar Raju Alluri
Guide

The thesis is satisfactory

Internal Examiner

Internal Examiner

Approved by

PROGRAMME CHAIR
B. Tech. CSE

DEAN
School of Computer Science and Engineering

ABSTRACT

The widespread advancements seen in computation power and data availability have facilitated Machine Learning algorithms like Deep Neural Networks (DNNs) to achieve almost human-level accuracy in various tasks, from Natural Language Processing (NLP) to Computer Vision. However, this escalation in accuracy, owing to model complexity, results in a major decline in interpretability. Despite the applications of DNNs being countless, certain critical use cases which demand models with transparent decision-making processes, still hesitate to implement such models due to the "black-box" nature of these models. The algorithms under Explainable Artificial Intelligence aim to overcome the barrier imposed by black-box models. These algorithms enhance user-trust and model transparency by attempting to decipher the underlying decision-making processes that result in the subsequent prediction. Limited research work aimed at developing quantitative approaches to evaluate the “explanations” obtained from Explainable AI techniques prompted us towards formulating an approach, comprising four novel metrics - Coherence, Correctness, Completeness, Congruency, to gain quantitative insights into these algorithms. This approach is implemented in evaluating the explanations obtained when Explainable AI is deployed in medical, synthetic media and person re-identification domains. The obtained results provide sufficient information to quantitatively estimate the explanations obtained in these critical use-cases.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr. BKSP Kumar Raju Alluri, Assistant Professor, School of Computer Science and Engineering, VIT-AP, for his constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavour. My association with him is not confined to academics only, but it is a great opportunity on my part to work with an intellectual and expert in the field of Data Science.

I would like to express my gratitude to the Chancellor, Vice President, Vice Chancellor, and the Dean of VIT-AP, for providing an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to Dr. Mehfooza Munavar Basa, Programme Chair, B. Tech CSE-DA, all teaching staff and members working as members of our university for their not-self-centred enthusiasm coupled with timely encouragement showered on me with zeal, which prompted the acquisition of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last, but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: Amaravati	**Pratyush Rokade**	**L. Satyajit**
Date: 16 June, 2022	**Name of the student**	**Name of the student**

TABLE OF CONTENTS

List of Figures

List of Tables

List of Acronyms

CNN	Convolutional Neural Network
DNN	Deep Neural Network
XAI	Explainable Artificial Intelligence
LIME	Local Interpretable Model-agnostic Explanations
SHAP	SHapley Additive exPlanations
IG	Integrated Gradience
Grad-CAM	Gradient-weighted Class Activation Mapping
TCAV	Testing with Concept Activation Vectors
SCS	System Causability Scale
DeepLIFT	Deep Learning Important Features

Chapter 1

Introduction

The rapid proliferation of ML/AI technologies resulting from abundant data and high computational power, ensured such models to be pervasive and entrenched in our day-to-day life. With such widespread integration of these models, humans have started relying on such "black-box" models to make vital decisions. Some use-cases like autonomous vehicle development and the medical field where human lives are at stake, prioritize model transparency owing to the sensitivity and the importance of their predictions. In such cases where the computer vision model is responsible for estimating the various factors while driving a vehicle, or in detecting the severity of a disease, even minutely inaccurate predictions can prove fatal. DNNs and its numerous variations regardless of being capable of providing utmost accuracy in a wide range of use-cases, are categorized as black-box models which proves to be an impediment towards demystifying the basis of the predictions of such models. This makes us incapable of understanding the reasoning behind the predictions, further inhibiting us from verifying the correctness of such predictions and if detected, developing modifications to counter any such fallacies.

Explainable Artificial Intelligence emerged as a solution to increase the interpretability of such highly accurate black-box models. Explainable AI algorithms improve model transparency and attempt at improving interpretability by providing insights into the model's decision-making process. "Explanations" provided by such XAI algorithms make the underlying working of the black-box models easily comprehensible to humans by highlighting the influential input features.

In recent years, research in the Explainable AI domain has made tremendous headway and countless algorithms have been formulated to decode the complex underlying processes. Model interpretability is improved by making such black-box models more comprehensible to humans. Understanding the underlying concepts of such XAI algorithms provides an insight into the obstacle for obtaining quantitative estimation to successfully evaluate and compare these algorithms.

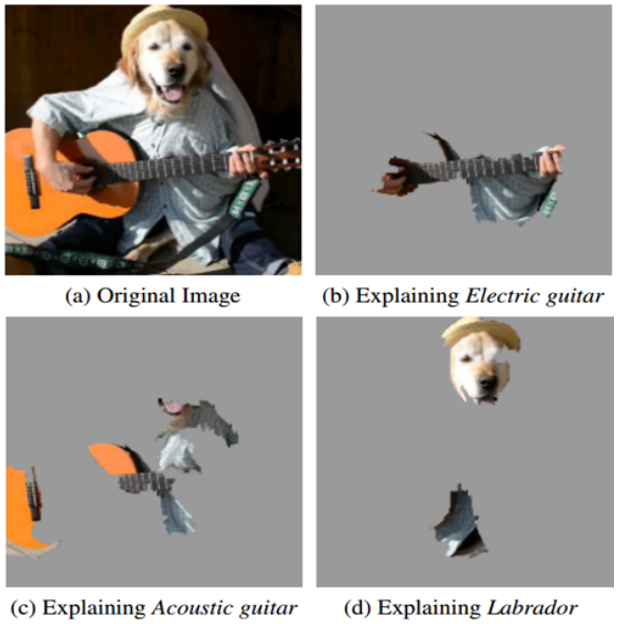


Figure 1. LIME XAI

Local Interpretable Model-Agnostic Explanations (LIME) Explainable AI algorithm [10] provides local model interpretability by approximating the underlying complex model on a local scale with an interpretable one on the assumption that at a local level, complex models exhibit linearity. This in turn enhances explainability by minimizing model complexity by estimating the prediction of a specific local instance. However, under the current implementation, the assumption that complex models cannot be simplified by approximating local behaviour using non-linear models might be a pitfall in cases where linear models prove to be incapable of providing good estimations of the model on a local scale. Interpretable explanations refer to the comprehensible nature in which the outcome of the algorithm relating to the relevant features is presented to the user. Model-agnosticism ensures consistency of results of the XAI algorithm across any given supervised learning ML models. The type of perturbations performed on the input features to observe changes in the prediction which helps explain the prediction depend on the use case.

Integrated Gradients (IG) technique [13] strives to suffice the two postulates of Sensitivity and Implementation Invariance. Sensitivity axiom dictates that a nonzero value be imparted to any feature which results in a difference in prediction regardless of the number of distinctions between the input and the corresponding baseline. The desired insensitivity of the attributions is captured when the function implemented by the deep neural network is mathematically independent of some variable. The attribution value of that specific variable is then considered to be zero. Implementation Invariance implies that essentially two networks are considered functionally equivalent if the outputs remain similar even when the

entire input subset is taken into account, disregarding the variety of implementations. Additionally, attributions obtained from methods satisfying Implementation Invariance should always have identical attributions for any such functionally equivalent networks. Integrated Gradients offers a visual representation of the salient regions by backtracking the model's output to its inputs. This resulting visualization has no requirement of modifying the subsequent black-box architecture.

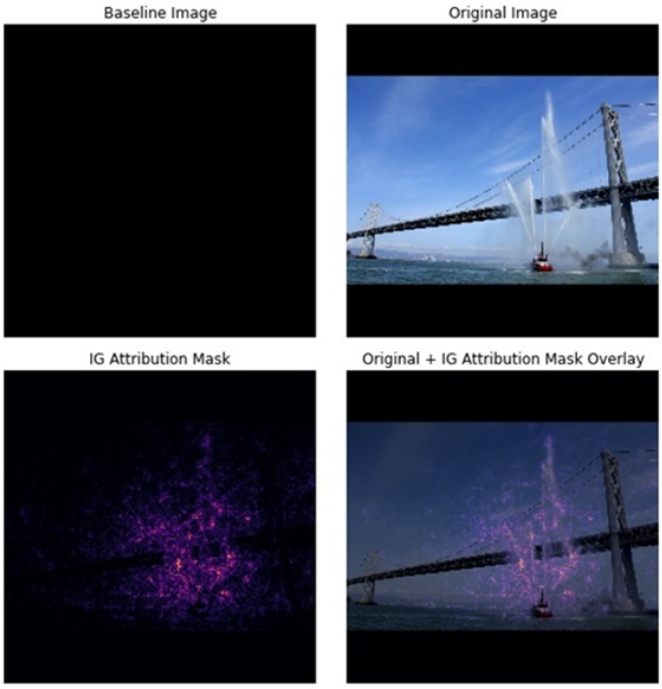


Figure 2. Integrated Gradients

Shapley Additive Explanation (SHAP) values [8] values encapsulate the influential features of the prediction with regard to the additive feature attribution method. For a certain instance, this method considers the average marginal contribution of an attribute across all the possible permutations combined into the model's prediction. Explanations of an instance can be obtained on a global scale by adding the SHAP values across every input feature. The surrogate nature of the SHAP model warrants model-agnostic explanations by noting the changes in prediction resulting from perturbing every entity in the attribute subset with a small value. The significance of an attribute is directly related to the change in prediction observed. The dissimilarity in values between the inputs and the predictions resulting from attribute contribution is captured by the SHAP values which are based on Shapley's properties of Symmetry - equally contributing attributes should get the same value, Dummy - attributes that do not contribute to the prediction at all are given a "0" value, and Additivity - attribute contributions sum up to the difference of the prediction and the average.

Gradient-weighted Class Activation Mapping (Grad-CAM) [11] is an Explainable AI algorithm standardized across various existing Convolutional Neural Network (CNN) architectures. Grad-CAM provides visual interpretability by superimposing a heatmap corresponding to the salient features among the attributes post-hoc model training. Gradients for any target attribute obtained from the last layer of the convolutional network are computed to further highlight salient features of the input image. These features help in predicting the concept by developing a coarse localization map. The authors of the paper singled out the ultimate layer of the CNN, reasoning that this layer corresponds to the optimum ratio of high-level visual features and detailed spatial information. Grad-CAM computes the significance of every neuron for a specific decision by calculating the gradient information contained in the final convolutional layer in order to distinguish the regions of the input which contribute significantly towards said prediction.

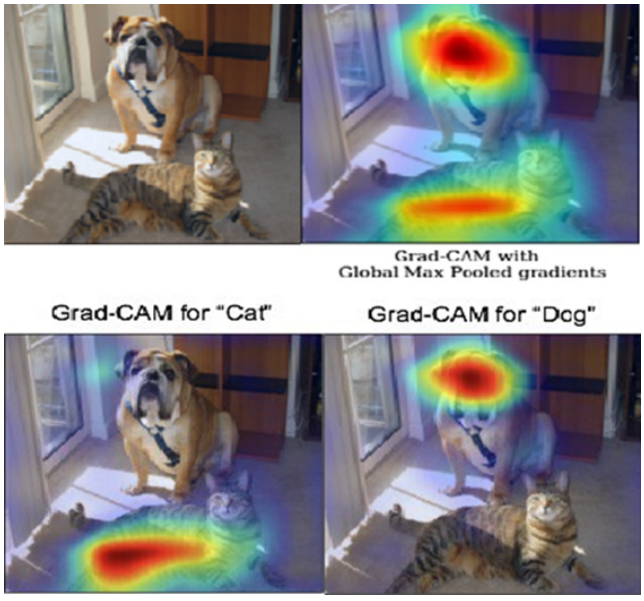


Figure 3. Grad-CAM

Deep Learning Important Features (DeepLIFT) [12] is a technique which is essentially based on the concept of backpropagation in neural networks. The distinction amid a neuron's activation and its reference activation is noted down to calculate the contribution score of that neuron. These scores make it possible to backtrack the contribution of all the neurons across every layer to each attribute of the input which corresponds to the output. DeepLIFT calculates importance scores by obtaining information even when the gradient is zero. The fundamental technique is predicated on the outcome being different from some reference output, which corresponds to the inputs being different from their reference inputs. DeepLIFT implements the concept of multipliers to penalize specific neurons owing to their difference in outputs. The paper defines multipliers as the ratio of the contribution of reference of the input neuron and the target neuron, to the referenced input neuron.

Testing with Concept Activation Vectors (TCAV) [7] intends to optimize the interpretability of a neural network by describing its intricate decision-making process in a comprehensible way to the user. The authors of the research recommend adopting TCAVs because they deploy directional derivatives to compute the scope of a user-defined characteristic in computing an accurate classification prediction. Sensitivity of a model towards a specific concept of a class can be quantified using TCAV for (a) a collection of instances that illustrate this concept, (b) dataset with labelled concept, and (c) trained network. Concept Activation Vector (CAV) is described as the vector orthogonal to a hyperplane distinguishing samples in the model's activations with and without a concept. The end objective of TCAV is to estimate the extent to which a concept was necessary for the prediction regardless of its involvement in the training phase. Current research work focused on developing evaluation rubrics for Explainable AI algorithms is clustered around the implementation of proxy methods and human intuition to further improve the robustness of these algorithms. However, approaches that make use of human intuition provide intuitive insights into the underlying working of the black box models rather than computing a quantifiable metric to evaluate their performance. The incapability of such approaches to provide robust quantitative estimates of the explanations obtained from the XAI algorithms fail to be unanimously accepted as metrics to quantify ExplainableAI algorithms.

- Despite human intuition spearheading the evaluation approaches for Explainable AI algorithms, they fail to provide concrete estimation of the explanations.
- Human interpretation falls short of ensuring consistent evaluation of the explanation since it depends on the end-user and the scope of their prior expertise in the specific field.
- Due to confirmation bias, humans are susceptible towards misinterpreting an explanation depending on what they anticipate from the prediction.

This could be misleading since the primary goal of the explanations is to demystify the underlying reasoning architecture.

1.1 MEDICAL DOMAIN

According to the World Health Organization's (WHO) World Cancer Report (2020) [15], cancer is one of the top causes of mortality, accounting for roughly 10 million deaths in 2020 (second only to cardiovascular disease). Cancer screening is a distinct and more sophisticated public health strategy

than other illnesses, requiring more resources, infrastructure, and coordination. Since human brain tumors are non-invasive, MRI is currently the best tool for their early identification. The interpretation of MRI,

on the other hand, is mostly based on the judgments of radiologists. Despite the widespread use of Deep Learning models in clinical decision-making, the lack of interpretability and transparency of algorithm-driven conclusions remains the most significant obstacle, particularly in medical contexts. This is a scenario where in the ideal world Explainable Artificial Intelligence (XAI) techniques could be implemented as a assistance tool to the radiologists and doctors as a solution to the current impediments posed by Artificial Intelligence. However in reality this is not being done as there is a lack of reliability on current XAI algorithms as well as the fact that brain tumor is a very sensitive issue. Any errors in detection of the tumor could lead to serious consequences, hence increasing the risk of involving Artificial Intelligence in this use case. Therefore, we aim to quantify the XAI results on suitable parameters in order to showcase the reliability of the current state of the art algorithms.

1.2 SYNTHETIC IMAGING DOMAIN

The name "deepfake" is derived from the underlying technology "deep learning," which is a sub-domain of artificial intelligence. Deep learning algorithms are used to swap faces in video and digital content to create realistic-looking false media. Deepfakes are the most well-known example of "synthetic media," which consists of pictures, sound, and video that look to have been made using traditional methods but were really created using sophisticated software. One of the most common methods is to capture video data from the two persons you're switching and process it on a powerful computer that you either own or (more likely) rent through a cloud service. The programme tries to learn how to recreate the face from all angles by comparing different pieces of footage. The software is fed a large amount of data, which it then utilizes to learn how to generate new data on its own. They are primarily based on autoencoders, but they can also be based on generative adversarial networks (GAN). Owing to its potentially dangerous effects, deep-fakes have generated worries throughout the world. It doesn't seem far off to imagine a gloomy future in which all kinds of digital media are possibly hacked and public faith in the government is waning. The matter might quickly escalate out of control if not handled with the necessary seriousness. Current Deep-Fake detection approaches try to precisely handle the problem at hand, but they may not be able to persuade a layperson of their dependability, resulting in a lack of public confidence. The creation of interpretable and simply explainable models is critical since the fundamental challenge is gaining the trust of humans. To accomplish this XAI algorithms are being implemented on Deep Fake images. We will formulate a

quantified approach to gain quantitative insights into the model. This will help us to assess reliability of the model to the human agents hence promote usability of XAI in such a sensitive use case.

1.3 PERSON RE-IDENTIFICATION DOMAIN

The process of associating images or videos of the same person taken from various angles and cameras is known as person re-identification. The key to the issue is to discover traits that describe a person. Many of the current models employ deeply trained models to extract features and achieve high performance. Due to its outstanding feature learning skills and fitting capability, some state-of-the-art approaches based on convolutional neural networks (CNNs) are proposed. When we want to assure security around a shopping mall, parking lot, university, or any other area, we use many cameras. We can trace a person's route and ensure that nothing unlawful or wrong is done by employing re-identification and tracking models. The primary worry with such technologies is privacy, but public surveillance is already functioning and active, so adding human tracking would make little difference. Vehicles and other items might be tracked as well. The road situation may be studied and improved in this way. This technology can also be used to track criminals which can assist the police investigation process to be much faster. However the police need to have a sense of reliability on these systems. The model needs to explain the reason for classifying a criminal with high precision. Hence implementing XAI algorithms on Person Re-Identification models will address the lack of interpretability as well as measure the correctness of the model.

Chapter 2

Related Work

The lack of agreement residing in the AI domain regarding the definition and approaches developed in evaluating the explanations obtained from the various XAI algorithms, in terms of a quantitative metric, stems from the fact that there has been minimal research work in this field of AI. Evaluation frameworks implementing human intuition to quantify Explainable AI algorithms depend largely on the in-field expertise of the user in the application pertaining to which the black box model is employed. Approaches which implement human perception to evaluate the correctness of the Explainable AI techniques do not offer a robust metric regarding the accuracy of the algorithm. However, these frameworks are insightful in understanding the impact of human intuition in Explainable AI quantification and cannot be overlooked.

In [3], the researchers emphasize that interpretability of a black box model is subjective rather than assuming it to be absolute in nature. The research therefore defines model interpretability in comparison with a base model. The framework proposed in this study establishes a comparative relationship between several interpretable procedures based on consistency and preciseness. The paper states that the extent of interpretation of a black box model is inversely proportional to the volume of data fed to the model. δ -Interpretability as defined in the paper can be stated as the capability of an Explainable AI algorithm to obtain such insights into the black box models that can help improve its performance. However, the explanations computed by the XAI algorithms should be comprehensible to the target model. δ computes the extent and impact of the interpretability evaluated from explanations of the XAI algorithm. Interpretability of the explanations along with its useful impact increases as the value of δ nears 0, on the other hand with $\delta > 1$, the impact of interpretability is construed as negative. Figure 4 visualizes their underlying concept. Their proposed framework can be understood in the following way, imagine the interpretable procedure to a teacher and the target model to be a student, with the observer being the student's parent. Here the teacher's capability can be evaluated by observing the change in performance she brings about in the student.

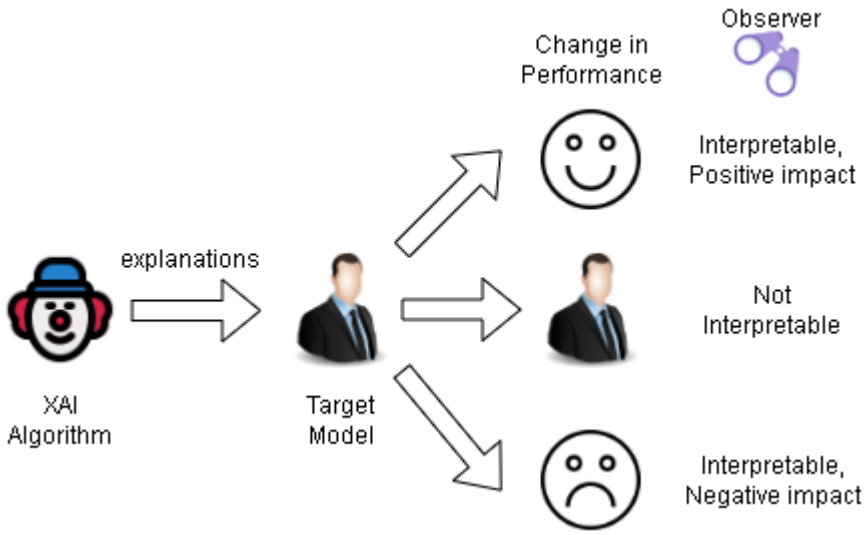


Figure 4. Visualizing the δ-Interpretability framework

In [4], the writers draw attention to the two definitions of interpretability – simulatability and "what if" native explainability. Simulatability refers to the capability of a user to efficiently implement the black box model on a specific input. "What if" native explainability assumes some prior expertise of the user in the preliminary prediction obtained from the model. This assumption facilitates the evaluation of a user's capability to precisely estimate the model's prediction with respect to domestic influence on the input. The study defines runtime operation on the simulation task as a parameter to gauge to estimate model interpretability. The authors substantiate this metric in the context of human perception. A human study involving 1000 volunteers provided extensive data to establish an indirect proportionality between the total number of operations and the volunteer's ability to evaluate local interpretable models. The user study results point towards the relationship that interpretability of a model decreases with increasing model complexity as decision trees perform better in terms of being locally interpretable than logistic regression models and neural networks. In case of decision trees, a clear inverse relationship can be observed between the total number of runtime operations and time, and between the accuracy and time. However, this trend fails to be observed in logistic regression models and neural networks, owing to the fact that noise might have been introduced due to the difficulty of the logistic regression and neural network simulation tasks.

In [9], the authors address the prevailing concern over poor interpretability of black box models by developing a framework (Figure 5) comprising the Predictive, Descriptive, and Relevant parameters to

evaluate interpretations obtained from the model's prediction. Predictive accuracy considers the flawed predictions of the model which directly affects the approximation of the Explainable AI technique. Descriptive accuracy captures the ability of the Explainable AI algorithm to demystify the underlying

processes of the convoluted network of the black box. Computed in the context of human perception, Relevancy is the measure to evaluate whether the particular evaluator can obtain insights from the explanation provided by the model. During model selection, practitioners often experience conflict due to the trade-off between predictive and descriptive accuracy. Higher predictive accuracy can be obtained from black box models, but on the other hand this inversely affects descriptive accuracy as model complexity increases. Relevancy, then comes into play as the deciding factor in the predictive and descriptive accuracy conflict. Model-based interpretability methods provide numerous approaches towards increasing descriptive accuracy by implementing a simpler model. The practitioner weighs under the chance of decreasing predictive accuracy, but depending on the data predictive accuracy might even remain unchanged. Post-hoc interpretability renders predictive accuracy invariant as these considerations are taken into account after the training phase of the model. Nevertheless, post-hoc interpretability methods could still enhance descriptive accuracy.

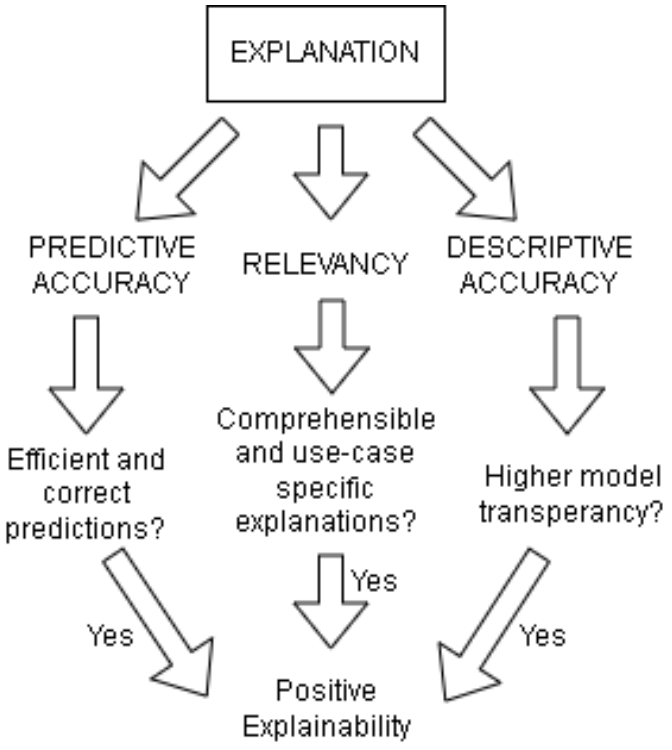


Figure 5. Visualizing the PDR framework

In [6], the authors attempt to define explainability and interpretability in the context of several domains corresponding to the comprehensive research work existing in various social sciences. Fields like

philosophy and psychology which try to interpret how humans construe and perceive explanations are considered. The paper hypothesizes that the explainability of an Explainable AI algorithm is dependent on the degree of feature interaction and the number of cognitive features of the respective black box model. An

expression for quantifying explainability is formulated by the authors to substantiate their hypothesis. However, these cognitive chunks sometimes display correlation and their interactions are not mutually exhaustive. Explainability (E), the proposed metric in this paper to quantify explanations provided by the XAI algorithms is inversely dependent on the interactions among the cognitive chunks. Interaction (I) ranges from 0 to 1, with improving explainability being observed on lowering interactions. For gaining further insights in the evaluation of the explanations, the number of cognitive chunks are broken down into the number of input cognitive chunks of the model and the number of output cognitive chunks included in the output subspace, which again are inversely proportional to explainability. Their experiments demonstrate evidence assisting their explainability quantification. However, this method which they implement is deemed to be an indirect approach. Newly constructed features provided the best explainability scores which was followed by scores obtained from domain related features, while original features experienced the least explainability scores.

In [1], the authors suggested three metrics concerning explanations for a black box model – explicitness/ intelligibility, faithfulness, and stability. Explicitness/ Intelligibility depicts the comprehensible and direct nature of the explanations. On the other hand, Faithfulness defines whether the pertinent accuracy really portrays “true” significance. While lastly, Stability checks whether the explanations remain consistent across similar attributes.

The authors in [5] formulated a system causability scale to qualitatively evaluate the explanations obtained from the Explainable AI model. The paper describes causability as the degree to which an explanation is intelligible for the end-user in the context of competency, efficacy, satisfaction, clarity and satisfaction. Explainability as defined by the authors, highlights relevant regions of the models which contributed to the specific prediction. Moreover, they formulate a System Causability Scale (SCS), using the Linkert scale which is similar to the System Usability Scale, and draws inspiration from the Framingham model. Their proposed SCS works towards determining to what extent an explanation itself is appropriate for the individual use case.

In [2], the authors have portrayed Pedestrian misalignment as a crucial challenge for a reliable person re-identification (re-ID) system. It is caused mostly by detector failures and stance fluctuations. Background noise will severely impede the feature learning and matching process if the alignment is poor. Their work proposes the posture invariant embedding (PIE) as a pedestrian descriptor to overcome this issue. To align

pedestrians to a standard posture, the PoseBox structure is first presented, which is created via pose estimation and affine transformations. Moreover, they create a PoseBox fusion (PBF) CNN architecture that takes the original picture, the PoseBox, and the pose estimation confidence as inputs to mitigate the effect of pose estimation mistakes and information loss during PoseBox building. The suggested PIE descriptor has been therefore defined as the retrieval task's fully connected layer of the PBF network. Market-1501, CUHK03, and VIPeR datasets are used in the experiments. They show that PoseBox alone can achieve reasonable re-ID accuracy, and that when combined with the PBF network, the learnt PIE descriptor can compete with state-of-the-art techniques.

In [14], the authors provide a unique image-based person identification challenge in this research. Traditional face-based person recognition systems have a poor tolerance for occluded situations, such as persons in a picture overlapping. They concentrate on an image captured by an above camera. Using an above camera eliminates the need for a camera to be installed in a specific area and eliminates the problem of obstructed pictures. They use background subtraction to identify the person's region in a taken image which then extracts four characteristics from the area: body size, hair colour, hairdo, and hair whorl.

Chapter 3

Proposed Methodology

With the aim focused towards obtaining robust quantifiable interpretations of the explanations provided by Explainable AI algorithms in an attempt to improve model interpretability by providing the reasoning behind the respective model predictions, we propose an approach to provide concrete and consistent evaluations of such XAI algorithms. Our proposed methodology, working towards quantifying the reliability of the explanations procured as outcome from these XAI algorithms, proceeds in the direction of building user trust in the black box models (Figure 6). The working ideology behind our proposed model is that salient regions, detected by XAI algorithms, itself should be sufficient enough to obtain approximately similar prediction accuracy, if not higher, than the accuracy obtained when the entire attribute set is provided to the model. Our model feeds the explanations of the input features obtained from the XAI algorithm to the black box model as input and notes its accuracy and prediction label. This information is then compared to the accuracy and prediction label computed when the original input feature subset is passed as input to the same black box model. Delivering the explanations will help us obtain quantitative assessment of model dependability, because the algorithm identifies the crucial features during model training. Initially, Prediction Accuracy (pa) is computed when the entire feature subset is passed to a black box model and the corresponding prediction class label is noted. The prediction is then utilized in computing the model's explanation and a feature subset. By sending the initial input subspace with its corresponding prediction to an Explainable AI algorithm, salient features of the original input are obtained. This obtained feature subset containing the most influential features in computing the prediction is then passed to the aforementioned black box model and the prediction label with its accuracy are observed and noted as Explanation Accuracy (ea). We formulate our four novel parameters with the two aforementioned accuracies (Prediction and Explanation) as base, which further helps compute the quantification metric. The related research in the field of evaluating Explainable AI provided us the inspiration for developing an evaluation methodology comprising the "4 C" parameters – (i) Coherence, (ii) Correctness, (iii) Completeness, and (iv) Congruency. This framework provides a quantifiable

understanding of the Explainable AI algorithms while facilitating assessment of the comparative characteristics of these algorithms by establishing a relationship between them. The novelty of this research lies in formulating quantified expressions for each of the proposed "4 C" metrics, which may be used to determine the robustness of explanations, hence enhancing model reliability of such black box algorithms.

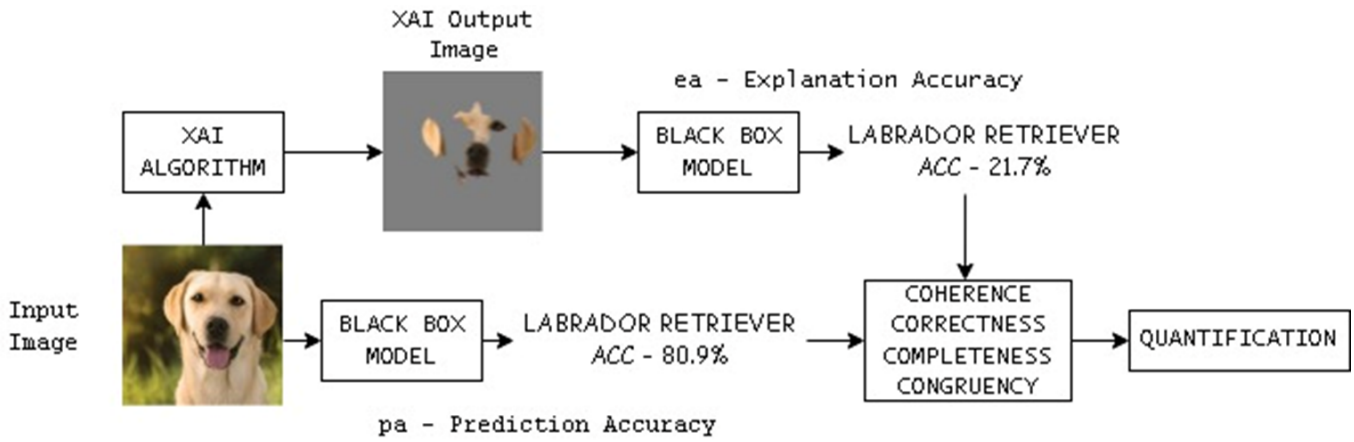


Figure 6. Visualizing the flow of the methodology proposed

3.1 FORMULATING 4C METRICS

Coherence focuses on calculating the consistency of the model by computing the absolute difference between the prediction and explanation accuracy. Moreover, Coherence captures the extent to which the explanations provided by XAI algorithms are truly influential towards the prediction.

$$\alpha_i = |pa - ea| \tag{1}$$

Whereas, pa = prediction accuracy; ea = explanation accuracy. α_i = coherence; Summing up the expression for α_i across all N inputs results in α.

$$\overline{\alpha} = \frac{\Sigma|pa-ea|}{N} \tag{2}$$

Correctness defines the reliability of the XAI algorithm in identifying and comparing the predicted class labels obtained from the prediction and explanation accuracy. The confusion matrix is employed as a base for adopting the sensitivity formula. The goal here however is to increase correctly classified positives whereas simultaneously minimizing incorrectly predicting samples as negatives. Correctness portrays the efficiency of the explanation in identifying the pertinent features which contributed the most to the prediction.

$$\beta = \frac{TP}{TP+FN} \tag{3}$$

Where, β =correctness; TP = total occurrences where the prediction label of the input is identical to the label obtained from the explanation accuracy; FN = total occurrences where the black box model predicted a different class label for the input than that for the explanation. Completeness computes the extent to which the explanation is proficient in representing the prediction score obtained when the entire input subset is

provided to the black box. This metric is calculated by multiplying the ratio between the explanation and prediction accuracy with hundred.

$$\gamma = \frac{ea \times 100}{pa} \tag{4}$$

Here, γ=completeness, pa=prediction Accuracy, ea= explanation accuracy.

Congruency (δ) measures the standard deviation of the difference observed in the predicted score of the model and the score obtained from the explanation across all samples. From equation (1), we obtain "α_i" the coherence for a particular input, aggregating this over all the samples and calculating the standard deviation offers the congruency of the explanations produced by the Explainable AI Models. Equation (2) provides the "α" coherence averaged across N samples.

$$\delta = \sqrt{\frac{\sum(\alpha_i - \bar{\alpha})^2}{N}} \tag{5}$$

Quantification combines the metrics – Correctness, Completeness, and Congruency – to acquire a succinct understanding of the explanations provided by the Explainable AI algorithms by computing a numerical value. We incorporate weights w1 in the below mentioned formula to penalize the Correctness parameter rigorously than the subsequent faults in Completeness and Congruency metrics. Correctness is prioritized owing to the fact that it corresponds to the cases wherein the explanation is incapable of identifying salient features for the particular class label which was predicted by the black box. The latter two parameters - Completeness and Congruency - are grouped under a single weight hyperparameter due to their similar impact on explainability. Additionally, Correctness and Completeness are directly proportional to Quantification whereas Congruency sustains an inverse trend. Due to the dependency observed in each of the aforementioned parameters to the Quantification value and their corresponding weights we propose the below equation.

$$Q = w_1 \bullet \beta + w_2 \bullet \frac{\gamma}{\delta} \tag{6}$$

Where Q=quantification; w1 , w2=weights (hyperparameter).

3.2 FRAMEWORK FOR BINARY PROBABILISTIC MODEL

The above framework works if the model employs a softmax distribution function to assign scores to classes from 0 to 1. However, the above framework falters if the model outputs a score between 0 and 1 which translates to the probability likelihood for a binary classification task. This encouraged us to tweak our existing framework to appropriately capture this change in model output. The algorithm we formulated (Figure. 7) assumes 0.5 as the neutral value and correspondingly rounds off the probability likelihood score to the nearest integer, either 0 or 1, which denotes the respective class in a binary classification task. To achieve this, prediction accuracy (pa) and explanation accuracy (ea) are updated before the "4C" metrics are computed. The updated accuracy score is the value obtained from computing the absolute difference between the corresponding accuracy (prediction or explanation) and the neutral value (0.5). Next, the predicted class label is compared to the actual class to identify whether the model's prediction was correct. For correctly classified samples, positive coherence (α_{pos}) and positive completeness (γ_{pos}) is calculated, while for incorrectly classified samples, negative coherence (α_{neg}) and negative completeness (γ_{neg}) is calculated. Since the updated values of prediction and explanation accuracy vary from 0 to 0.5, we formulate a way to penalize incorrect classifications by coming up with the two different approaches for calculating correct and incorrect classifications. The simple concept behind the two approaches visible in the algorithm below is that the difference in the probability likelihood scores of the two accuracies should be more for incorrect classifications than the correct classifications.

$$pa = | 0.5 - pa |$$

$$ea = | 0.5 - ea |$$

if Predicted == Actuals:

$$\alpha_{pos} = | pa - ea |$$

$$\gamma_{pos} = ea * 100 / pa$$

else:

$$\alpha_{neg} = | pa + ea |$$

$$pa = pa + 0.5$$

$$\gamma_{neg} = ea * 100 / pa$$

Figure 7. Algorithm to calculate the prediction accuracy, explanation accuracy, coherence and completeness

After having calculated and updated the prediction and explanation accuracy, which further aid in computing the positive and negative types of coherence and completeness. With the similar formulas employed in section 3.1, we proceed to formulating the similar metrics for a model yielding probabilistic results.

Positive and negative coherence averaged across N inputs amounts to the following two variations of Coherence.

$$\bar{\alpha}_{pos} = \frac{\Sigma \alpha_{pos}}{N} \qquad \bar{\alpha}_{neg} = \frac{\Sigma \alpha_{neg}}{N} \tag{7}$$

The total completeness is formulated by computing the two variations - positive and negative completeness – by taking the mean over all N samples.

$$\bar{\gamma}_{pos} = \frac{\Sigma \gamma_{pos}}{N} \qquad \bar{\gamma}_{neg} = \frac{\Sigma \gamma_{neg}}{N} \tag{8}$$

The two coherence variations are utilized in calculating the respective positive and negative congruency.

$$\delta_{pos} = \sqrt{\frac{\Sigma(\alpha_{pos_i} - \bar{\alpha}_{pos})^2}{N}} \quad \delta_{neg} = \sqrt{\frac{\Sigma(\alpha_{neg_i} - \bar{\alpha}_{neg})^2}{N}} \tag{9}$$

Correctness metric remains unchanged as it points to the true positive rate. However, we have included the Fault metric as the true negative rate from the confusion matrix to prioritize the number of negative samples which are correctly classified as negative.

$$\sigma = \frac{TN}{TN+FP} \tag{10}$$

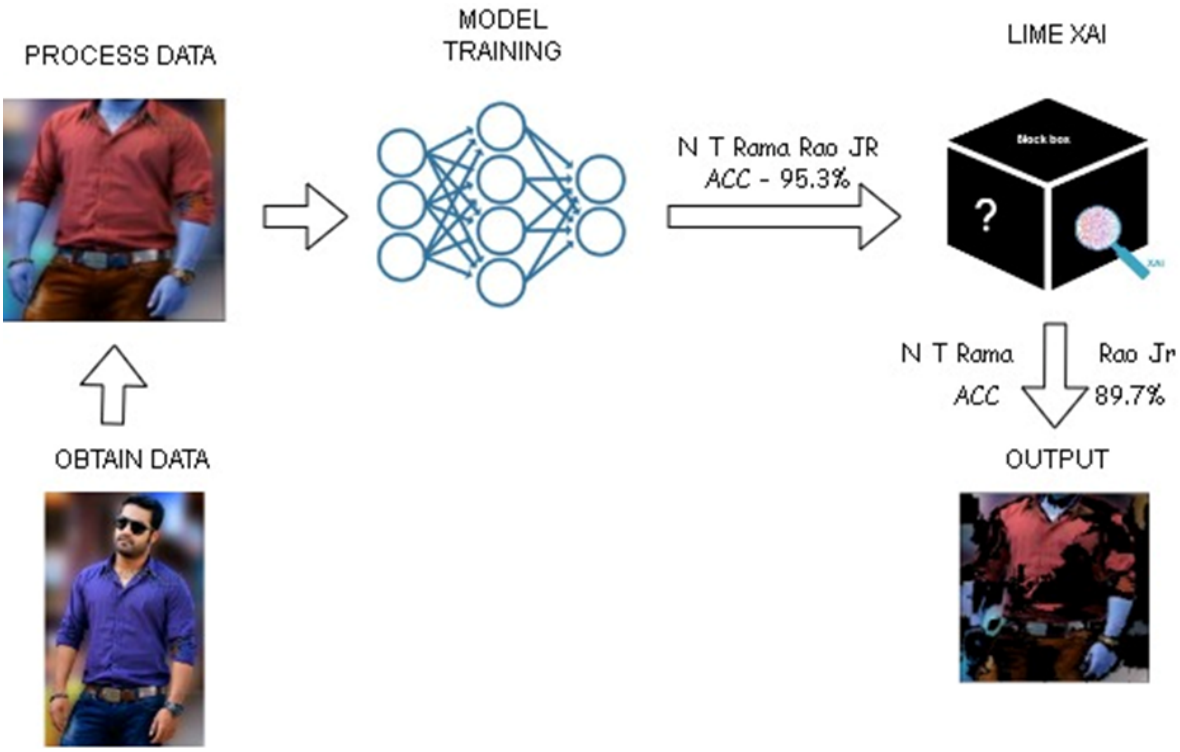
A different expression is formulated for Quantification owing to the variations in the framework and the new metrics introduced. The incorporated weights w1 and w2 are split into two variations each ($w_1^{'}$, $w_1^{''}$, $w_2^{'}$, $w_2^{''}$) to accommodate the positive and negative variations of all the parameters proposed above. After following the same proportionality present in the equation (number) we arrive at the following expression for quantifying models providing probability likelihood scores.

$$Q = w_1^{'} \bullet \beta + w_1^{''} \cdot \sigma + w_2^{'} \bullet \frac{\bar{\gamma}_{pos}}{\delta_{pos}} + w_2^{''} \bullet \frac{\bar{\gamma}_{neg}}{\delta_{neg}} \tag{11}$$

3.3 FRAMEWORK FOR PERSON RE-ID DOMAIN

The working pipeline constitutes data collection, data pre-processing, data augmentation, model training, implementing LIME XAI, and lastly, quantifying obtained explanations. Relevant data is collected by taking screenshots of four actors in various angles and poses with their head cropped out. After performing transformations like transposing and rotating to augment the dataset, the total size of the dataset is increased to 4800 images. The data is then fed to a mobilenet v2 pretrained model for training which then employs LIME XAI to generate explanations. Since the model implements a softmax distribution function in the final layer, this model follows the framework introduced in section 3.1.

Figure 8. Person Re-Id domain methodology



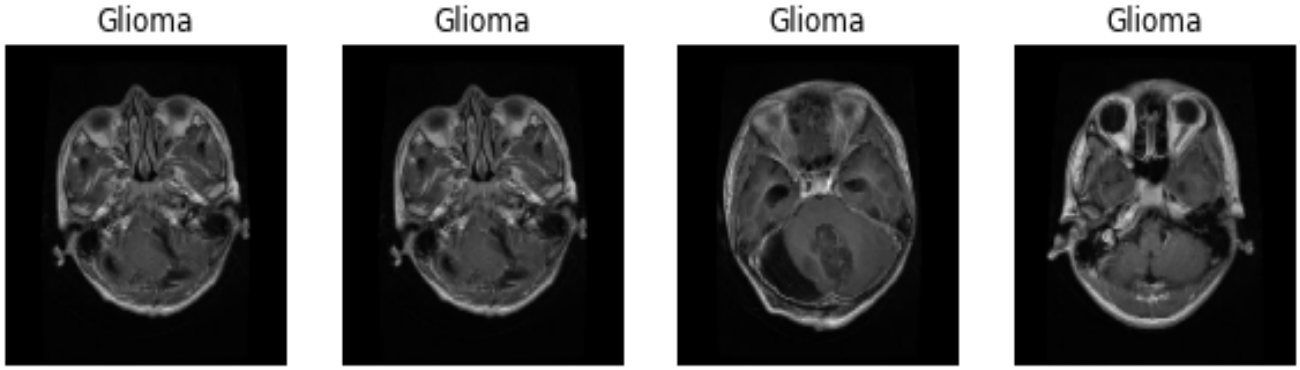
Chapter 4

Datasets

Data plays a crucial role in Data Science and procuring relevant data is the preliminary step of any ML/AI project. In this project we have implemented Explainable AI across three critical use-cases – medical, synthetic imaging, and person re-identification. The corresponding datasets on which we implemented our quantification framework for the respective three domains are briefly discussed below. The sources and the means of creating these datasets along with its contents are stated in the following sections.

4.1 MEDICAL DOMAIN

Classification tasks concerning whether a disease or defect persists in a patient's particular medical image are emerging to be one of the key applications of Artificial Intelligence in the medical domain. This encouraged us towards choosing the Brain Tumor MRI Dataset (REFERENCE). This dataset was created to train a classification model to identify brain tumors. This Kaggle dataset is a combined dataset formed based on three datasets – figshare(REFERENCE), SARTAJ(REF), and Br35H(REF). The dataset comprises a total of 7022 images in .jpg format of the human brain MRI scans. These images are then further classified into 4 different categories – glioma, meningioma, pituitary, and no tumor. SARTAJ dataset provides the images labelled under the meningioma and pituitary category. Br35H contains no tumor images. Whereas the glioma category images are obtained from the figshare website.



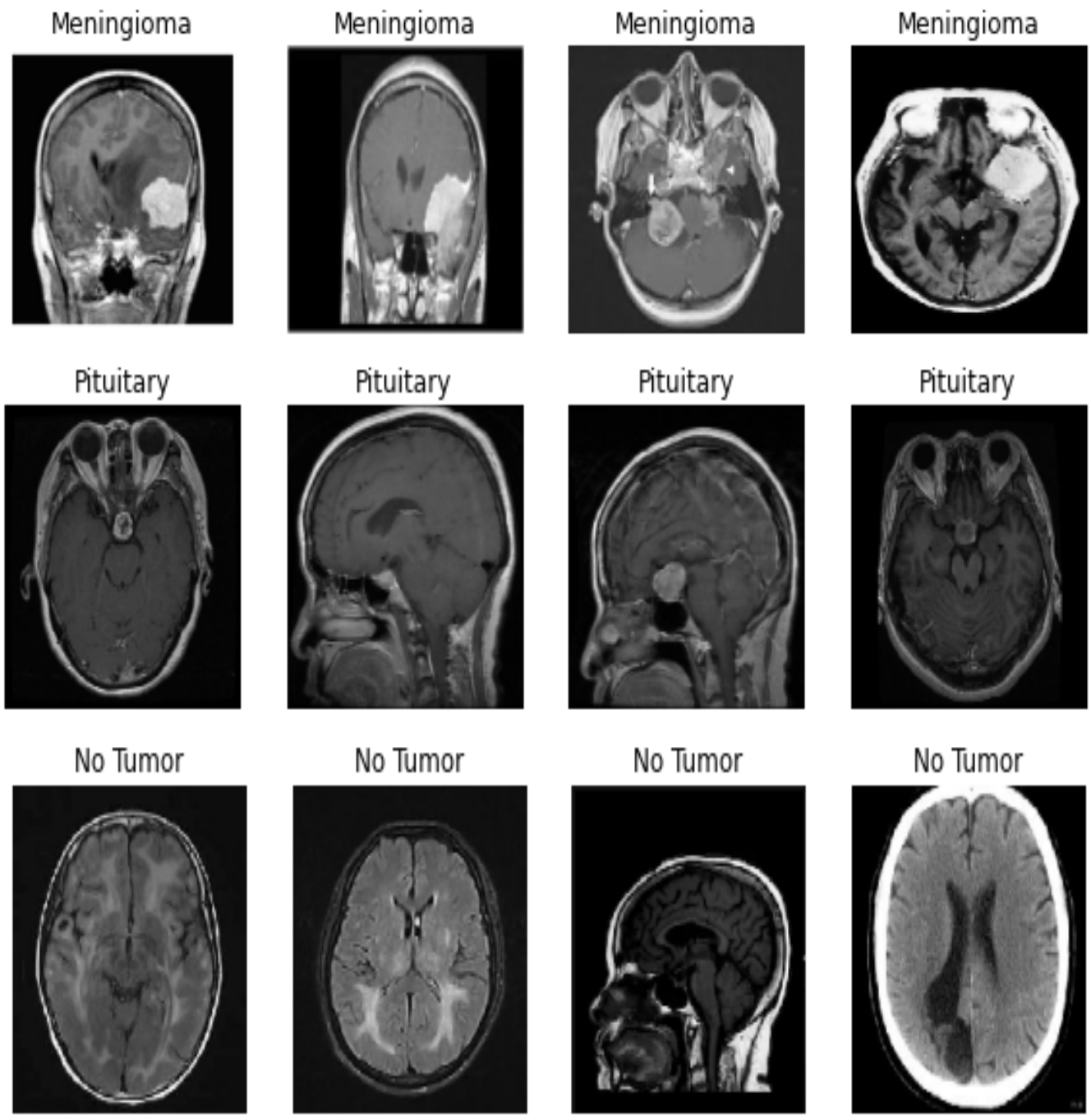


Figure 9. Medical domain dataset samples

4.2 SYNTHETIC IMAGING DOMAIN

DeepFake is yet another emerging field wherein deep fakes produced by some GANs are even indistinguishable from real images. Image classification between Real and DeepFake is another sensitive use-case where Explainable AI can help identify the regions of the image which are synthetically integrated.

The dataset comprises two categories - Real and DeepFake containing 4259 and 2845 images respectively. These images are of various celebrity faces.

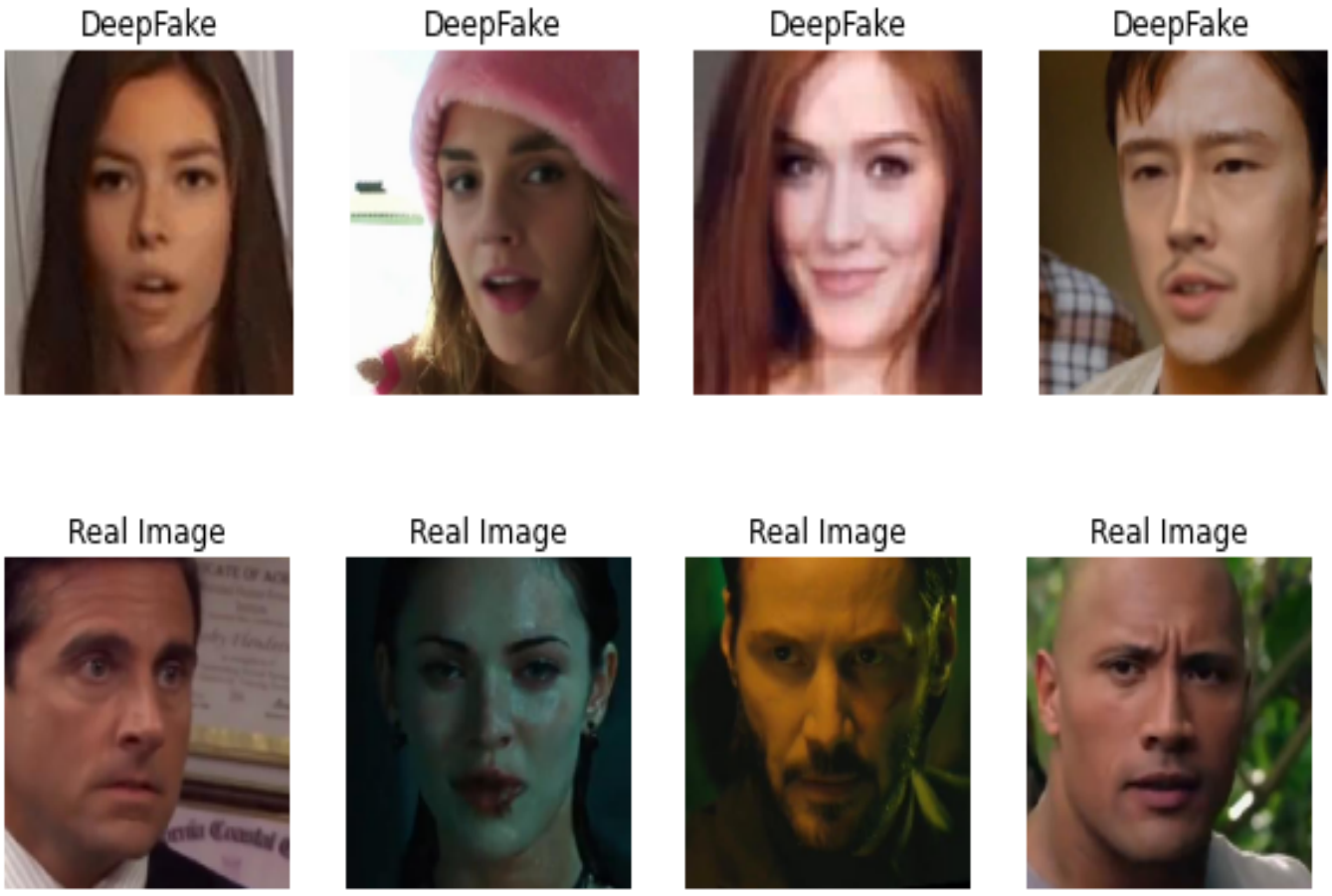


Figure 10. Synthetic Imaging domain dataset samples

4.3 PERSON RE-IDENTIFICATION DOMAIN

Due to the unavailability of a dataset which satisfied the specific requirements of our model, we were compelled to create a dataset. The model required images of people in multiple angles but with the head part cropped out. This was done to facilitate person re-identification regardless of what headgear or masks a person might wear so as to hide his/her identity. The dataset comprises full body images which are obtained from movies in various angles and several different attires. We categorized the dataset into 4 classes – Allu Arjun, Mahesh Babu, Nani and N T Rama Rao Jr. After data augmentation, we have a total of 4800 images in the dataset equally divided into the aforementioned four class labels with each class amounting to a total of 1200 images.

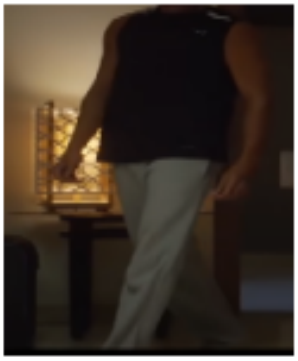
ALLU ARJUN

ALLU ARJUN

ALLU ARJUN

ALLU ARJUN

MAHESH BABU



MAHESH BABU

MAHESH BABU

MAHESH BABU

NANI

NANI

NANI

NANI

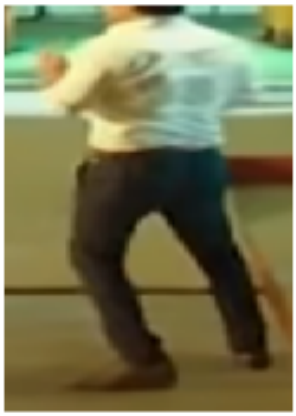


Figure 11. Person Re-Id domain dataset samples

Chapter 5

Experimental Results

We substantiate our formulated methodology with the Explainable AI algorithm explanations of LIME XAI. The underlying concept is that the explanations obtained from the XAI algorithm for a specific input subspace, if provided as input to the black box model should result in a similar value for prediction label and accuracy since the explanations obtained from the algorithm correspond to the salient features of the original output.

5.1 MEDICAL DOMAIN

We implemented our approach mentioned in Section 3.1 over 25 images of each category of the Brain Tumor MRI dataset - glioma, meningioma, pituitary and no tumor. The obtained results corresponding to each class are displayed below in separate tables.

Table 1 displays the values of prediction accuracy and label, and explanation accuracy and label for the glioma class label. It is visible that in some cases explanations have offered more clarity to the black box model than the entire input. This is evident from the samples where explanation accuracy has improved over the prediction accuracy.

Table 1. Predictions and Explanations for Glioma class label

glioma			
pa	pl	ea	el
0.9985935	glioma	0.9988111	glioma
0.9999999	glioma	1.00E+00	glioma
0.9810352	glioma	0.9947102	glioma
0.99975485	glioma	0.99976677	glioma
0.9966556	glioma	0.8588218	glioma
0.9998869	glioma	0.99996555	glioma
0.9999993	glioma	0.99999917	glioma
0.99999857	glioma	0.99999857	glioma
0.9999995	glioma	0.9999982	glioma
0.7653178	glioma	0.99670947	glioma
0.99999976	glioma	0.9999999	glioma
0.9547902	glioma	0.99998546	glioma
0.99988997	glioma	0.9999664	glioma

0.9999944	glioma	0.99918205	glioma
0.9896109	glioma	0.98448074	glioma
0.9832644	glioma	0.9953668	glioma
0.99865675	glioma	0.99930024	glioma
0.99988294	glioma	0.9998048	glioma
0.61241037	glioma	0.6610542	glioma
0.9999989	glioma	0.9999814	glioma
0.999824	glioma	0.999348	glioma
0.99999726	glioma	0.99999213	glioma
0.8310514	glioma	0.9760669	glioma
0.9399894	glioma	0.9112067	glioma
0.96729994	glioma	0.97841257	glioma

Meningioma tumor category is summarized in Table 2, it displays the corresponding scores of prediction accuracy and explanation accuracy while also stating the respective class label the black box model predicted.

Table 2. Predictions and Explanations for Meningioma class label

meningioma			
pa	pl	ea	el
0.99999356	meningioma	0.98893434	meningioma
0.89145464	meningioma	0.9981363	meningioma
0.9995172	meningioma	0.998618	meningioma
0.9976157	meningioma	0.9689755	meningioma
0.999977	meningioma	0.9994363	meningioma
0.99990463	meningioma	0.9999379	meningioma
0.73711693	meningioma	0.63314146	meningioma
0.99947935	meningioma	0.9997998	meningioma
0.9964888	meningioma	0.9697187	meningioma
0.9986265	meningioma	0.9998609	meningioma
0.98798406	meningioma	0.9851702	meningioma
0.98963004	meningioma	0.98858625	meningioma
0.9964994	meningioma	0.96262985	meningioma
0.93119276	meningioma	0.6474508	meningioma
0.5207528	meningioma	0.99561596	meningioma
0.99981695	meningioma	0.99998987	meningioma
0.91223544	meningioma	0.6343925	meningioma
0.987887	meningioma	0.97257406	meningioma
0.8192801	meningioma	0.9376177	meningioma

0.9995096	meningioma	0.9996183	meningioma
0.99952793	meningioma	0.99790984	meningioma
0.9743533	meningioma	0.977123	meningioma
0.9173454	meningioma	0.95116234	meningioma
0.9955244	meningioma	0.9654966	meningioma
0.9955817	meningioma	0.9684967	meningioma

The resulting values of the pituitary tumor class label are shown in Table 3. Here the explanations have faltered in two cases where they were incorrectly classified as meningioma tumor.

Table 3. Predictions and Explanations for Pituitary class label

pituitary			
pa	pl	ea	el
0.9999974	pituitary	0.31255692	meningioma
0.99946445	pituitary	0.9798306	pituitary
0.9999995	pituitary	0.99999917	pituitary
0.9996799	pituitary	0.09935685	meningioma
0.9989936	pituitary	0.9996432	pituitary
0.9634065	pituitary	0.98013765	pituitary
0.99998784	pituitary	0.99999166	pituitary
0.999734	pituitary	0.93265575	pituitary
0.99352515	pituitary	0.99571437	pituitary
0.99929833	pituitary	0.85731703	pituitary
0.99998283	pituitary	0.9978345	pituitary
0.9799129	pituitary	0.9885839	pituitary
0.9996207	pituitary	0.9885937	pituitary
0.99856853	pituitary	0.8914525	pituitary
0.98423094	pituitary	0.9954613	pituitary
0.99019694	pituitary	0.99289024	pituitary
0.99996245	pituitary	0.7805186	pituitary
0.99708074	pituitary	0.9921028	pituitary
0.9914778	pituitary	0.63120794	pituitary
0.92351323	pituitary	0.9607527	pituitary
0.9888114	pituitary	0.85657287	pituitary
0.999481	pituitary	0.9995592	pituitary
0.99998784	pituitary	0.99999535	pituitary
0.9994729	pituitary	0.999361	pituitary
0.42955554	pituitary	0.49241322	pituitary

Lastly, the no tumor class label values for prediction accuracy and label alongside the corresponding explanation accuracy and label are listed in Table 4.

Table 4. Predictions and Explanations for No Tumor class label

notumor			
pa	pl	ea	el
0.99899346	notumor	0.979884	notumor
0.96244884	notumor	0.85396254	notumor
0.9998192	notumor	0.9998416	notumor
0.9998323	notumor	0.9970702	notumor
0.99999833	notumor	0.9999957	notumor
0.9999989	notumor	0.9981273	notumor
0.9178403	notumor	0.86378497	notumor
0.9817228	notumor	0.96615577	notumor
0.99998987	notumor	0.9999138	notumor
0.690685	notumor	0.89517635	notumor
0.9813579	notumor	0.29855883	meningioma
0.9953538	notumor	0.82530624	notumor
0.9998661	notumor	0.9999964	notumor
0.9985667	notumor	0.9939658	notumor
0.99970067	notumor	0.9922281	notumor
0.95855814	notumor	0.93260026	notumor
0.99999785	notumor	0.99991655	notumor
0.99997294	notumor	0.9662319	notumor
0.99999905	notumor	0.961769	notumor
0.9999999	notumor	0.99999523	notumor
0.6004237	notumor	0.90715665	notumor
0.9993932	notumor	0.9233628	notumor
0.9992576	notumor	0.99798167	notumor
0.9999436	notumor	0.99582565	notumor
0.99945945	notumor	0.9997676	notumor

These values then facilitate the computation of the parameters discussed in section 3.1 and after incorporating the weights w1 = 0.9 and w2 = 0.1 we obtain the value for Quantification. Table 5 displays the values corresponding to Coherence, Correctness, Completeness, Congruency and Quantification (Q) for all the four class labels of the Brain Tumor MRI dataset.

Table 5. 4C metric values alongside Quantification values for Medical Domain

Class	Coherence	Correctness	Completeness	Congruency	Q
pituitary	0.1118457	0.92	89.63754185	0.219850558	41.60003
notumor	0.070319	0.96	96.94971884	0.145859323	67.33196
meningioma	0.0633432	1	99.5479313	0.114005204	88.21876
glioma	0.0272521	1	101.3949071	0.056824219	179.3361

5.2 SYNTHETIC IMAGING DOMAIN

The MesoNet architecture trained on the Celeb-DF-v1 dataset is used as the black box model on which the Explainable AI quantification is implemented. LIME XAI provides the explanations to help identify the salient regions of the images corresponding to either of the two class labels - Real or DeepFake. The following table (Table 6) showcases the values of each of the parameters formulated in section NUMBER for the respective class labels.

Table 6. Updated 4C metric values alongside Quantification values for Synthetic Image Domain

Class	DeepFake	Real
Coherence +ve	0.158868541	0.12949857
Coherence -ve	0.791405332	0.478229206
Completeness +ve	112.6132659	368.3010185
Completeness -ve	52.13794329	33.15980896
Congruency +ve	0.123277103	0.135441702
Congruency -ve	0.169670906	0.197703722
Specificity	0.40299	0.095271
Sensitivity	0.095271	0.40299
Quantification	549.3785	1299.167

5.3 PERSON RE-IDENTIFICATION DOMAIN

Table 7 displays the scores for the “4C” parameters and the subsequent Quantification value when LIME XAI is implemented on the custom person re-id dataset comprising 25 images for each of the class labels - N T Rama Rao JR, Mahesh Babu, Allu Arjun and Nani.

Table 7. 4C metric values alongside Quantification values for Person Re-Identification Domain

Class	Coherence	Congruency	Correctness	Completeness	Q
N T Rama Rao Jr	0.3811975	0.42877886	0.6	76.06821696	18.28066
Mahesh Babu	0.4428109	0.4293956	0.48	86.11041917	20.48587
Allu Arjun	0.453883	0.43430126	0.48	119.6977432	27.99299
Nani	0.3119643	0.39850232	0.12	28.95428243	7.373775

Chapter 6

Conclusion and Future Work

The purpose of this study is not to formulate original, ingenious algorithms to give insights into the black box models but rather to obtain a quantitative understanding of the existing Explainable AI algorithms commonly implemented currently. Furthermore, the paper also attempts to establish a quantitative comparison between the Explainable AI models. The “4C” framework developed in this research calculates the four metrics – Coherence, Correctness, Completeness, and Congruency. The aforementioned metrics were combined to compute the novel quantification expression for explainability which helped us to successfully quantify LIME Explainable AI algorithm. Moreover, the Explainable AI technique by identifying and visualizing the key features of the images resulting in the prediction of the model, helped us improve model transparency.

Limited by our software and hardware specifications we were constrained from experimenting this approach on a huge dataset, nevertheless the experimental results we obtained clearly proved the validity of our proposed approach. In future research, we would extend the scope of our “4C” framework by implementing it on various other Explainable AI techniques and alongside test it out on various other hyperparameters and use-cases.

Appendix

The code snippets for all the respective objectives are provided below:

Appendix 1 – MEDICAL DOMAIN

```
[ ] ### RUN
    # For Data Processing & ML Models
    import numpy as np
    from sklearn.utils import shuffle
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import classification_report
    from tensorflow import keras
    from tensorflow.keras.layers import *
    from tensorflow.keras.losses import *
    from tensorflow.keras.models import *
    from tensorflow.keras.metrics import *
    from tensorflow.keras.optimizers import *
    from tensorflow.keras.applications import *
    from tensorflow.keras.preprocessing.image import load_img
    from PIL import Image, ImageEnhance

    # For Data Visualization
    import matplotlib.pyplot as plt
    import seaborn as sns

    # Miscellaneous
    from tqdm import tqdm
    import os
    import random
```

```
[ ] train_dir = '/home/guest/Pratyush/Brain Tumor MRI/data/Training/'
    test_dir = '/home/guest/Pratyush/Brain Tumor MRI/data/Testing/'
```

```
[ ] train_paths = []
    train_labels = []

    for label in os.listdir(train_dir):
        for image in os.listdir(train_dir+label):
            train_paths.append(train_dir+label+'/'+image)
            train_labels.append(label)

    train_paths, train_labels = shuffle(train_paths, train_labels)
```

```
[ ] ### RUN

    test_paths = []
    test_labels = []

    for label in os.listdir(test_dir):
        for image in os.listdir(test_dir+label):
            test_paths.append(test_dir+label+'/'+image)
            test_labels.append(label)

    test_paths, test_labels = shuffle(test_paths, test_labels)
```

```
[ ] ### RUN

    def augment_image(image):
        image = Image.fromarray(np.uint8(image))
        image = ImageEnhance.Brightness(image).enhance(random.uniform(0.8,1.2))
        image = ImageEnhance.Contrast(image).enhance(random.uniform(0.8,1.2))
        image = np.array(image)/255.0
        return image
```

```
[ ] ### RUN

    IMAGE_SIZE = 128
```

```
[ ] ### RUN

    def open_images(paths):
        '''
        Given a list of paths to images, this function returns the images as arrays (after augmenting them)
        '''
        images = []
        for path in paths:
            image = load_img(path, target_size=(IMAGE_SIZE,IMAGE_SIZE))
            image = augment_image(image)
            images.append(image)
        return np.array(images)
```

```
[ ] ### RUN

    unique_labels = os.listdir(train_dir)
```

```
[ ] ### RUN

    def encode_label(labels):
        encoded = []
        for x in labels:
            encoded.append(unique_labels.index(x))
        return np.array(encoded)

    def decode_label(labels):
        decoded = []
        for x in labels:
            decoded.append(unique_labels[x])
        return np.array(decoded)
```

```
[ ] ### RUN

    def data_gen(paths, labels, batch_size=12, epochs=1):
        for _ in range(epochs):
            for x in range(0, len(paths), batch_size):
                batch_paths = paths[x:x+batch_size]
                batch_images = open_images(batch_paths)
                batch_labels = labels[x:x+batch_size]
                batch_labels = encode_label(batch_labels)
                yield batch_images, batch_labels
```

```
[ ] base_model = VGG16(input_shape=(IMAGE_SIZE,IMAGE_SIZE,3), include_top=False, weights=vgg16_weights_path)
    # Set all layers to non-trainable
    for layer in base_model.layers:
        layer.trainable = False
    # Set the last vgg block to trainable
    base_model.layers[-2].trainable = True
    base_model.layers[-3].trainable = True
    base_model.layers[-4].trainable = True
```

```
[ ] model = Sequential()
    model.add(Input(shape=(IMAGE_SIZE,IMAGE_SIZE,3)))
    model.add(base_model)
    model.add(Flatten())
    model.add(Dropout(0.3))
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(len(unique_labels), activation='softmax'))

    model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 4, 4, 512)         14714688

 flatten (Flatten)           (None, 8192)              0

 dropout (Dropout)           (None, 8192)              0

 dense (Dense)               (None, 128)               1048704

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 4)                 516

=================================================================
Total params: 15,763,908
Trainable params: 8,128,644
Non-trainable params: 7,635,264
_________________________________________________________________
```

```
[ ] model.compile(optimizer=Adam(learning_rate=0.0001), loss='sparse_categorical_crossentropy', metrics=['sparse_categorical_accuracy'])
```

```
[ ] batch_size = 20
    steps = int(len(train_paths)/batch_size)
    epochs = 4
    history = model.fit(data_gen(train_paths, train_labels, batch_size=batch_size, epochs=epochs),
                        epochs=epochs, steps_per_epoch=steps)
```

```
Epoch 1/4
285/285 [==============================] - 195s 681ms/step - loss: 0.4495 - sparse_categorical_accuracy: 0.8296
Epoch 2/4
285/285 [==============================] - 195s 686ms/step - loss: 0.2257 - sparse_categorical_accuracy: 0.9115
Epoch 3/4
285/285 [==============================] - 195s 683ms/step - loss: 0.1428 - sparse_categorical_accuracy: 0.9457
Epoch 4/4
285/285 [==============================] - 195s 683ms/step - loss: 0.1064 - sparse_categorical_accuracy: 0.9605
```

```
[ ] plt.figure(figsize=(8,4))
    plt.grid(True)
    plt.plot(history.history['sparse_categorical_accuracy'], '.g-', linewidth=2)
    plt.plot(history.history['loss'], '.r-', linewidth=2)
    plt.title('Model Training History')
    plt.xlabel('epoch')
    plt.xticks([x for x in range(epochs)])
    plt.legend(['Accuracy', 'Loss'], loc='upper left', bbox_to_anchor=(1, 1))
    plt.show()
```


```
[ ] batch_size=32
    steps = int(len(test_paths)/batch_size)
    y_pred = []
    y_true = []
    for x,y in tqdm(data_gen(test_paths, test_labels, batch_size=batch_size, epochs=1), total=steps):
        pred = model.predict(x)
        pred = np.argmax(pred, axis=-1)
        for i in decode_label(pred):
            y_pred.append(i)
        for i in decode_label(y):
            y_true.append(i)
```

```
41it [00:36,  1.11it/s]
```

```
[ ] print(classification_report(y_true, y_pred))
```

```
              precision    recall  f1-score   support

      glioma       0.99      0.82      0.90       300
  meningioma       0.86      0.92      0.89       306
     notumor       0.94      1.00      0.97       405
   pituitary       0.97      0.98      0.98       300

    accuracy                           0.94      1311
   macro avg       0.94      0.93      0.93      1311
weighted avg       0.94      0.94      0.94      1311
```

```
[ ] model.save("brain_tumor_classif.h5")
```

```
[ ]
```

```
[ ] model = keras.models.load_model("brain_tumor_classif.h5")
```

```
2022-04-08 15:15:07.521490: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-04-08 15:15:07.534255: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcusolver.so.11'; dlerror: libcusolver.so.11: cannot open shared object file: No such file or direc
2022-04-08 15:15:07.536328: W tensorflow/core/common_runtime/gpu/gpu_device.cc:1850] Cannot dlopen some GPU libraries. Please make sure the missing libraries mentioned above are installed properly if you would like to use GPU
Skipping registering GPU devices...
2022-04-08 15:15:07.537961: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critic
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

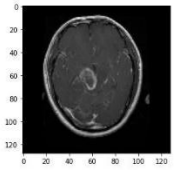
```
[ ] xinput = np.zeros((128,128,3))
```

```
batch_size=32
steps = int(len(test_paths)/batch_size)
y_pred = []
y_true = []
for x,y in tqdm(data_gen(test_paths, test_labels, batch_size=batch_size, epochs=1), total=steps):
    for i in range(batch_size):
        if i <= 5:
            continue
        xinput = x[i]
        flag = i

        break

    plt.imshow(xinput)
    pred = model.predict(x)
    print("Prediction Accuracy----",np.max(pred[6]))
    pred = np.argmax(pred, axis=-1)
    for l in decode_label(pred):
        y_pred.append(l)
    print("Prediction Label----",y_pred[6])
    for l in decode_label(y):
        y_true.append(l)
    break
```

```
  0%|                                                                                                | 0/40 [00:00<?, ?it/s]Prediction Accuracy---- 0.99999666
Prediction Label---- glioma
```



```
[ ] from lime import lime_image
```

```
[ ] explainer = lime_image.LimeImageExplainer()
```

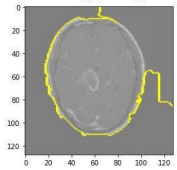
```
explanation = explainer.explain_instance(xinput, model.predict, top_labels=5, hide_color=0, num_samples=1000)
```

```
100%|██████████| 1000/1000 [00:28<00:00, 34.82it/s]
```

```
[ ] from skimage.segmentation import mark_boundaries
```

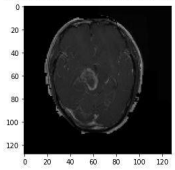
```
[ ] temp, mask = explanation.get_image_and_mask(explanation.top_labels[0], positive_only=True, num_features=5, hide_rest=True)
    plt.imshow(mark_boundaries(temp / 2 + 0.5, mask))
```

```
<matplotlib.image.AxesImage at 0x7fa99c6ce370>
```



```
[ ] plt.imshow(temp)
```

```
<matplotlib.image.AxesImage at 0x7fa99c6b4070>
```



```
[ ] lime_temp = np.expand_dims(temp, axis=0)
    lime_temp.shape
```

```
(1, 1, 128, 128, 3)
```

```
[ ] temp.shape
```

```
(1, 128, 128, 3)
```

```
[ ] lime_pred = model.predict(temp)
```

```
[ ] lime_pred
```

```
array([[1.4977902e-09, 4.4509325e-09, 9.9999607e-01, 3.8766580e-06]],
      dtype=float32)
```

```
[ ] print("Explanation Accuracy----",np.max(lime_pred))
    print("Explanation Label----",y_pred[np.argmax(lime_pred)])
```

```
Explanation Accuracy---- 0.99999607
Explanation Label---- glioma
```

Appendix 2 – SYNTHETIC IMAGING DOMAIN

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

```
[ ] pip install lime
```

```
      Building wheel for lime (setup.py) ... done
      Created wheel for lime: filename=lime-0.2.0.1-py3-none-any.whl size=283857 sha256=85ff9c851861b097a9ed4a843ee01b43140686 4fb77975189cf054b0e7e00751
      Stored in directory: /root/.cache/pip/wheels/ca/cb/e5/ac701e12d365a08917bf4c6171c0961bc880a8181359c66aa7
    Successfully built lime
    Installing collected packages: lime
    Successfully installed lime-0.2.0.1
```

```
[ ] import warnings
    warnings.filterwarnings('ignore')
```

```
[ ] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    from tensorflow.keras.layers import Input, Dense, Flatten, Conv2D, MaxPooling2D, BatchNormalization, Dropout, LeakyReLU
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    from tensorflow.keras.optimizers import Adam
    from tensorflow.keras.models import Model
    # import lime
    # from lime import lime_image
    # from skimage.segmentation import mark_boundaries

    import matplotlib.pylab as plt
    import numpy as np
    import tensorflow as tf
    import tensorflow_hub as hub
    from PIL import Image
```

```
[ ] class Classifier:
        def __init__():
            self.model = 0

        def predict(self, x):
            return self.model.predict(x)

        def flt(self, x, y):
            return self.model.train_on_batch(x, y)

        def get_accuracy(self, x, y):
            return self.model.test_on_batch(x, y)

        def load(self, path):
            self.model.load_weights(path)
```

```
[ ] class Meso4(Classifier):
        def __init__(self, learning_rate=0.001):
            self.model = self.init_model()
            optimizer = Adam(lr = learning_rate)
            self.model.compile(optimizer = optimizer,
                               loss = 'mean_squared_error',
                               metrics = ['accuracy'])

        def init_model(self):
            x = Input(shape = (image_dimensions['height'],
                               image_dimensions['width'],
                               image_dimensions['channels']))

            x1 = Conv2D(8, (3,3), padding='same', activation='relu')(x)
            x1 = BatchNormalization()(x1)
            x1 = MaxPooling2D(pool_size=(2,2), padding='same')(x1)

            x2 = Conv2D(8, (5,5), padding='same', activation='relu')(x1)
            x2 = BatchNormalization()(x2)
            x2 = MaxPooling2D(pool_size=(2,2), padding='same')(x2)

            x3 = Conv2D(16, (5,5), padding='same', activation='relu')(x2)
            x3 = BatchNormalization()(x3)
            x3 = MaxPooling2D(pool_size=(2,2), padding='same')(x3)

            x4 = Conv2D(16, (5,5), padding='same', activation='relu')(x3)
            x4 = BatchNormalization()(x4)
            x4 = MaxPooling2D(pool_size=(4,4), padding='same')(x4)

            y = Flatten()(x4)
            y = Dropout(0.5)(y)
            y = Dense(16)(y)
            y = LeakyReLU(alpha=0.1)(y)
            y = Dropout(0.5)(y)
            y = Dense(1, activation='sigmoid')(y)

            return Model(inputs=x, outputs=y)
```

```
[ ] image_dimensions = {'height':256, 'width': 256, 'channels':3}
```

```
[ ] meso = Meso4()
    meso.load('./drive/MyDrive/DeepFakeXAI/weights/Meso4_DF')
```

```
[ ] dataGenerator = ImageDataGenerator(rescale=1./255)

    generator = dataGenerator.flow_from_directory(
                './drive/MyDrive/DeepFakeXAI/data/',
                target_size=(256,256),
                batch_size=1,
                class_mode=None
    )

    generator.class_indices
```

```
    Found 5698 images belonging to 2 classes.
    {'DeepFake': 0, 'Real': 1}
```

```
[ ] p_plh = []
    p_label = []
    p_flag = []
    e_plh = []
    e_label = []
    e_flag = []
    count = 0
```

```
[ ] while(1):


      print(count)

      if X == "end":
        print("Calculation Over.")
      else:
        p_plh.append(meso.predict(X)[0][0])
        p_label.append(int(y[0]))
        p_flag.append(round(meso.predict(X)[0][0])==y[0])
        print("Prediction Details: ", p_plh[count], p_label[count], p_flag[count])

        explainer = lime_image.LimeImageExplainer()
        explanation = explainer.explain_instance(np.squeeze(np.array(X)).astype('double'), meso.predict, hide_color=0, num_samples=1000)
        temp, mask = explanation.get_image_and_mask(explanation.top_labels[0], positive_only=True, num_features=5, hide_rest=True)

        x = np.expand_dims(temp, axis=0)
        e_plh.append(meso.predict(x)[0][0])
        e_label.append(int(y[0]))
        e_flag.append(round(meso.predict(x)[0][0])==y[0])
        print("Explanation Details: ", e_plh[count], e_label[count], e_flag[count])

        if count % 2 == 0:
          p_plh_df = pd.DataFrame(p_plh)
          p_plh_df.to_csv('p_plh.csv')
          p_label_df = pd.DataFrame(p_label)
          p_label_df.to_csv('p_label.csv')
          p_flag_df = pd.DataFrame(p_flag)
          p_flag_df.to_csv('p_flag.csv')

          e_plh_df = pd.DataFrame(e_plh)
          e_plh_df.to_csv('e_plh.csv')
          e_label_df = pd.DataFrame(e_label)
          e_label_df.to_csv('e_label.csv')
          e_flag_df = pd.DataFrame(e_flag)
          e_flag_df.to_csv('e_flag.csv')

        count = count + 1
```

```
[ ] from openpyxl import load_workbook

    data_file = 'mesoxai.xlsx'

    wb = load_workbook(data_file)
    ws = wb['Sheet1']
    all_rows = list(ws.rows)

    p_plh = []
    p_label = []
    p_flag = []

    e_plh = []
    e_label = []
    e_flag = []

    acc = []

    for cell in all_rows[1::4]:
        pred = cell[0].value
        pred = pred.split()
        p_plh.append(pred[2])
        p_label.append(pred[3])
        p_flag.append(pred[4])

    for cell in all_rows[3::4]:
        exp = cell[0].value
        exp = exp.split()
        e_plh.append(exp[2])
        e_label.append(exp[3])
        e_flag.append(exp[4])
```

```
[ ] i_pos = []
    i_neg = []
    x1 = 0
    x2 = 0
```

```
[ ] import numpy as np
    for i in range(len(p_plh)):
      pplh = np.absolute(float(p_plh[i])-0.5)
      eplh = np.absolute(float(e_plh[i])-0.5)
      if (p_label[i] == '0' and e_label[i] == '0') and (p_flag[i] == 'True' and e_flag[i] == "True"):
        i = eplh*100/pplh
        i_pos.append(i)

      elif ((p_label[i] == '0' and e_label[i] == '1') or ((p_label[i] == '0' and e_label[i] == '0') and (p_flag[i] == 'True' and e_flag[i] == "False"))):
        pplh = pplh + 0.5
        i = eplh*100/pplh
        i_neg.append(i)
```

Appendix 3 – PERSON RE-IDENTIFICATION DOMAIN

In [1]:
```
import numpy as np
import cv2

import PIL.Image as Image
import os

import matplotlib.pyplot as plt

import tensorflow as tf
import tensorflow_hub as hub

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import *
from tensorflow.keras.models import Sequential

from lime import lime_image
from skimage.segmentation import mark_boundaries
```

```
/home/guest/.pyenv/versions/3.8.6/envs/person-reid2/lib/python3.8/site-packages/tqdm/auto.py:22: TqdmWarning: IProgress not fou
nd. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

In [2]:
```
IMAGE_SHAPE = (224,224)

classifier = tf.keras.Sequential([
    hub.KerasLayer("model/", input_shape = IMAGE_SHAPE+(3,))
])
```

In [3]:
```
import pathlib
data = pathlib.Path("data/")
data
```

Out[3]:
```
PosixPath('data')
```

In [4]:
```
list(data.glob('*/*'))[-5:]
```

Out[4]:
```
[PosixPath('data/Mahesh Babu/97frr.png'),
 PosixPath('data/Mahesh Babu/82rl.png'),
 PosixPath('data/Mahesh Babu/106frl.png'),
 PosixPath('data/Mahesh Babu/152f.png'),
 PosixPath('data/Mahesh Babu/11f.png')]
```

In [5]:
```
image_count = len(list(data.glob('*/*')))
image_count
```

Out[5]:
```
4800
```

In [6]:
```
ntrjr = list(data.glob('N T Rama Rao Jr/*'))
```

In [7]:
```
mahesh = list(data.glob('Mahesh Babu/*'))
```

In [8]:
```
arjun = list(data.glob('Allu Arjun/*'))
```

In [9]:
```
nani = list(data.glob('Nani/*'))
```

In [10]:
```
person_images_dict = {
    'N T Rama Rao Jr': ntrjr,
    'Mahesh Babu': mahesh,
    'Allu Arjun': arjun,
    'Nani': nani
}
```

In [11]:
```
person_labels_dict = {
    'N T Rama Rao Jr': 0,
    'Mahesh Babu': 1,
    'Allu Arjun': 2,
    'Nani': 3
}
```

In [17]:
```
X, y = [], []

count = 1
for person, images in person_images_dict.items():
    for image in images:
        img = cv2.imread(str(image))
        resized_img = cv2.resize(img,(224,224))
        X.append(resized_img)
        y.append(person_labels_dict[person])
        count = count + 1
```

In [18]:
```
X = np.array(X)
y = np.array(y)
```

In [19]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=0)
```

In [20]:
```
u, f = np.unique(y_test
                 , return_counts=True)
print(u, f)
```

```
[0 1 2 3] [240 240 240 240]
```

In [21]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[21]:
```
((3840, 224, 224, 3), (960, 224, 224, 3), (3840,), (960,))
```

In [22]:
```
X_train_scaled = X_train / 255
X_test_scaled = X_test / 255
```

In [72]:
```
X_train_scaled[0].shape, X_train_scaled[0]
```

Out[72]:
```
((224, 224, 3),
 array([[[0.21568627, 0.42745098, 0.21176471],
         [0.21960784, 0.43137255, 0.21568627],
         [0.21960784, 0.43137255, 0.21568627],
         ...,
         [0.37647059, 0.36470588, 0.30196078],
         [0.37647059, 0.36078431, 0.30588235],
         [0.38039216, 0.36470588, 0.31372549]],

        [[0.19215686, 0.40392157, 0.18823529],
         [0.19215686, 0.40392157, 0.18823529],
```

In [23]:
```
pretrained_model_without_top_layer = hub.KerasLayer(
    "model/feature vector/", input_shape=(224,224,3), trainable=False
)
```

In [24]:
```
num_of_people = 4

model = tf.keras.Sequential([
    pretrained_model_without_top_layer
])

model.add(Dense(256,activation=('relu')))
model.add(Dense(128,activation=('relu')))
model.add(Dropout(.4))
model.add(Dense(4,activation=('softmax')))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_1 (KerasLayer)  (None, 1280)              2257984

 dense (Dense)               (None, 256)               327936

 dense_1 (Dense)             (None, 128)               32896

 dropout (Dropout)           (None, 128)               0

 dense_2 (Dense)             (None, 4)                 516

=================================================================
Total params: 2,619,332
Trainable params: 361,348
Non-trainable params: 2,257,984
_________________________________________________________________
```

In [25]:
```
model.compile(
    optimizer="adam",
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['acc']
)
```

In [26]:
```
from tensorflow.keras.callbacks import ReduceLROnPlateau
lrr= ReduceLROnPlateau(monitor='loss', factor=.01, patience=3, min_lr=1e-5)
```

In [27]:
```
history = model.fit(X_train_scaled, y_train, epochs=10, callbacks=[lrr])
```

```
Epoch 1/10
/home/guest/.pyenv/versions/3.8.6/envs/person-reid2/lib/python3.8/site-packages/tensorflow/python/util/dispatch.py:1082: UserWarning: "`sparse_categorical_crossentropy` received `from_logits=True`, but the `output` argument was produced by a sigmoid or softmax activation and thus does not represent logits. Was this intended?"
  return dispatch_target(*args, **kwargs)
120/120 [==============================] - 29s 221ms/step - loss: 1.0507 - acc: 0.5526 - lr: 0.0010
Epoch 2/10
120/120 [==============================] - 26s 220ms/step - loss: 0.7050 - acc: 0.7266 - lr: 0.0010
Epoch 3/10
120/120 [==============================] - 26s 220ms/step - loss: 0.4933 - acc: 0.8159 - lr: 0.0010
Epoch 4/10
120/120 [==============================] - 26s 220ms/step - loss: 0.3435 - acc: 0.8758 - lr: 0.0010
Epoch 5/10
120/120 [==============================] - 26s 220ms/step - loss: 0.2340 - acc: 0.9180 - lr: 0.0010
Epoch 6/10
120/120 [==============================] - 26s 220ms/step - loss: 0.1738 - acc: 0.9367 - lr: 0.0010
Epoch 7/10
120/120 [==============================] - 26s 218ms/step - loss: 0.0981 - acc: 0.9680 - lr: 0.0010
Epoch 8/10
120/120 [==============================] - 26s 218ms/step - loss: 0.0868 - acc: 0.9742 - lr: 0.0010
Epoch 9/10
120/120 [==============================] - 26s 219ms/step - loss: 0.0483 - acc: 0.9872 - lr: 0.0010
Epoch 10/10
120/120 [==============================] - 26s 219ms/step - loss: 0.0443 - acc: 0.9872 - lr: 0.0010
```

In [28]:
```
model.evaluate(X_test_scaled, y_test) #dropout = 0.4
```

```
30/30 [==============================] - 7s 220ms/step - loss: 0.3571 - acc: 0.8854
```

Out[28]:
```
[0.35712364315986633, 0.8854166865348816]
```

In [ ]:
```
from keras.models import load_model
model.save('model.h5')
```

In [1]:
```
from keras.models import load_model
import tensorflow_hub as hub
import PIL.Image as Image
import numpy as np
import matplotlib.pyplot as plt
import cv2

import lime
from lime import lime_image
```

```
/home/guest/.pyenv/versions/3.8.6/envs/person-reid2/lib/python3.8/site-packages/tqdm/auto.py:22: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

In [2]:
```
model = load_model("model.h5",custom_objects={'KerasLayer':hub.KerasLayer})
```

```
2022-06-02 15:55:08.161169: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-06-02 15:55:08.164634: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcusolver.so.11'; dlerror: libcusolver.so.11: cannot open shared object file: No such file or directory; LD_LIBRARY_PATH: /home/guest/.pyenv/versions/3.8.6/envs/person-reid2/lib/python3.8/site-packages/cv2/../../lib64::/usr/local/cuda/lib64
2022-06-02 15:55:08.165132: W tensorflow/core/common_runtime/gpu/gpu_device.cc:1850] Cannot dlopen some GPU libraries. Please make sure the missing libraries mentioned above are installed properly if you would like to use GPU. Follow the guide at https://www.tensorflow.org/install/gpu for how to download and setup the required libraries for your platform.
Skipping registering GPU devices...
2022-06-02 15:55:08.165345: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

In [3]:
```
names = ["N T Rama Rao Jr", "Mahesh Babu", "Allu Arjun", "Nani"]
```

In [4]:
```
pa = 0
ea = 0
diff = []
count = 0
```

In [5]:
```
for i in range(1,26):
    img = cv2.imread("lime test/N T Rama Rao Jr/{}.png".format(i))
    resized_img = cv2.resize(img,(224,224))
    model_img = resized_img[np.newaxis,...]
    model_img = model_img / 255
    pred = model.predict(model_img)
    pa = pa + pred[0][0]
    pl = names[np.argmax(pred)]
    print(pl,"----",pred[0][0])

    explainer = lime_image.LimeImageExplainer()
    explanation = explainer.explain_instance(resized_img.astype('double'), model.predict, hide_color=0, num_samples=1000)
    temp, mask = explanation.get_image_and_mask(0, positive_only=True, num_features=1000, hide_rest=True)
    mask_stacked = np.stack((mask,mask,mask), axis = 2)
    final = np.multiply(resized_img, mask_stacked)
    final_fit = final[np.newaxis,...]
    final_fit = final_fit / 255
    exp = model.predict(final_fit)
    ea = ea + exp[0][0]
    el = names[np.argmax(exp)]
    print(el,"----",exp[0][0])
    print("--------------------------------------------------------------------------------")
```

```
    diff.append(np.absolute(pred[0][0]-exp[0][0]))

    if pl == names[0] and el == names[0]:
        count = count + 1

print("Cons----",np.mean(diff))
print("Prec----",np.std(diff))
print("Eff----",count/25)
print("Int----",ea*100/pa)

E = count/25
P = np.std(diff)
I = ea*100/pa

Q = 0.9 * E + 0.1 * ( I / P )
print("Q----",Q)
```

REFERENCES

[1] David Alvarez-Melis and Tommi S. Jaakkola. 2018. Towards Robust Interpretabil- ity with Self-Explaining Neural Networks. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (Montréal, Canada) (NIPS'18). Curran Associates Inc., Red Hook, NY, USA, 7786–7795.

[2] L. Zheng, Y. Huang, H. Lu and Y. Yang, "Pose-Invariant Embedding for Deep Person Re-Identification," in IEEE Transactions on Image Processing, vol. 28, no. 9, pp. 4500-4509, Sept. 2019, doi: 10.1109/TIP.2019.2910414.

[3] A.Dhurandhar,VijayIyengar,RonnyLuss,andKarthikeyanShanmugam.2017. TIP: Typifying the Interpretability of Procedures. ArXiv abs/1706.02952 (2017).

[4] SorelleAFriedler,ChitradeepDuttaRoy,CarlosScheidegger,andDylanSlack. 2019. Assessing the Local Interpretability of Machine Learning Models. arXiv preprint arXiv:1902.03501 (2019).

[5] Andreas Holzinger, André Carrington, and Heimo Müller. 2020. Measuring the Quality of Explanations: The System Causability Scale (SCS). KI - Künstliche Intel- ligenz 34, 2 (01 Jun 2020), 193–198. https://doi.org/10.1007/s13218-020-00636-z

[6] S. R. Islam, W. Eberle, and S. Ghafoor. 2020. Towards Quantification of Ex- plainability in Explainable Artificial Intelligence Methods. ArXiv abs/1911.10104 (2020).

[7] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fer- nanda Viegas, and Rory Sayres. 2018. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). arXiv:1711.11279 [stat.ML]

[8] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In Proceedings of the 31st International Conference on Neural Information Processing Systems (Long Beach, California, USA) (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 4768–4777.

[9] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. 2019. Definitions, methods, and applications in inter- pretable machine learning. Proceedings of the National Academy of Sci- ences 116, 44 (2019), 22071–22080. https://doi.org/10.1073/pnas.1900654116 arXiv:https://www.pnas.org/content/116/44/22071.full.pdf

[10] Marco Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations. Association for Computational Linguistics, San Diego, California. https://doi.org/10.18653/v1/N16- 3020

[11] RamprasaathR.Selvaraju,MichaelCogswell,AbhishekDas,RamakrishnaVedan- tam, Devi Parikh, and Dhruv Batra. 2017. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In 2017 IEEE International Con- ference on Computer Vision (ICCV). 618–626. https://doi.org/10.1109/ICCV.2017.74

[12] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning Im- portant Features through Propagating Activation Differences. In Proceedings of the 34th International Conference on Machine Learning - Volume 70 (Sydney, NSW, Australia) (ICML'17). JMLR.org, 3145–3153.

[13] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. In Proceedings of the 34th International Conference on Machine Learning - Volume 70 (Sydney, NSW, Australia) (ICML'17). JMLR.org, 3319–3328.

[14] Nakatani, Ryota & Kouno, Daichi & Shimada, Kazutaka & Endo, Tsutomu. (2012). A Person Identification Method Using a Top-view Head Image from an Overhead Camera. Journal of Advanced Computational Intelligence and Intelligent Informatics. 16.

[15] Sung, H., Ferlay, J., Siegel, R. L., Laversanne, M., Soerjomataram, I., Jemal, A., et al. (2021). Global Cancer Statistics 2020: GLOBOCAN Estimates of Incidence and Mortality Worldwide for 36 Cancers in 185 Countries. CA A. Cancer J. Clin. 71, 209–249. doi:10.3322/caac.21660