HUMAN ABNORMAL BEHAVIOUR DETECTION BASED ON DL and ML ALGORITHMS

MINI PROJECT REPORT

IPG M.Tech

in

Information Technology

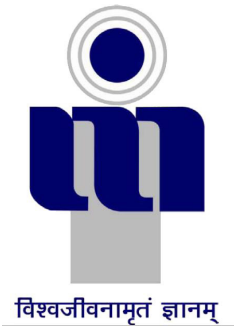
by

Sankalp (2021IMT-084)

Lata (2021IMT-059)

Gaurang Sondur (2021IMT-036)

Astitva Kamble (2021IMT-048)



ABV INDIAN INSTITUTE OF INFORMATION TECHNOLOGY AND MANAGEMENT

GWALIOR - 474015

May 2024

CANDIDATES DECLARATION

We hereby certify that the work, which is being presented in the report, entitled **Abnormal Behaviour of People in Heritage Site**, in partial fulfillment of the requirement for Mini-BTech Project for **Bachelor of Technology in Information Technology** and submitted to the institution is an authentic record of our own work carried out during the period *Jan 2024* to *April 2024* under the supervision of **Dr. Mahendra Shukla**. We also cited the reference about the text(s)/figure(s)/table(s) from where they have been taken.

Date: Signature of the Candidate

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Date: Signature of the Supervisor(s)

ABSTRACT

Our project focuses on developing an intelligent system to detect and deter inappropriate behavior, such as smoking and spitting, at national heritage sites. To achieve this, we've employed a dynamic approach to fine-tuning how our system learns during training. This method helps optimize the performance of our model efficiently. We've also tapped into the power of the pre-trained DenseNet architecture, a sophisticated computer brain, to aid our system in understanding images better. Building upon this architecture, we've added extra layers with dense connections, special functions that help the model understand patterns, and techniques to prevent it from getting too smart and making mistakes.

Moreover, we've paid careful attention to how we teach our system. We've decided the best number of images to show it at once and how many times it should see them. We've closely monitored its progress during training and made adjustments when needed. We've even listened to feedback from experts in heritage site management to improve our system further.

Our approach offers a comprehensive framework for automatically monitoring and preserving cultural heritage sites. By leveraging smart technology and straightforward solutions, we aim to ensure the continued significance and beauty of these cherished locations for generations to come.

ACKNOWLEDGEMENTS

We are highly indebted to **Dr. Mahendra Shukla**, and are obliged for giving us the autonomy of functioning and experimenting with ideas. We would like to take this opportunity to express our profound gratitude to them not only for their academic guidance but also for their personal interest in our project and constant support coupled with confidence boosting and motivating sessions which proved very fruitful and were instrumental in infusing self-assurance and trust within us. The nurturing and blossoming of the present work is mainly due to their valuable guidance, suggestions, astute judgment, constructive criticism and an eye for perfection. Our mentor always answered myriad of our doubts with smiling graciousness and prodigious patience, never letting us feel that we are novices by always lending an ear to our views, appreciating and improving them and by giving us a free hand in our project. It's only because of their overwhelming interest and helpful attitude, the present work has attained the stage it has.

Finally, we are grateful to our Institution and colleagues whose constant encouragement served to renew our spirit, refocus our attention and energy and helped us in carrying out this work.

Sankalp (2021IMT-084)
Lata (2021IMT-059)
Gaurang Sondur(2021IMT-036)
Astitva Kamble(2021IMT-048)

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

This chapter delves into the salient predicament of safeguarding heritage sites within our societal framework, elucidating the formidable challenges they confront and the scholarly endeavors undertaken to fathom these intricacies. Amidst the epoch of burgeoning Machine Learning and Deep Learning, where the pervasive influence of Artificial Intelligence (AI) pervades every facet of existence, leveraging its capabilities has become essential for optimizing operations and augmenting efficacy across multifarious domains. Consequently, our endeavor has been to scrutinize a panoply of Machine Learning and Deep Learning paradigms to discern the most efficacious avenues towards realizing our objectives.

1.1 MOTIVATION

India boasts a wealth of cultural heritage, including 42 UNESCO World Heritage sites and over 3,500 archaeological sites[1] and monuments under the protection of the Archaeological Survey of India (ASI). These sites not only represent our history and culture but also draw numerous tourists. Despite their importance, some individuals engage in careless behavior, such as smoking and spitting, disregarding the rules established to preserve these sites. This disregard diminishes the beauty and sanctity of these historical treasures.

Interestingly, people often spit in areas where signs explicitly prohibit such behavior, demonstrating a lack of respect for rules and penalties. Additionally, smoking is a prevalent issue at these sites, creating discomfort for visitors.

Efforts by the government and local authorities to address these problems include educational campaigns and increased surveillance by guards. However, the sheer volume of visitors makes it challenging to enforce regulations manually. As a result, there is a need for automated solutions to mitigate these issues effectively

1.2 OBJECTIVES

The main goal of this research is to propose an effective solution to the previously mentioned issue that requires less human labour. Our ultimate goal is to create and put into practice a methodology that will assist in preventing the integrity of our heritage sites by lowering the quantity of aberrant behaviours and identifying the people who engage in them. It is true that doing so would raise awareness and deter others from doing such things. Over time, it would have a significant effect, perhaps draw more tourists, and contribute to the effective maintenance and preservation of our cultural heritage.

1.3 OVERVIEW OF MACHINE LEARNING AND DEEP LEARNING

Within the larger subject of artificial intelligence, machine learning (ML) and deep learning (DL) are two essential domains that are distinguished by unique approaches and applications.

The goal of machine learning, a branch of artificial intelligence, is to create models and algorithms that can learn from data and use it to anticipate or make judgments. It includes a range of methods, including reinforcement learning, unsupervised learning, and supervised learning. While the algorithm finds patterns and structures in unlabeled data in unsupervised learning, it learns from labeled data in supervised learning. Through trial and error, an agent is trained to interact with its surroundings and discover the best actions through reinforcement learning.

However, Deep Learning is a specific kind of machine learning that uses multiple-layered neural networks (hence the name "deep") to extract information from data and make intricate predictions or judgments. Deep learning models have shown impressive results in tasks including speech recognition, natural language processing, and image recognition. Examples of these models are convolutional neural networks (CNNs) for image recognition and recurrent neural networks (RNNs) for sequential data. Deep learning models frequently outperform conventional machine learning techniques in applications requiring unstructured data because they are exceptionally good at automatically identifying complex patterns and representations in big datasets.

1.4 METHODOLGY

The methodology involves collecting a diverse dataset of spitting and smoking images healthy samples. Data preprocessing ensures uniformity and quality. The dataset is

split into training, validation, and test sets. Different models are designed and trained using an optimizer and loss function. Model performance is evaluated on the validation set, and hyperparameter tuning is conducted. Comaprisons of result of different models were done.

1.4.1 DATA COLLECTION AND PREPROCESSING

1. Gather a comprehensive dataset comprising images depicting smoking, spitting, and healthy behaviors. Ensure diversity in terms of demographics, environments, and scenarios.

2. Thoroughly inspect and validate the dataset for correct labeling, high resolution, and consistency in image quality.

3. Normalize the images to a standard scale to maintain uniformity in pixel values across the dataset. This step enhances comparability and facilitates effective model training.

1.4.2 DATA LABELLING AND SPLITTING:

1. Divide the dataset into training, validation, and test sets in an appropriate ratio (e.g.,70-15-15) for model training, tuning, and evaluation.

1.4.3 MODEL TRAINING

1. Utilize various machine learning algorithms such as Support Vector Machines (SVM), Random Forest, and k-Nearest Neighbors (KNN) classifiers for training on the dataset. Implement appropriate hyperparameter tuning techniques to optimize model performance.

2. Employ deep learning architectures including EfficientNetB3, MobileNetV2, and DenseNet, coupled with standard preprocessing steps like normalization and image augmentation. Train these models on the dataset to capture complex patterns and features inherent in the images.

1.4.4 MODEL EVALUATION AND TESTING

1. After the training phase, where various machine learning and deep learning models were trained on the dataset. The subsequent step involved a meticulous analysis and comparison of the results obtained from these models. This analysis was crucial for discerning the efficacy and performance of each model in tackling the specific task at hand.

2. Finally, deploy the best-performing model(s) on the test set to evaluate their generalization capabilities and assess real-world performance. Validate the model's effectiveness in accurately classifying smoking, spitting, and healthy behaviors in unseen data samples.

1.4.5 DOCUMENTATION

1. Document all steps, findings, and decisions made throughout the methodology. Prepare comprehensive reports summarizing the experimental setup, results, and recommendations for future research or application.

Chapter 2

DESIGN DETAILS AND IMPLEMENTATION

In this section, we outline the design details and implementation steps of the various model. We describe the data preprocessing techniques, model training setup, and evaluation metrics used for assessing performance. The implementation is carried out using Python, leveraging popular deep learning libraries such as TensorFlow and Keras and Scikit-learn for Machine Learning. The details provided here will offer a comprehensive understanding of the system's structure and how it addresses the challenges of spitting and smoking in our heritage monuments.

2.1 DATA COLLECTION

The code begins by gathering information. We have used the Kaggle Smoking images dataset[ ]. Additionally, we have curated our own dataset for spitting by using images from Google and other sources.

2.2 DATA PREPROCESSING

The images are preprocessed to ensure uniformity and prepare them for training the machine learning and deep learning model. The following steps are applied to each image in the dataset:

2.2.1 IMAGE RESIZING

Each image is resized to a square with a side length of 224 pixels. This step is necessary to standardize the image dimensions and reduce computational complexity.

2.2.2 LABEL GENERATION

We have store all the images according to their label folder. Hence all images were labeled according to the folder in which they reside.

2.2.3 DATA SPLITTING

From the entire dataset, 70 percent of data were used for training, 15 percent were used for validation and remaining 15 percent were used for testing.

2.3 SUMMARY OF DATASET:

Total number of images: 2668
Image size: 224x224 pixels (each image is square)
Classes in dataset : (Smoking, spitting, and Normal)
Class distribution: The distribution of images among different classes in both the training and testing sets.
The preprocessed data (X_train, X_test, y_train, y_test) is now ready for use in training and evaluating a model to predict the abnormal behaviour of person in heritage site. This data can be utilized with various model of Deep Learning and Machine Learning, such as convolutional neural networks (CNNs).

2.4 IMPLEMENTATION/EXECUTION OF PROJECT

2.4.1 TRAINING SET LABEL ENCODING

The original y_train array contains the categorical stage labels for each corresponding image in the training set. A new empty list y_train_new is created to store the encoded numerical labels. For each label in y_train, the code finds the index of that label in the labels list (Smoking, Spitting, Normal) The index value represents the numerical encoding of the corresponding stage label. The y_train_new list now contains the numerical representations of the stage labels for the training set. Next, the y_train list is reassigned to the one-hot encoded version of y_train_new using keras.

2.4.2 TESTING SET LABEL ENCODING

The same procedure is applied to the testing set labels (y_test). A new empty list y_test_new is created to store the encoded numerical labels. For each label in y_test, the code finds the index of that label in the labels list. The y_test_new list now contains the numerical representations of the stage labels for the testing set. Similar to the

training set, the y_test list is reassigned to the one-hot encoded version of y_test_new. Now, the training and testing sets are represented as numerical one-hot encoded arrays, where each row corresponds to an image, and the columns represent class(Smoking, spitting, and Normal) of an image . These encoded labels are suitable for training and evaluating neural network models, particularly for multi-class classification tasks.

2.4.3 DATA AUGMENTATION

Data augmentation is applied using the ImageDataGenerator from Keras[5]to artificially increase the diversity of training samples. Augmentation techniques include rotation, width and height shifts, shearing, zooming, and horizontal flipping. Data augmentation is useful for training deep learning models when the dataset is limited, as it can prevent overfitting and improve model performance.

2.4.4 TRAINING AND EVALUATION

The model is compiled using the Adam optimizer and categorical cross-entropy loss function (as it's a multi-class classification problem). The accuracy metric is used to monitor the model's performance during training. Callbacks like early stopping are used with patience of 5 to prevent overfitting and restore the best model weights during training. The model is trained for 10 epochs with a batch size of 32 and a validation split of 15 percent. The training and validation accuracy and loss are recorded after each epoch.

2.5 LITERATURE SURVEY

Not much work has been done on abnormal behavior detection with respect to heritage sites. Spit detection was earlier explored in [6]; however, it primarily focused on COVID and related aspects. In a recent study [8], various machine learning approaches were compared for predicting improper behaviors in specific situations that may endanger people, such as smoking in a gas station. The study collected datasets related to behavior detection, including categories such as smoking, calling, and normal behaviors. Experiments were conducted using several popular algorithms, including Linear Support Vector Machine (LSVM), Kernel Support Vector Machine (KSVM), Decision Tree Classifier (DT), Random Forest Classifier (RF), K-nearest Neighbors (KNN), and K-Means Clustering. Performance evaluation metrics such as Confusion Matrix and Mean Squared Error (MSE) were utilized to assess the algorithms, and Principal Component Analysis (PCA) was employed to visualize the outcomes. The results indicated that the Random Forest Classifier (RF) achieved the best performance, with an accu-

racy of 82 percentage in predicting people's abnormal behaviors.
After training using three deep learning base models—EfficientNet[8], DenseNet (Huang et al., 2016), and MobileNetV2 [7]—on our dataset, we achieved the following accuracies for each model: DenseNet: 99.002 percentage, EfficientNetB3: 98.254 percentage, and MobileNetV2: 91.272 percentage. DenseNet emerged as the top performer on our dataset, surpassing both the machine learning models discussed in Chapter 3 and other deep learning models explored in this chapter. This observation highlights the effectiveness of DenseNet for our specific task. The architecture we followed during model training is depicted in Figure 4.1. Additionally, in a comparison study of machine learning approaches by [8], it was found that the Random Forest Classifier (RF) achieved the best performance for predicting people's abnormal behaviors with an accuracy of 82 percentage. This indicates the significance of employing appropriate machine learning algorithms for behavior detection tasks.

Chapter 3

RESULTS AND DISCUSSION FOR ML ALGORITHMS

3.1 RESULTS

In this section we discuss the results obtained by applying three Machine Learning models[8] on our dataset. First we have tried to apply SVM model on our dataset as it is the most commonly used classification algorithm, upon trying the appropriate hyperparameters we found out the accuracy to be 79.97%. Next model we had tried was the K-Nearest Neighbours which is also a well known classification algorithm. The best accuracy that we could obtain was 69.54% which was even lesser than our SVM model. Next we have tried the Random Forest Classifier which gave us the best results with the accuracy of 84.45%. We have analyzed these algorithm on basis of classification reports which consists of metrics like accuracy, f1score, recall. We have also plotted the confusion matrix and ROC curve for each of them. In the further subsections we discuss each of the algorithms and the results obtained by them in detail.

3.1.1 SUPPORT VECTOR MACHINE(SVM)

Support Vector Machines (SVMs) are popular in classification tasks for their ability to separate two classes using an optimal hyperplane that maximizes the margin between the closest data points of different classes. The dimensionality of the input data determines whether the hyperplane is a line (2-D) or a plane (n-dimensional). By maximizing the margin, SVMs can generalize well to new data and make accurate predictions. The support vectors are the data points that determine this maximal margin.
Figure 3.1 explains the working of the SVM algorithm
Figure 3.2 shows the confusion matrix for the SVM classifier
Figure 3.3 shows the ROC curve as well as the AUC values for each of the classes
Table 3.1 describes the classification report of the SVM classifier

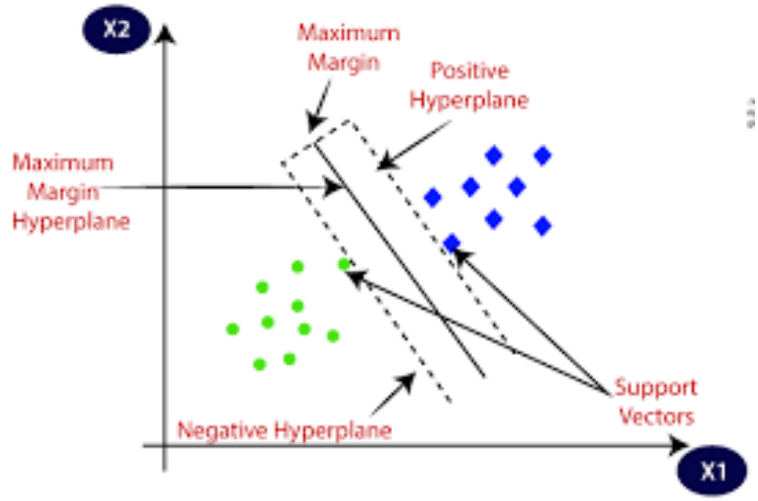


Figure 3.1: Working of SVM[8]

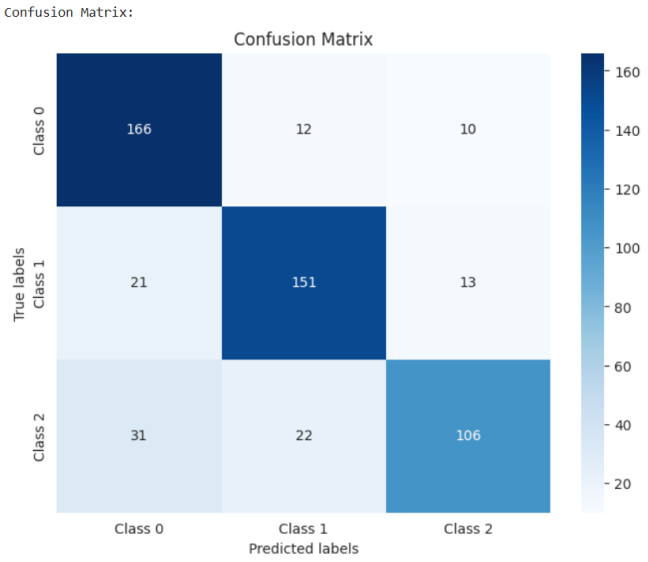


Figure 3.2: Confusion Matrix for SVM

SVMs are versatile, handling both linear and nonlinear tasks. For nonlinear data, kernel functions are used to transform the data into a higher-dimensional space for linear separation, known as the "kernel trick". The choice of kernel function (e.g., linear, polynomial, RBF, sigmoid) depends on the data and the task at hand. In our model we have used the parameters C = 1, kernel = 'poly' and gamma = 'auto'.

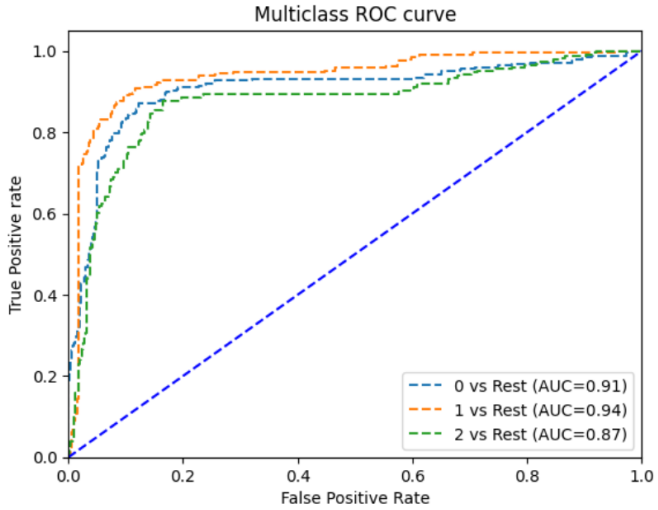


Figure 3.3: ROC curve for SVM

Table 3.1: Classification report for SVM classifier

	Precision	Recall	f1 - score	support
0	0.76	0.88	0.82	188
1	0.82	0.82	0.82	185
2	0.82	0.67	0.74	159
accuracy			0.80	532
macro avg	0.80	0.79	0.79	532
weighted avg	0.80	0.79	0.79	532

3.1.2 RANDOM FOREST ALGORITHM

Decision trees are a popular and intuitive tool used in machine learning for both classification and regression tasks. They work by recursively partitioning the input space into regions, with each partition represented by a tree node. At each node, the algorithm selects the feature and the split point that best separates the data based on certain criteria, typically aiming to maximize information gain or minimize impurity. This process continues until a stopping criterion is met, such as reaching a maximum tree depth, minimum number of data points in a node, or when no further splits lead to significant improvement in purity.

Figure 3.4 explains the working of the Random Forest algorithm

Figure 3.5 shows the confusion matrix for the Random Forest classifier

Figure 3.6 shows the ROC curve as well as the AUC values for each of the classes

Table 3.2 describes the classification report of the Random Forest classifier

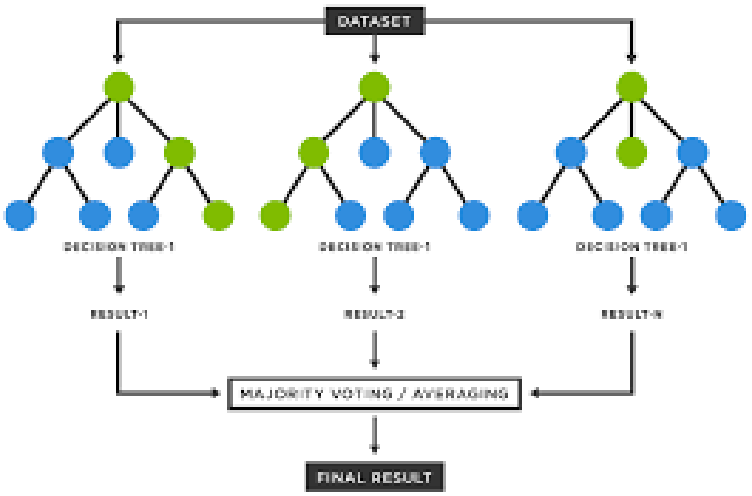


Figure 3.4: Working of Random Forest[8]

Table 3.2: Classification report for Random Forest classifier

	Precision	Recall	f1 - score	support
0	0.83	0.90	0.87	188
1	0.82	0.83	0.83	185
2	0.88	0.77	0.82	159
accuracy			0.84	532
macro avg	0.84	0.84	0.84	532
weighted avg	0.84	0.84	0.84	532

Random forests are an ensemble learning method that builds upon the idea of decision trees. Instead of relying on a single decision tree, random forests train multiple trees using different subsets of the training data and different subsets of the features. This randomness in data sampling and feature selection helps to reduce overfitting and improve the generalization of the model. When making predictions, random forests aggregate the predictions of individual trees to arrive at a final prediction, typically using a majority voting scheme for classification or averaging for regression. The randomness in the training process and the aggregation of predictions make random forests robust and often more accurate than individual decision trees, especially in complex datasets. n_estimators are the number of trees you want to build before taking the maximum voting or averages of predictions. In our model we have used n_estimators = 250.

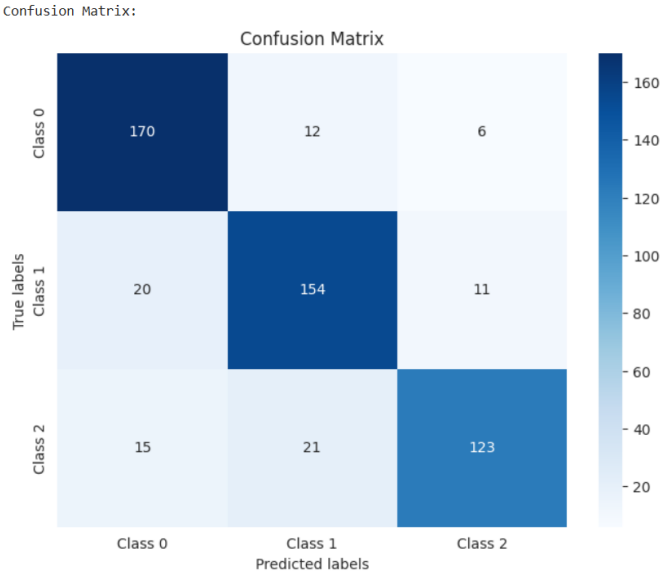


Figure 3.5: Confusion Matrix for Random Forest

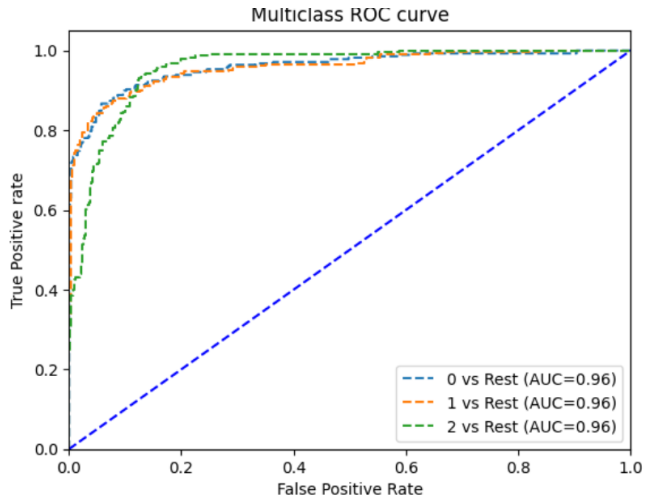


Figure 3.6: ROC curve for Random Forest

3.1.3 K-NEAREST NEIGHBOURS

The k-Nearest Neighbors (kNN) algorithm is a simple yet effective method used for both classification and regression tasks in machine learning. In kNN, the prediction for a new data point is made based on the majority class among its k nearest neighbors. The "nearest" neighbors are defined by some distance metric, often Euclidean distance, though other metrics can also be used depending on the data characteristics.

Figure 3.7 explains the working of the KNN algorithm

Figure 3.8 shows the confusion matrix for the KNN classifier

Figure 3.9 shows the ROC curve as well as the AUC values for each of the classes

Table 3.3 describes the classification report of the KNN classifier

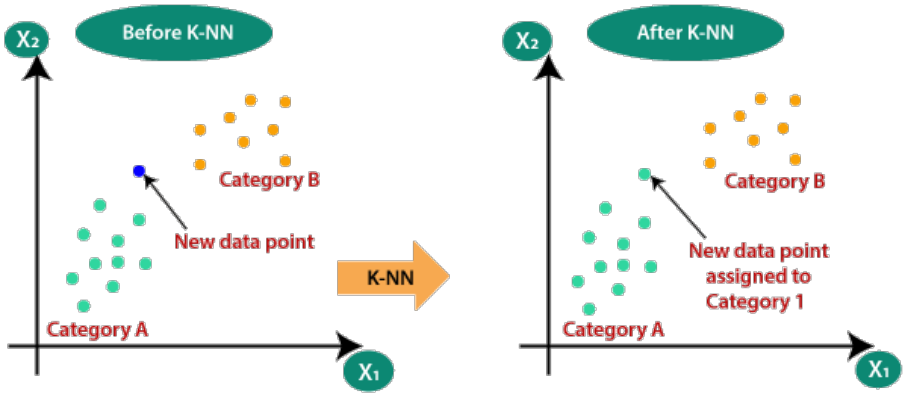


Figure 3.7: Working of KNN[8]

Table 3.3: Classification report for KNN classifier

	Precision	Recall	f1 - score	support
0	0.65	0.82	0.73	188
1	0.79	0.62	0.69	185
2	0.71	0.67	0.69	159
accuracy			0.70	532
macro avg	0.72	0.70	0.70	532
weighted avg	0.72	0.70	0.70	532

For classification, the algorithm assigns the class label that is most common among the k nearest neighbors. In the case of regression, the algorithm predicts the average of the target values of the k nearest neighbors. The choice of k is a hyperparameter that needs to be tuned; a small k can lead to noisy decisions, while a large k may smooth out important patterns in the data. The algorithm is simple to understand and implement, making it a popular choice for beginners and for baseline comparison with more complex models. However, it can be computationally expensive, especially with large datasets, as it requires computing distances between the new data point and all training data points.

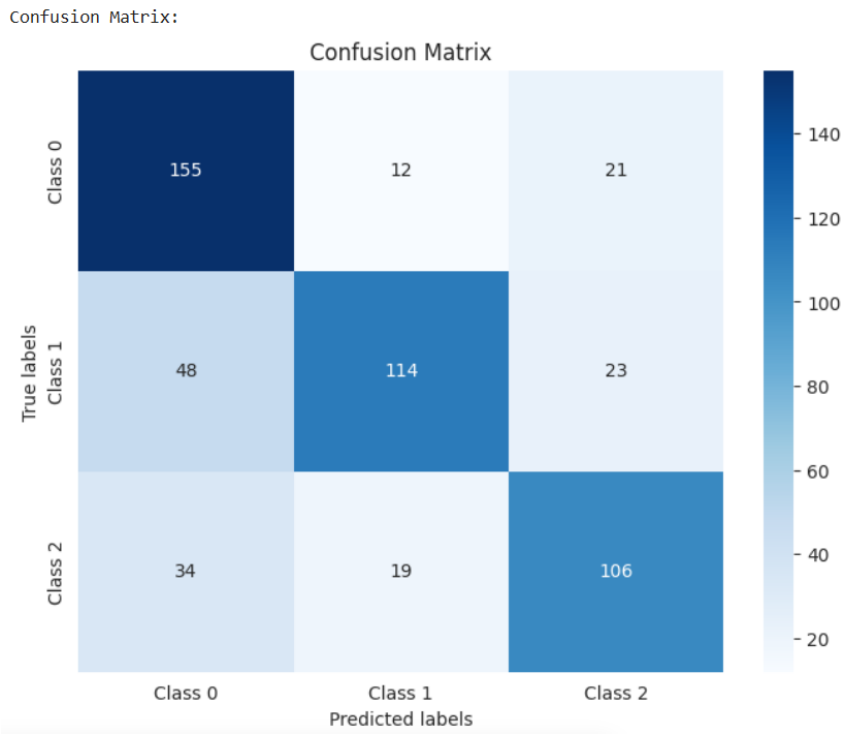


Figure 3.8: Confusion Matrix for KNN

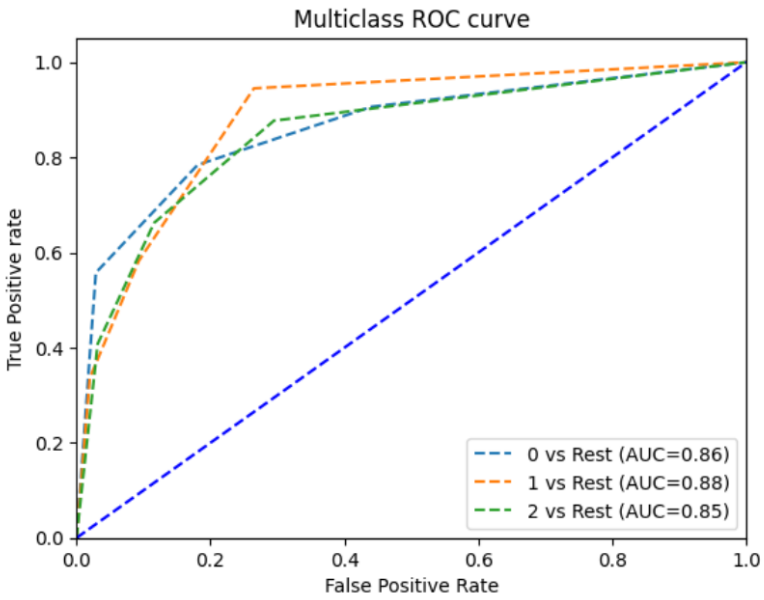


Figure 3.9: ROC curve for KNN

Chapter 4

RESULTS AND DISCUSSION FOR DL ALGORITHMS

4.1 RESULTS

In this study, we evaluated the performance of three deep learning models—DenseNet[1], EfficientNetB3[9], and MobileNetV2[7]—on our dataset for image classification tasks. After training each model, we compared their accuracies, with DenseNet achieving the highest accuracy of 99.002%, outperforming both the discussed machine learning models and other deep learning architectures. We provided insights into the architectures of DenseNet, MobileNetV2 and EfficientNetB3, highlighting their unique features for efficient image classification. Specifically, MobileNetV2 incorporates depthwise separable convolutions and other optimization techniques to reduce computational complexity while maintaining accuracy. On the other hand, EfficientNetB3 employs a compound scaling method to uniformly scale network dimensions, resulting in efficient models with fewer parameters. Our experiments revealed that using EfficientNetB3 as a base model in the architecture mentioned in Figure 4.1 achieved a training accuracy of 98.7% and a test accuracy of 98.2% after ten epochs of training. We visualized the training progress, presented confusion matrices, and provided classification reports for all three models to further analyze their performance.

Hence it was observed as DenseNet performs the best on our dataset and outperforms the machine learning models discussed in Chapter 3 and also other deep learning models discussed in this Chapter. The General architecture that we followed while training our models is given in figure 4.1 .

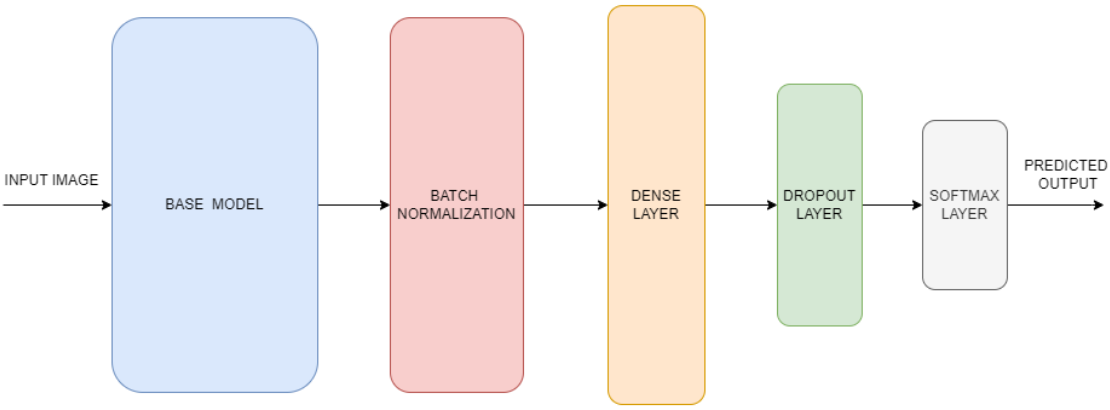


Figure 4.1: Architecture followed while training the model [3]

4.1.1 MOBILENETV2 MODEL

MobileNet[3] is a computer vision model open-sourced by Google and designed for training classifiers. MobileNetV2[7] incorporates several key features that contribute to its efficiency and effectiveness in image classification tasks. The General architecture of MobileNet Base Model is given in figure 4.2

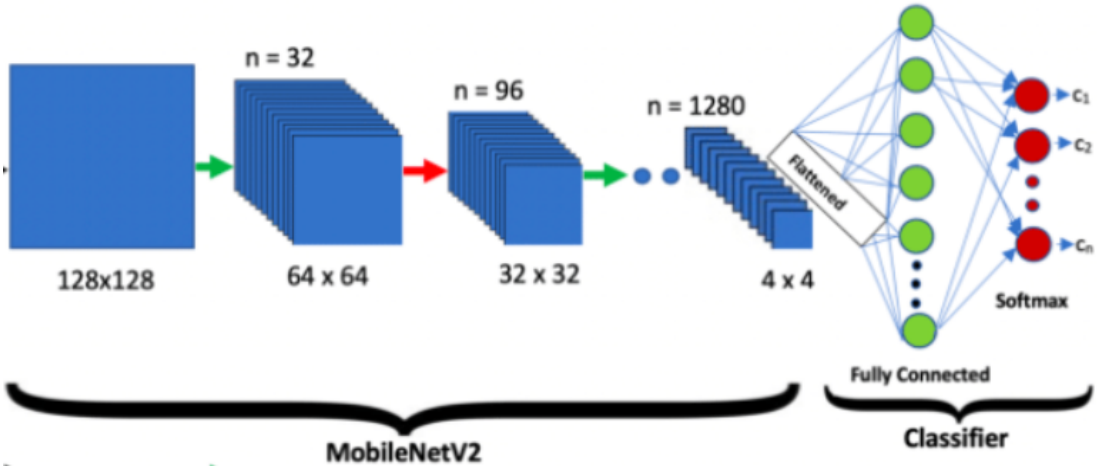


Figure 4.2: General Architecture of MobileNetV2 model[3]

It consists of a series of convolutional layers, followed by depthwise separable convolutions[2], inverted residuals, bottleneck design, linear bottlenecks, and squeeze-and-excitation (SE) blocks[4]. Each of these features plays a crucial role in reducing the computational complexity of the model while maintaining high accuracy.
We trained using MobileNetV2 as base model , on our dataset for 10 epochs that took nearly 8 minutes and gave training accuracy of about 98% and test accuracy of 93%. Figure 4.3 depicts the plot of training and validation loss and accuracy of the model. Figure 4.4 shows the confusion matrix for the model. Table 4.1 is the classification report for the same.

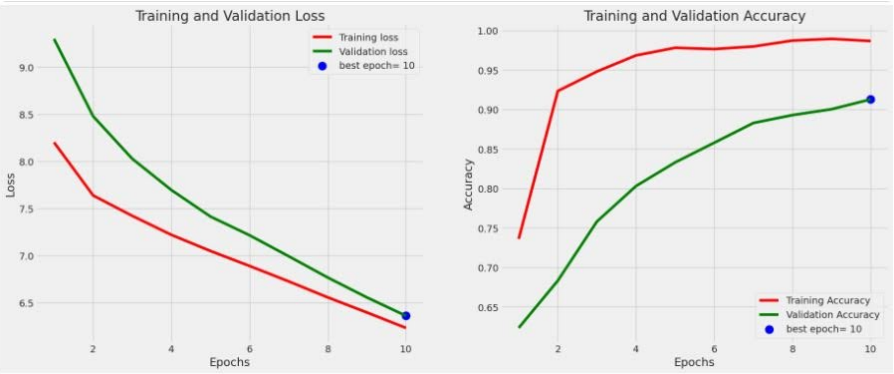


Figure 4.3: Traning and validation loss and accuracy of the MobileNetV2

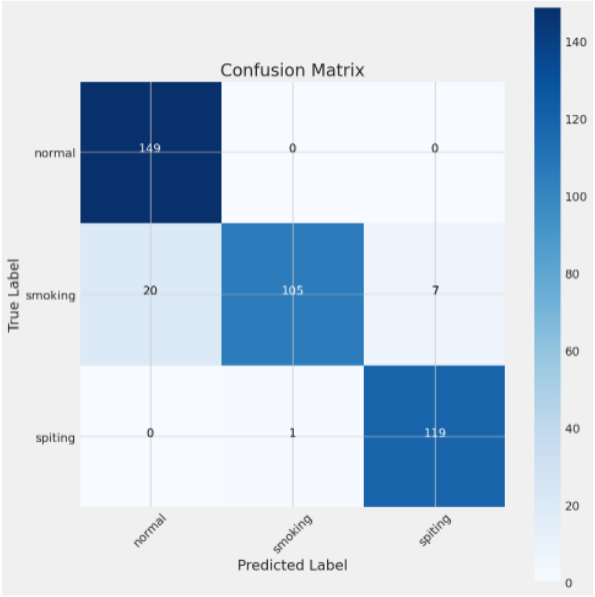


Figure 4.4: Confusion Matrix of MobileNetV2

4.1.2 EFFICIENTNETB3 MODEL

EfficientNet[9] is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. Unlike conventional practice that arbitrarily scales these factors, the EfficientNet scaling method uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients. The compound scaling method is justified by the intuition that if the input image is bigger, then the network needs more layers to increase the receptive field and more channels to capture more fine-grained patterns on the bigger image. Figure 4.5 shows the scaling methods used in EfficientNetB3.
With considerably fewer numbers of parameters, the EfficientNet models are efficient and also provide better results. All the EfficientNet models have two set of layers called the 'stem' and the 'final layers' at the start and end respectively.

After the each of them EfficientNet Architectures contains 7 blocks. These blocks

Table 4.1: Classification report for MobileNetV2

	Precision	Recall	f1 - score	support
normal	0.88	1.00	0.94	149
smoking	0.99	0.80	0.88	132
spitting	0.94	0.99	0.97	120
accuracy			0.93	401
macro avg	0.94	0.93	0.93	401
weighted avg	0.94	0.93	0.93	401

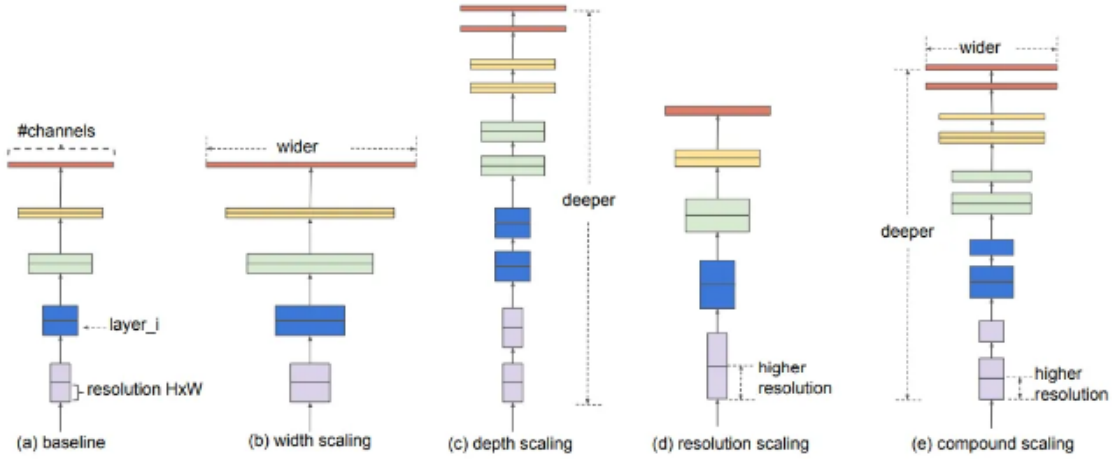


Figure 4.5: Model Scaling. (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is the compound scaling method proposed by EfficientNet that uniformly scales all three dimensions with a fixed ratio. [9]

further have a varying number of sub-blocks whose number is increased as we move from EfficientNetB0 to EfficientNetB7. We trained using EfficientNetB3 as base model, on our dataset for 10 epochs that took nearly 8 minutes and gave training accuracy of about 98.7% and test accuracy of 98.2%.

Figure 4.6 depicts the traning and validation loss and accuracy of the EfficientNetB3 Figure 4.7 shows the confusion matrix of the model Table 4.2 represents the classification report for the same:

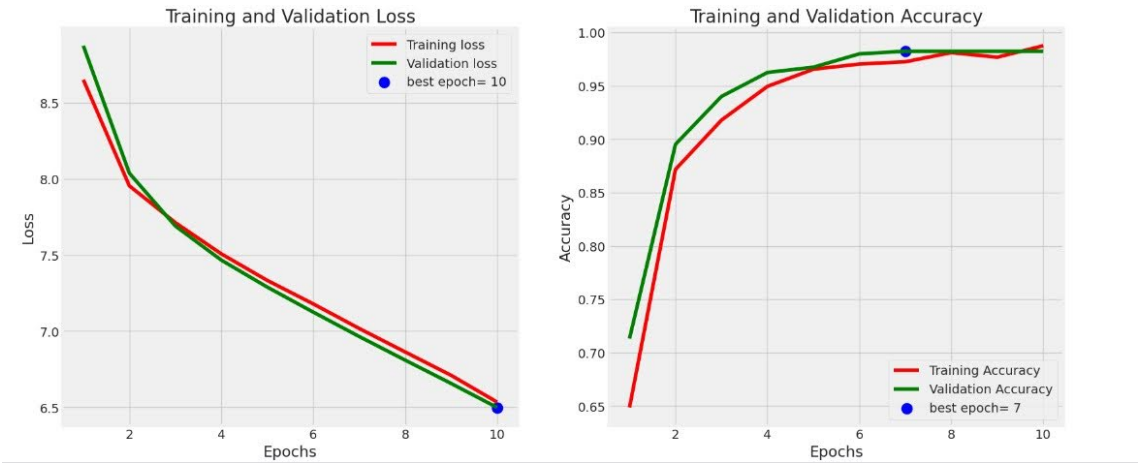


Figure 4.6: Traning and validation loss and accuracy of the EfficientNetB3

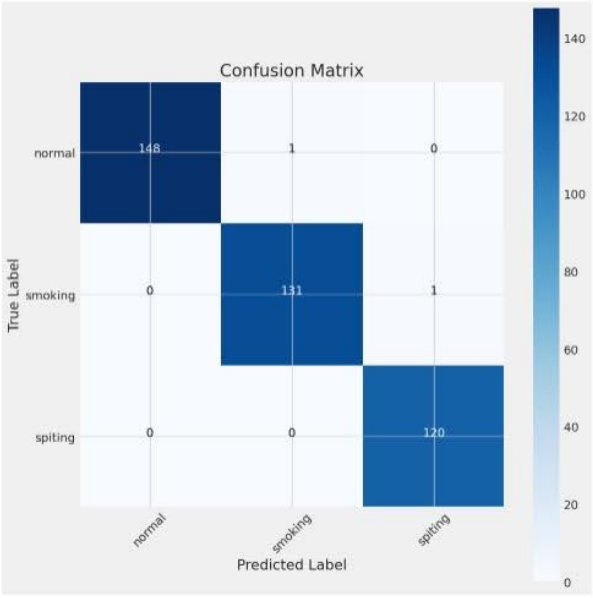


Figure 4.7: Confusion Matrix For EfficientNetB3 Architecture

4.1.3 DENSENET MODEL

DenseNet[2] is a type of CNN architecture that is known for its efficiency and accuracy in image classification tasks. DenseNet has two key features that make it stand out from other CNN architectures. First, it has a dense block structure, where each layer is connected to every other layer in a feedforward fashion. Second, it uses bottleneck layers that help reduce the number of parameters without reducing the number of features learned by the network. Figure 4.8 shows the architecture of the DenseNet121 model that we have used as base model.

A dense block consists of multiple bottleneck layers connected to each other in a feedforward fashion. Each bottleneck layer takes the output of the previous layer as input and produces a fixed number of output feature maps. The transition layer consists

Table 4.2: Classification report for EfficientNetB3

	Precision	Recall	f1 - score	support
normal	1.00	0.99	1.00	149
smoking	0.99	0.99	0.99	132
spitting	0.99	1.00	1.00	120
accuracy			1.00	401
macro avg	0.99	1.00	0.99	401
weighted avg	1.00	1.00	1.00	401

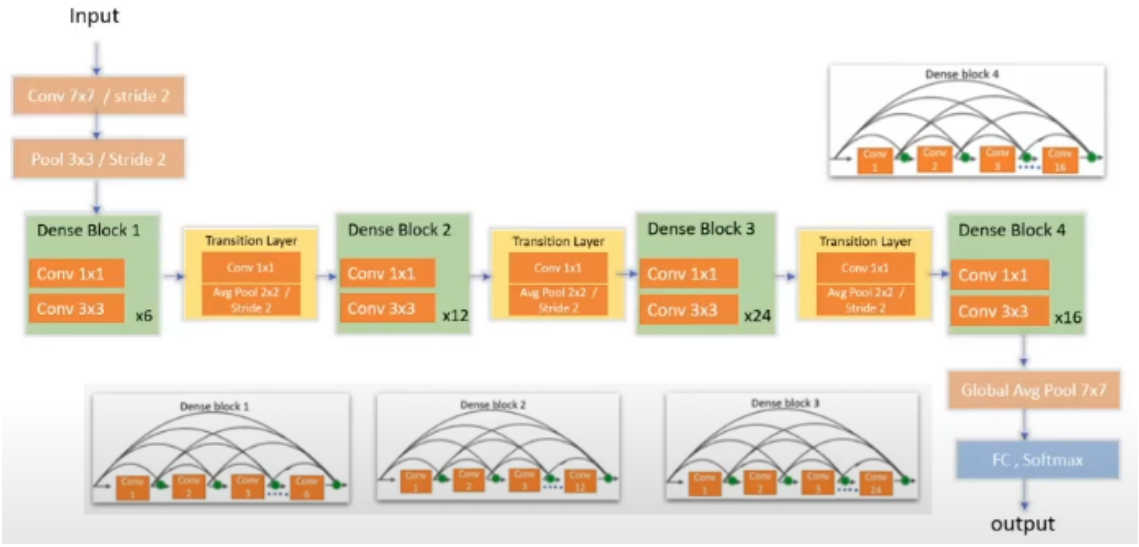


Figure 4.8: Architecture of DenseNet121 model[2]

of a batch normalization layer, a 1x1 convolutional layer, and a 2x2 average pooling layer. The batch normalization layer and the convolutional layer reduce the number of feature maps, and the pooling layer reduces the spatial dimensionality of the feature maps. After passing through all of the blocks in the network, the output is then passed through a global average pooling layer. This layer averages the values across all of the feature maps, resulting in a single value for each feature. Finally, the output of the global average pooling layer is passed through a fully connected layer, which performs a linear transformation of the input and produces the final output.

We trained using DenseNet as base model , on our dataset for 10 epochs that took nearly 2 hours and gave training accuracy of about 99.5% and test accuracy of 99.02%. Figure 4.9 depicts the traning and validation loss and accuracy of the DenseNet. Figure 4.10 shows the confusion matrix for the DenseNet Table 4.3 represents the classification report for the same:

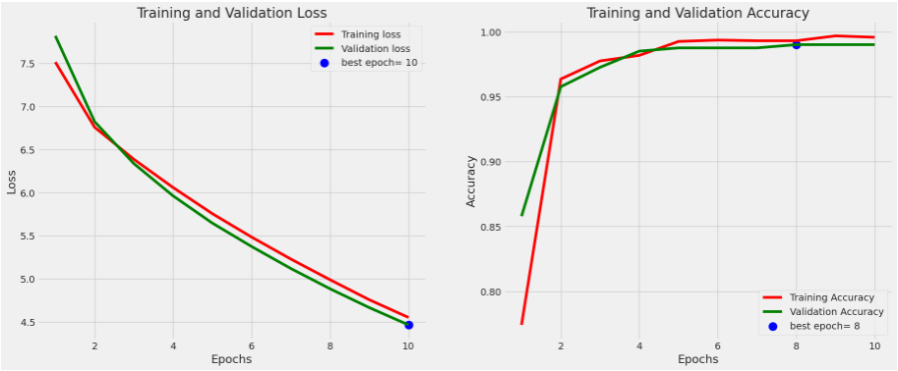


Figure 4.9: Traning and validation loss and accuracy of the EfficientNetB3

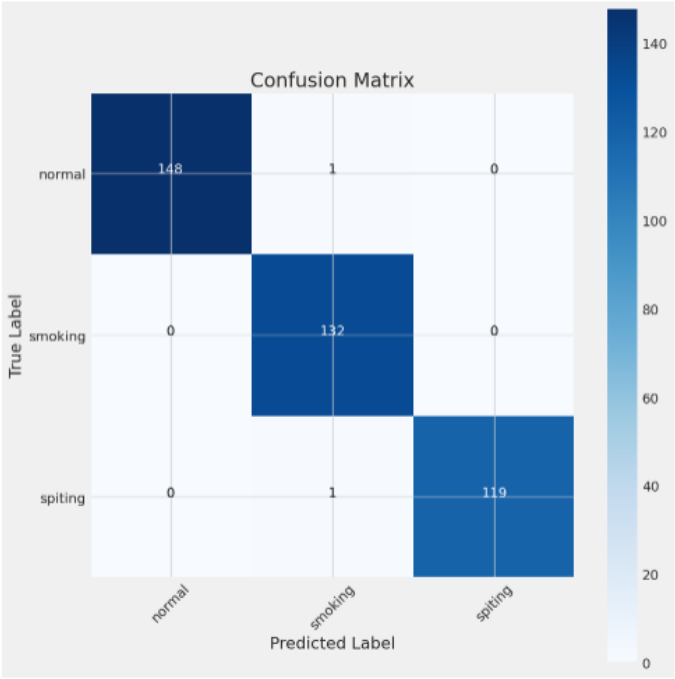


Figure 4.10: Confusion Matrix of the DenseNet Architecture

Table 4.3: Classification report for DenseNet

	Precision	Recall	f1 - score	support
normal	1.00	0.99	1.00	149
smoking	0.99	1.00	0.99	132
spitting	1.00	0.99	1.00	120
accuracy			1.00	401
macro avg	1.00	0.99	0.99	401
weighted avg	1.00	1.00	1.00	401

Chapter 5

DEPLOYMENT

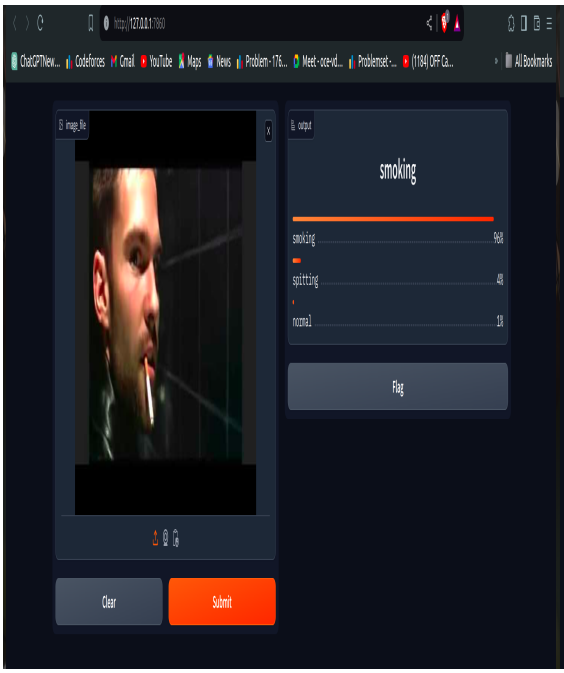
We've integrated Gradio, a sophisticated Python library developed by the RTC team, to enhance our project with interactive capabilities for our machine learning models. Gradio stands out as an optimal choice due to its user-friendly interface and comprehensive feature set.

With Gradio, we are able to effortlessly design dynamic interfaces without requiring extensive expertise in web development. Its robust support for a diverse range of input types, including images, text, and audio, aligns seamlessly with our varied model architectures and application scenarios. Furthermore, Gradio provides a variety of output components, enabling intuitive visualization of model predictions and outputs for users.

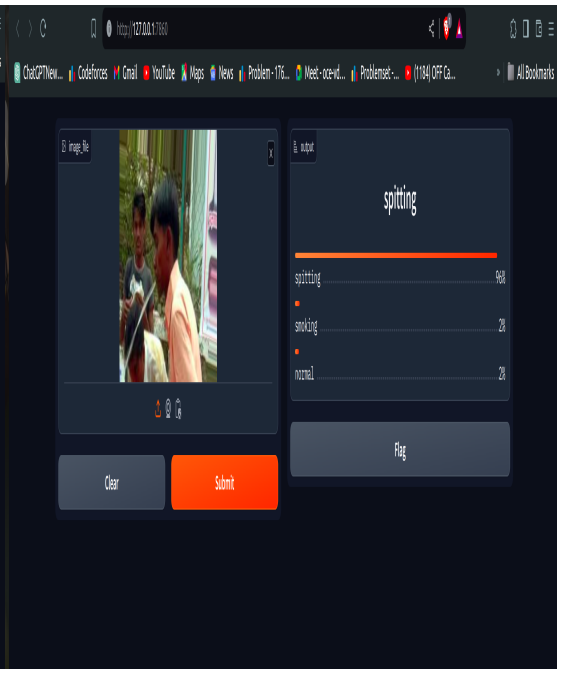
Gradio simplifies the interface creation process through its intuitive Python syntax. By defining our model's prediction function and input/output components, Gradio automates the intricate details of web interface generation, resulting in polished and user-friendly interfaces. It offers extensive customization options, allowing us to tailor the appearance and functionality of our interfaces to suit our project's specific requirements precisely. Whether adjusting styling elements, layout configurations, or integrating custom JavaScript code, Gradio provides the flexibility needed to create a tailored user experience.

Additionally, Gradio facilitates seamless deployment across various platforms, including local servers, cloud services, and standalone desktop applications. This versatility in deployment options ensures widespread accessibility and engagement with our machine learning models.

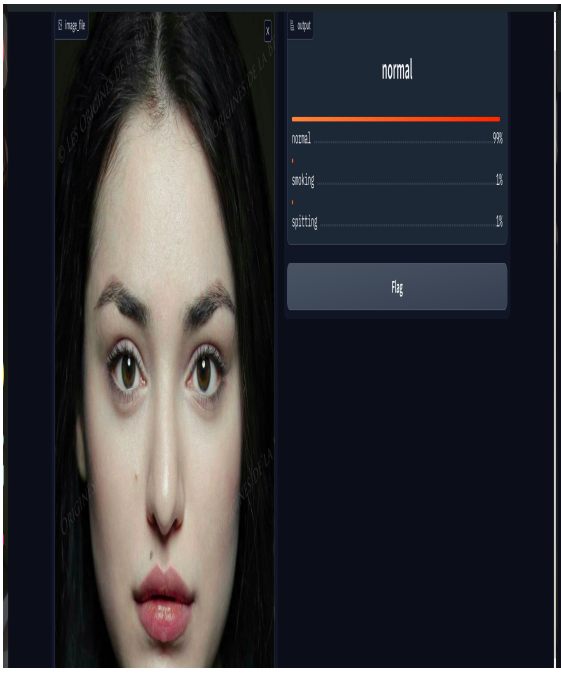
In summary, Gradio serves as an invaluable tool for enhancing the presentation and usability of our machine learning models. Its intuitive interface creation process, comprehensive customization capabilities, and seamless deployment options contribute to a compelling user experience, ultimately amplifying the impact and accessibility of our work.



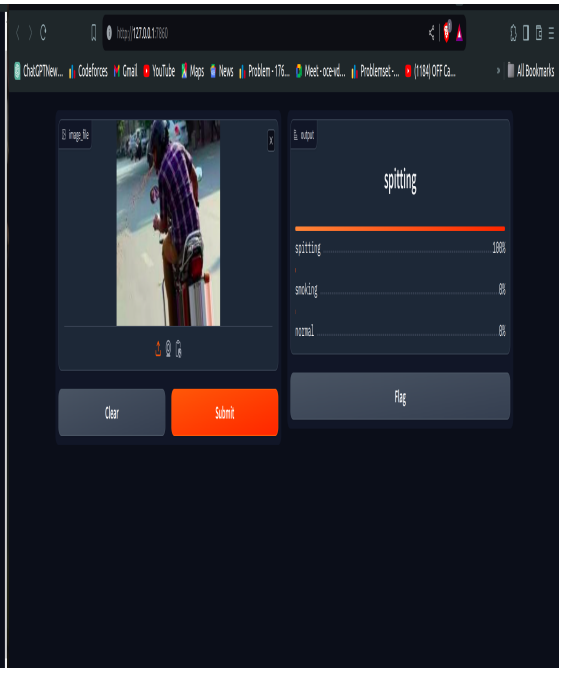
(a) The model successfully predicted with 96 percent probability that the person was smoking.



(b) The model successfully predicted with 94 percent probability that the person was spitting in public place.



(c) The model successfully predicted with 94 percent probability that the person was not doing any abnormal behaviour



(d) The model successfully predicted with 98 percent probability that the person was spitting on public place

Figure 5.1: Various result were obtained when following images were uploaded on our platform

Chapter 6

CONCLUSION

We successfully collected datasets from various sources. The dataset for smoking and images of normal people was extracted from Kaggle. However, since a dataset for spitting was not available on any sites, we collected images of spitting from various sources such as Google, Yahoo, and other platforms.

Firstly, we ensured that all the images were of equal size. Then, the task was to augment these images, as augmentation improves the training of the data. These images were properly labeled.

The next task was to train our ML models on the dataset created by us. We tried various models on the data. SVM showed an accuracy of 79.97%, while KNN's accuracy was even lower at 69.54%. We then attempted Random Forest, which achieved a maximum accuracy of 84.45%. Unfortunately, such accuracy levels are inadequate for deployment in public places due to the potential for both false positives and false negatives. False positives could wrongly accuse tourists, negatively impacting their experience, while false negatives could result in perpetrators going undetected, leading to damage to heritage sites. Consequently, Machine Learning models are deemed unsuitable for this purpose.

We then shifted our focus to Deep Learning. Initially, we trained our data using the base model MOBILENETV2, which achieved an accuracy of 91.27%, better than Random Forest. However, we decided to explore more base models of Deep Learning. We applied the EFFICIENTNETB3 base model, which gave us an accuracy of 98.254%, significantly better than all the methods we had tried until then. Finally, we also utilized the DENSENET base model, which achieved an accuracy of 99.002%, the highest so far. However, DENSENET took more time than other models as it is resource-heavy.

Hence, we concluded that DENSENET should be used for practical purposes in Heritage sites. This model is highly effective in capturing culprits while minimizing false positives, thereby preserving the tourist experience. However, if resources are constrained, we can use the EFFICIENTNETB3 base model, as it achieves an accuracy of 98.254% while utilizing fewer resources.

Chapter 7

FUTURE SCOPE

Even if the performance of our present model seems promising, improvement and refinement are still being worked on. A crucial path forward for its development is to expand the dataset, especially with relation to pictures of people spitting. The model can get a more sophisticated knowledge of this behaviour and increase its predicted accuracy by adding more spitting cases to the dataset from a variety of settings and contexts.

We used traditional machine learning algorithms in our first experiment, such as KNN, SVM, and Random Forest, and obtained 79.97%, 69.54%, and 84.45% accuracy rates, respectively. Still, investigating other algorithms might reveal better models that are suited to the subtleties of our dataset, which would improve performance all around.

With accuracies ranging from 93% to 99.002%, our experiments with well-known base models in the field of Deep Learning, including DENSENET, EFFICIENTNETB3, and MOBILENETV2, produced promising results. However, there is a wide range of new architectures and methods in the field of deep learning. Investigating these innovative approaches may reveal substitutes that meet or even exceed DENSENET's performance while using computational resources more effectively.

The final objective is to smoothly incorporate our model into practical applications, especially those related to public space monitoring infrastructure. This means putting in place a strong system that allows security cameras to take live pictures or videos of people. Our software then receives these visual inputs and instantly analyses them to find and highlight instances of anomalous behaviour. We can protect the comfort and well-being of people using public areas while also strengthening their security and integrity by utilising AI in this way.

Chapter 8

CONTRIBUTION

As this was a team project, everyone was involved in each and every part of it. The project served as a learning curve for our group, as we gained many insights throughout its development. As mentioned earlier, each section had the involvement of every member of the team. However, if we were to discuss the major individual contributions in each section.

Sankalp: Responsible for gathering data, augmentation, cleaning and preprocessing of data.

Gaurang: Responsible for applying ML algorithms on the preprocessed dataset and analysis based on various metrics.

Astitva: Responsible for applying DL algorithms on the preprocessed dataset and analysis based on various metrics.

Lata: Deploying of the made model and creating the UI for better visualisation and testing.

REFERENCES

[1] Wikipedia list of world heritage sites in india, 2023.

[2] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, July 2017.

[3] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[4] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

[5] Keras contributors. Keras documentation: Image data augmentation. https://keras.io/api/layers/preprocessing$_{l}ayers/image_{a}ugmentation/$, 2024.

[6] Sujata Khedkar, Anish Vaidya, Monika R Thorat, Motwani, and Chaitanya R Shinde. Swachh ai: Real-time spit detection using yolov3 darknet model. *SSRN Electronic Journal*, 2021.

[7] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[8] Xiaoyu Song. Prediction of people's abnormal behaviors based on machine learning algorithms. In *2022 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE)*, pages 406–409, 2022.

[9] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *ArXiv*, abs/1905.11946, 2019.