SMART-INTERNZ INTERNSHIP AI&ML

PROJECT REPORT:

End-To-End Deep Learning Project For Detecting Melanoma Diseases.

TEAM NO:

592277

TEAM MEMBERS: VIT-AP

1. Gattu Prabhat Sushrut-21BCE8542
2. Alex Dandamudi-21BCE9937
3. SHAIK SAIFULLA- 21BEC7068

Project Report

Detecting Melanoma Diseases using Deep Learning.

1. INTRODUCTION

1.1 Project Overview:

The objective of this end-to-end deep learning project is to develop a and accurate system for the early detection of melanoma, a deadly form of skin cancer. Melanoma is known for its rapid progression, and early diagnosis is critical for effective treatment. Current methods for detecting melanoma involve visual inspection by dermatologists, which can be time-consuming and subject to human error therefore project aims to create an automated system that can assist medical professionals in diagnosing melanoma more quickly and accurately.The proposed model simplifies the classification process by directly using raw images as input. It automatically learns important features from the images, removing the need for complicated lesion segmentation and feature extraction steps. This approach streamlines the classification process and improves efficiency

1.2 Purpose:

To detect melanoma disease with accuracy for early treatment and make the process easily accessible to everyone the novelty of the present methodology lies in its ability to perform skin lesion detection in a very quick time, which in turn helps technicians enhance their diagnostic skills. . The speed at which the methodology can accurately identify and classify skin lesions allows technicians to efficiently analyze a large number of images and make quicker and more accurate diagnostic decisions . Firstly, early detection allows for interventions that can significantly impact disease progression. It provides a window of opportunity for targeted therapies, lifestyle modifications, and cognitive interventions that might slow down the advancement of symptoms. Moreover, it empowers individuals and their families with critical information, enabling them to plan for the future, make informed decisions about care, and access support networks that specialize in melanoma care and management. This early awareness can alleviate stress, facilitate better coping mechanisms, and foster a proactive approach to handling the challenges associated with the disease.

2. LITERATURE SURVEY

2.1 Existing problem

Deep learning models require large and diverse datasets for training. In the case of melanoma detection, obtaining a sufficiently large dataset with diverse melanoma cases can be challenging. Limited data can lead to overfitting, where the model performs well on the training data but fails to generalize to new, unseen cases. Collecting high-quality, labeled skin lesion images for melanoma can be a labor-intensive and time-consuming process. The need for expert dermatologists to annotate images further complicates the data collection process. This can result in datasets that are limited in size and may not adequately represent the diversity of melanoma cases. Melanomas can manifest in various forms, colors, and stages. Capturing this heterogeneity in the dataset is essential for training a model that can generalize well to different presentations of the disease. However, obtaining a diverse set of melanoma cases that covers the spectrum of variations is challenging. Compounding these challenges are limitations in data accessibility and quality. Predictive models heavily depend on comprehensive and diverse datasets that capture the multifaceted aspects of melanoma, including genetic predispositions, lifestyle factors, and biomarkers. However, the availability of such datasets is often limited, impacting the accuracy and generalizability of predictive algorithms. Ethical concerns regarding data privacy and security add another layer of complexity, necessitating stringent measures to ensure the responsible handling of sensitive medical information required for effective prediction models. Addressing these multifaceted challenges demands collaborative efforts across disciplines, technological advancements, and a deeper understanding of the disease's underlying mechanisms.

2.2 References

Ahmed, Bilal, Muhammad Imran Qadir, and Saba Ghafoor. "Malignant Melanoma: Skin Cancer− Diagnosis, Prevention, and Treatment." *Critical Reviews™ in Eukaryotic Gene Expression* 30.4 (2020).

Kanavy, H. E., & Gerstenblith, M. R. (2011, December). Ultraviolet radiation and melanoma. In *Seminars in cutaneous medicine and surgery* (Vol. 30, No. 4, pp. 222-228). WB Saunders.

Soenksen, L.R., Kassis, T., Conover, S.T., Marti-Fuster, B., Birkenfeld, J.S., Tucker-Schwartz, J., Naseem, A., Stavert, R.R., Kim, C.C., Senna, M.M. and Avilés-Izquierdo, J., 2021. Using deep learning for dermatologist-level detection of suspicious pigmented skin lesions from wide-field images. *Science Translational Medicine*, *13*(581), p.eabb3652.

M, J. S., P, M., Aravindan, C., & Appavu, R. (2022, October 12). Classification of skin cancer from dermoscopic images using deep neural network architectures - Multimedia Tools and Applications. SpringerLink. https://doi.org/10.1007/s11042-022-13847-3 [16]

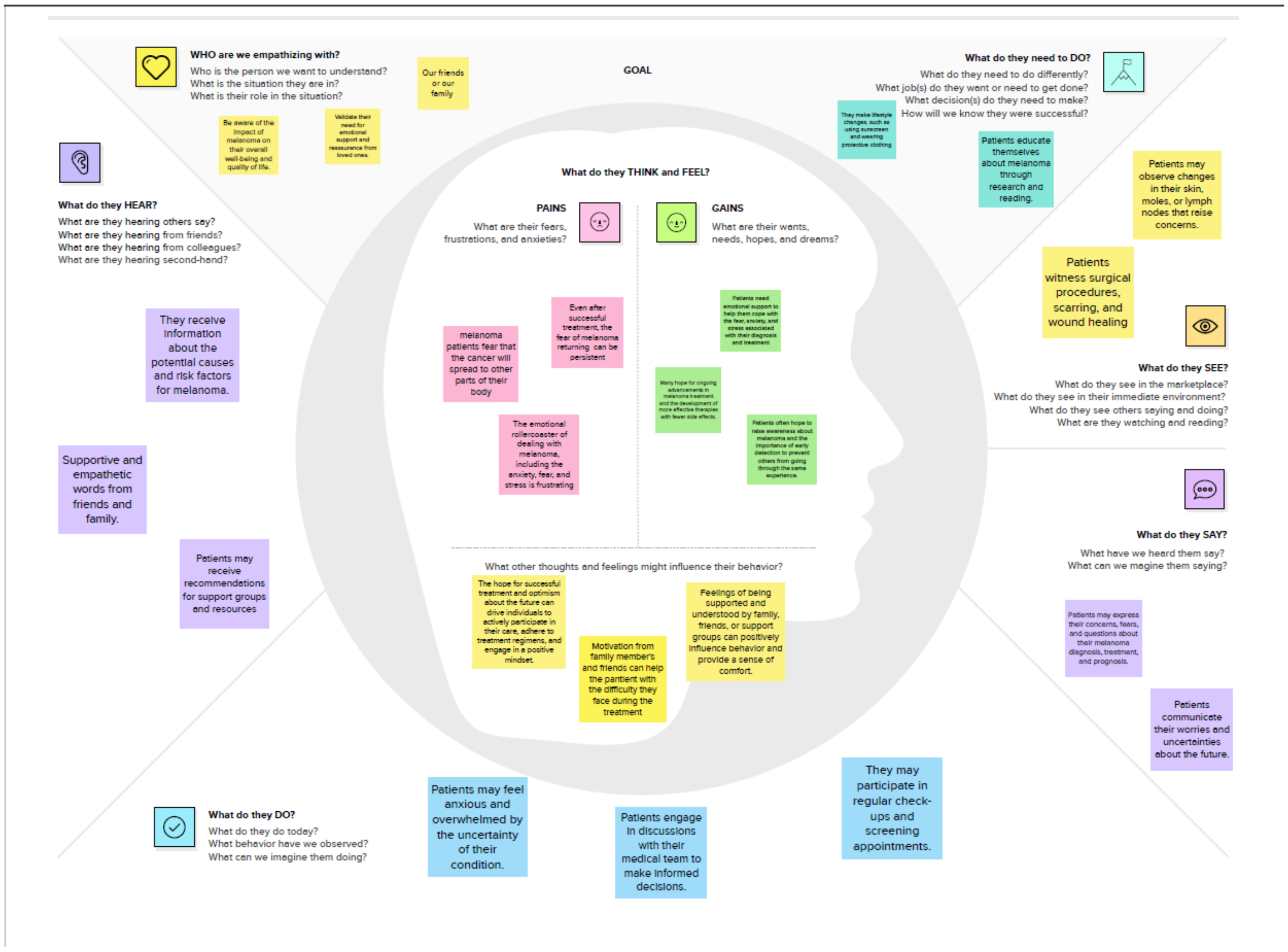
Kalouche S. Vision-Based Classification of Skin Cancer Using Deep Learning. [(accessed on 10 January 2021)];2016 Available online: https://www.semanticscholar.org/paper/Vision-Based-Classification-of-Skin-Cancer-using-Kalouche/b57ba909756462d812dc20fca157b3972bc1f533

2.3 Problem Statement Definition

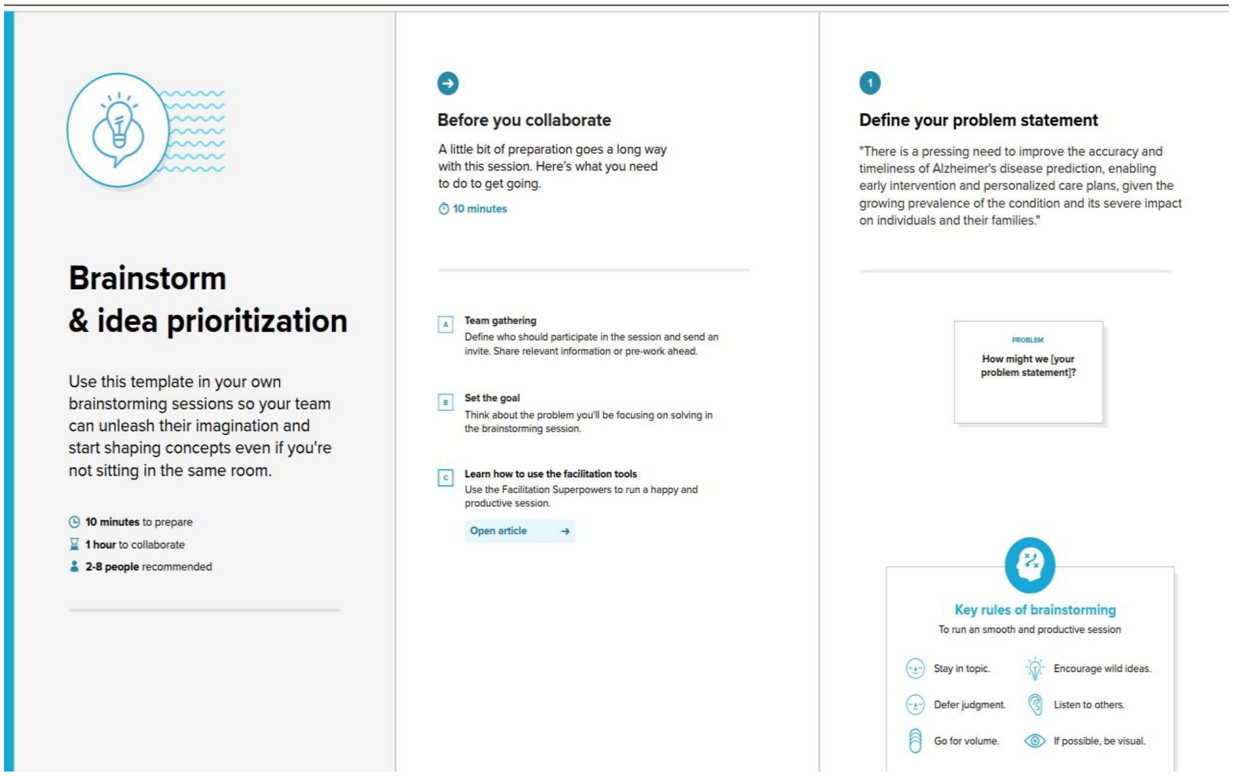
The accurate and timely detection of melanoma, a deadly form of skin cancer, remains a formidable challenge in healthcare. Leveraging deep learning for melanoma detection holds immense promise, yet several critical obstacles impede the development of robust and clinically applicable solutions. Limited access to large, diverse datasets containing high-quality annotated melanoma images hinders the ability of deep learning models to generalize effectively. The rarity of melanoma cases, coupled with the need for expert annotation, contributes to class imbalance and challenges the model's capacity to discern subtle but crucial features indicative of malignancy. Variability in imaging conditions and the heterogeneous nature of melanoma lesions further complicates the training process, impacting the model's ability to handle real-world scenarios. Moreover, issues related to the interpretability of deep learning models, their integration into clinical workflows, and the imperative for real-time processing in dermatological examinations demand comprehensive solutions. Addressing these challenges is paramount for the successful deployment of deep learning models in the early detection of melanoma, thereby improving patient outcomes and reducing the burden of this life-threatening disease.

3. <u>IDEATION & PROPOSED SOLUTION</u>

3.1 Empathy Map Canvas


WHO are we empathizing with?
Who is the person we want to understand?
What is the situation they are in?
What is their role in the situation?
Our friends or our family
Be aware of the impact of melanoma on their overall well-being and quality of life.
Validate their need for emotional support and reassurance from loved ones.
GOAL
What do they need to DO?
What do they need to do differently?
What job(s) do they want or need to get done?
What decision(s) do they need to make?
How will we know they were successful?
They make lifestyle changes, such as using sunscreen and wearing protective clothing
Patients educate themselves about melanoma through research and reading.
Patients may observe changes in their skin, moles, or lymph nodes that raise concerns.
What do they HEAR?
What are they hearing others say?
What are they hearing from friends?
What are they hearing from colleagues?
What are they hearing second-hand?
They receive information about the potential causes and risk factors for melanoma.
Supportive and empathetic words from friends and family.
Patients may receive recommendations for support groups and resources
What do they THINK and FEEL?
PAINS
What are their fears, frustrations, and anxieties?
melanoma patients fear that the cancer will spread to other parts of their body
Even after successful treatment, the fear of melanoma returning can be persistent
The emotional rollercoaster of dealing with melanoma, including the anxiety, fear, and stress is frustrating
GAINS
What are their wants, needs, hopes, and dreams?
Patients need emotional support to help them cope with the fear, anxiety, and stress associated with their diagnosis and treatment
Many hope for ongoing advancements in melanoma treatment and the development of more effective therapies with fewer side effects.
Patients often hope to raise awareness about melanoma and the importance of early detection to prevent others from going through the same experience.
What other thoughts and feelings might influence their behavior?
The hope for successful treatment and optimism about the future can drive individuals to actively participate in their care, adhere to treatment regimens, and engage in a positive mindset.
Motivation from family member's and friends can help the pantient with the difficulty they face during the treatment
Feelings of being supported and understood by family, friends, or support groups can positively influence behavior and provide a sense of comfort.
Patients witness surgical procedures, scarring, and wound healing
What do they SEE?
What do they see in the marketplace?
What do they see in their immediate environment?
What do they see others saying and doing?
What are they watching and reading?
What do they SAY?
What have we heard them say?
What can we magine them saying?
Patients may express their concerns, fears, and questions about their melanoma diagnosis, treatment, and prognosis.
Patients communicate their worries and uncertainties about the future.
What do they DO?
What do they do today?
What behavior have we observed?
What can we imagine them doing?
Patients may feel anxious and overwhelmed by the uncertainty of their condition.
Patients engage in discussions with their medical team to make informed decisions.
They may participate in regular check-ups and screening appointments.

3.2 Ideation & Brainstorming



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article

1 **Define your problem statement**

"There is a pressing need to improve the accuracy and timeliness of Alzheimer's disease prediction, enabling early intervention and personalized care plans, given the growing prevalence of the condition and its severe impact on individuals and their families."

PROBLEM
How might we [your problem statement]?

Key rules of brainstorming
To run an smooth and productive session

Stay in topic.	Encourage wild ideas.
Defer judgment.	Listen to others.
Go for volume.	If possible, be visual.

2

Brainstorm

Alex

Gather information about melanoma, its prevalence, and the current methods of detection.

- Outline your approach for splitting data into training, validation, and test sets.
- Define evaluation metrics (e.g., accuracy, precision, recall, F1-score) for model performance.

Consider various sources of data:
- Dermatological images
- Patient history
- Biopsy results

Prabhat

Brainstorm various machine learning and deep learning models:
- Convolutional Neural Networks (CNNs)
- Support Vector Machines (SVM)
- Decision Trees
- Ensemble methods

Explore the importance of early detection and its impact on patient outcomes.

- Explore techniques for hyperparameter tuning.
- Consider data augmentation methods to improve model performance.

Saifulla

- Explore techniques for hyperparameter tuning.
- Consider data augmentation methods to improve model performance.

- Describe the testing process, including user testing and clinical validation.
- Address any feedback and make necessary improvements.

- Create documentation for the project, including code, model details, and user guides.
- Prepare a final report summarizing the project's findings and outcomes.

3

Group ideas

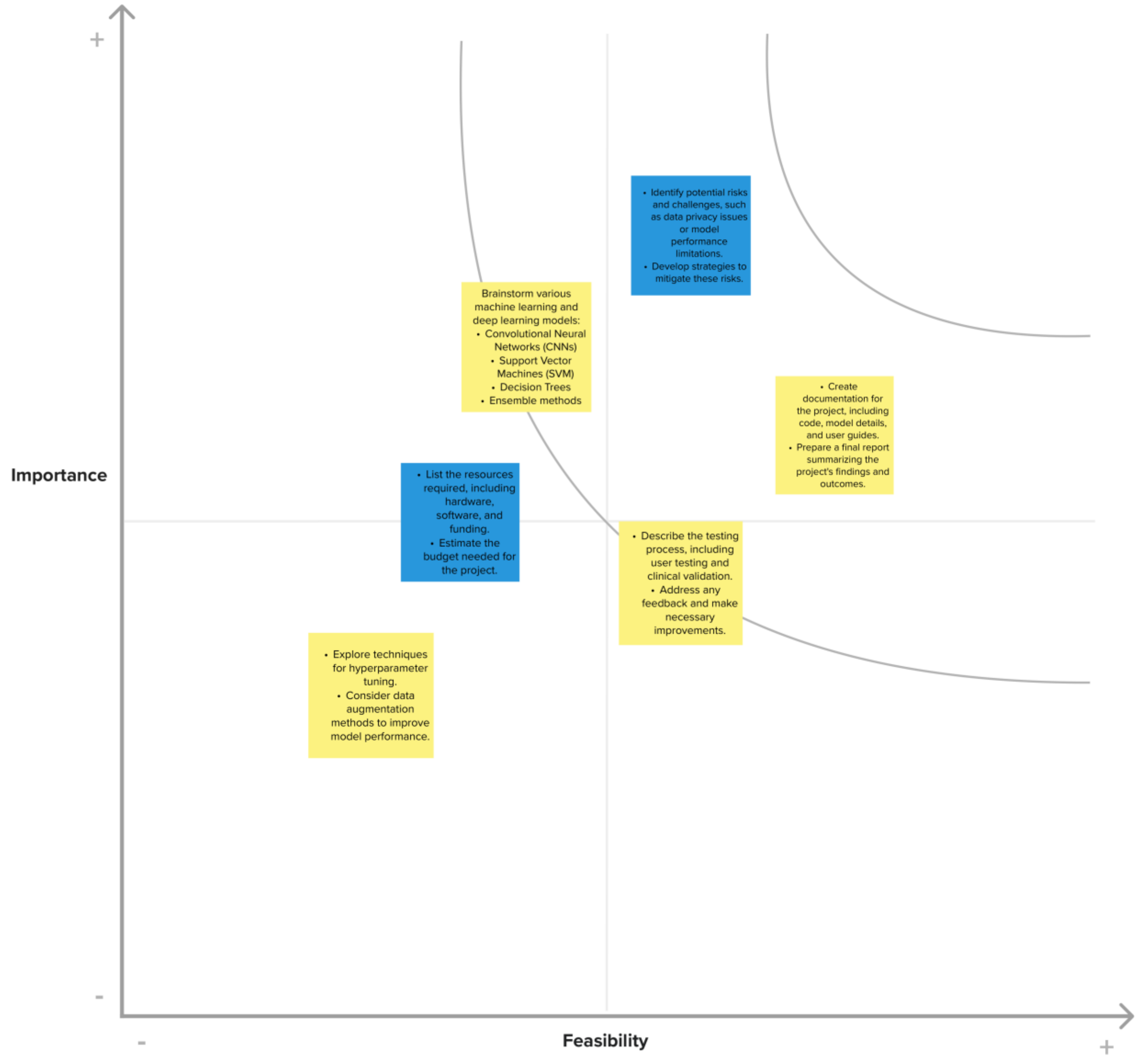
- Create a timeline with milestones for different project phases.
- Ensure a realistic and achievable project schedule.

- List the resources required, including hardware, software, and funding.
- Estimate the budget needed for the project.

- Identify potential risks and challenges, such as data privacy issues or model performance limitations.
- Develop strategies to mitigate these risks.

4

Prioritize


+
Importance
-
-
Feasibility
+
Identify potential risks and challenges, such as data privacy issues or model performance limitations.
Develop strategies to mitigate these risks.
Brainstorm various machine learning and deep learning models:
Convolutional Neural Networks (CNNs)
Support Vector Machines (SVM)
Decision Trees
Ensemble methods
Create documentation for the project, including code, model details, and user guides.
Prepare a final report summarizing the project's findings and outcomes.
List the resources required, including hardware, software, and funding.
Estimate the budget needed for the project.
Describe the testing process, including user testing and clinical validation.
Address any feedback and make necessary improvements.
Explore techniques for hyperparameter tuning.
Consider data augmentation methods to improve model performance.

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

- Comparative analysis of previous model is done
- Identifying and classifying the type of skin cancer
- Data Collection and Integration
- User Interface for Data Input
- Prediction Reporting

4.2 Non-Functional requirements

- Security: Ensure robust security measures for patient data, complying with healthcare data privacy regulations (e.g., HIPAA).
- Performance: The system should handle a large volume of data efficiently and provide timely predictions without significant delays.
- Reliability: Maintain high accuracy in predictions while minimizing false positives/negatives to instill confidence in healthcare professionals using the system.
- Scalability: Design the system to accommodate future expansions, allowing for the addition of new data sources or improvements in prediction models.
- Usability: Ensure ease of use for healthcare professionals interacting with the system, providing clear interfaces and intuitive functionalities.
- Interoperability: Enable compatibility with existing healthcare systems or databases to facilitate data exchange and integration.

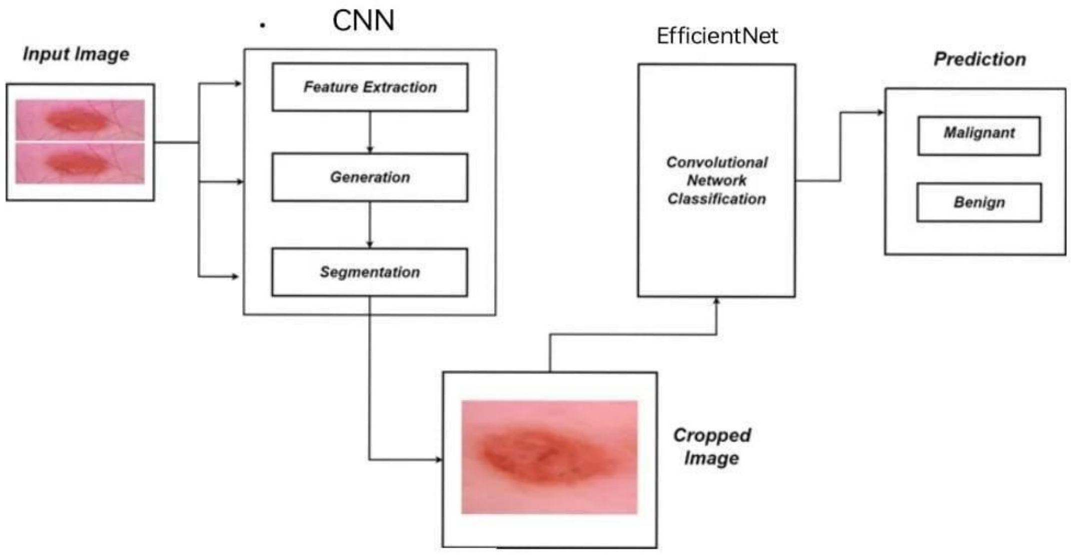
5 .PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

User Stories

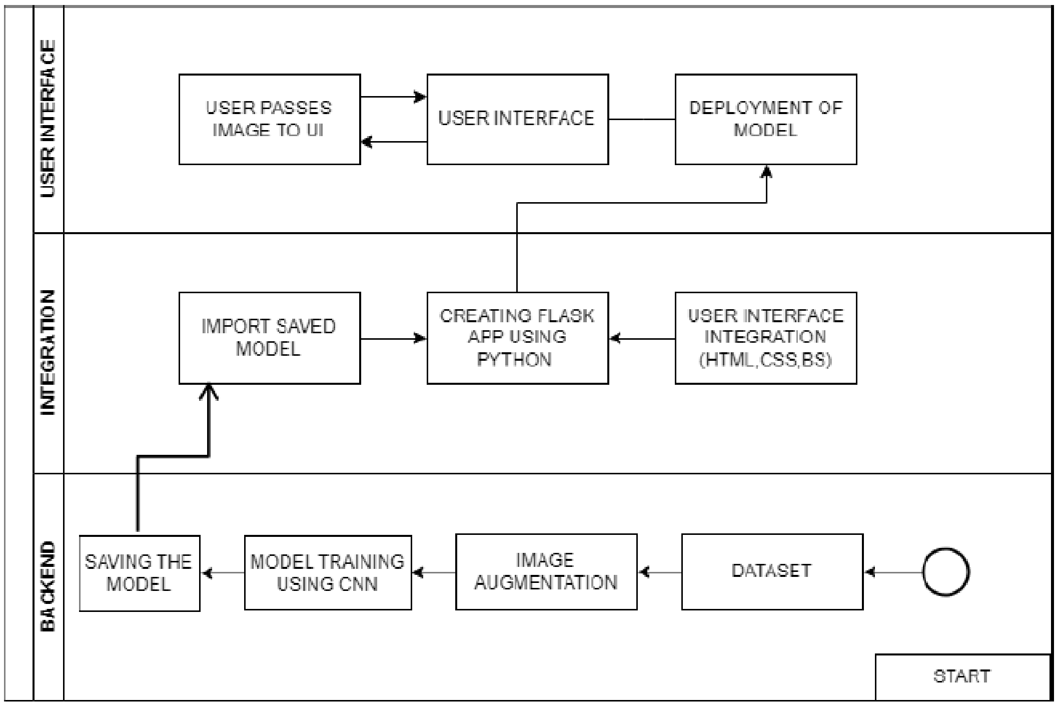
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria
Healthcare	Project setup & Infrastructure	USN-1	Set up the development environment with the required tools, frameworks and libraries to start the detection or predicting	completed setting all necessary tools, frameworks and libraries
Researchers	Developing environment	USN-2	Gather a diverse dataset of lesions in different parts of the body for training the Deep learning model	Gathered a diverse dataset of images depicting various stages of disease
Public Health Officials	Feature extraction	USN-3	Feature extraction involves automatically identifying and selecting significant patterns or attributes from raw data, enabling models to focus on prediction.	We could remove the unwanted from the dataset and focus on the patterns
Educational Institutions	Data preprocessing	USN-4	Evaluate different deep learning architectures (e.g., CNNs) to select the most suitable model for predicting	Discovering various deep learning algorithms and focusing on the best one
Different Company's	Model development	USN-5	Train the selected deep learning model using the preprocessed dataset and monitor its performance on the validation set.	To increase the models performance
Specific Hospitals (Beta testing)	Training	USN-6	Train the selected deep learning model using the preprocessed dataset and monitor its performance under doctors and other trained professionals	To train the model to the max
Specific Hospitals (Beta testing)	Prediction	USN-7	When the particular image scan is inserted into the model it will classify it with the pre-trained model and predict the stage of cancer	To get the best prediction
General Public	Model deployment	USN-8	Deploy the trained deep learning model as an web service which can be accesd by genral public to check whether they have Melanoma	Acceptance of the public

5.2 Solution Architecture



6 PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture



6.2 Sprint Planning & Estimation

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	3	1 Days	30 Oct 2023	30 Oct 2023	20	3 Nov 2023
Sprint-2	5	4 Days	31 Oct 2023	3 Nov 2023		
Sprint-3	10	5 Days	4 Nov 2023	8 Nov 2023		
Sprint-4	1	10 Days	9 Nov 2023	18 Nov 2023		
Sprint-5	1	4 Days	19 Nov 2023	22 Nov 2023		

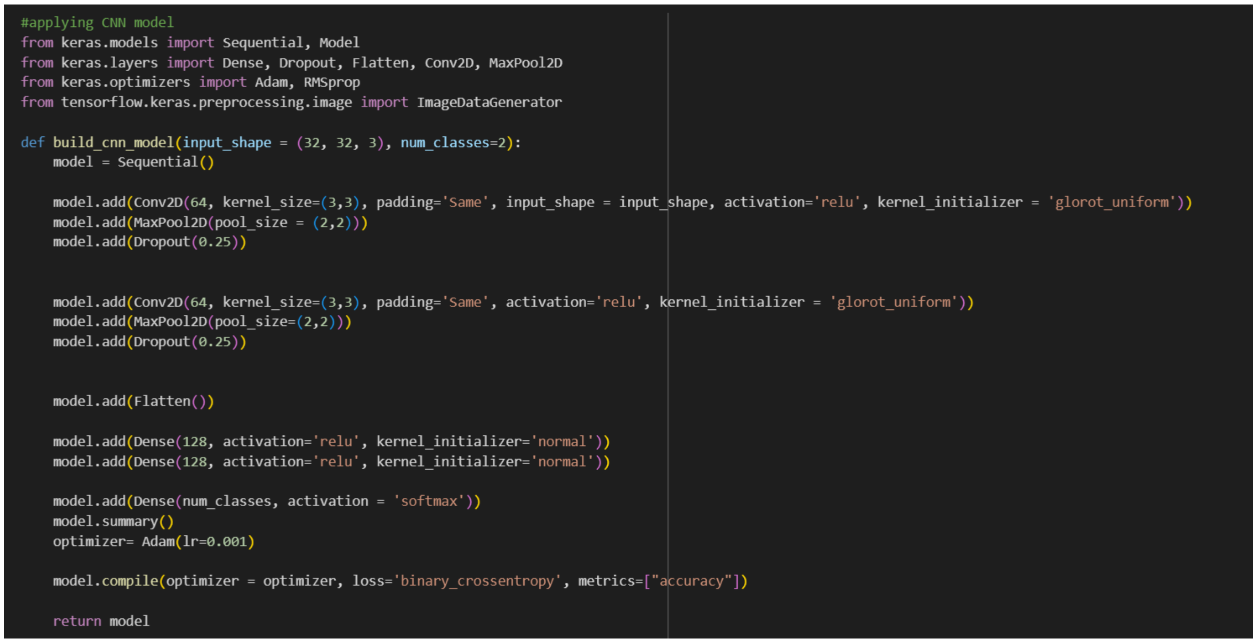
6.3 Sprint Delivery Schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Project setup & Infrastructure	USN-1	Set up the development environment with the required tools and frameworks to start Project For Detecting Melanoma Diseases	1	High	Alex
Sprint-2	Data collection	USN-2	Gather a diverse dataset of dermatological images, Ensure dataset contains labelled images for benign and malignant melanoma, Ensure data is representative of different skin types, ages, and genders.	2	High	Prabhat
Sprint-3	Data pre-processing	USN-3	Prepare the dataset for deep learning model training. Split the dataset into training, validation, and testing sets.	3	High	Alex
Sprint-4	Model development	USN-4	Create a deep learning model for melanoma detection. Choose a suitable deep learning architecture like CNN, Inception, ResNet.	4	High	Saifulla
Sprint-5	Training	USN-5	Train the model on the pre-processed dataset. Monitor training progress and adjust hyperparameters as needed.	6	Medium	Saifulla
Sprint-6	Model evaluation	USN-6	Assess the model's performance. Evaluate the model on the validation set using appropriate metrics (e.g., accuracy, F1 score, ROC AUC).	1	Medium	Prabhat
Sprint-7	Model deployment	USN-7	Develop a user-friendly interface for uploading images for prediction. Ensure security and privacy measures for sensitive medical data. Monitor and maintain the deployed model.	1	Medium	Alex

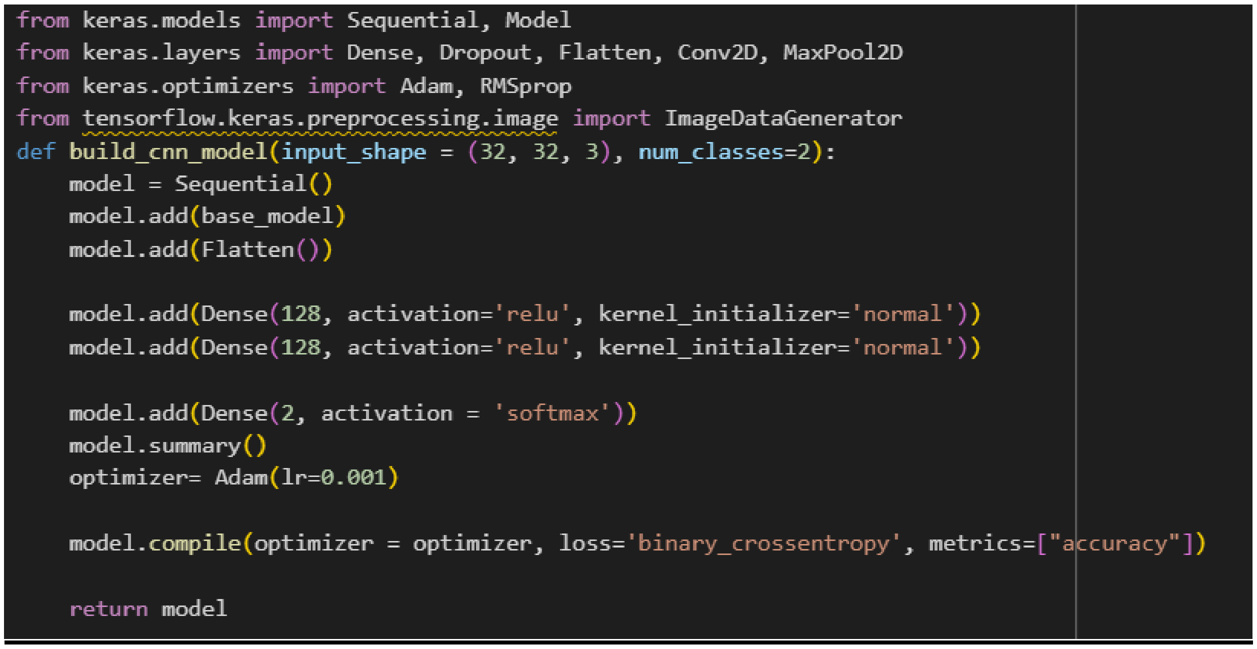
7. CODING & SOLUTIONING (Explain the features added in the project along with code)

Dataset: https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000

The CNN model of our project

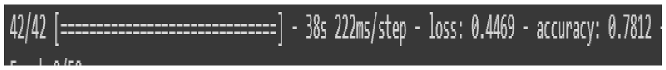


```python
#applying CNN model
from keras.models import Sequential, Model
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras.optimizers import Adam, RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator

def build_cnn_model(input_shape = (32, 32, 3), num_classes=2):
    model = Sequential()

    model.add(Conv2D(64, kernel_size=(3,3), padding='Same', input_shape = input_shape, activation='relu', kernel_initializer = 'glorot_uniform'))
    model.add(MaxPool2D(pool_size = (2,2)))
    model.add(Dropout(0.25))


    model.add(Conv2D(64, kernel_size=(3,3), padding='Same', activation='relu', kernel_initializer = 'glorot_uniform'))
    model.add(MaxPool2D(pool_size=(2,2)))
    model.add(Dropout(0.25))


    model.add(Flatten())

    model.add(Dense(128, activation='relu', kernel_initializer='normal'))
    model.add(Dense(128, activation='relu', kernel_initializer='normal'))

    model.add(Dense(num_classes, activation = 'softmax'))
    model.summary()
    optimizer= Adam(lr=0.001)

    model.compile(optimizer = optimizer, loss='binary_crossentropy', metrics=["accuracy"])

    return model
```

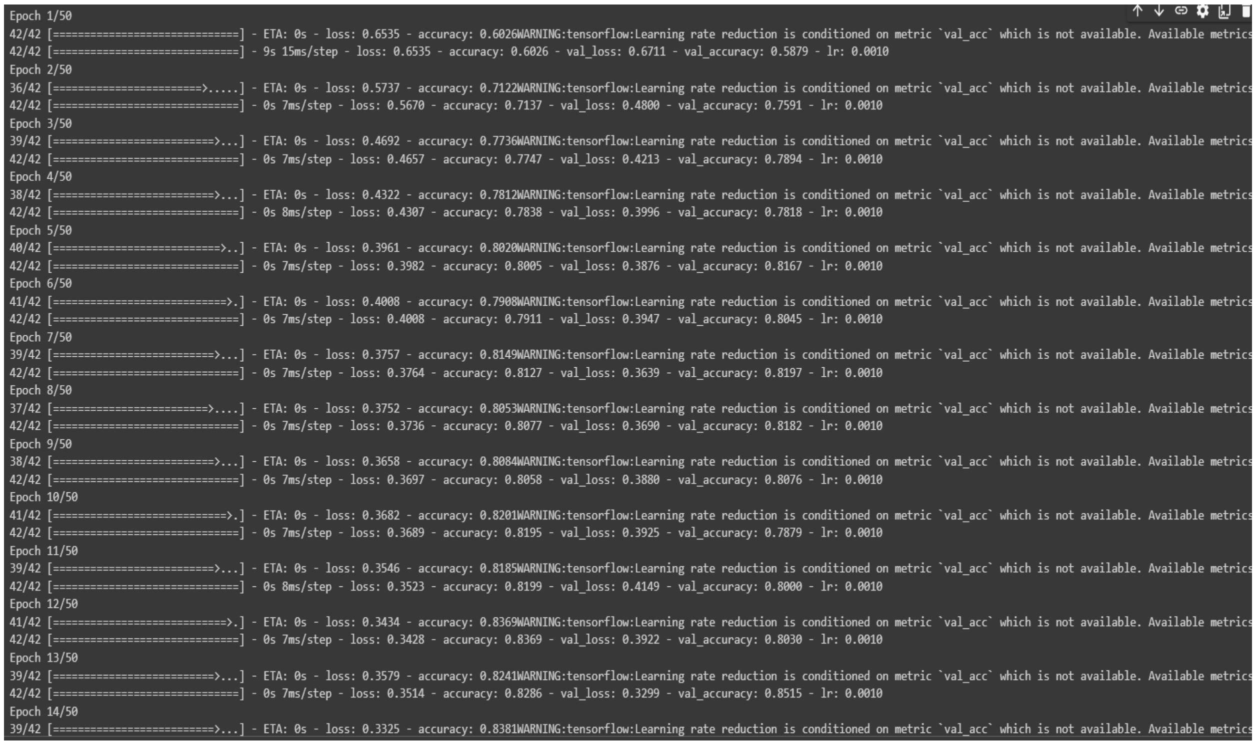


```python
from keras.models import Sequential, Model
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras.optimizers import Adam, RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
def build_cnn_model(input_shape = (32, 32, 3), num_classes=2):
    model = Sequential()
    model.add(base_model)
    model.add(Flatten())

    model.add(Dense(128, activation='relu', kernel_initializer='normal'))
    model.add(Dense(128, activation='relu', kernel_initializer='normal'))

    model.add(Dense(2, activation = 'softmax'))
    model.summary()
    optimizer= Adam(lr=0.001)

    model.compile(optimizer = optimizer, loss='binary_crossentropy', metrics=["accuracy"])

    return model
```

8. PERFORMANCE TESTING



```
42/42 [==============================] - 38s 222ms/step - loss: 0.4469 - accuracy: 0.7812
```



```
Epoch 1/50
42/42 [==============================] - ETA: 0s - loss: 0.6535 - accuracy: 0.6026WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 9s 15ms/step - loss: 0.6535 - accuracy: 0.6026 - val_loss: 0.6711 - val_accuracy: 0.5879 - lr: 0.0010
Epoch 2/50
36/42 [========================>.....] - ETA: 0s - loss: 0.5737 - accuracy: 0.7122WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.5670 - accuracy: 0.7137 - val_loss: 0.4800 - val_accuracy: 0.7591 - lr: 0.0010
Epoch 3/50
39/42 [==========================>...] - ETA: 0s - loss: 0.4692 - accuracy: 0.7736WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.4657 - accuracy: 0.7747 - val_loss: 0.4213 - val_accuracy: 0.7894 - lr: 0.0010
Epoch 4/50
38/42 [==========================>...] - ETA: 0s - loss: 0.4322 - accuracy: 0.7812WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 8ms/step - loss: 0.4307 - accuracy: 0.7838 - val_loss: 0.3996 - val_accuracy: 0.7818 - lr: 0.0010
Epoch 5/50
40/42 [===========================>..] - ETA: 0s - loss: 0.3961 - accuracy: 0.8020WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.3982 - accuracy: 0.8005 - val_loss: 0.3876 - val_accuracy: 0.8167 - lr: 0.0010
Epoch 6/50
41/42 [============================>.] - ETA: 0s - loss: 0.4008 - accuracy: 0.7908WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.4008 - accuracy: 0.7911 - val_loss: 0.3947 - val_accuracy: 0.8045 - lr: 0.0010
Epoch 7/50
39/42 [==========================>...] - ETA: 0s - loss: 0.3757 - accuracy: 0.8149WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.3764 - accuracy: 0.8127 - val_loss: 0.3639 - val_accuracy: 0.8197 - lr: 0.0010
Epoch 8/50
37/42 [=========================>....] - ETA: 0s - loss: 0.3752 - accuracy: 0.8053WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.3736 - accuracy: 0.8077 - val_loss: 0.3690 - val_accuracy: 0.8182 - lr: 0.0010
Epoch 9/50
38/42 [==========================>...] - ETA: 0s - loss: 0.3658 - accuracy: 0.8084WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.3697 - accuracy: 0.8058 - val_loss: 0.3880 - val_accuracy: 0.8076 - lr: 0.0010
Epoch 10/50
41/42 [============================>.] - ETA: 0s - loss: 0.3682 - accuracy: 0.8201WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.3689 - accuracy: 0.8195 - val_loss: 0.3925 - val_accuracy: 0.7879 - lr: 0.0010
Epoch 11/50
39/42 [==========================>...] - ETA: 0s - loss: 0.3546 - accuracy: 0.8185WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 8ms/step - loss: 0.3523 - accuracy: 0.8199 - val_loss: 0.4149 - val_accuracy: 0.8000 - lr: 0.0010
Epoch 12/50
41/42 [============================>.] - ETA: 0s - loss: 0.3434 - accuracy: 0.8369WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.3428 - accuracy: 0.8369 - val_loss: 0.3922 - val_accuracy: 0.8030 - lr: 0.0010
Epoch 13/50
39/42 [==========================>...] - ETA: 0s - loss: 0.3579 - accuracy: 0.8241WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.3514 - accuracy: 0.8286 - val_loss: 0.3299 - val_accuracy: 0.8515 - lr: 0.0010
Epoch 14/50
39/42 [==========================>...] - ETA: 0s - loss: 0.3325 - accuracy: 0.8381WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
```

10. ADVANTAGES & DISADVANTAGES

Advantages:

- **Early Detection Potential:** Deep learning models excel at identifying subtle patterns within complex data, enabling the detection of early-stage biomarkers or indicators of melanoma disease before noticeable symptoms appear.
- **Transfer Learning Capabilities:** Pre-trained ResNet50 models on large datasets (e.g., ImageNet) can be fine-tuned with melanoma's-specific data. This transfer learning helps leverage the knowledge gained from broader datasets, enhancing performance in disease prediction tasks.
- **Accurate Predictive Capabilities:** Leveraging convolutional neural networks (CNNs) and other deep learning architectures, these models can process diverse datasets, including neuroimaging scans and genetic information, leading to more precise and accurate predictions compared to traditional methods.
- **Robustness to Overfitting:** The skip connections in ResNet50 aid in mitigating overfitting issues, allowing the model to generalize well even with a limited dataset, which is crucial in medical applications where extensive datasets might be unavailable.
- **Potential for Automation:** These models hold the potential to automate certain aspects of diagnosis and prediction, easing the burden on healthcare professionals and improving efficiency in healthcare delivery.

Disadvantages:

- **Complexity and Interpretability:** Deep learning models, especially deep neural networks, are often viewed as "black boxes," making it challenging to interpret the reasoning behind their predictions. This lack of interpretability can hinder trust and understanding, crucial in medical decision-making.
- **Overfitting and Generalization**: deep learning models can overfit to the training data, resulting in poor generalization to new, unseen data. This is a concern in healthcare where models need to perform reliably on diverse patient populations.
- **Deployment Challenges:** Translating ResNet50 models into practical clinical use might be challenging due to their computational demands and the need for specialized expertise in deploying and maintaining such systems within healthcare environments.

11. CONCLUSION

The results indicate that the ensemble model outperformed the individual models across various metrics. Specifically, the ensemble model achieved higher precision, recall, f1-score, and accuracy compared to ResNet, InceptionV3, AlexNet, GoggleNet, VGG-16, which had accuracy scores of 82%, 72%, 79%, 81%, and 75%, respectively.various techniques and methods for identifying the melanoma skin cancer. The findings highlight the importance of automated approaches, specifically deep learning and feature extraction, in enhancing both the efficiency and accuracy of the melanoma skin cancer identification process.Expanding the dataset size to include more diverse and representative samples of skin lesions would reduce the risk of overfitting, enhancing the model's generalization capabilities.

12. FUTURE SCOPE

Deep learning is the field of constable's development. Accuracy can be improved because of the advancements in deep learning and machine learning. Very wide multi modal approaches provide a more comprehensive view of the disease progression. We can also evolve in the monitoring of the person vitals and also helps in the personalized medicine checking their can also evolve in the monitoring of the person vitals and also helps in the personalized medicine checking their stages. Large scale collaborative datasets can be among institutions for further detailed research

i) Using a larger dataset: Expanding the dataset size to include more diverse and representative samples of skin lesions would reduce the risk of overfitting, enhancing the model's generalization capabilities.

ii) Performing additional regularization and hyperparameter tuning: Further refining the model through regularization techniques and fine-tuning hyperparameters can improve its performance and robustness.

iii) Training with Dermnet dataset: Instead of using a general dataset, training the architecture with Dermnet, a specialized skin-related dataset, can potentially lead to better performance and more accurate predictions for skin lesion classification.

13. APPENDIX

GitHub Link:
https://github.com/smartinternz02/SI-GuidedProject-613379-1698914602

Source Code Mount Goggle Drive:

```
#extracting dataset from zip file by unzipping
import zipfile
a = zipfile.ZipFile('/content/skin-cancer-malignant-vs-benign.zip','r')
a.extractall('/content')
a.close()
```

Import the Required libraries:

```
#importing libraries
#numpy for arrays
import numpy as np
#pandas for data analaysis
import pandas as pd
#os for directories
import os
#matplot for displaying images and plotting images
import matplotlib.pyplot as plt
import seaborn as sns
#glob is used for files and determing path
from glob import glob
#random seed to generate random values. Sending images to training
np.random.seed(21)
```

load the training and testing files

```
from PIL import Image
#load the training and testing files
directory_benign_train = '/content/train/benign'
directory_malignant_train = '/content/train/malignant'
directory_benign_test = '/content/test/benign'
directory_malignant_test = '/content/test/malignant'
read = lambda imname:
np.asarray(Image.open(imname).convert('RGB').resize((32,32)))

#reading files
img_benign_train = [read(os.path.join(directory_benign_train, filename)) for
filename in os.listdir(directory_benign_train)]
img_malignant_train = [read(os.path.join(directory_malignant_train, filename)) for
filename in os.listdir(directory_malignant_train)]


img_benign_test = [read(os.path.join(directory_benign_test, filename)) for
filename in os.listdir(directory_benign_test)]
img_malignant_test = [read(os.path.join(directory_malignant_test, filename)) for
filename in os.listdir(directory_malignant_test)]

type(img_benign_train)
```

converting images to arrays of unsigned 8 bit integer

```
#converting images to arrays of unsigned 8 bit integer
X_benign_train = np.array(img_benign_train, dtype='uint8')
X_malignant_train = np.array(img_malignant_train, dtype='uint8')

X_benign_test = np.array(img_benign_test, dtype='uint8')
X_malignant_test = np.array(img_malignant_test, dtype='uint8')

type(X_benign_train)
```

```
#labeling dieases as 1 or 0
y_benign_train = np.zeros(X_benign_train.shape[0])
y_malignant_train = np.ones(X_malignant_train.shape[0])

y_benign_test = np.zeros(X_benign_test.shape[0])
y_malignant_test = np.ones(X_malignant_test.shape[0])

y_malignant_train
```

```
#combiining all images
X_train = np.concatenate((X_benign_train, X_malignant_train), axis=0)
```

```
y_train = np.concatenate((y_benign_train, y_malignant_train), axis=0)

X_test = np.concatenate((X_benign_test, X_malignant_test), axis=0)
y_test = np.concatenate((y_benign_test, y_malignant_test), axis=0)

print("Shape of X_train: ", X_train.shape)
print("Shape of y_train: ", y_train.shape)
print("Shape of X_test: ", X_test.shape)
print("Shape of y_test: ", y_test.shape)

y_test
```

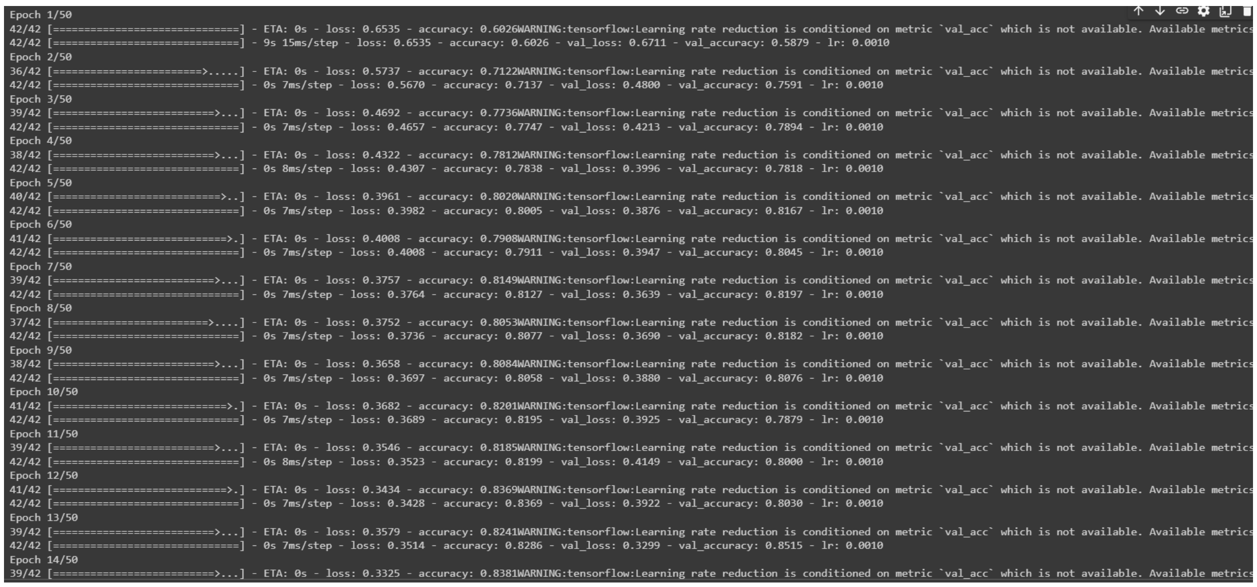
CNN model

```
#applying CNN model
from keras.models import Sequential, Model
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras.optimizers import Adam, RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator

def build_cnn_model(input_shape = (32, 32, 3), num_classes=2):
    model = Sequential()

    model.add(Conv2D(64, kernel_size=(3,3), padding='Same', input_shape = 
input_shape, activation='relu', kernel_initializer = 'glorot_uniform'))
    model.add(MaxPool2D(pool_size = (2,2)))
    model.add(Dropout(0.25))


    model.add(Conv2D(64, kernel_size=(3,3), padding='Same', activation='relu', 
kernel_initializer = 'glorot_uniform'))
    model.add(MaxPool2D(pool_size=(2,2)))
    model.add(Dropout(0.25))


    model.add(Flatten())

    model.add(Dense(128, activation='relu', kernel_initializer='normal'))
    model.add(Dense(128, activation='relu', kernel_initializer='normal'))

    model.add(Dense(num_classes, activation = 'softmax'))
    model.summary()
    optimizer= Adam(lr=0.001)

    model.compile(optimizer = optimizer, loss='binary_crossentropy', 
metrics=["accuracy"])
```

```
    return model
```

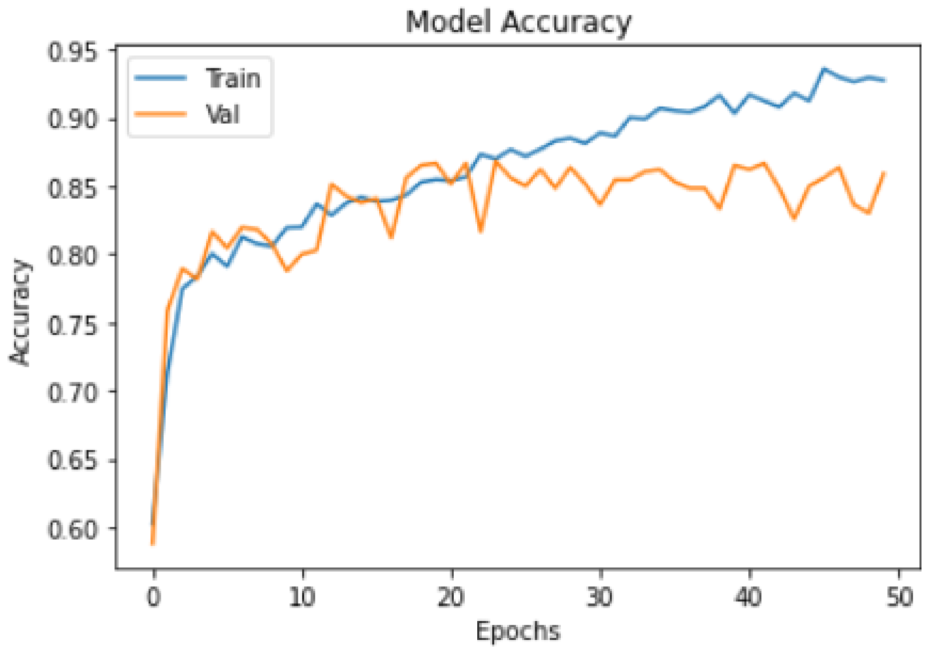
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 64)        1792

 max_pooling2d (MaxPooling2D  (None, 16, 16, 64)       0
 )

 dropout (Dropout)           (None, 16, 16, 64)        0

 conv2d_1 (Conv2D)           (None, 16, 16, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 dropout_1 (Dropout)         (None, 8, 8, 64)          0

 flatten (Flatten)           (None, 4096)              0

 dense (Dense)               (None, 128)               524416

 dense_1 (Dense)             (None, 128)               16512

 dense_2 (Dense)             (None, 2)                 258

=================================================================
Total params: 579,906
Trainable params: 579,906
Non-trainable params: 0
_________________________________________________________________
/usr/local/lib/python3.8/dist-packages/keras/optimizers/optimizer_v2/adam.py:110: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
```

```
from keras.callbacks import ReduceLROnPlateau
learning_rate_annealer = ReduceLROnPlateau(monitor='val_acc',mode='max',
                                            patience=3,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr = 1e-4)



history = model_cnn.fit(X_train,
                        y_train,
                        validation_split=0.2,
                        epochs=50,
                        batch_size = 64,
                        verbose=1,
                              callbacks=[learning_rate_annealer],validation_data=(X_test,y_t
est))



print(history.history.keys())
```

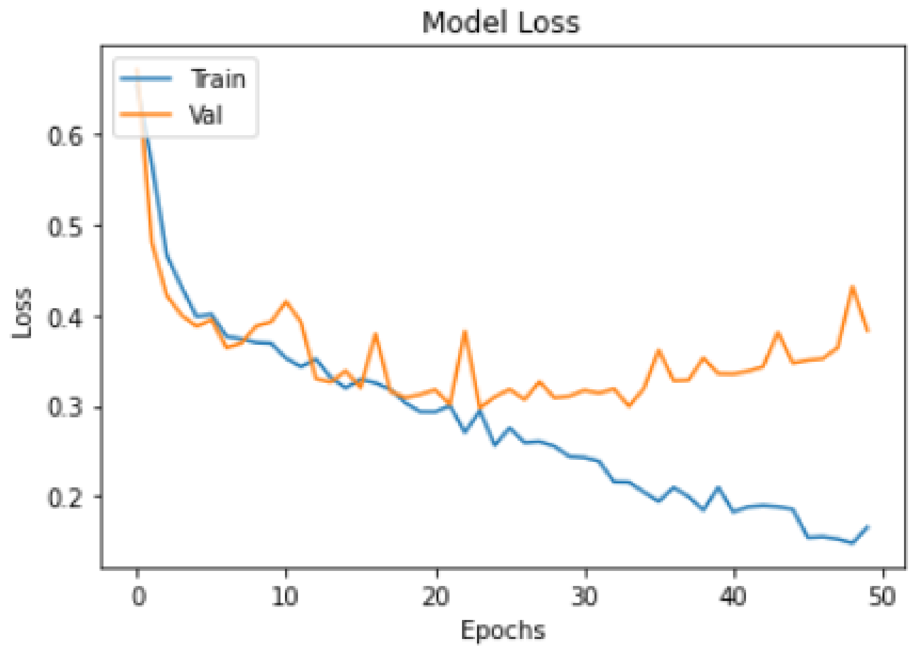


```
Epoch 1/50
42/42 [==============================] - ETA: 0s - loss: 0.6535 - accuracy: 0.6026WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 9s 15ms/step - loss: 0.6535 - accuracy: 0.6026 - val_loss: 0.6711 - val_accuracy: 0.5879 - lr: 0.0010
Epoch 2/50
36/42 [========================>.....] - ETA: 0s - loss: 0.5737 - accuracy: 0.7122WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.5670 - accuracy: 0.7137 - val_loss: 0.4800 - val_accuracy: 0.7591 - lr: 0.0010
Epoch 3/50
39/42 [==========================>...] - ETA: 0s - loss: 0.4692 - accuracy: 0.7736WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.4657 - accuracy: 0.7747 - val_loss: 0.4213 - val_accuracy: 0.7894 - lr: 0.0010
Epoch 4/50
38/42 [==========================>...] - ETA: 0s - loss: 0.4322 - accuracy: 0.7812WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 8ms/step - loss: 0.4307 - accuracy: 0.7838 - val_loss: 0.3996 - val_accuracy: 0.7818 - lr: 0.0010
Epoch 5/50
40/42 [===========================>..] - ETA: 0s - loss: 0.3961 - accuracy: 0.8020WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.3982 - accuracy: 0.8005 - val_loss: 0.3876 - val_accuracy: 0.8167 - lr: 0.0010
Epoch 6/50
41/42 [============================>.] - ETA: 0s - loss: 0.4008 - accuracy: 0.7908WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.4008 - accuracy: 0.7911 - val_loss: 0.3947 - val_accuracy: 0.8045 - lr: 0.0010
Epoch 7/50
39/42 [==========================>...] - ETA: 0s - loss: 0.3757 - accuracy: 0.8149WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.3764 - accuracy: 0.8127 - val_loss: 0.3639 - val_accuracy: 0.8197 - lr: 0.0010
Epoch 8/50
37/42 [=========================>....] - ETA: 0s - loss: 0.3752 - accuracy: 0.8053WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.3736 - accuracy: 0.8077 - val_loss: 0.3690 - val_accuracy: 0.8182 - lr: 0.0010
Epoch 9/50
38/42 [==========================>...] - ETA: 0s - loss: 0.3658 - accuracy: 0.8084WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.3697 - accuracy: 0.8058 - val_loss: 0.3880 - val_accuracy: 0.8076 - lr: 0.0010
Epoch 10/50
41/42 [============================>.] - ETA: 0s - loss: 0.3682 - accuracy: 0.8201WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.3689 - accuracy: 0.8195 - val_loss: 0.3925 - val_accuracy: 0.7879 - lr: 0.0010
Epoch 11/50
39/42 [==========================>...] - ETA: 0s - loss: 0.3546 - accuracy: 0.8185WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 8ms/step - loss: 0.3523 - accuracy: 0.8199 - val_loss: 0.4149 - val_accuracy: 0.8000 - lr: 0.0010
Epoch 12/50
41/42 [============================>.] - ETA: 0s - loss: 0.3434 - accuracy: 0.8369WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.3428 - accuracy: 0.8369 - val_loss: 0.3922 - val_accuracy: 0.8030 - lr: 0.0010
Epoch 13/50
39/42 [==========================>...] - ETA: 0s - loss: 0.3579 - accuracy: 0.8241WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
42/42 [==============================] - 0s 7ms/step - loss: 0.3514 - accuracy: 0.8286 - val_loss: 0.3299 - val_accuracy: 0.8515 - lr: 0.0010
Epoch 14/50
39/42 [==========================>...] - ETA: 0s - loss: 0.3325 - accuracy: 0.8381WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics
```

```
#plotting acuracy and loss
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])

plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')

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


plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')

plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```



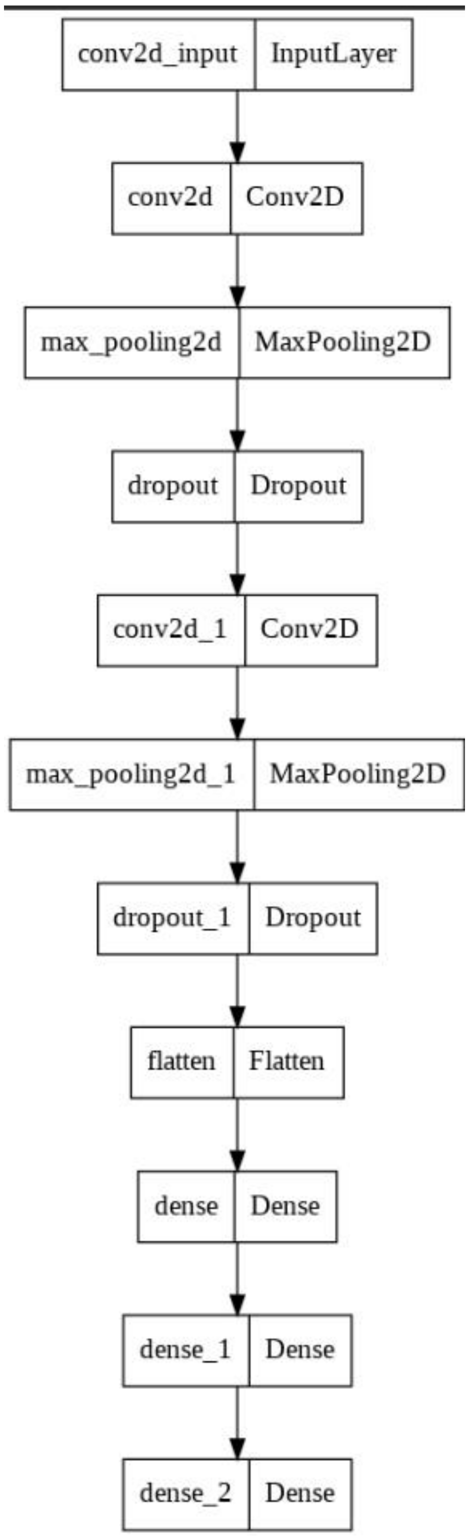
Model Accuracy
Train
Val
Accuracy
Epochs
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
10
20
30
40
50



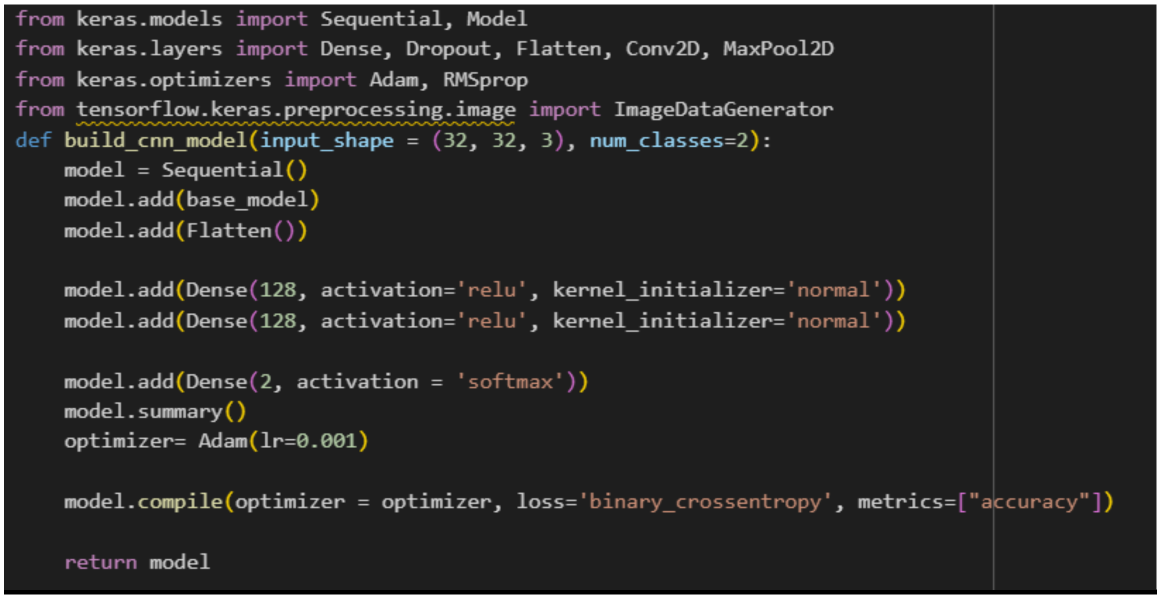
Model Loss
Train
Val
Loss
Epochs
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50

Tensorflow

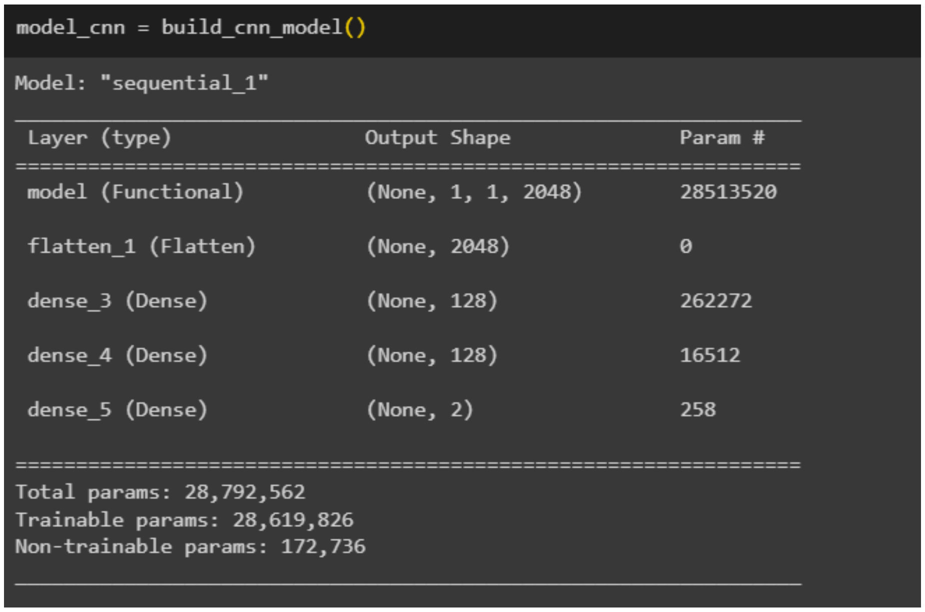
```
import tensorflow as tf
tf.keras.utils.plot_model(model_cnn)
```



second CNN Layers

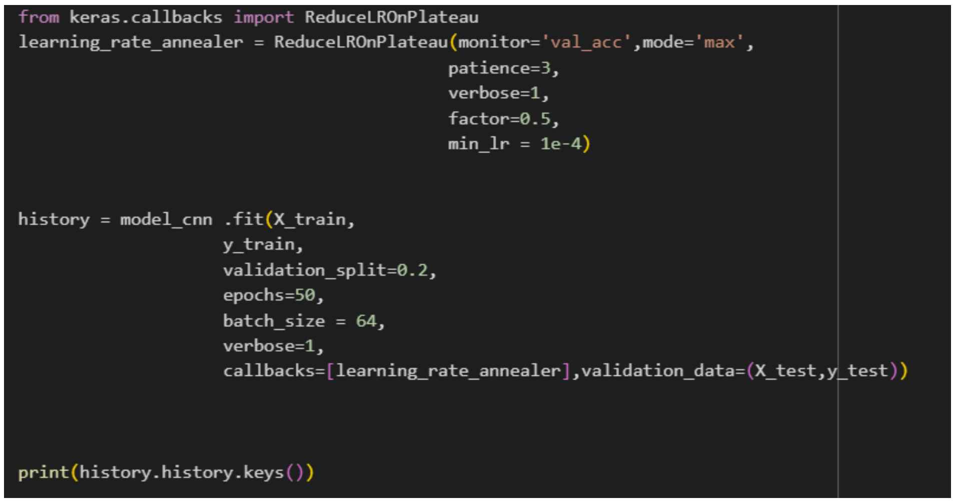


```python
from keras.models import Sequential, Model
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras.optimizers import Adam, RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
def build_cnn_model(input_shape = (32, 32, 3), num_classes=2):
    model = Sequential()
    model.add(base_model)
    model.add(Flatten())

    model.add(Dense(128, activation='relu', kernel_initializer='normal'))
    model.add(Dense(128, activation='relu', kernel_initializer='normal'))

    model.add(Dense(2, activation = 'softmax'))
    model.summary()
    optimizer= Adam(lr=0.001)

    model.compile(optimizer = optimizer, loss='binary_crossentropy', metrics=["accuracy"])

    return model
```

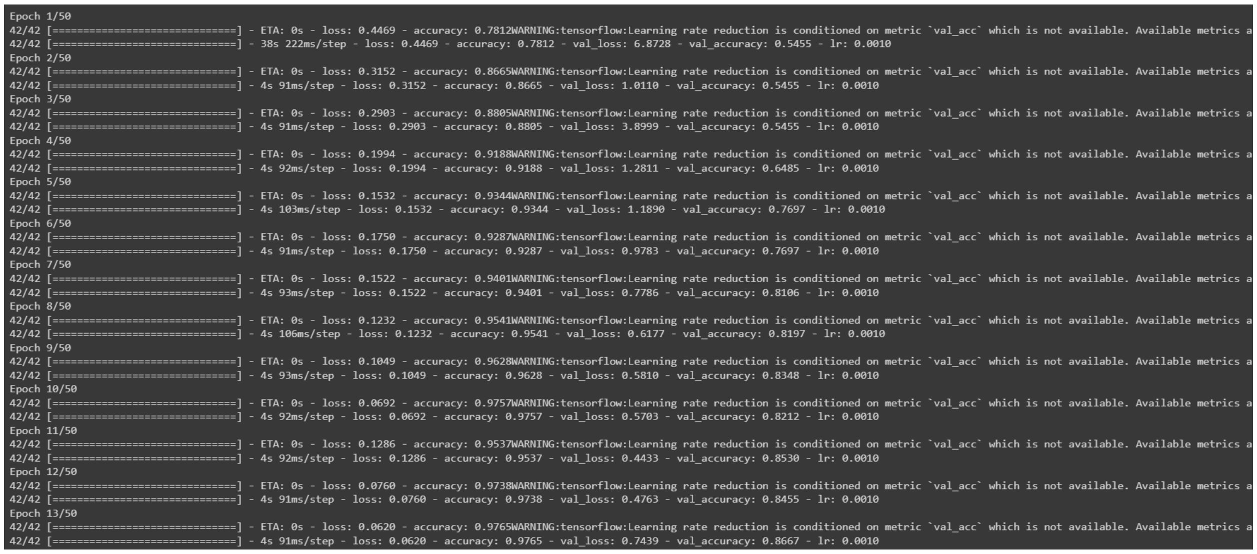


```python
model_cnn = build_cnn_model()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 model (Functional)          (None, 1, 1, 2048)        28513520

 flatten_1 (Flatten)         (None, 2048)              0

 dense_3 (Dense)             (None, 128)               262272

 dense_4 (Dense)             (None, 128)               16512

 dense_5 (Dense)             (None, 2)                 258

=================================================================
Total params: 28,792,562
Trainable params: 28,619,826
Non-trainable params: 172,736
_________________________________________________________________
```



```python
from keras.callbacks import ReduceLROnPlateau
learning_rate_annealer = ReduceLROnPlateau(monitor='val_acc',mode='max',
                                          patience=3,
                                          verbose=1,
                                          factor=0.5,
                                          min_lr = 1e-4)


history = model_cnn .fit(X_train,
                    y_train,
                    validation_split=0.2,
                    epochs=50,
                    batch_size = 64,
                    verbose=1,
                    callbacks=[learning_rate_annealer],validation_data=(X_test,y_test))



print(history.history.keys())
```



```
Epoch 1/50
42/42 [==============================] - ETA: 0s - loss: 0.4469 - accuracy: 0.7812WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics a
42/42 [==============================] - 38s 222ms/step - loss: 0.4469 - accuracy: 0.7812 - val_loss: 6.8728 - val_accuracy: 0.5455 - lr: 0.0010
Epoch 2/50
42/42 [==============================] - ETA: 0s - loss: 0.3152 - accuracy: 0.8665WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics a
42/42 [==============================] - 4s 91ms/step - loss: 0.3152 - accuracy: 0.8665 - val_loss: 1.0110 - val_accuracy: 0.5455 - lr: 0.0010
Epoch 3/50
42/42 [==============================] - ETA: 0s - loss: 0.2903 - accuracy: 0.8805WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics a
42/42 [==============================] - 4s 91ms/step - loss: 0.2903 - accuracy: 0.8805 - val_loss: 3.8999 - val_accuracy: 0.5455 - lr: 0.0010
Epoch 4/50
42/42 [==============================] - ETA: 0s - loss: 0.1994 - accuracy: 0.9188WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics a
42/42 [==============================] - 4s 92ms/step - loss: 0.1994 - accuracy: 0.9188 - val_loss: 1.2811 - val_accuracy: 0.6485 - lr: 0.0010
Epoch 5/50
42/42 [==============================] - ETA: 0s - loss: 0.1532 - accuracy: 0.9344WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics a
42/42 [==============================] - 4s 103ms/step - loss: 0.1532 - accuracy: 0.9344 - val_loss: 1.1890 - val_accuracy: 0.7697 - lr: 0.0010
Epoch 6/50
42/42 [==============================] - ETA: 0s - loss: 0.1750 - accuracy: 0.9287WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics a
42/42 [==============================] - 4s 91ms/step - loss: 0.1750 - accuracy: 0.9287 - val_loss: 0.9783 - val_accuracy: 0.7697 - lr: 0.0010
Epoch 7/50
42/42 [==============================] - ETA: 0s - loss: 0.1522 - accuracy: 0.9401WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics a
42/42 [==============================] - 4s 93ms/step - loss: 0.1522 - accuracy: 0.9401 - val_loss: 0.7786 - val_accuracy: 0.8106 - lr: 0.0010
Epoch 8/50
42/42 [==============================] - ETA: 0s - loss: 0.1232 - accuracy: 0.9541WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics a
42/42 [==============================] - 4s 106ms/step - loss: 0.1232 - accuracy: 0.9541 - val_loss: 0.6177 - val_accuracy: 0.8197 - lr: 0.0010
Epoch 9/50
42/42 [==============================] - ETA: 0s - loss: 0.1049 - accuracy: 0.9628WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics a
42/42 [==============================] - 4s 93ms/step - loss: 0.1049 - accuracy: 0.9628 - val_loss: 0.5810 - val_accuracy: 0.8348 - lr: 0.0010
Epoch 10/50
42/42 [==============================] - ETA: 0s - loss: 0.0692 - accuracy: 0.9757WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics a
42/42 [==============================] - 4s 92ms/step - loss: 0.0692 - accuracy: 0.9757 - val_loss: 0.5703 - val_accuracy: 0.8212 - lr: 0.0010
Epoch 11/50
42/42 [==============================] - ETA: 0s - loss: 0.1286 - accuracy: 0.9537WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics a
42/42 [==============================] - 4s 92ms/step - loss: 0.1286 - accuracy: 0.9537 - val_loss: 0.4433 - val_accuracy: 0.8530 - lr: 0.0010
Epoch 12/50
42/42 [==============================] - ETA: 0s - loss: 0.0760 - accuracy: 0.9738WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics a
42/42 [==============================] - 4s 91ms/step - loss: 0.0760 - accuracy: 0.9738 - val_loss: 0.4763 - val_accuracy: 0.8455 - lr: 0.0010
Epoch 13/50
42/42 [==============================] - ETA: 0s - loss: 0.0620 - accuracy: 0.9765WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics a
42/42 [==============================] - 4s 91ms/step - loss: 0.0620 - accuracy: 0.9765 - val_loss: 0.7439 - val_accuracy: 0.8667 - lr: 0.0010
```

For the final output

```python
import numpy as np
import matplotlib.pyplot as plt


# creating the dataset
data = { 'CNN ':98 , 'EfficientNET':99}
courses = list(data.keys())
values = list(data.values())

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(courses, values, color ='orange',
        width = 0.4)

plt.xlabel("Architectures used")
plt.ylabel("Accuracy")
plt.title("Accuracies of different Methods  in train data")
plt.show()
```

```
import numpy as np
import matplotlib.pyplot as plt


# creating the dataset
data = { 'CNN ':75,  'EfficientNET ':79}
courses = list(data.keys())
values = list(data.values())

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(courses, values, color ='orange',
        width = 0.4)

plt.xlabel("Architectures used")
plt.ylabel("Accuracy")
plt.title("Accuracies of different Methods  in test data")
plt.show()
```

```
#highlighing the image details by converting RGB TO BGR
img_preprocessed = preprocess_input(img_batch)
print(img_preprocessed)
```

```
a=history.model.predict(img_preprocessed)
```

```
print(a)
```

```
def classify(a):
  if (a[0][0]==1 and a[0][1]==0):
    print("benign")
  else:
    print("malignant")
```

```
print(classify(a))
```