DETECTION OF SKIN CANCER USING DEEP LEARNING MODEL MOBILENET.

TEAM 3

AQDUS CHAROLIA

AFFAN CHAROLIA

VYOMA DESAI

MUHAMMAD SHAIK

Sunday, May 8th, 2022

AGENDA

PROBLEM STATEMENT

OUR SOLUTION

ABOUT DATASET

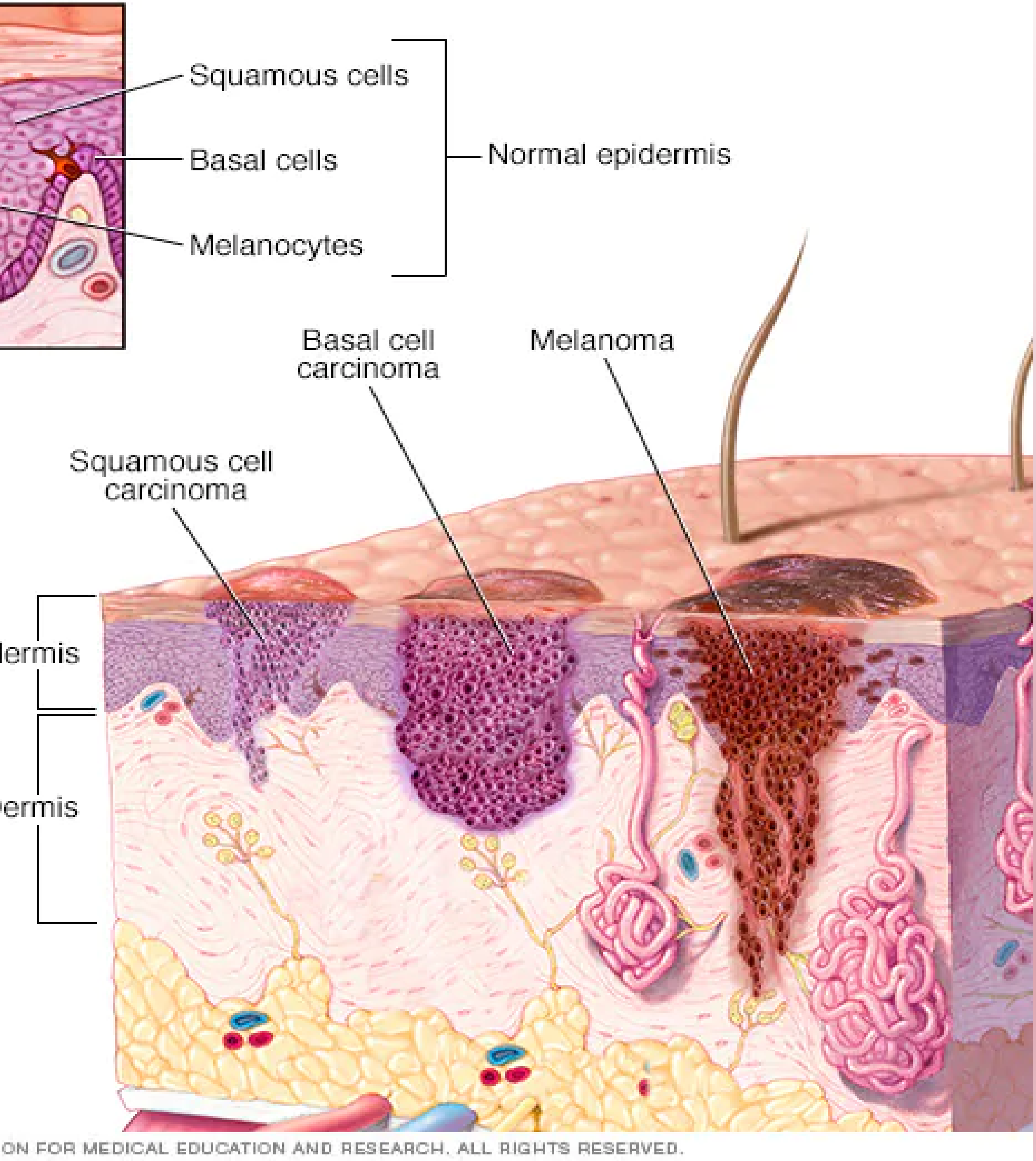
METHODOLOGY

WEB APPLICATION

RESULTS

CONCLUSION

FUTURE WORK



PROBLEM STATEMENT

DETECTION OF SKIN CANCER

- Skin cancer is one of the most common cancers in the United states with nearly **5.4 million** cases.
- At a glance, by old age, 1 in 5 Americans will develop skin cancer. Every hour a couple of people die because of skin cancer.
- When detected in the late stages its survival rate is **less than 14%.** If detected in the early stages the chances for survival are about **97%,** but for this detection to occur the diagnosis has to be done really fast and luck should be on one’s side.
- People are reluctant to get themselves checked when they notice any unusual lesion developing on their skin.

OUR SOLUTION

Let's solve it together

- Presented a Web based application where Skin-Cancer images can be uploaded to get the closest prediction between 7 different classes of skin caner using Deep Learning to perform the prediction.
- Resnet50, MobileNet, VGG16, CNN models have been trained on the HAM10000 Dataset after which the model with best validation accuracy has been employed in the webapp.
- Classes of Skin Cancer Images are Melanoma, Benign Keratosis, Basal Cell Carcinoma, Actinic Keratosis, Vascular Skin Lesions, Dermatofibroma and Neoplasms
- The application is designed to be used by medical professionals for a quick result.

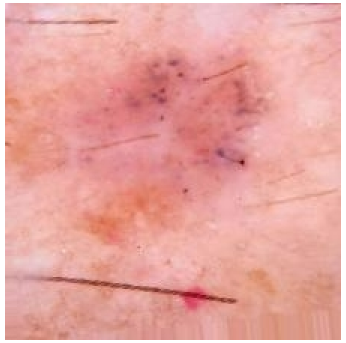
ABOUT DATASET

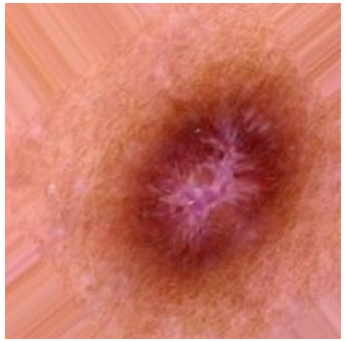
Skin Cancer HAM10000 Dataset

- Dataset consists of 10015 dermatoscopic images which can serve as a training set for academic machine learning/ deep learning purposes.
- Was part of the 2018 ISIC Challenge where Lesion Segmentation, Lesion attribute Selection and Disease Classification were the tasks to be performed.
- The dataset contains 6705 nv, 1113 mel, 1099 bkl, 327 akiec, 142 vasc, and 24 df classes of Skin Cancer images.
- The dermatoscopic images are collected from different populations, acquired and stored by different modalities by the authors.

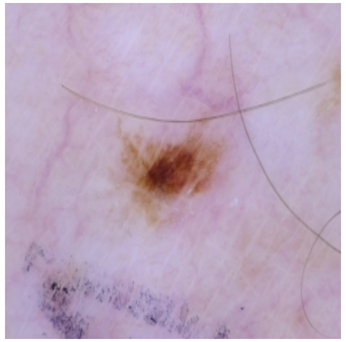
SKIN CANCER IMAGE CLASSES


Actinic Keratosis

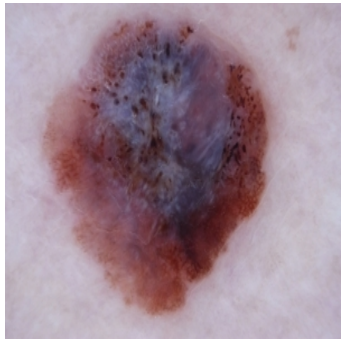

Basal Cell Carcinoma


Dermatofibroma


Benign Keratosis


Melanocytic Nevi


Vascular Skin Lesions


Melanoma

DATA PRE-PROCESSING

- Dataset split first into train and test images, each with the further classification based on the skin cancer type (7 classes) using the ham10000.csv file provided which details which image label belongs to which class.
- Data Augmentation performed on the training dataset to increase the overall size of the image dataset which will train the deep learning models better and yield better results in terms of accuracy and classification.
- For performing this task used Tensorflow's ImageDataGenerator library.
- Performed a rotation_range=180, width_shift_range=0.1, height_shift_range=0.1, zoom_range=0.1, horizontal_flip=True, vertical_flip=True.
- The batch size of the training and validation data was set to 10 for training the model.
- Normalized the images by rescaling them to 1./255 and the size fixed to (224,224,3).
- Pre-processed 38568 training images and 938 validation images.

TRAINING PHASE

NEXT STEPS

```
=================================================================
input_7 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 222, 222, 32)      896
_________________________________________________________________
max_pooling2d_28 (MaxPooling (None, 111, 111, 32)      0
_________________________________________________________________
dropout_34 (Dropout)         (None, 111, 111, 32)      0
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 109, 109, 64)      18496
_________________________________________________________________
max_pooling2d_29 (MaxPooling (None, 54, 54, 64)        0
_________________________________________________________________
dropout_35 (Dropout)         (None, 54, 54, 64)        0
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 52, 52, 128)       73856
_________________________________________________________________
max_pooling2d_30 (MaxPooling (None, 26, 26, 128)       0
_________________________________________________________________
dropout_36 (Dropout)         (None, 26, 26, 128)       0
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 24, 24, 256)       295168
_________________________________________________________________
max_pooling2d_31 (MaxPooling (None, 12, 12, 256)       0
_________________________________________________________________
dropout_37 (Dropout)         (None, 12, 12, 256)       0
_________________________________________________________________
flatten_6 (Flatten)          (None, 36864)             0
_________________________________________________________________
dense_12 (Dense)             (None, 7)                 258055
=================================================================
```

CNN

MODEL SUMMARY

- A custom Convolutional Neural Network
- Convolution layers starting from 16 layers to 256 layers followed by maxpooling and dropout layers
- Initiate with batch normalization layer to avoid overfitting
- CNN consists of 5 conv2d layers, 1 fully connected layer, 5 dropout and maxpool layers and 1 flatten layer.
- A total of 646,471 params were trained

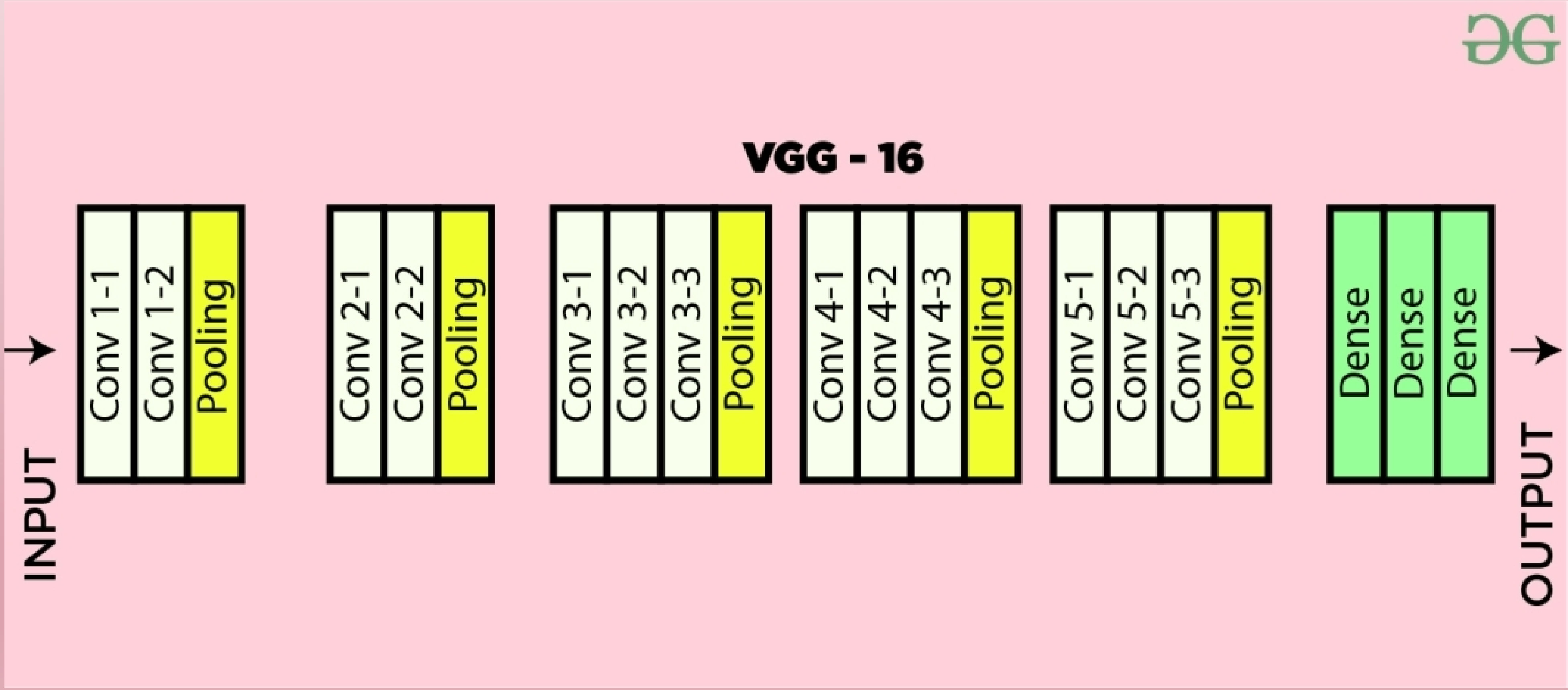
```
block3_pool (MaxPooling2D)    (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)         (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)         (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)         (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)    (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)         (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)         (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)         (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)    (None, 7, 7, 512)         0
_________________________________________________________________
global_average_pooling2d (Gl (None, 512)                0
_________________________________________________________________
dense (Dense)                 (None, 128)               65664
_________________________________________________________________
dense_1 (Dense)               (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)               (None, 32)                2080
_________________________________________________________________
dense_3 (Dense)               (None, 7)                 231
=================================================================
Total params: 14,790,919
Trainable params: 76,231
Non-trainable params: 14,714,688
```

VGG16

MODEL SUMMARY

- The model achieves 92.7% top-5 test accuracy in ImageNet
- ImageNet dataset of over 14 million images belonging to 1000 classes
- It has multiple 3×3 kernel-sized filters one after another
- Using transfer learning, freezing the previous layers of the model (keep the same trained weights) and performing global_avg_pool followed by adding custom fully connected layers to train our data.

VGG16 ARCHITECTURE



Source (Internet)

ock2_out (Activation)	(None, 7, 7, 2048)	0	conv5_block2_add[0][0]
ock3_1_conv (Conv2D)	(None, 7, 7, 512)	1049088	conv5_block2_out[0][0]
ock3_1_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block3_1_conv[0][0]
ock3_1_relu (Activation	(None, 7, 7, 512)	0	conv5_block3_1_bn[0][0]
ock3_2_conv (Conv2D)	(None, 7, 7, 512)	2359808	conv5_block3_1_relu[0][0]
ock3_2_bn (BatchNormali	(None, 7, 7, 512)	2048	conv5_block3_2_conv[0][0]
ock3_2_relu (Activation	(None, 7, 7, 512)	0	conv5_block3_2_bn[0][0]
ock3_3_conv (Conv2D)	(None, 7, 7, 2048)	1050624	conv5_block3_2_relu[0][0]
ock3_3_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block3_3_conv[0][0]
ock3_add (Add)	(None, 7, 7, 2048)	0	conv5_block2_out[0][0] conv5_block3_3_bn[0][0]
ock3_out (Activation)	(None, 7, 7, 2048)	0	conv5_block3_add[0][0]
verage_pooling2d_3 (Glo	(None, 2048)	0	conv5_block3_out[0][0]
(Dense)	(None, 128)	262272	global_average_pooling2d_
(Dense)	(None, 64)	8256	dense_12[0][0]
(Dense)	(None, 32)	2080	dense_13[0][0]
(Dense)	(None, 7)	231	dense_14[0][0]

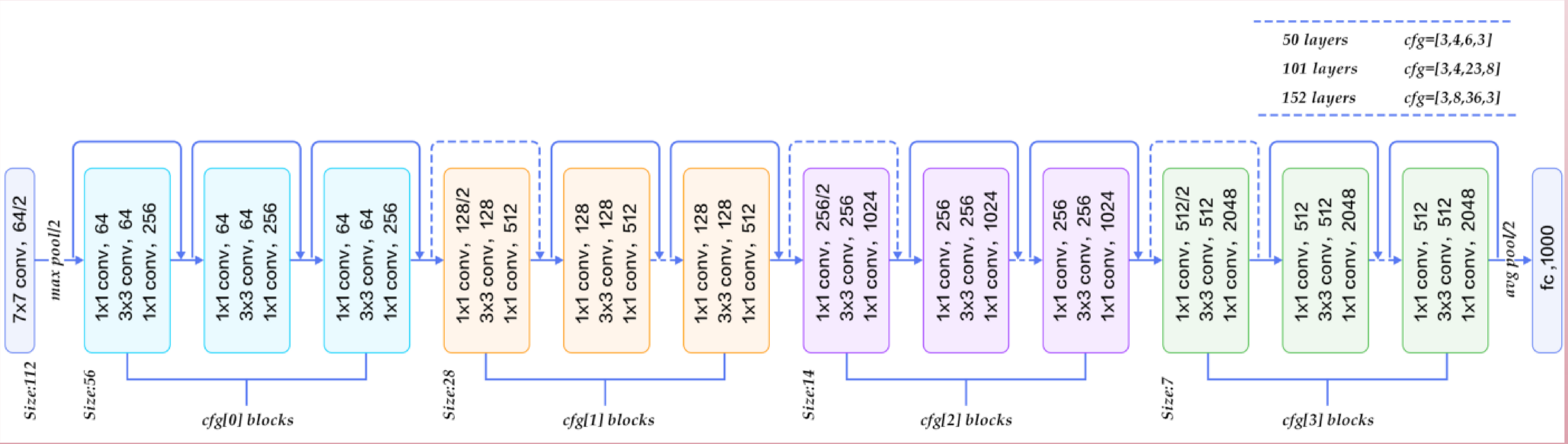
rams: 23,860,551
e params: 272,839
nable params: 23,587,712

RESNET50

MODEL SUMMARY

- Deep residual network - ResNet-50
- Convolutional neural network (CNN) which is 50 layers deep
- Powerful backbone model that is used very frequently in many computer vision tasks
- Uses a skip connection to add the output from an earlier layer to a later layer
- Helps mitigate the vanishing gradient problem
- We have performed transfer learning, by freezing the previous layers to learn as false, followed by global_avg_maxpool and then added our custom dense layers to train our data.

RESNET50 ARCHITECTURE



Source (Internet)

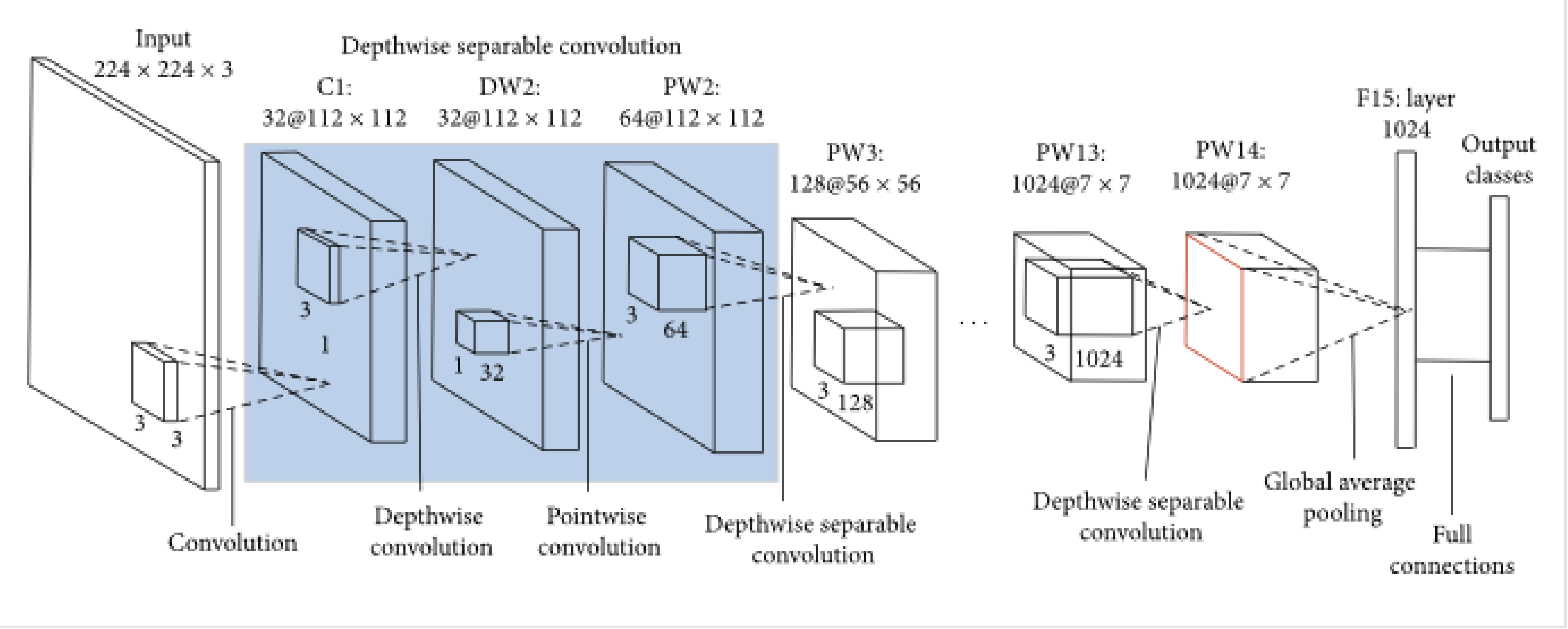
```
conv_pw_12_bn (BatchNormaliz (None, 7, 7, 1024)        4096
_________________________________________________________________
conv_pw_12_relu (ReLU)       (None, 7, 7, 1024)        0
_________________________________________________________________
conv_dw_13 (DepthwiseConv2D) (None, 7, 7, 1024)        9216
_________________________________________________________________
conv_dw_13_bn (BatchNormaliz (None, 7, 7, 1024)        4096
_________________________________________________________________
conv_dw_13_relu (ReLU)       (None, 7, 7, 1024)        0
_________________________________________________________________
conv_pw_13 (Conv2D)          (None, 7, 7, 1024)        1048576
_________________________________________________________________
conv_pw_13_bn (BatchNormaliz (None, 7, 7, 1024)        4096
_________________________________________________________________
conv_pw_13_relu (ReLU)       (None, 7, 7, 1024)        0
_________________________________________________________________
global_average_pooling2d (Gl (None, 1024)              0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              1049600
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 7175
=================================================================
Total params: 4,285,639
Trainable params: 2,914,823
Non-trainable params: 1,370,816
_________________________________________________________________
```

MOBILENET

MODEL SUMMARY

- MobileNet model is designed to be used in mobile applications, and it is TensorFlow's first mobile computer vision model.
- MobileNet uses depthwise seperable convolutions.
- This significantly reduces the number of parameters when compared to the network with regular convolutions with the same depth in the nets. This results in lightweight deep neural networks.
- A depthwise separable convolution is made from two operations.
  1. Depthwise convolution.
  2. Pointwise convolution.

MOBILENET ARCHITECTURE



Source (Internet)

RESULTS FROM MODELS

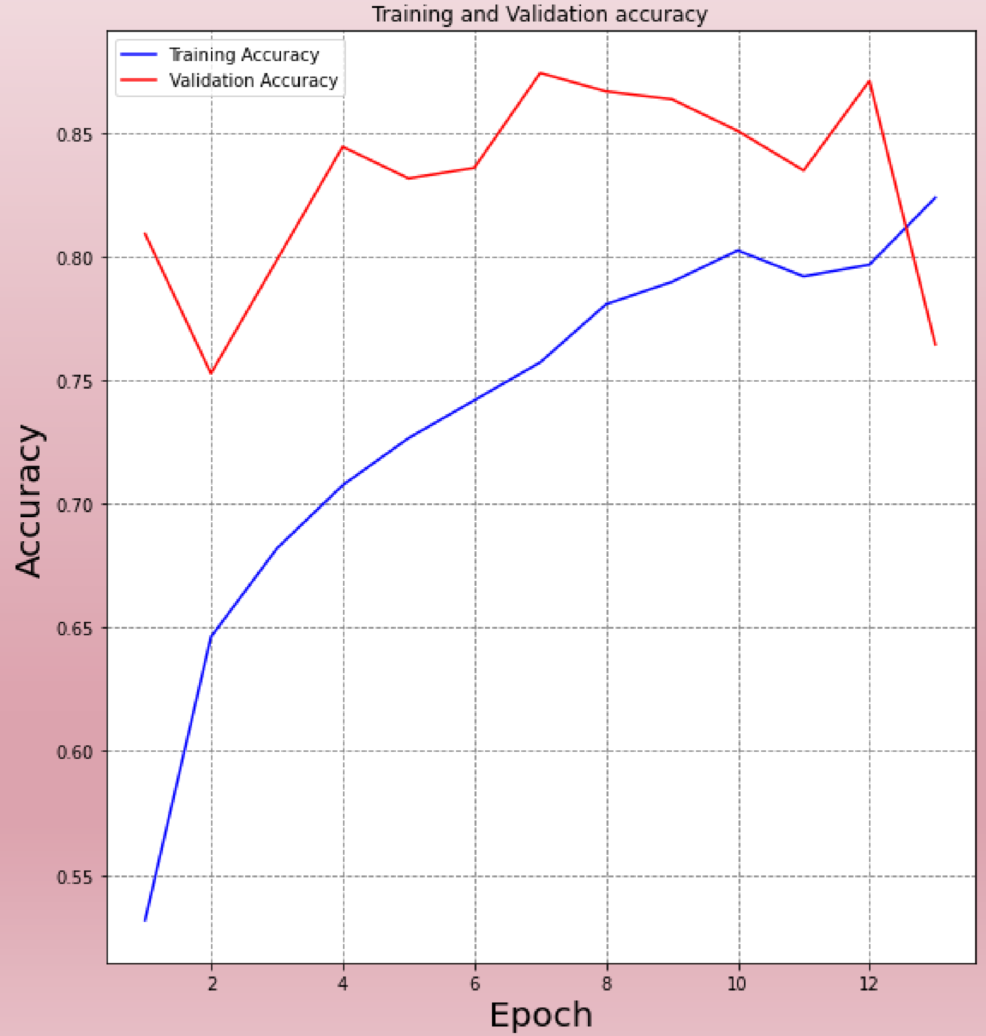
LET'S DIVE IN

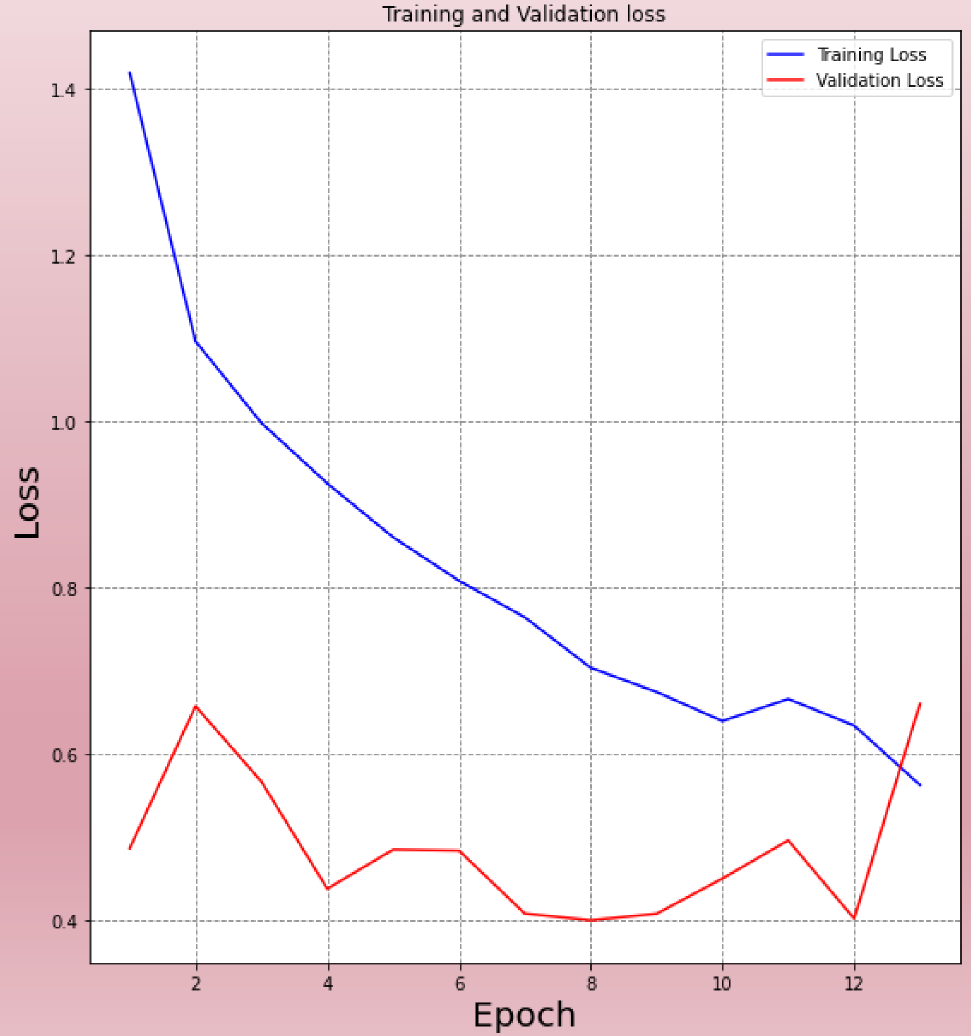
MODEL PERFORMANCE

Characteristics\Model	CNN(10 Epochs)	VGG16(10 Epochs)	ReNet50(20 Epochs)	MobileNet(20 Epochs)
Timeframe	15.01 Minutes	38.83 Minutes	44.6 Minutes	14.56 Minutes
Training Accuracy	66.64%	75.37%	77.60%	95.37%
Training Loss	86.37%	76.80%	57.28%	12.58%
Validation Accuracy	82.4%	83.05%	86.67%	89.77%
Validation Loss	55.93%	47.78%	40.00%	35.20%

RESNET50

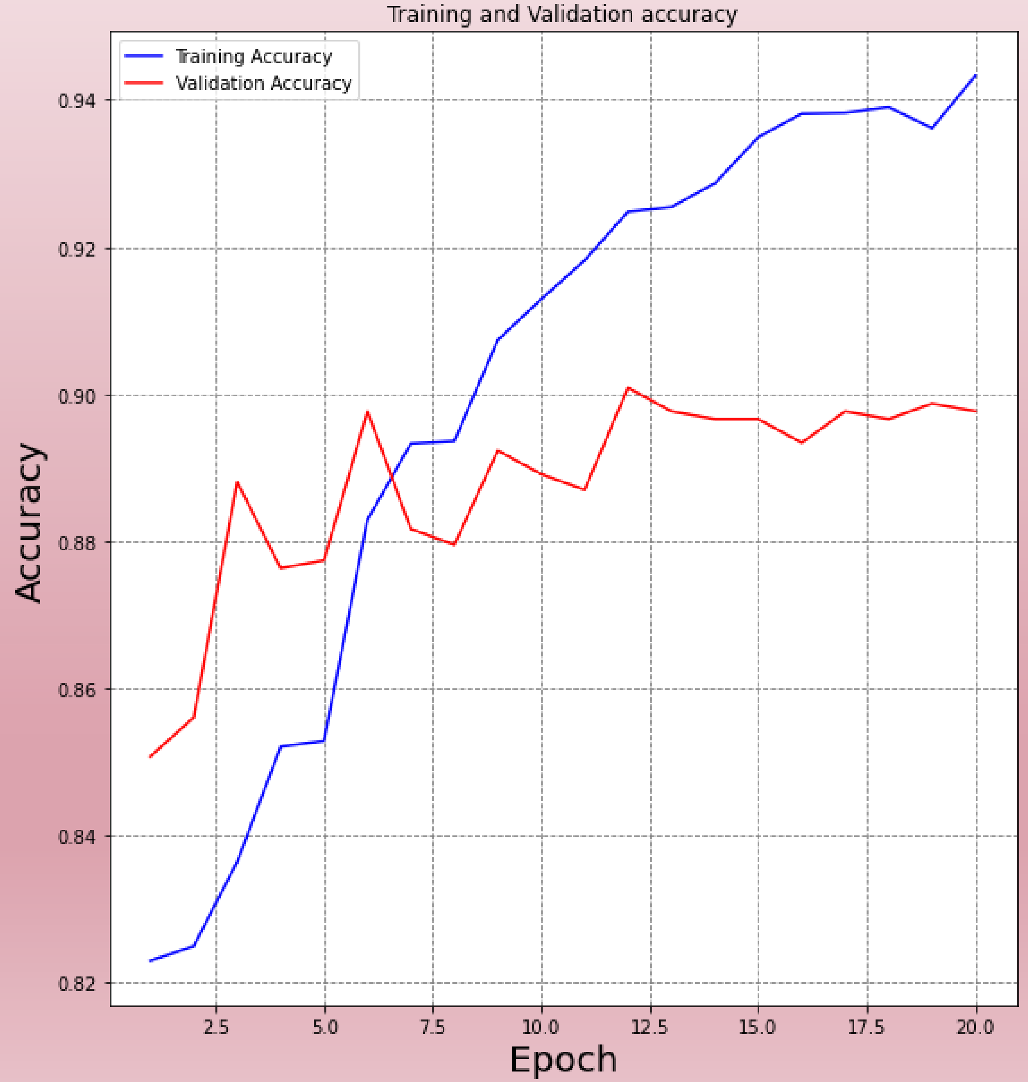
MODEL ACCURACY AND LOSS PLOT

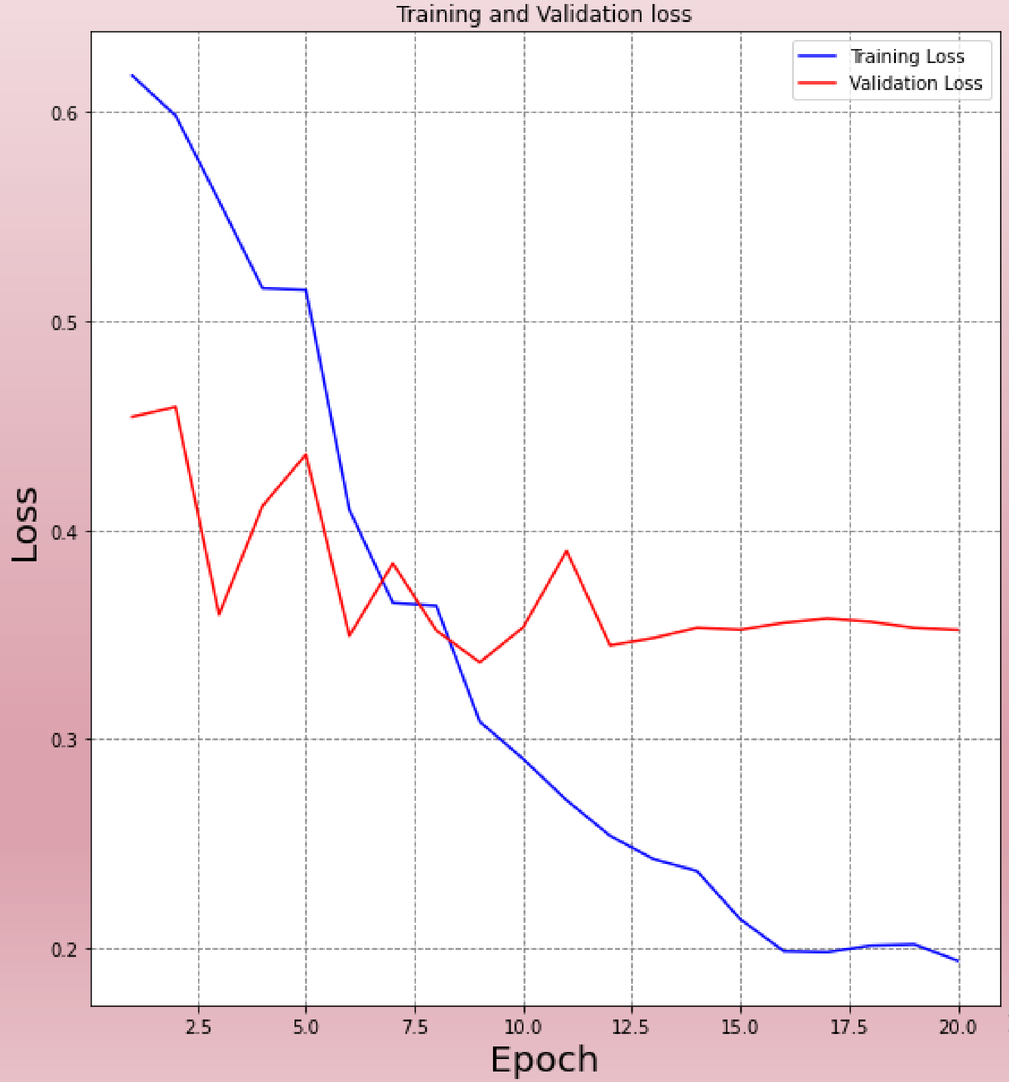




MOBILENET

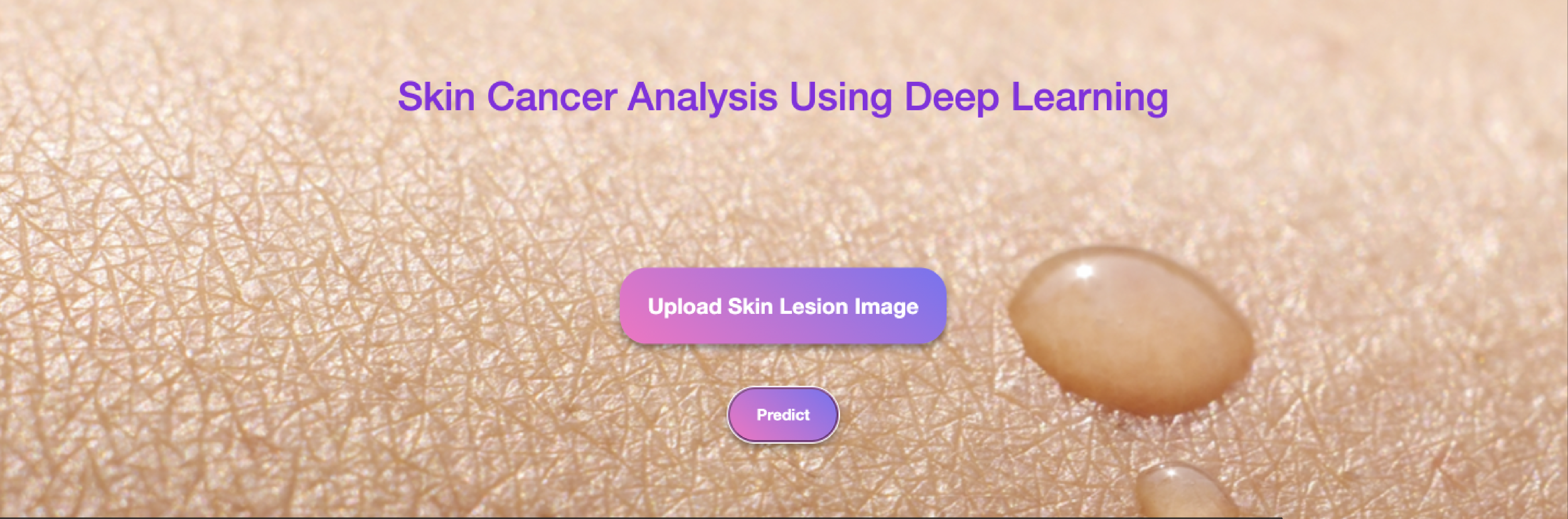
MODEL ACCURACY AND LOSS PLOT





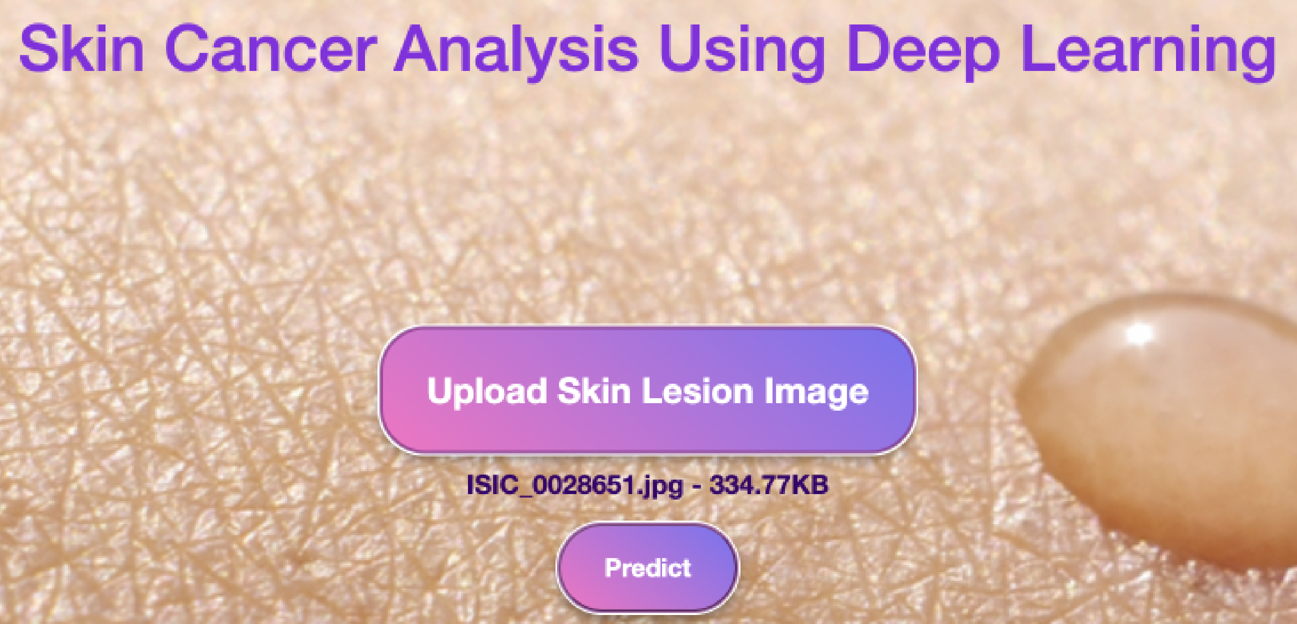
WHAT’S NEXT

WEB APPLICATION



HOME PAGE

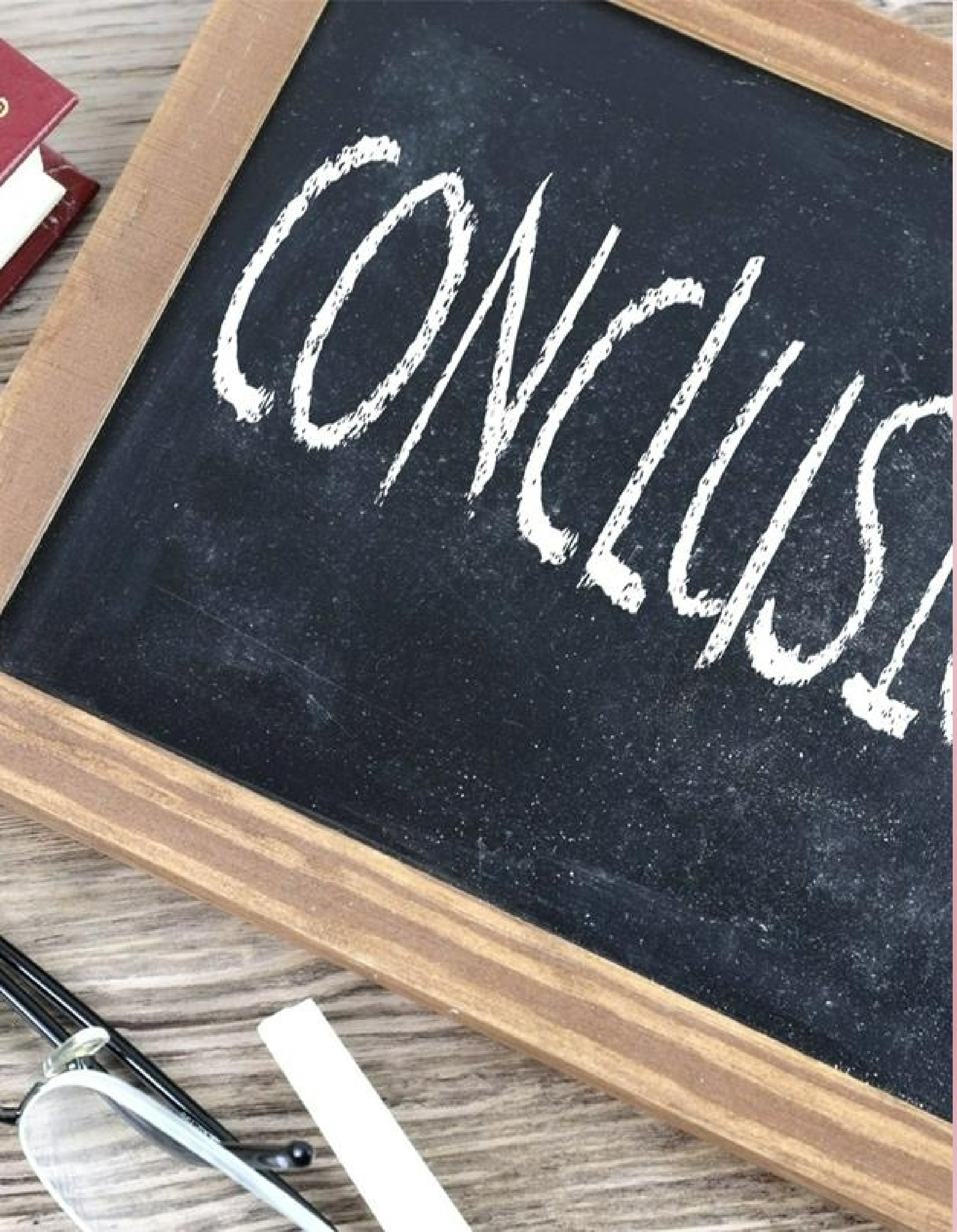
- Click on the Upload Skin Lesion Image label to select image.
- Click on the Predict Button to start the prediction and display the classification result.



UPLOADED IMAGE

- Preview of Upload Skin Lesion

PREDICTED RESULT USING TRAINED MOBILENET MODEL



CONCLUSION

- Training Accuracy of 95.37% and Validation Accuracy of 89.77% is achieved using MobileNet Deep Learning Model.
- MobileNet performed well in classifying all the Skin Cancer Lesions.
- MobileNet performance much faster and more accurate compared to ResNet50, VGG16 and cutom CNN model.
- MobileNet suited best suited for mobile applications as small size and faster performance.


QUESTIONS
WE APPRECIATE FEEDBACK

THANK YOU!

STAY SAFE AND HEALTHY

TEAM 21