Skin Lesions Classification using Computer Vision and Convolutional Neural Networks

Machine Learning Project

SATISH PILLA

B210472095

Contents

1 Introduction

Skin cancer effects many people around the world. Skin cancer is the most common malignancy in humans. Skin cancer is considered one of the deadliest dermatological diseases and is caused by unregulated cell growth on the skin surface, primarily on skin exposed to the sun, primarily the scalp, face, lips, ears, neck, chest, arms, and legs [1]. It is usually diagnosed visually, beginning with a clinical screening, followed by a dermoscopic examination, a biopsy, and a histopathological evaluation. So, identifying someone with or at a risk of skin cancer could help to take measures right away to lower their risk or destroy any cancer (if developed) at an early stage. Therefore, building an automatic system for the classification of skin lesions would help detect a malignancy.

In the project, we are using Convolutional Neural Networks (CNN) to accurately classify pigmented skin lesion in dermoscopic images to detect the malignant skin lesions as early as possible. CNNs are a class of deep neural network that use convolution instead of general matrix multiplication in at least one of their layers. They excel in analyzing visual imagery because they are fully-connected (FC) feed-forward networks that reduce the number of parameters very efficiently without losing out on the quality of models [2]. Two convolutional neural networks with varied architecture and/or depth, as well as data pre-processing methods, are examined in the project to see how they affect classification performance of skin lesions. The models used are CNN architecture VGG16Net and InceptionNet-V3.

VGG16Net: VGG16Net [3] is a 16-layer CNN model with very small convolution filters and an architecture of increasing depths. In 2014, the model won first place in the ImageNet recognition competition, with a top-5 test accuracy of 92.7 %.

InceptionNet-V3: InceptionV3 [4] is the third version of Google's Inception CNN, which is frequently utilized. It's a 48-layer model that first appeared in the ImageNet identification challenge in 2015, where it came in second place. In the challenge, it achieved an accuracy of 78.1 percent.

2 Dataset

The dataset used in the project was "The HAM10000 dataset, a large collection of multisource dermoscopic images of common pigmented skin lesions" [5], which were released as a training set for academic ML purposes and are publicly available through the ISIC archive [6]. (https://www.kaggle.com/kmader/. It consisted of 10,015 dermoscopic skin pigmented lesion 600 by 450-pixel images, digitized and stored as JPEG images. Initially, they were manually cropped and centered around the lesion, as well as adjusted for contrast and color reproduction.

In this project, we aim to classify skin lesion categories in images by building the most accurate machine learning model for the HAM10000 dataset ("Human Against Machine with 10000 training images"). This will help doctors to quickly identify high priority patients and speed up their work-process.

The dataset includes 7 attributes associated with each image and patient:

- a lesion_id [lesion_id]
- a unique image_id [image_id]
- a diagnostic skin lesion category [dx] [To be predicted in our tests]
- a technical validation field type, which indicates how the skin lesion diagnosis was made [dx_type]
- the patient's age [age]
- the patient's sex [sex]
- the localization of the skin lesion [localization]

The main descriptive statistics of the HAM10000 dataset:

index	**Lesion_id**	**Image_id**	**dx**	**dx_type**	**sex**	**localization**	**celltype**
count	10015	10015	10015	10015	10015	10015	10015
unique	7470	10015	7	4	3	15	7
top	HAM_0001863	ISIC_0027411	nv	histo	male	back	Melanocytic nevi
freq	6	1	6705	5340	5406	2192	6705

index	**age**	**cell_type_idx**
std	16.96861369249538	1.2088585476070266
min	0.0	0.0
mean	51.863828077927295	3.623964053919121
max	85.0	6.0
count	9958.0	10015.0
75%	65.0	4.0
50%	50.0	4.0
25%	40.0	4.0

Table 1: Main descriptive statistics of the data set contents

From above we can see that there is unique image id for each entry, but not a unique lesion id. This meant that there were duplicate photos with the same lesion id but distinct distortions, such as angle, shear, or zoom distortion. Furthermore, class Melanocytic Nevi [nv] dominated the skin lesion categories with a frequency of 6,705 out of the 10,015 photos we received, indicating a class imbalance problem in the data set.

Now let's look at different skin lesion present in the dataset. Figure 2 shows sample images from the dataset for each class. The seven categories are:

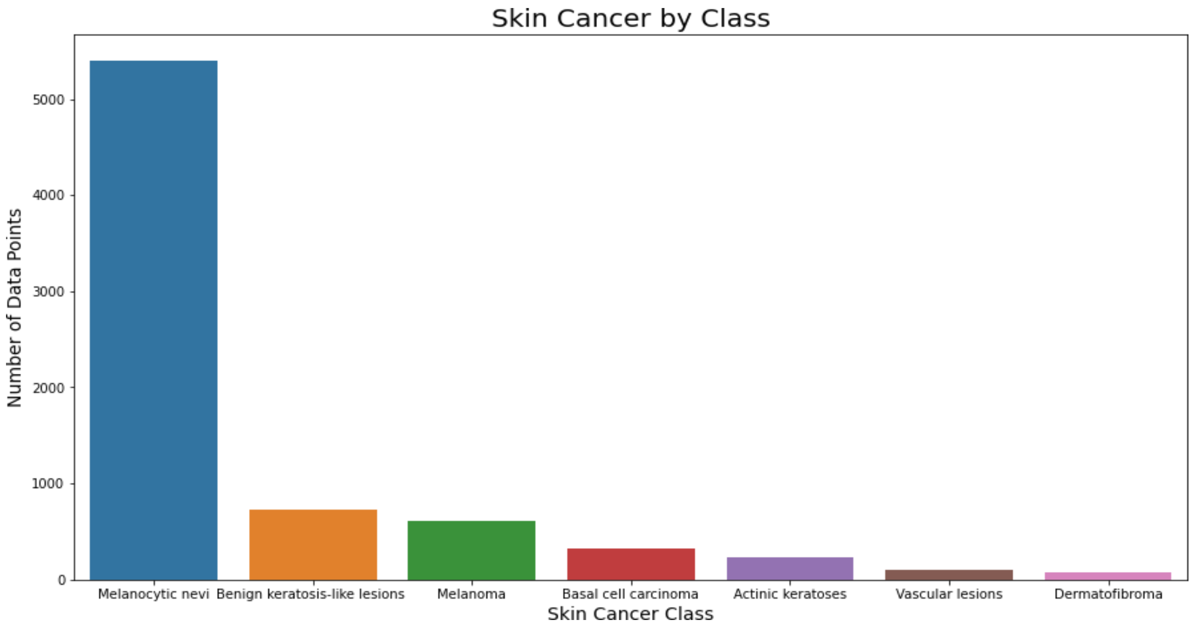


Fig. 1: Skin Lesion Categories Distribution

- Melanocytic Nevi [nv] are benign melanocyte neoplasms that come in a variety of forms.

From a dermatoscopic perspective, the variants may differ dramatically. [6705 photos]

- Melanoma [mel] is a malignant tumor that arises from melanocytes and comes in a variety of forms. It can be treated by simple surgical excision if caught early enough. [1113 photos]
- Benign Keratosis-like Lesions [bkl]: A flat variety of seborrheic keratosis and lichen-planus-like keratoses (LPLK), which is a seborrheic keratosis or solar lentigo with inflammation and regression. [1099 photos]
- Basal Cell Carcinoma [bcc]: is a type of epithelial skin cancer that seldom metastasizes but can be deadly if left untreated. [514 photos]
- Actinic Keratoses [akiec]: A type of squamous cell carcinoma that is non-invasive and can be treated locally without surgery. [327 photos]
- Vascular Lesions [vasc]: These can be benign or malignant and vary from cherry angiomas to angiokeratomas and pyogenic granulomas. [142 photos]
- Dermatofibroma [df]: A benign skin lesion that can be classified as either a benign growth or an inflammatory response to minor trauma. [115 photos]

Let's now check and compare each feature with the target variable:

Skin Cancer by Hispathology:

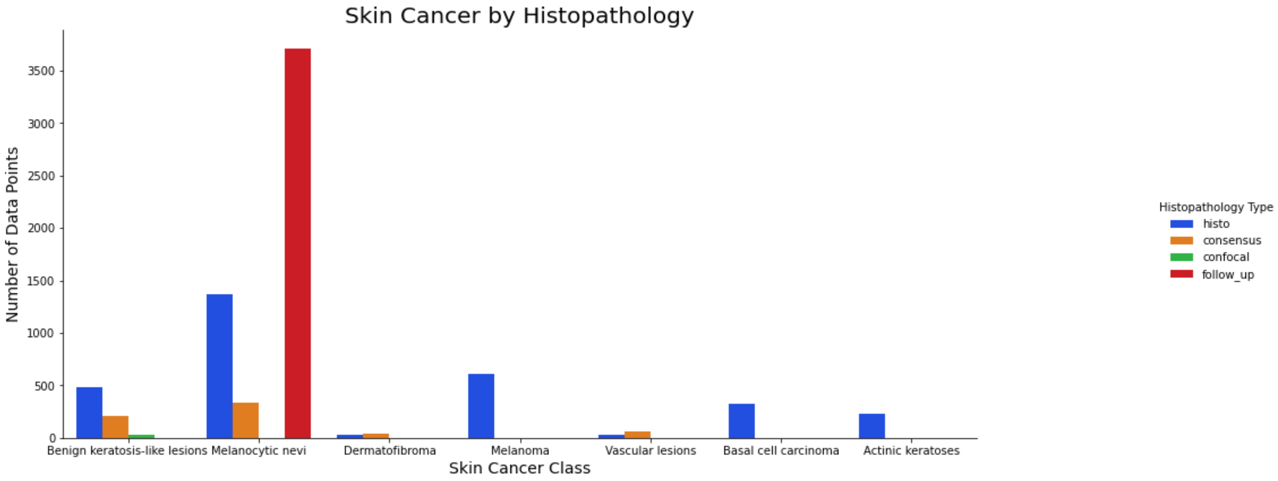


Fig 2: Skin cancer by Hispathology

From above we can clearly see that the majority of the technical validation were either by histograms or follows-up. For the Melanocytic nevi [nv] the confirmation is done majority through follow-ups.

Skin Cancer by Body Localization:

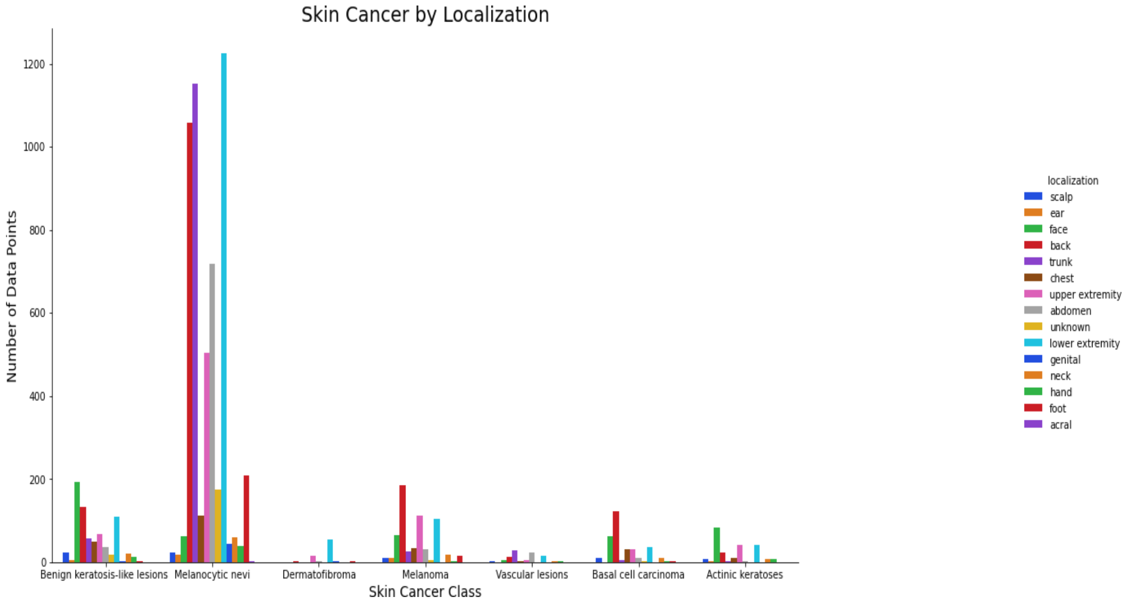


Fig 3: Skin cancer by body localization

We can see that skin cancer as more occurrence in the back, lower extremity and trunk of the people.

Skin Cancer by Gender:

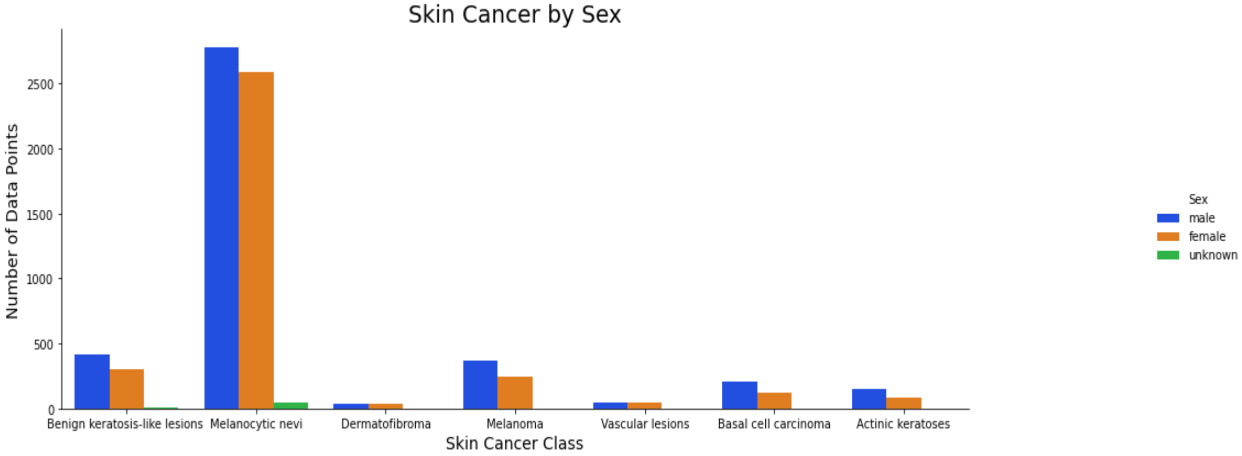


Fig 4: Skin cancer by sex

We can see that there is equal distribution of cancer types among male and female, Melanocytic nevi [nv] is the most common cancer among both male and female.

Skin Cancer by Age:

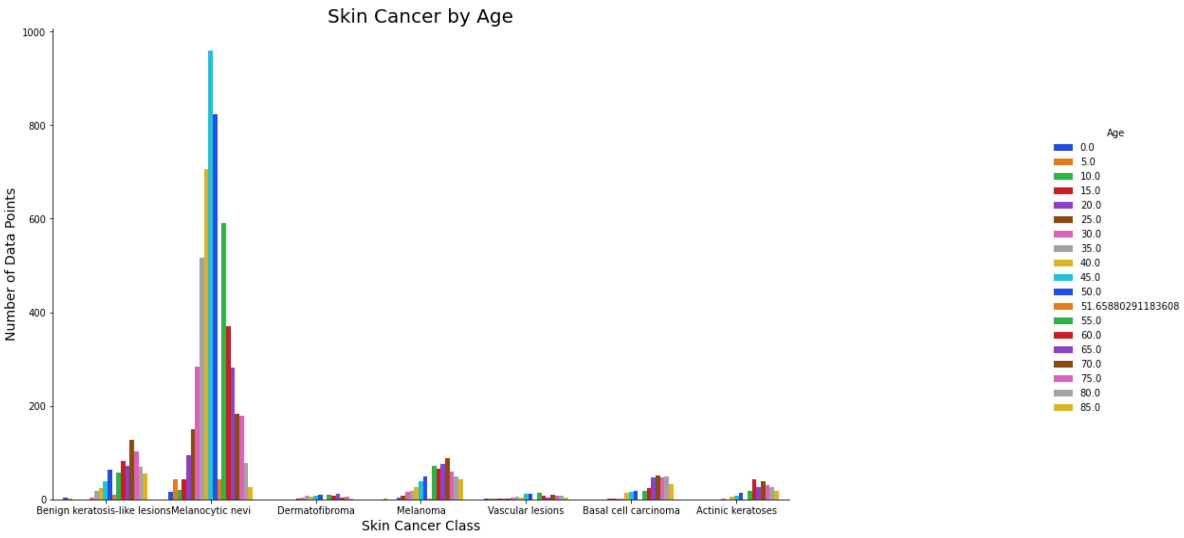


Fig 5: Skin cancer by age

Again, from above we can clearly see that most common cancer is Melanocytic nevi [nv] in the dataset, it seems to occur at middle age for other cancer class, it seems to occur more at a later age.

3 Transfer Learning

Transfer learning is a method of training a deep neural network model on a problem that is identical to the one being solved. It gives us a significant increase in terms of speed and performance. Instead of beginning from scratch, pre-trained models were employed as a starting point. The trained model's layers are then employed in a new model that is trained on the problem of interest. Transfer learning, to put it simply, is a process in which a model learned on one problem is used in some way on a second, related problem.

4 Convolution Neural Network

Convolutional Neural Network (CNN) [7] was created as a result of advancements in deep neural networks. The spatial properties of visual data are not captured by a standard neural network. By comprehending the nearby pixels, CNN can learn local properties. When the data in question is image representations, CNN becomes a highly valuable tool. Convolutional layers, Activation Functions, Pooling layer, and Fully Connected (FC) layer are the basic building parts of a CNN based system.

A CNN is created by stacking these layers one after the other. A CNN is made by stacking layers one by one. Due to the availability of massive amounts of labelled data and computational power, CNNs have advanced at a fast pace since 2012. Various designs, including as AlexNet, ZFNet, VGGNet, GoogLeNet, and ResNet, have established computer vision benchmarks.

The CNN layers are as follows:

- Input Layer: The pixel values of the input image are stored in the input layer. Because we're using full-color photographs in our project, Initially the image has a resolution of 600 x 450 pixels. In data preprocessing we are reducing the resolution to 1:4 ratio.

- Convolutional Layer: A CNN's main building block is this layer. The parameters of the layer are learnable filters which extend through the full depth of the input. By picking a small section of the image at a time, these filters traverse the full image area. The kernel output for that region is determined by multiplying the filter values by the pixel values, and the sum of the products is calculated as the kernel output for that region. Similarly, moving the kernel throughout the whole image space produces a single feature map [8]. Many feature maps are created when multiple such kernels are utilized, and these feature maps form the convolution layer's output. The model complexity is reduced since the weight vector that generates a feature map is shared.

- Non-Linearity Layer: This layer of neurons uses activation functions to introduce non-linearities that are useful in multi-stage neural networks. Sigmoid, rectified linear unit (RELU), or tanh activation functions could be used.

- Pooling Layer-After the convolution layer, the pooling layer takes the little boxes created by the convolution layer and outputs a single maximum or average of the values contained within the box. This aids in the reduction of overfitting.

- Fully Connected Layer-At the end of the model, one or two fully connected layers are added, which connect all neurons from the previous layer to all neurons in the current layer. The final layer of FC estimates probability for each class. Softmax is a common multiclass categorization system. Softmax teaches the last layer to properly predict each image with the highest level of confidence.

4.1 VGG16Net Architecture

VGG16Net is a convolutional neural network (CNN) architecture that won the 2014 ILSVR(Imagenet) competition. It is regarded as one of the best vision model architectures ever created. The most distinctive feature of VGG16 is that, rather than having a huge number of hyper-parameters, they focused on having 3x3 filter convolution layers with a stride 1 and always used the same padding and maxpool layer of 2x2 filter stride 2. Throughout the architecture, the convolution and max pool layers are arranged in the same way. It has two FC (completely connected layers) in the end, followed by a softmax for output. The 16 in VGG16 alludes to the fact that it contains 16 layers with different weights [9]. This is a huge network with a lot of traffic.

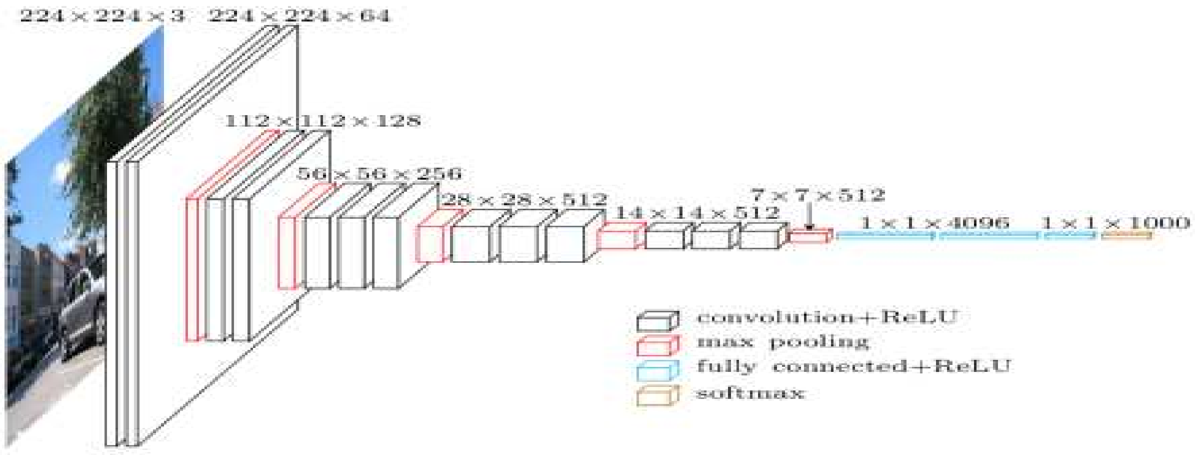


Fig 6: VGG16Net Architecture

In the project we used the pre-trained ImageNet model weights, and then fine tune all network layers with our dataset. The images are all resized in the pre-processing steps. The first layer of the CNN is a function extractor, while the final layer is a softmax classifier that sorts the images into one of the specified categories.

4.2 InceptionV3 architecture

On the ImageNet dataset, Inception v3 is a commonly used image recognition model that has been demonstrated to achieve higher than 78.1 percent accuracy. The model represents the result of several ideas explored over time by a number of researchers [10]. It is based on the original paper: "Rethinking the Inception Architecture for Computer Vision" by Szegedy, et. al.

Convolutions, average pooling, max pooling, concatenations, dropouts, and fully linked layers are among the symmetric and asymmetric building components in the model. Batch normalization is done to activation inputs and is used extensively throughout the model. Softmax is used to calculate loss.

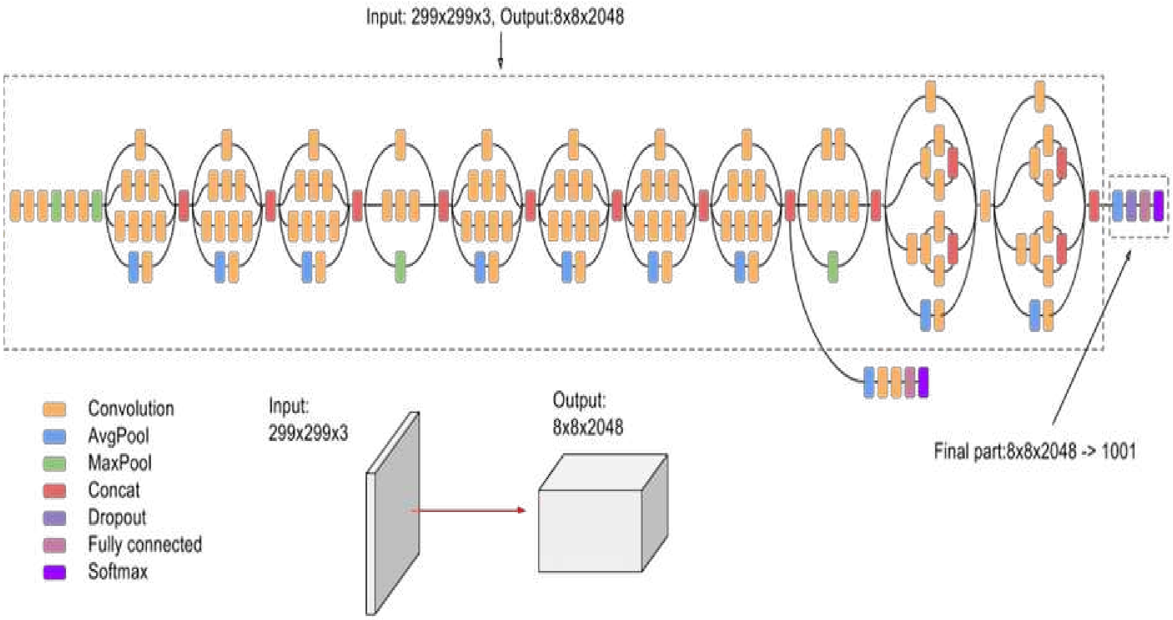


Fig 7: InceptionV3 Architecture

5 Data Pre-processing

5.1 Data Editing and Cleansing

The data was directly loaded from Kaggle into google colab. For the images and their labels, I created two dictionaries. The first dictionary contained image names extracted from the downloaded kaggle data set's multiple image folders. A second dictionary was then built to match the diagnostic skin lesion categories code to the entire name of the category.

Data cleansing: This involved removing duplicate photos from lesion id's and preserving only one image per lesion id. These images show the same exact lesion, but from a different perspective, zoom level, and so on. In addition, the age records contained fifty-two NA values. These were replaced with the data set's age mean. Unique numeric codes were created for each skin lesion category to assist with the predictions to be made later on, as integers were easier to handle than strings.

The images were scaled and processed. Due to the large number of images on hand, width and height sizes would have been a challenge for training our CNN models. Images were scaled by a ratio of 1:4 to speed up the process and guarantee that our CNN models performed properly. The new images had a resolution of 150 by 112 pixels. Afterwards, the image was flattened and stored as a numeric image list.

One-hot encoding was used to establish unique number codes for the skin lesion categories. Because we had seven distinct label integer values, the integer encoded variable was removed and a new binary variable was inserted for each unique integer value, yielding the (1,7) row vector in our case. There was no ordinal relationship between the integer numbers of the label, therefore this was required. It ruled out any natural integer ordering assumptions that the CNN algorithm would have reached.

5.2 Data Splitting

Feature and target split: The feature used in the project was the flattened numeric images list and the target was the one-hot encoding created for the skin lesion categories. The data was split 70:10:20 respectively across each class individually to ensure there was enough samples from each class in each split for accurate modeling.

5.3 Feature Normalization

Normalization is a scaling technique in which values are shifted and rescaled so that they end raging between 0 and 1. The normalization of each image was done by subtracting its values from the training's mean value and then dividing by the training's standard deviation.

$$\boldsymbol{x_{mean}} = (\boldsymbol{x} - \mu)/\boldsymbol{\sigma}$$

5.4 Data Augmentation

To avoid overfitting, all of the original images were modified and enhanced at each epoch and then used for training. Because the model was trained on numerous versions of the same image, it became more resilient and accurate. Each epoch had the same number of photos as the original images. The images were:

Randomly rotated by 20%
Randomly shifted horizontally by 20%
Randomly shifted vertically by 20%
Randomly sheared by 10%
Randomly zoomed by 10%
Randomly channel shifted by 10%

6 Modelling and Results:

VGG16Net and InceptionV3 models were pre-build and loaded via keras application library package. Extra layers top (last layers) was added to align parameter numbers and outputs.

Models Hyperparameters:

Hyperparameter	Value
Optimizer	Adam
Loss Function	Categorical Cross-Entropy
Epochs	50
Batch Size	10

Table 2: Model Hyperparameters

- Optimizer: Adam is the most widely used optimization algorithm for training deep neural networks today because it is simple to use, computationally efficient, and successful when dealing with vast amounts of data and parameters.
- Loss Function: Categorical cross-entropy is a loss function for categorizing single labels. This is when only one category is applicable for each data point. This worked nicely in this case because one example could only belong to one of the seven types of skin lesions.
- Epochs: After several initial trials with values of 20, 25,50, 100, 150, and 200, 50 epochs were found to be sufficient to provide the best results.
- Batch Size: After a series of trails with batch sizes of 5, 10, 20, and 40, a batch size of 10 yielded the best results.

Evaluation Metrics: A classification task can be evaluated using a variety of metrics, but we'll focus on accuracy, precision, and recall.

- Accuracy: The ratio of correct predictions to total predictions is used to define a model's accuracy. The more accurate the model is, the more accurate forecasts it made for that class.
- Recall: The ratio of true positives to total true positives and false negatives is the recall of a model.
- Precision: The ratio of true positives to total true positives and false positives is the

precision of a model. The lower the false positive predictions the model generates for that class, the higher the precision.

6.1 Results:

VGG16Net: Default VGG16 model except image size to be (224,244,3), but our image sizes are different, so we need to change the parameter image size in the first layer. Our size will be (150,112,3). Our problem consists of 7 classifications, so we are excluding the top layer and customizing it.

InceptionV3: Similarly, Inception V3 model excepts size to be (229,229,3), so again we can need to change the parameter image size in the first layer same as above.

Figure 9 and 10 shows the custom architecture of VGG16Net and InceptionV3, where we have added one flatten layer and one dense layer.

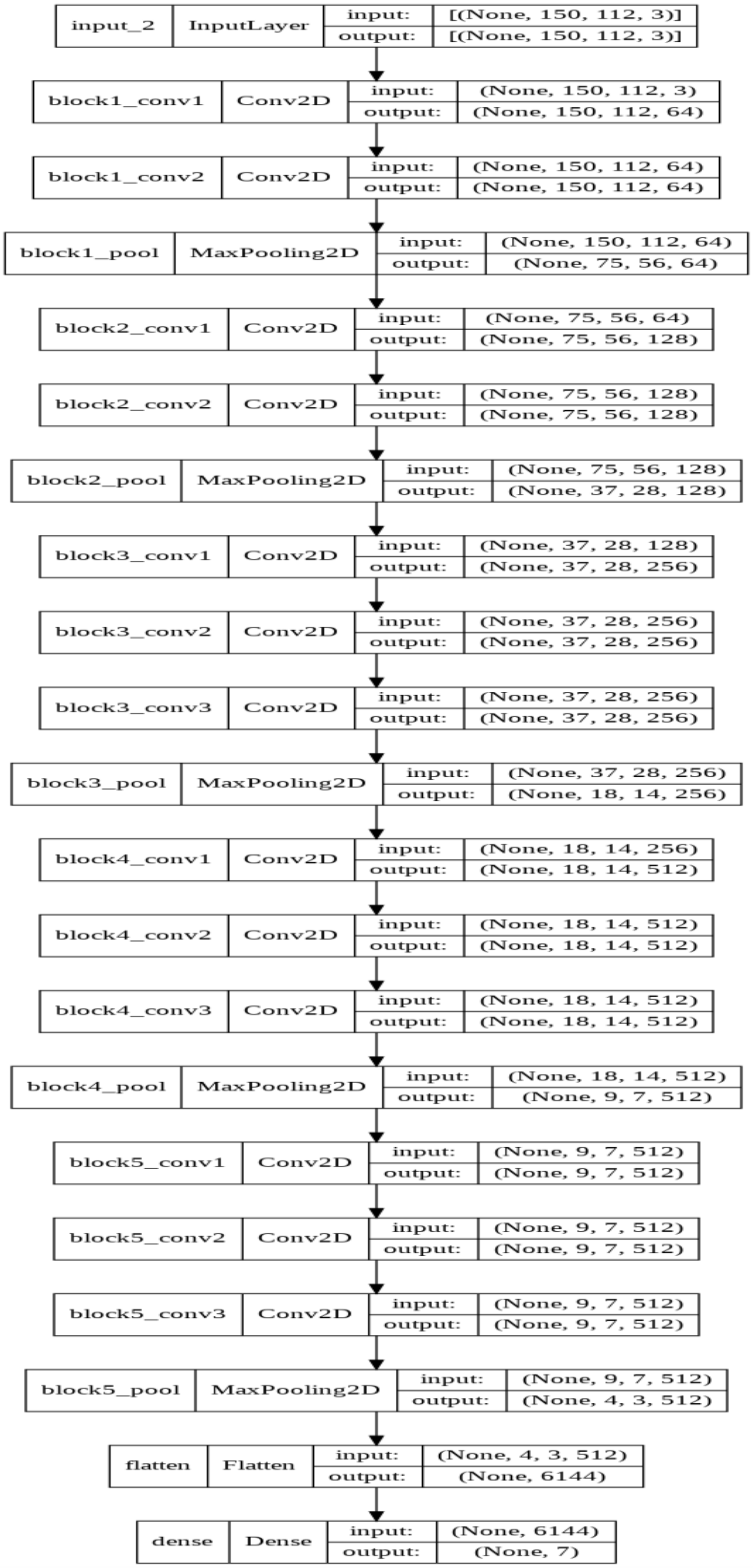

input_2 | InputLayer | input: [(None, 150, 112, 3)] | output: [(None, 150, 112, 3)]
block1_conv1 | Conv2D | input: (None, 150, 112, 3) | output: (None, 150, 112, 64)
block1_conv2 | Conv2D | input: (None, 150, 112, 64) | output: (None, 150, 112, 64)
block1_pool | MaxPooling2D | input: (None, 150, 112, 64) | output: (None, 75, 56, 64)
block2_conv1 | Conv2D | input: (None, 75, 56, 64) | output: (None, 75, 56, 128)
block2_conv2 | Conv2D | input: (None, 75, 56, 128) | output: (None, 75, 56, 128)
block2_pool | MaxPooling2D | input: (None, 75, 56, 128) | output: (None, 37, 28, 128)
block3_conv1 | Conv2D | input: (None, 37, 28, 128) | output: (None, 37, 28, 256)
block3_conv2 | Conv2D | input: (None, 37, 28, 256) | output: (None, 37, 28, 256)
block3_conv3 | Conv2D | input: (None, 37, 28, 256) | output: (None, 37, 28, 256)
block3_pool | MaxPooling2D | input: (None, 37, 28, 256) | output: (None, 18, 14, 256)
block4_conv1 | Conv2D | input: (None, 18, 14, 256) | output: (None, 18, 14, 512)
block4_conv2 | Conv2D | input: (None, 18, 14, 512) | output: (None, 18, 14, 512)
block4_conv3 | Conv2D | input: (None, 18, 14, 512) | output: (None, 18, 14, 512)
block4_pool | MaxPooling2D | input: (None, 18, 14, 512) | output: (None, 9, 7, 512)
block5_conv1 | Conv2D | input: (None, 9, 7, 512) | output: (None, 9, 7, 512)
block5_conv2 | Conv2D | input: (None, 9, 7, 512) | output: (None, 9, 7, 512)
block5_conv3 | Conv2D | input: (None, 9, 7, 512) | output: (None, 9, 7, 512)
block5_pool | MaxPooling2D | input: (None, 9, 7, 512) | output: (None, 4, 3, 512)
flatten | Flatten | input: (None, 4, 3, 512) | output: (None, 6144)
dense | Dense | input: (None, 6144) | output: (None, 7)

Fig 8: Architecture of VGG16Net used for image classification

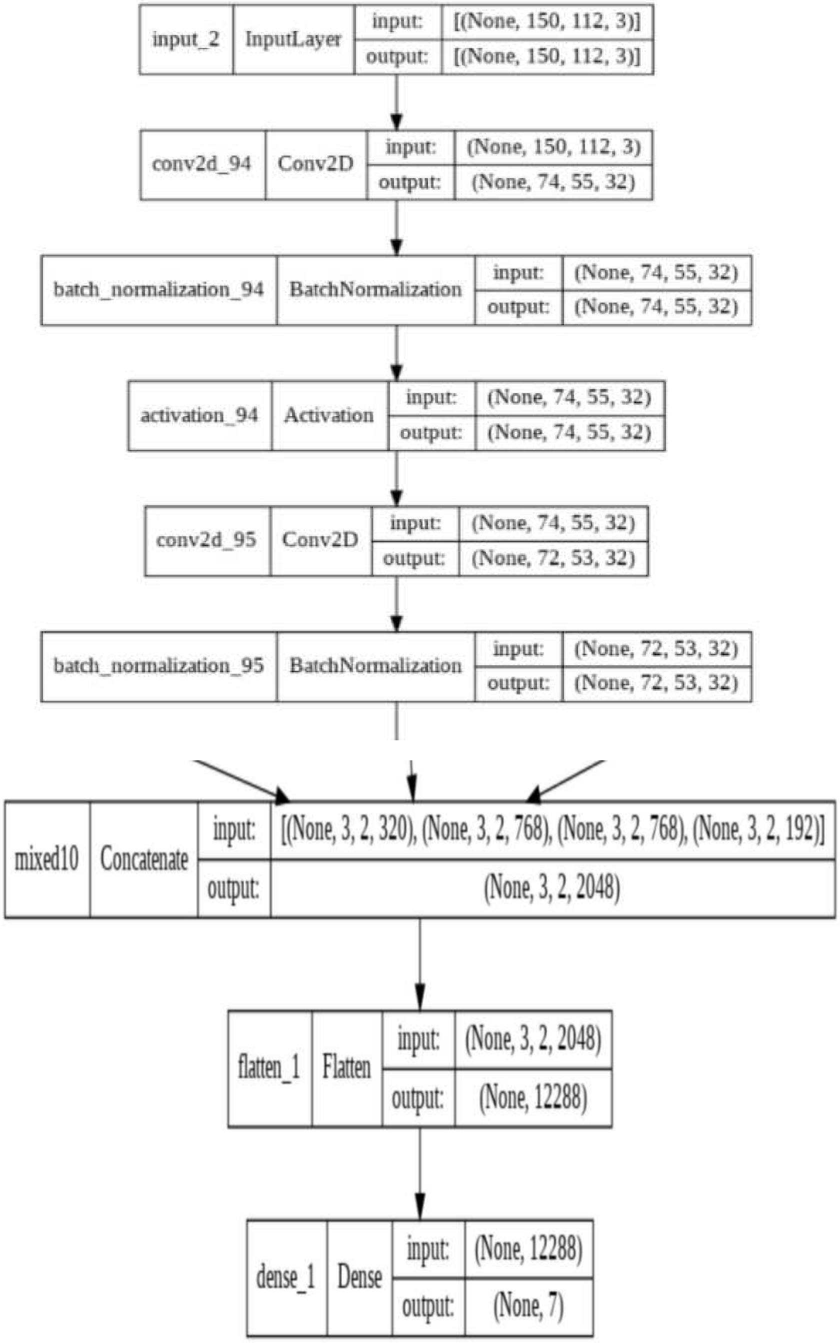

input_2 | InputLayer | input: [(None, 150, 112, 3)] | output: [(None, 150, 112, 3)]
conv2d_94 | Conv2D | input: (None, 150, 112, 3) | output: (None, 74, 55, 32)
batch_normalization_94 | BatchNormalization | input: (None, 74, 55, 32) | output: (None, 74, 55, 32)
activation_94 | Activation | input: (None, 74, 55, 32) | output: (None, 74, 55, 32)
conv2d_95 | Conv2D | input: (None, 74, 55, 32) | output: (None, 72, 53, 32)
batch_normalization_95 | BatchNormalization | input: (None, 72, 53, 32) | output: (None, 72, 53, 32)
mixed10 | Concatenate | input: [(None, 3, 2, 320), (None, 3, 2, 768), (None, 3, 2, 768), (None, 3, 2, 192)] | output: (None, 3, 2, 2048)
flatten_1 | Flatten | input: (None, 3, 2, 2048) | output: (None, 12288)
dense_1 | Dense | input: (None, 12288) | output: (None, 7)

Fig 9: Architecture of inceptionV3(snippet of last layer and first layer) used for image classification (Entire architecture is present in the google colab notebook)

The plot for the accuracy and loss obtained during the training and testing process for VGG16Net and inceptionV3 are shown in figure 11 and 12.

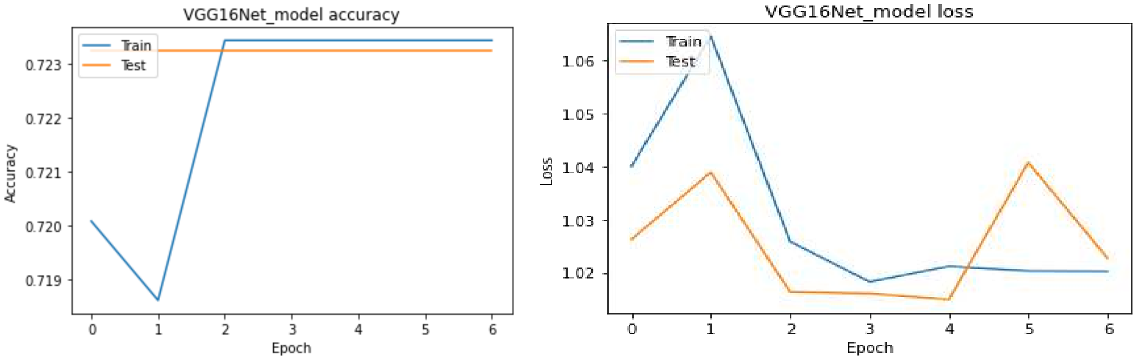


Fig 11: Plots of loss and accuracy for VGG16Net model.

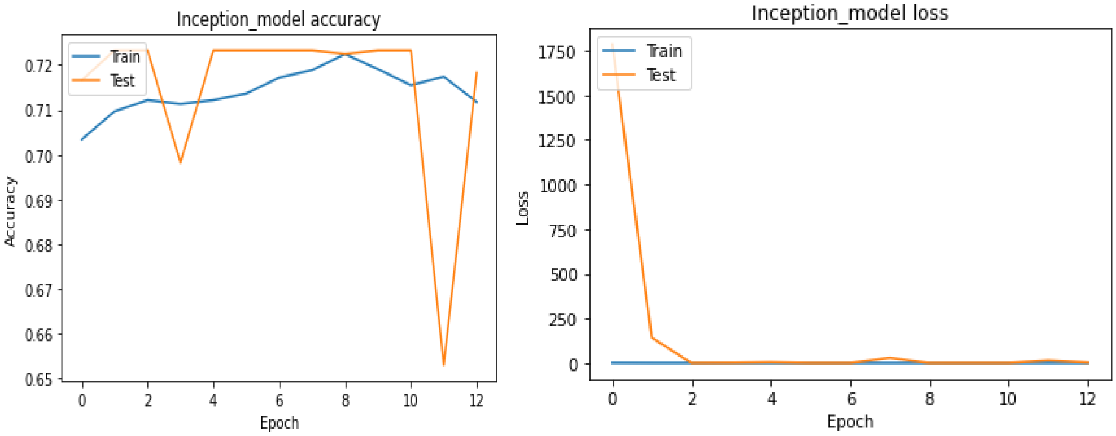


Fig 11: Plots of loss and accuracy for InceptionV3 model.

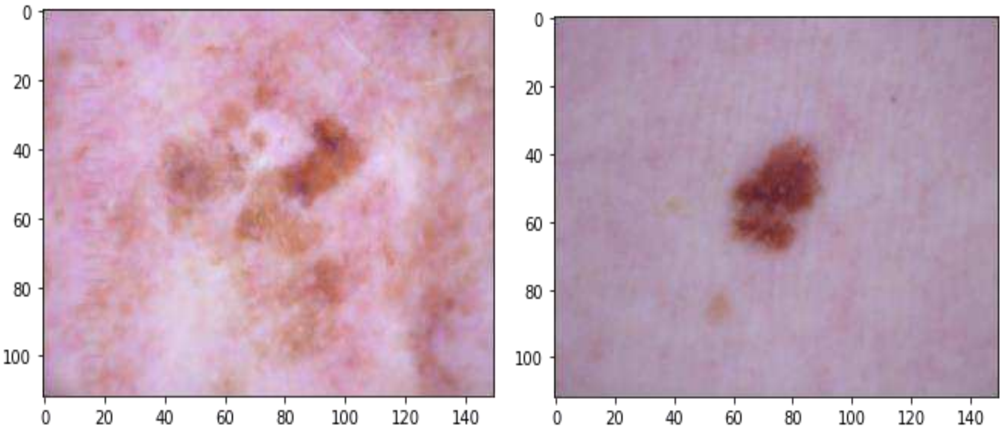
It was discovered that after training the model for 25 epochs, it had a test accuracy of 72% for both the model (VGG16Net and InceptionV3). For the VGG16Net model the accuracy of training data for the first epoch increased, for second epoch it got decreased and form third epoch it remains same throughout the process. The test accuracy for VGG16Net was same for all the epochs. Training and test loss of VGG16Net first increased for first 2 epoch then it gradually decreased up to 4 epochs. Then from there it started to increase again. Similarly, for InceptionV3 the training and test accuracy had a constant increase up to 11 epochs and for the 12th epochs it dipped to it lowest accuracy. The training and test loss of inceptionV3 decreased after 1st epochs and after that it remained constant throughout the process.

Performance results of transfer learning:

CNN models	**Accuracy**	**Precision**	**Recall**
VGG16Net	0.722892	0.722892	0.7228
InceptionV3	0.721553	0.550818	0.721553

Table 3: Performance evaluation of VGG16Net and InceptionV3 models

Testing for VGG16 Model:

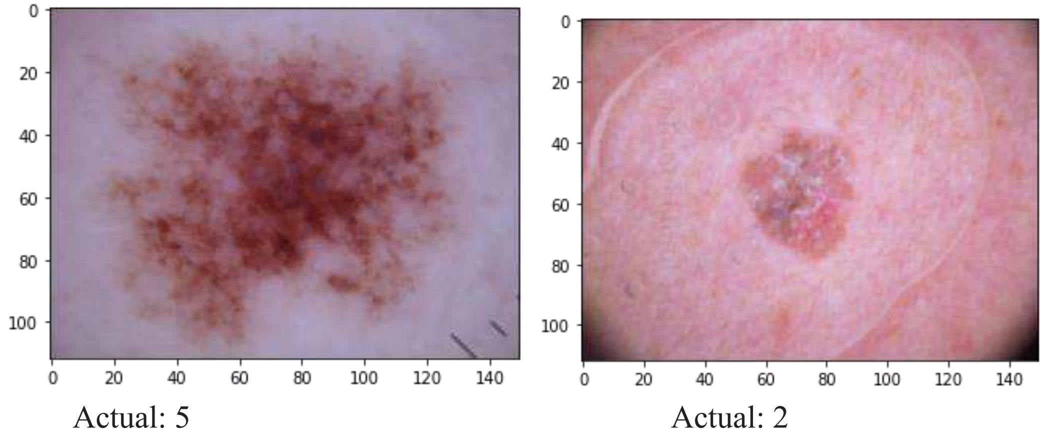


Actual: 2
Predicted: 4

Actual: 4
Predicted: 4

Fig 12: Testing for VGG16 model

Testing for Inception V3 model:



Actual: 5
Predicted: 2

Actual: 2
Predicted: 2

Fig 13: Testing for InceptionV3 model

7 Conclusion and Future work:

In the project we investigated the use of two different CNN architecture (VGG16Net, InceptionV3) to predict the skin lesion categories based on skin lesion images. Both the models gave identical Accuracy, precision and recall scores (0.722892, 0.722892, 0.722892), The two models were saved as VGG16_model and InceptionV3_model. While testing on random images both the models were able to predict Melanocytic nevi correctly while predicting other skin lesion it failed sometimes. We were able to successfully build and applied two of the most used CNN models for image classification (VGG16Net and InceptionV3).

Future work:

- Utilize all the data in the dataset, such as age, sex and others via machine learning technique to extract useful information.
- Explore other skin dataset for more data.
- Explore other pre-trained CNN models such as ResNet50, Xception etc.

.

8 Reference:

[1] T. Emara, H. M. Afify, F. H. Ismail and A. E. Hassanien, "A Modified Inception-v4 for Imbalanced Skin Cancer Classification Dataset," 2019 14th International Conference on Computer Engineering and Systems (ICCES), 2019, pp. 28-33, doi: 10.1109/ICCES48960.2019.9068110.

[2] W. F. Chabala and I. Jouny, "Comparison of Convolutional Neural Network Architectures on Dermastopic Imagery," 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 2020, pp. 0928-0931, doi: 10.1109/UEMCON51285.2020.9298059.

[3] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[4] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2818-2826, doi: 10.1109/CVPR.2016.308.

[5] Tschandl, Philipp, Cliff Rosendahl, and Harald Kittler. "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions." *Scientific data* 5.1 (2018): 1-9.

[6] *The International Skin Imaging Collaboration (ISIC)*, Dec. 2018, [online] Available: https://www.isicarchive.com/#!/topWithHeader/onlyHeaderTop/gallery.

[7] S. Bassi and A. Gomekar, "Deep Learning Diagnosis of Pigmented Skin Lesions," 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2019, pp. 1-6, doi: 10.1109/ICCCNT45670.2019.8944601.

[8] K. Pai and A. Giridharan, "Convolutional Neural Networks for classifying skin lesions," TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), 2019, pp. 1794-1796, doi: 10.1109/TENCON.2019.8929461.

[9] https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c

[10] https://cloud.google.com/tpu/docs/inception-v3-advanced