Classification of Skin Cancer Using Convolution Neural network Architecture

UG Project

By
Dev kumar Seth(19124047)
and
Manprit (19124022)

Under the guidance of Professor LP Singh

Department of Mathematical Sciences Indian Institute of Technology (BHU), Varanasi Varanasi-221005, India

Abstract:Skin cancer turns out to be one of the most frequent type of cancer in the world whose risk is increasing day by day.Thus classifying the type of skin cancer is a topic of major concern for dermatologists and this has also been an area of research.Classification of skin cancer using computation and artificial intelligence is a challenging task as they can be of varying shapes and sizes,skin color,texture ,also the lightning conditions and contrast of the image may also be a matter of concern.Skin Images are filtered and 8000 images per class (total 7 classes)has been generated from Ham10000 dataset.A model that uses resnet Architecture has been trained to predict the type of lesion into 7 classes of cancer.Complex methods such as transfer learning and Residual Neural Networks(ResNet)are being used to make training process faster and accurate.Analysis of this work can help medical personals in advanced Diagnosis,to know the type of infection so that proper treatment can be given.

Introduction:

Skin cancer turns out to be one of the most frequent diseases in the world. A recent study shows that the risk of skin cancer has increased in the past few years. According to 2018 cancer recordings, the no. of deaths due to cancer was estimated to be 9.6 million (*Cancer*, n.d.)(101/1 per 100000). Detection and classification of cancer are essential in the early-stage as it has a significant impact on patients' survival.

Skin cancer primarily has malignant and benign classes; they both look similar. Its classification is done by Dermoscopy imaging which is done by 3D convolution neural networks with feature maps along with softmax.

Various methods such as transfer learning, segmentation and detection have been used for classification and detection, but this is always a difficult task. Due to the higher image size, it turns out to be heavy work for the feature and activation maps. Further challenges to this task are skin cancer can be of different shapes, and sizes moreover, the images can have distinct contrast, magnification, and augmentation.

In this project, we have made a model that can classify skin cancer into seven categories (Radhakrishnan, 2022) these are:

- Dermatofibroma(DF)
- Benign Keratosis(BKL)
- Intraepithelial Carcinoma(AKIEC)
- Actinic Keratosis
- Basal Cell Carcinoma(BCC)
- Melanocytic Nevi(NV)
- Melanoma(MEL)

In this paper Convolutional neural networks (*Index of /Slides/2017*, n.d.) are used with the help of resnet and denseness architectures, which tries to eliminate the above challenges by augmentation and mathematical computations. A convolution neural network is a part of artificial intelligence which can be used to extract representational features from an image by using various filters that slides through the complete image input and extracts the features. The image is then classified into a certain class on the basis of this feature information.

A significant accuracy of this model can help medical personnel identify and classify skin cancer. Further, With more advanced algorithms and a better dataset, we can make models with better accuracy. As well as, we can apply our algorithms to various other medical fields with decent accuracy.

Related Work:

In the past techniques like K nearest neighbors, Support vector machine, and Cosine similarity Matrix have been used to do the classification job. At present with greater computation power, new advanced algorithms, and a large dataset, models with greater accuracy and efficiency can be developed that can serve the medical industries with new technologies.

Different architectures and techniques have been used in past to classify skin cancer, As and examples. Pomponiu et al (V. Pomponiu, n.d., #) has used only 399 images in his dataset that was taken from a normal standard camera.In his work Alexnet model was applied to the augmented and processed images. The classification was done using cosine similarity matrices. The algorithm was not tested on an independent dataset through an accuracy of 93.64% was achieved. But the training of the algorithm on a small dataset might be a matter of concern in practical usage.

Similar work was also done by Codella et al (V. Pomponiu, n.d., 15).Through the dataset was large 2354 dermatoscopic images from a dataset by the International Skin Imaging Collaboration (ISIC) database, In this work, classification was done into 2 classes namely of melanoma versus non-melanoma lesions. This model gave an accuracy of 93%, moreover, the major findings of this model were deep and silent features had a significant role in the performance of the model.

In 2018 Valle(*[1703.04819] RECOD Titans at ISIC Challenge 2017*, 2017) worked found the importance of data augmentation on model efficiency,similar observations were confirmed by Perez’s 2018 paper ”Data Augmentation for Skin Lesion Analysis”.

A resnet34 model has been applied by (Niharika Gouda, n.d., 15) Which classifies cancer into 9 different classes From a dataset of is International Skin Imaging Collaboration (ISIC) that had 25331 images. This model gave a recall score of 0.83 and an accuracy of 0.92.

Challenges In Past:

The challenges that occur in classifying the type of skin cancer are:

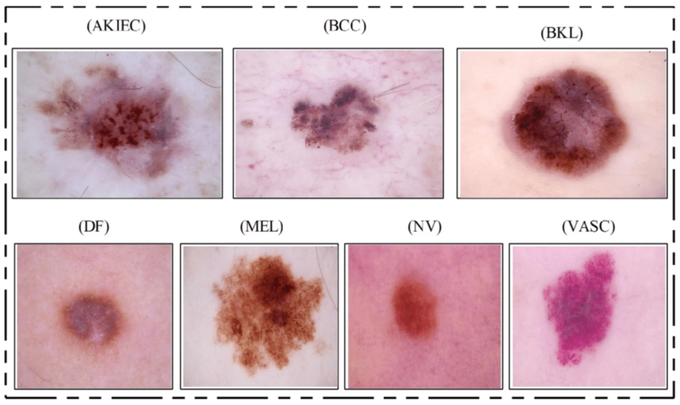
- Skin cancers are of several types, shapes and sizes that can be tough to classify, to handle this, preprocessing is required for accurate prediction.
- The useless data and noise must be removed that may alter the result.
- Low contrast neighboring tissues possess additional difficulties to predict the type of cancer.
- Image could be taken in different lighting conditions and can have multiple colour and texture variations
- Sometimes moles present on the human skin make it difficult to detect skin cancer accurately.

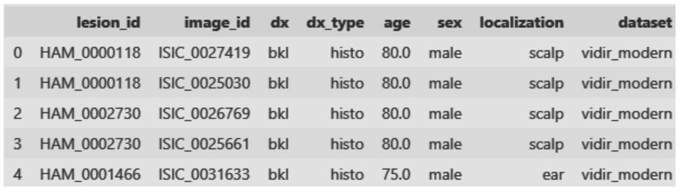
Objectives:

Our objective is to use different deep learning architectures (resnet) for the classification of the skin lesion into 7 types of cancer(Described above) and also to obtain a classification score which validates on accuracy.Analysis of this investigation can help the medical specialist to know the kind of infection so that begin with any treatment if required.

Dataset and materials:

The dataset used in this project is Ham10000 (Tschandl, n.d.) it takes the problem of lack of diversity and small data size by providing 10015 (of size 450 X600) dermatoscopic images over the period of 20 years.All the images are resized into 224 X 224 which is suitable for our architecture. Oversampling and undersampling methods are used so that the no. of Images per classed can be similar .Finally, the images are normalised so that the pixel values comes in range of 0 to 1.





	lesion_id	image_id	dx	dx_type	age	sex	localization	dataset
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp	vidir_modern
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp	vidir_modern
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp	vidir_modern
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp	vidir_modern
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear	vidir_modern

Fig 1 Ham10000 Dataset Samples

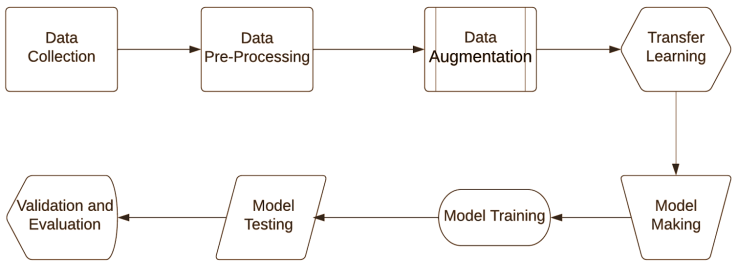
Because of the large data size, we used cloud services for storage and computations like google collab notebook and google Drive.

Transfer learning is a standard method for training the CNN models, as the publicly available dataset is limited. Therefore, an extensive database, Imagenet, is designed consisting of 1000 classes and 1.5 million Images. Thus the models pre-trained image dataset via ImageNet and the actual training parameters are found during training.

Artificial Neural Network (ANN), having one of the types called Deep Residual Network (ResNet) which are made with the point of solving the issue of less precision while not preferring a shallower ANN which has a deep layer but constructing a plain ANN. Resnet was made so that we can add significant numbers of layers in an ANN,that may give rise to better accuracy.

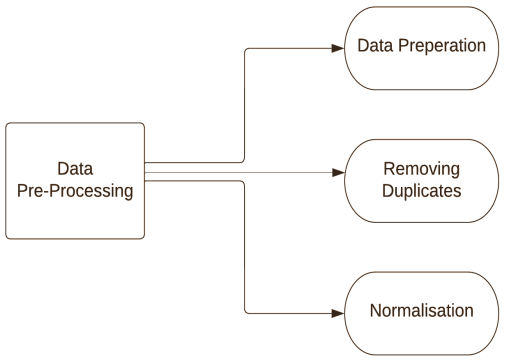
The motivation behind the ResNet is modifying ANN with more profound layers with high exactness. The weight updation of the identical layers (to lessen debasement slope) can happen by creating ANN. The idea is implemented by utilizing an "alternate route association".

Theory and Methods:



Data Collection: We have collected the Ham10000 dataset from the Harvard Dataverse description of which is given above.

Data Preprocessing:



- Data loading and preparation have been done on the cloud through Google colab notebook.
- Duplicates and unnecessary data have been removed.
- In last we change the pixel values from 0-255 to 0-1, this helps in better efficiency during the execution of the model.

We are going to divide the Model Making part of study into these parts and study each of these parts separately:

Neural Networks(NN) or artificial neural networks (ANNs) :

NNs (*What Are Neural Networks?*, 2020) or ANNs are DL(Deep Learning) methods, whose nodal layer contains 1 input, 1 or more hidden, and 1 output layers.

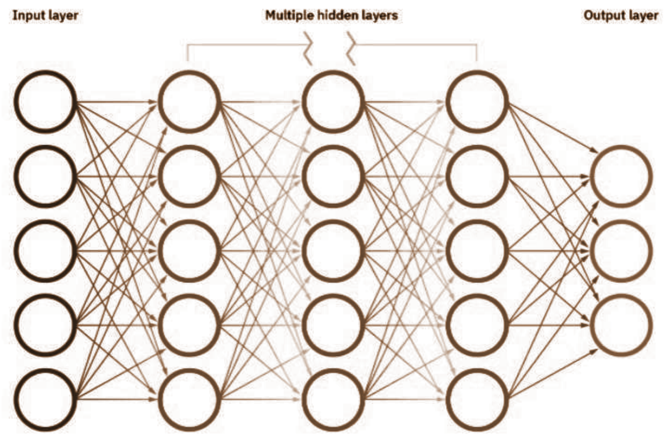


Fig 2 Neural Network

What makes neural networks work?

Visualize each of the nodes as a distinct linear regression model containing input, weights, bias, and an output. This is what the formula would look like:

$$k = \sum_{i=1}^{n} w_i x_i + b = w_1 x_1 + w_2 x_2 + \cdots + b$$

where b is bias term.

After defining an input layer, weights are assigned and these weights would be used to regulate each variable's significance such that the larger weights contribute more to the output than smaller ones. Each value in the input is multiplied by the corresponding weight in the weight matrix. Finally, the total sum is by ‘k’.

$$f(k) = output$$

The activation function of a particular node takes K as input and gives an output which is then passed as input to the next node.

Machine Learning for Image Classification:

Machine learning for image classification makes use of algorithms' ability to learn hidden information from a dataset of organized and disorganized samples.

A Data-Driven Approach to Machine Learning:
1. Gather a picture and label the dataset.
2. Train a classifier using Machine Learning
3. Use new photos to test the classifier.

Linear Classifier:
Based on a linear combination of its explanatory factors, a linear classifier divides a set of data points into discrete classes. A model, for example, might use facts about a horse's weight, height, colour, and other characteristics to determine its species. The efficacy of these models is determined by their ability to uncover a mathematical combination of attributes that classifies data points according to their actual class value, providing unambiguous boundaries for classification.

$$f(x, W) = Wx + b$$

It is the formula for a linear score function.
The input picture data is x, the parameters or weights is W, and the applied bias is b.

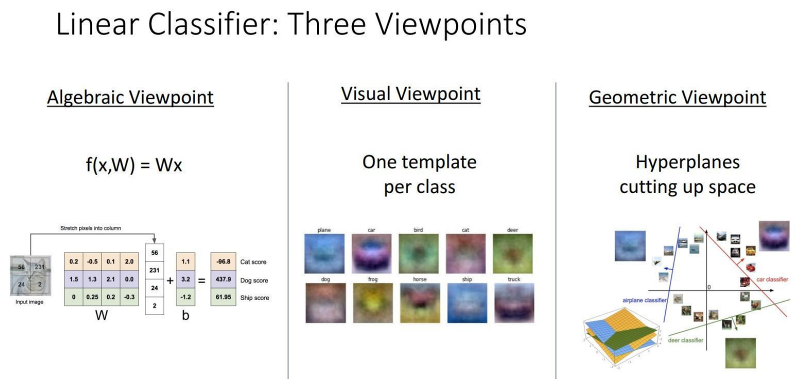


Fig 3 Interpretation of Classifier

The linear classifier has one “template” per category.
However, the issue here is that a single template cannot capture numerous data modalities.

TO so classify a dataset:
1. Use a loss function to quantify how good a value of W is
2. Find a W that minimizes the loss function (optimization)
A classifier will be a good classifier if it has low loss and vice versa.
Loss Functions quantify preferences.
For
x_i *corresponding to image*,
y_i *corresponding to label*
And dataset is

$$\{(x_i, y_i)\}_{i=1}^{N}$$

Loss for single example is given by:

$$L_i(f(x_i, W), y_i)$$

Loss for dataset is average of each of the examples:

$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

For some dataset (x,y) :

We have score function as :-

$$s = f(x; W) = Wx$$

SoftMax Function as :-

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

Full Loss Function as:

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + R(W)$$

Optimization:

Gradient Descent:

It is basically a first-order iterative optimization procedure to locate loss function's local minimum whose output is a partial derivative of a set of input parameters. The higher the slope, the greater the gradient.

Gradient Descent is used to determine the optimal values of the parameters to find the smallest feasible value of the given cost function, starting with an initial value.

The learning rate must be initialized as it controls the size of each iteration's step.

The smaller the α is taken, the larger the convergence time costlier the computation and larger the α is taken, the greater the chance that convergence might even fail and overshoot the minimum.

Types of Gradient Descent:

Mostly, there are three types of Gradient Descent:

1. Batch Gradient Descent
2. Stochastic Gradient Descent
3. Mini-batch Gradient Descent

Mini-batch Gradient Descent: In Mini-batch Gradient Descent we don't go through the complete dataset in one epoch rather we iterate through a particular batch of the data which is selected from the dataset.Suppose the size of the batch is b then, learning occurs as

$$w = w - \alpha \nabla_w J(x^{\{i:i+b\}}, y^{\{i:i+b\}}; w)$$

where w is the weight, α is learning rate, b is the batch size, and J is the loss function

We can avoid the pre-existing order of examples by shuffling the training data set.If the batch size is b then the dataset will have size(dataset)/b batches along with a batch with the remaining data.

We should adjust the batch size and usually pick any power of 2 such as 32, 64 etc because GPUs like hardware get better run time.

Stochastic Gradient Descent (SGD):

The SGD (Stochastic Gradient Descent) adjusts the parameters accordingly on each case of (x^i,y^i) in place of going through all of them which results in learning in every example:

$$w = w - \alpha \nabla_w J(x^i, y^i; w)$$

Stochastic Gradient Descent (SGD):

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(x_i, y_i, W) + \lambda R(W)$$

$$\nabla_W L(W) = \frac{1}{N}\sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

Where L(W) is the loss function, λ is hyperparameter, R(W) regularization.

Adam Optimizer

For the optimization of GD(Gradient Descent) algorithms, the Adaptive Moment Estimation technique would be used. When working with huge problems with a lot of data or parameters, the method is quite efficient. It takes minimal memory. It appears to be a hybrid of the 'gradient descent with momentum and the 'RMSP' algorithms.

What is Adam's approach?

The Adam optimizer employs a hybrid of two gradient descent techniques:

Momentum:

This approach is used to speed up the gradient descent algorithm by considering the gradient's 'exponentially weighted average.' Using averages accelerates the algorithm's convergence to the minima.

$$w_{t+1} = w_t - \alpha m_t$$

$$m_t = \beta m_{t-1} + (1-\beta)\left[\frac{\delta L}{\delta w_t}\right]$$

m_t = aggregate of gradients at time t [current] (initially, m_t = 0)

m_{t-1} = aggregate of gradients at time t-1 [previous]

W_t = weights at time t

W_{t+1}= weights at time t+1

α_t = learning rate at time t

∂L = derivative of Loss Function

∂W_t = weights' derivative at time t

β = Parameter for the moving average(const, 0.9)

Root Mean Square Propagation (RMSP):

$$w_{t+1} = w_t - \frac{\alpha_t}{(v_t+\varepsilon)^{1/2}} * \left[\frac{\delta L}{\delta w_t}\right]$$

$$v_t = \beta v_{t-1} + (1-\beta) * \left[\frac{\delta L}{\delta w_t}\right]^2$$

W_t = weights at time t

W_{t+1} = weights at time t+1

α_t = learning rate at time t

∂L = derivation of loss function

∂W_t = derivation of weights at time t

V_t =sum of square of past gradients. [i.e. sum(∂ L/ ∂W_{t-1})] (initially, V_t= 0)

β = Moving average parameter (const, 0.9)

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\left[\frac{\delta L}{\delta w_t}\right] v_t = \beta_2 v_{t-1} + (1-\beta_2)\left[\frac{\delta L}{\delta w_t}\right]^2$$

ϵ = A small positive constant (10^{-8})

Mathematical Aspect of Adam Optimizer:

Taking the formulas used in the above two methods, we get

Parameters Used :

1. ϵ = a small +ve constant to avoid 'division by 0' error when (vt -> 0). (10^{-8})
2. β1 & β2 = decay rates of an average gradients in the above two methods. (β1 = 0.9 & β2 = 0.999)
3. α — parameter step size/learning rate (0.001)

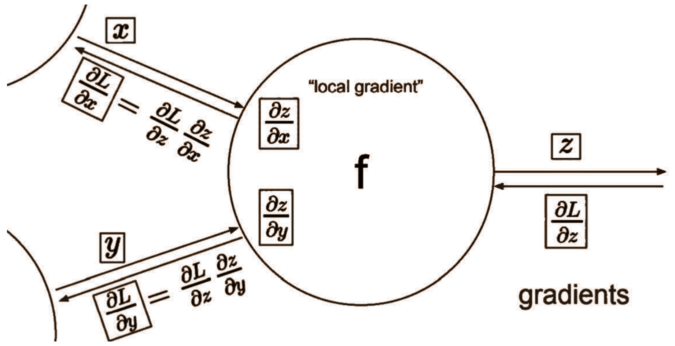
Activation Function:

The Activation Function assists the neural network in utilizing critical data at the same time suppressing irrelevant data and It determines whether or not a neuron should be stimulated. This means that it will use simpler mathematical operations to determine whether the neuron's input to the network is essential or not throughout the prediction phase.

It's task is to generate output from a set of input values that are provided to a node.

Backpropagation:

It is a common type of artificial network training that aids in the calculation of the gradient loss function in accordance with all the weights of the network by the help of the chain rule method, to train a neural network more successfully using the backpropagation algorithm. That is, after each forward run, the backpropagation algorithm does a backward pass over the network by modifying the model's parameters. It is utilised to figure out the loss function gradient with regard to all of the weights of the network.



x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
$\frac{\partial z}{\partial y}$
z
$\frac{\partial L}{\partial z}$
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
gradients

Fig4 Backpropagation

The Forward Pass:

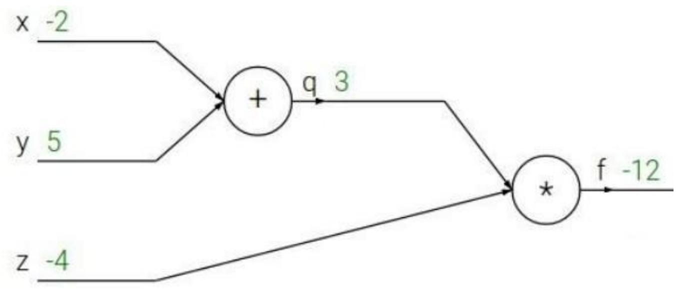
Output of the previous layer is received by the next node in the connection as its input. Input is customized by using actual weights 'W', where the weights are selected randomly. Each node in the hidden layer, Evaluates the output of the previous layer and passes the output to the node on the next layer This operation will end when the output layer is reached.

The Backwards Pass:

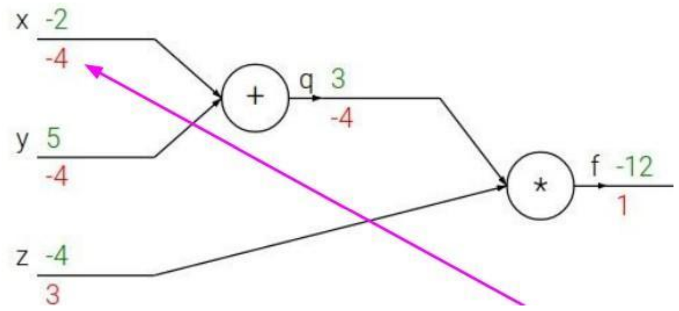
Backpropagation's purpose is to update each of the network's weights in such a way that the actual output is closer to the desired output, thus minimising the error for each output neuron and the network as a whole.

a simple Backpropagation example:

$$f(x, y, z) = (x + y)z$$ for x=-2,y=5,z=-4



Forward Propagation



Backward Propagation

The way neuron i sends output to neuron j is governed by connections, weights, and biases. Propagation adds the predecessor neuron function with the weight and computes the input and output and the learning rule changes the weights and thresholds of the network's variables.

Convolutional Neural Networks (CNN):

CNN is also called ConvNet, is a prevalent DL(deep learning) architecture and is a type of ANN. In comparison to others, it recognises significant properties without the need for human intervention and is computationally efficient. Here parameter sharing is done and special convolution and pooling procedures are used.

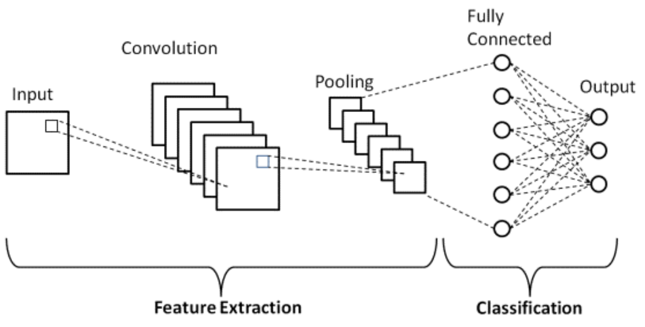
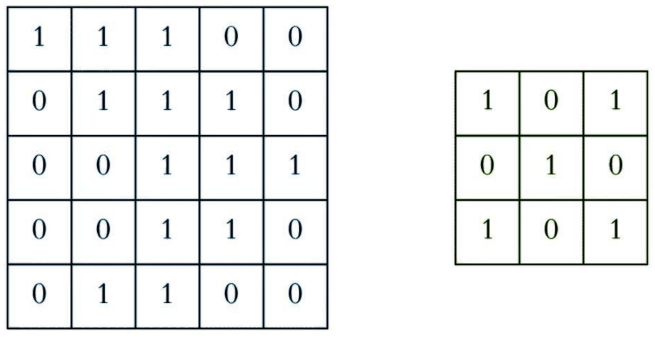


Fig 5 CNN

Layers of CNN:
CNN layers are basically a series of convolution and pooling operations and then at the end comes fully connected layers.

Convolutional layers:

Convolutional layer is the first layer of the sequence whose task is to extract the distinct traits from the input photos. To do the process of convolution between the input image and a filter of a certain size NxN in this layer, take the dot product between the filter matrix and the different portions of the input image by sliding the filter across the input image.



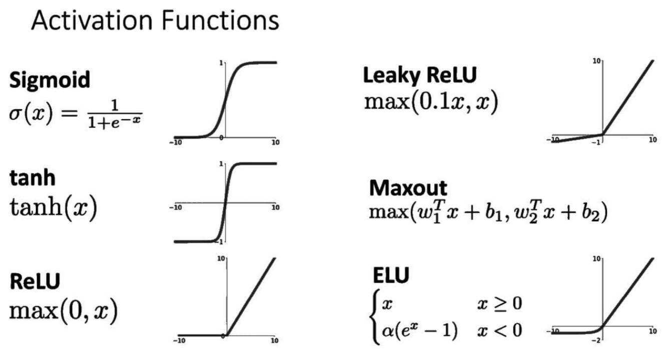
$$\begin{array}{|c|c|c|c|c|}\hline 1&1&1&0&0\\\hline 0&1&1&1&0\\\hline 0&0&1&1&1\\\hline 0&0&1&1&0\\\hline 0&1&1&0&0\\\hline\end{array}\qquad\begin{array}{|c|c|c|}\hline 1&0&1\\\hline 0&1&0\\\hline 1&0&1\\\hline\end{array}$$

Input Filter / Kernel

The result is a Feature map which contains information about the image like its corners and edges, which is then forwarded to further layers to learn about the furthermore types of features from the input image.
Stride decides the step size of the movement of the convolutional filter.
Padding means basically bordering our input with zeros, which is applied in the convolution layer to maintain feature map size.
For m*m input and f*f filter size and stride value equal to 's' and padding value be p,
[{(m + 2p − f + 1) / s} + 1] $*$ [{(m + 2p − f + 1) / s} + 1] will be our output image dimension after applying convolution.
Max Pooling Layer: It reduces the amount of data in each acquired element in the convolutional layer at the same time important data is maintained.

Dropouts: It is a normalization technique for which each cycle, the dropout randomly drops neurons with predefined layers and then doesn't utilize them in any next layer.
Fully connected (FC) layer:Last few layers of CNN architecture is basically the FC layer in which before inserting input, the input needs to be flattened (flattening the 3D matrix into the vector form) . The process of classification begins at this phase.

Activation Function:For any CNN architecture, activation functions are one of the essential parameters which are used to establish any form of complex & continuous connections among the network variables and are helpful in making our network non-linear. Examples of activation functions are Sigmoid, ReLU, tanh,etc and for our model we would be using ReLU (Rectified Linear Activation function).For positive input, ReLU outputs the value equal to input but for negative input it outputs zero value.



ResNet Architecture: Resnet or Residual Networks is a CNN architecture that was introduced by Microsoft Research in 2015. It is based on the fact that if there are too many layers in a CNN , the model tries to overfit the train data and accuracy on the test data is reduced,this may lead to a problem of vanishing gradient.This model tries to solve this problem by using the method of skip connections.The basic approach in this model is we use network fit residual mapping instead of layer learning the underlying mapping.

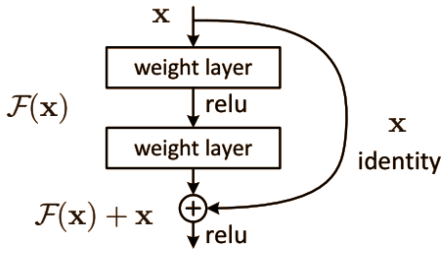

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Fig 6 Skip connection

Resnet50 (Boesch, n.d.) is a version of resnet architecture, that consists of

- 1 layer with 64 different kernels all of size 7 X 7 and size 2.
- Max Pooling layer with stride 2.
- Next we have 3 layers with 1 X 1 type 64 filters,3 X 3 type 64 filters,1 X1 type 256 filters.These 3 layers are repeated thrice so total 9 layers.
- Further we have 3 layers with 1 X 1 type 128 filters,3 X 3 type 128 filters,1 X1 type 512 filters.These 3 layers are repeated four times so total 9 layers.
- After this we have 3 layers with 1 X 1 type 256 filters,3 X 3 type 256 filters,1 X1 type 108 filters.These 3 layers are repeated six times so total 18 layers.
- Next we have 3 layers with 1 X 1 type 512 filters,3 X 3 type 512 filters,1 X1 type 2048 filters.These 3 layers are repeated thrice so total 9 layers.
- Finally we have a fully connected layer with softmax function at last which contains 1000 nodes. Thus doing Average Pool.
- So total we have $1 + 9 + 12 + 18 + 9 + 1 = 50$ Layers Deep CNN.

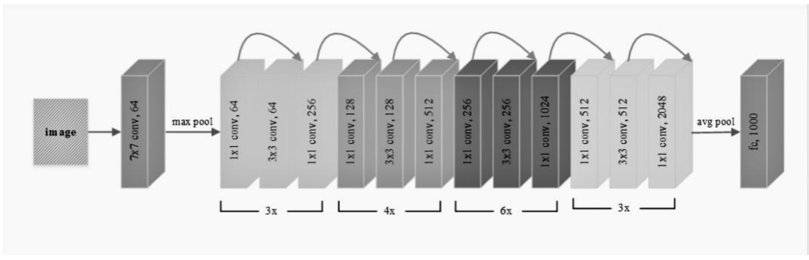
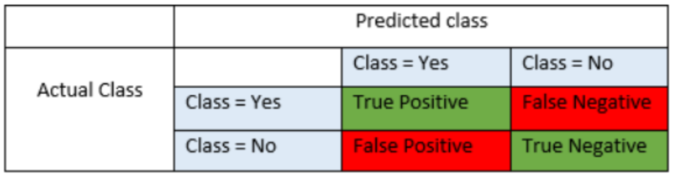


Fig 7 Resnet Architecture

Transfer learning: Transfer learning is basically reusing the learning and insights of previously trained models for weights of hyperparameters in any other associated problem of the project.Its use is so common that mostly user goes for pretrained models like Imagenet for better efficiency.

Validation Techniques:

- **True Positive(TP):** Correctly predicted values that are positive.
- **True Negative(TN):** Correctly predicted values that are negative.
- **False Positive (FP):** Falsely predicted values that are positive.
- **False Negative(NP):** Falsely predicted values that are negative.
- **Confusion matrix:**



	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

The values with the colour green are the values for which our model predicts correctly, so we want a model that has more values in the green colour class.

- **Accuracy:**

$$ACCURACY = \frac{TP + TN}{TP + TN + FP + FN}$$

It is the ratio of total correct predictions to the no of predictions.

- **Precision:**

$$PRECISION = \frac{TP}{TP+FP}$$

It is the ratio of correct positive predictions to the no of positive predictions.

- **Recall:**

$$RECALL = \frac{TP}{TP+FN}$$

Ratio of correctly predicted positive values to actual observations. It has correspondence to true positive rates.

- **F1 score:**

$$F1SCORE = 2 \cdot \frac{RECALL \cdot PRECISION}{RECALL + PRESION}$$

It is the weighted average of recall and precision, It is useful for uneven class distribution.

- **True positive Rate:**

$$TPR = \frac{TP}{TP+FN}$$

- **False positive rate:** Probability of classifying actual negative observation positive.

$$FPR = \frac{FP}{FP+TN}$$

- **ROC curve:** An ROC curve is made when we calculate and plot TPR vs FPR for a variable no of Thresholds. It helps in visualizing how choices affect the Classifier's performance. An ideal classifier thus will be a classifier with 1(100%)true positive rate and 0(0%)false-positive rate.So what we want is our classifier is somewhere close to the ideal classifier.

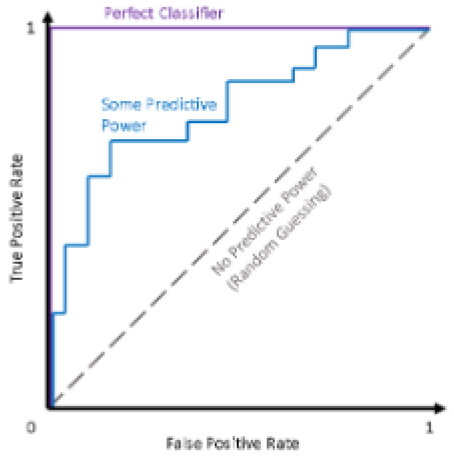


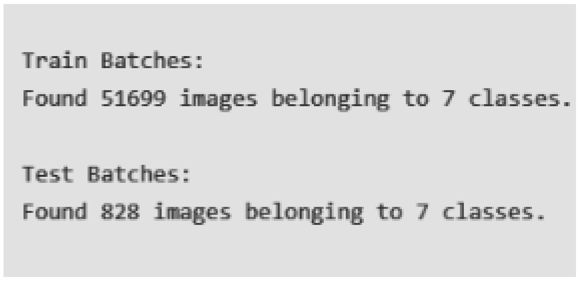
Fig 8 ROC Curve

- **ROC area Under the curve Or ROC score:** It is the measure of closeness of a classifier to an ideal classifier. As the area of an ideal classifier is 1 so area of the ROC curve must be close to 1 for an accurate classifier

Computation:

- Computation is performed on python with libraries that have specific functions for CNN and deep learning.As a high computation power is required,cloud-based services are used to train and test the model.
- At first, the dataset(HAM10000) is being loaded, all the unnecessary information is being removed and all the duplicates are also removed, along with normalization.
- Then the dataset is divided into two parts the test and train dataset.

- Augmentation is performed on the following criteria:
  - Rotation: Range 0° to 180°.
  - Width Shift: Range $0.1X$
  - Height Shift: $0.1X$
  - Zoom $0.1X$
  - Horizontal Flip
  - Vertical Flip
- After Augmentation 8000 image per class is generated. SO finally we have

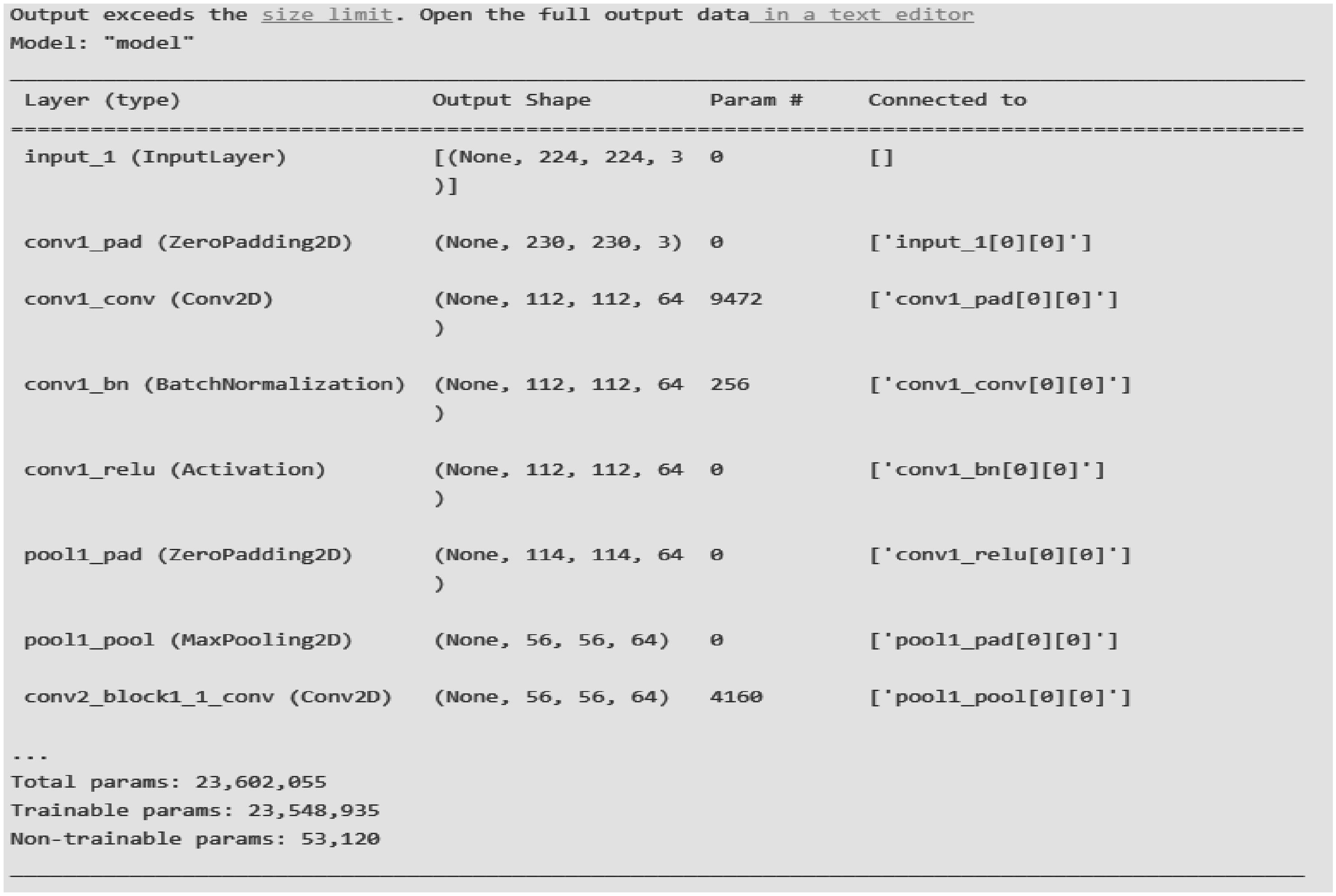


```
Train Batches:
Found 51699 images belonging to 7 classes.

Test Batches:
Found 828 images belonging to 7 classes.
```

- As transfer learning is being performed, data is then loaded through ImageNet with the bottom 3 layers removed, that we integrated through ResNet architecture.

Finally model looks like this



```
Output exceeds the size limit. Open the full output data in a text editor
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 conv1_pad (ZeroPadding2D)      (None, 230, 230, 3)  0           ['input_1[0][0]']

 conv1_conv (Conv2D)            (None, 112, 112, 64  9472        ['conv1_pad[0][0]']
                                )

 conv1_bn (BatchNormalization)  (None, 112, 112, 64  256         ['conv1_conv[0][0]']
                                )

 conv1_relu (Activation)        (None, 112, 112, 64  0           ['conv1_bn[0][0]']
                                )

 pool1_pad (ZeroPadding2D)      (None, 114, 114, 64  0           ['conv1_relu[0][0]']
                                )

 pool1_pool (MaxPooling2D)      (None, 56, 56, 64)   0           ['pool1_pad[0][0]']

 conv2_block1_1_conv (Conv2D)   (None, 56, 56, 64)   4160        ['pool1_pool[0][0]']

...
Total params: 23,602,055
Trainable params: 23,548,935
Non-trainable params: 53,120
__________________________________________________________________________________________________
```

- Hyperparameters are being set as shown below and the models is set to train.
  - Learning Rate:0.01
  - Epsilon:0.1
  - optimiser:Adam
  - Loss: Categorical cross Entropy
  - Metrics: Accuracy
  - Class Weights: { 0: 1.0, # akiec 1: 1.0, # bcc 2: 1.0, # bkl 3: 1.0, # df 4: 5.0, # mel 5: 1.0, # nv 6: 1.0, # vasc }
  - Patience:40
  - Min delta:0.001
  - Epochs:3000

- Steps per Epoch:Length of train data frame/10
- Verbose::2

- Training terminated at:

```
...
Epoch 53/300
918/918 - 102s - loss: 0.0441 - accuracy: 0.9890 - val_loss: 0.5523 - val_accuracy: 0.8841 - 102s/epoch - 111ms/step
Epoch 54/300
918/918 - 103s - loss: 0.0517 - accuracy: 0.9864 - val_loss: 0.4952 - val_accuracy: 0.8973 - 103s/epoch - 112ms/step
```

Results and Discussion:

Validation and accuracy:

- Classification Report:

```
Classification Report:
              precision    recall  f1-score   support

       akiec       0.55      0.52      0.53        23
         bcc       0.71      0.65      0.68        26
         bkl       0.71      0.70      0.70        66
          df       0.62      0.83      0.71         6
         mel       0.58      0.62      0.60        34
          nv       0.96      0.96      0.96       663
        vasc       1.00      0.80      0.89        10

    accuracy                           0.90       828
   macro avg       0.73      0.73      0.73       828
weighted avg       0.90      0.90      0.90       828
```

- Weighted Statistics:

```
Precision: 0.9025777172230262
Recall: 0.9021739130434783
Accuracy: 0.9021739130434783
weighted Roc score: 0.9760559013047851
```

- Macro Statistics:

```
Precision: 0.7327445737220174
Recall: 0.7265468546798469
Accuracy: 0.9021739130434783
Macro Roc score: 0.976046390819996
```

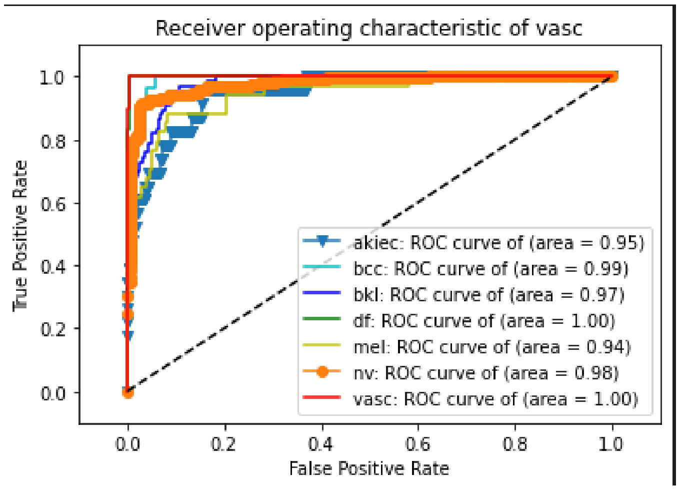
- Micro Statistics:

```
Precision: 0.9021739130434783
Recall: 0.9021739130434783
Accuracy: 0.9021739130434783
Micro Roc score: 0.992505416306876
```

- ROC Area Under curve for Each class:

```
The ROC AUC score of akiec is: 0.9482041587901702
The ROC AUC score of bcc is: 0.9889219259543449
The ROC AUC score of bkl is: 0.9749264296508391
The ROC AUC score of df is: 0.9989862124898621
The ROC AUC score of mel is: 0.9440657875240777
The ROC AUC score of nv is: 0.9777092188856895
The ROC AUC score of vasc is: 0.9995110024449878
```

- ROC characteristic curve:



Findings:

- Resnet Architecture along with transfer learning to predict the type of skin cancer present on the body.An overall accuracy of 90.21% is achieved.
- This method of classification can be applied in various Fields of medical use

Future Scope::

- The proposed model can be advanced with modern techniques like soft attention that may result in better efficiency and accuracy.
- The proposed method can be used in various other tasks of detection and classification used in the medical field. Like covid detection, tumor classification, etc.

References

1. *[1703.04819] RECOD Titans at ISIC Challenge 2017*. (2017, March 14). arXiv. Retrieved May9, 2022, from https://arxiv.org/abs/1703.04819
2. *Applied Deep Learning - Part 4: Convolutional Neural Networks | by Arden Dertat*. (2017, November 8). Towards Data Science. Retrieved May 9, 2022, from https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2
3. Boesch, G. (n.d.). *Deep Residual Networks (ResNet, ResNet50) - Guide in 2022 - viso.ai*. Viso Suite. Retrieved May 9, 2022, from https://viso.ai/deep-learning/resnet-residual-neural-network/
4. *Cancer*. (n.d.). WHO | World Health Organization. Retrieved May 9, 2022, from https://www.who.int/health-topics/cancer#tab=tab_1
5. *Index of /slides/2017*. (n.d.). CS231n. Retrieved May 9, 2022, from http://cs231n.stanford.edu/slides/2017/
6. Niharika Gouda. (n.d.). Skin Cancer Classification using ResNet. https://ieeexplore.ieee.org/document/9250855
7. Noel C. F. Codella. (n.d.). Skin lesion analysis toward melanoma detection: A challenge at the 2017 International symposium on biomedical imaging (ISBI), hosted by the international skin imaging collaboration (ISIC). https://ieeexplore.ieee.org/document/8363547
8. Radhakrishnan, R. (2022, February 23). *What Are the Main Types of Skin Cancer? 7 Types*. MedicineNet. Retrieved May 9, 2022, from

https://www.medicinenet.com/what_are_the_main_types_of_skin_cancer/article.htm

9. Tschandl, P. (n.d.). *The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions*. Harvard Dataverse. Retrieved May 9, 2022, from https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T
10. V. Pomponiu. (n.d.). Deepmole: Deep neural networks for skin mole lesion classification. https://ieeexplore.ieee.org/abstract/document/7532834
11. *What are Neural Networks?* (2020, August 17). IBM. Retrieved May 9, 2022, from https://www.ibm.com/cloud/learn/neural-networks