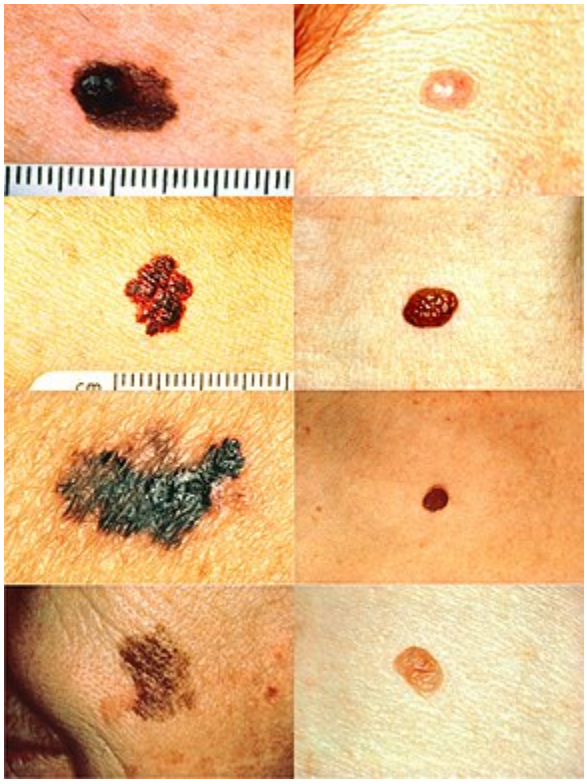
Chapter 1

Introduction

1.1 Motivation

People with melanoma will increase their chances of survival by having a timely diagnosis and treatment. Melanoma is a form of skin cancer that occurs when melanocytes, which produce pigment, mutate and proliferate uncontrollably. The skin is where most pigment cells form. The chest and back are the most common areas affected in men. The most popular site for women is their legs. Melanoma is mostly found on the forehead. Melanoma can, however, develop in the eyes and other parts of the body, including — in sporadic cases — the intestines.



The American Cancer Society estimated melanoma 5-year median survival rates. These charts compare the chances of a person with melanoma living for five years to someone who does not

have cancer. The 5-year relative survival rate for melanoma is 98 percent if a doctor detects and treats it until it spreads. However, if it applies to deeper tissues or neighboring lymph nodes, the survival rate decreases to 64%. If it spreads to distant organs or tissues, the chances of living for five years drop to 23%.

This project deals with the prediction of melanoma from images using computer vision techniques. Through this model, early detection of melanoma will help dermatologists diagnose skin cancer and take the necessary action to cure their patients.

1.2 Approach and Methodology

This work focuses on using deep learning methods and algorithms in order to classify lesion images as benign or malignant. There are two different problems that will be solved within this thesis. First, new images are added from the given dataset to prevent models from overfitting using data augmentation techniques. Secondly, different CNN classification models were trained on the dataset to find the best performing architecture(best validation accuracy) out of all those. The concept of transfer learning was used in this case to train the models.

At the beginning we did data preprocessing which involved using Data Augmentation techniques like Flipping, Shearing, Cropping, Rotation, Translation on the given images. These methods are used to prevent overfitting of the model architectures on the training set. After all the preprocessing is done our model was initialized by adding some layers to the end of an already trained standard CNN model. The model was trained after that on the training dataset, validated on the validation dataset and accuracy was noted. In the training it was ensured that only the last few layers of the model were trained since the backbone of the model was trained already.

Chapter 2

Theoretical Background

2.1 Machine Learning

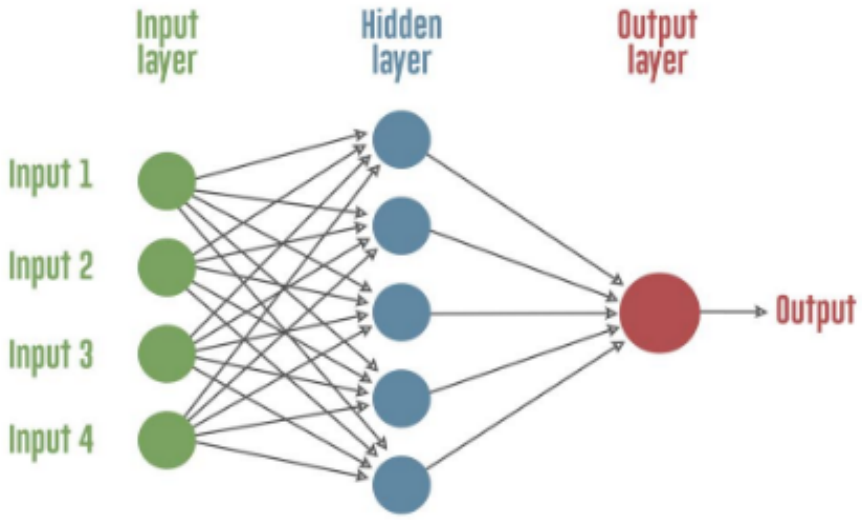
A major difference between humans and computers has been for a long time that human beings tend to automatically improve their way of tackling a problem. Humans learn from the previous and try to solve them by correcting them or looking for new approaches to address the problem. Traditional computer programs do not look at the outcome of their tasks and are therefore unable to improve their behavior. The field of machine learning addresses this exact problem and involves the creation of computer programs that are able to learn and therefore improve their performances by gathering more data and experience. The first scientist to create a self-learning program was A. Samuel in 1952, who created a program that became better at playing the game checkers with the number of games played. In 1967, the first pattern recognition program was able to detect patterns in data by comparing new data to known data and finding similarities between them. Since the 1990's machine learning has been used in data mining areas, adaptive software systems as well as text and image learning fields. As an example: A computer program that gathers data concerning the customers of an e-commerce shop and creates better personalized advertisements out of these pieces of information has the ability to acquire new knowledge and comes close to being artificial intelligence.

2.1.1 Artificial Neural Networks

Singh and Chauhan define Artificial Neural Networks as "a mathematical model that is based on biological neural networks and therefore is an emulation of a biological neural system". Compared to conventional algorithms, neural networks can solve problems that are rather complex, on a substantially easier level in terms of algorithm complexity. Therefore, the main reason to use Artificial Neural Networks is their simple structure and self-organizing nature which allows them to be used in a wide range of problems without any further interference by the programmer. An Artificial Neural Network consists of nodes, also called neurons, weighted connections between these neurons that can be adapted during the learning process of the network and an activation function that defines the output value of each node depending on its

input values. Every neural network consists of different layers. The input layer receives information from external sources, such as attribute values of the corresponding data entry, the output layer produces the output of the network and hidden layers connect the input and the output layer with one another. The input value of each node in every layer is calculated by the sum of all incoming nodes multiplied with the respective weight of the interconnection between the nodes.

The output value of each node is calculated by using all input values on a predefined function that is the same for every node in the network. This function is commonly referred to as an activation function.



In a supervised learning scenario, Artificial Neural Networks can be trained using the backpropagation algorithm, which readjusts the weights of the interconnections in the neural network based on local error rates.

- **Backpropagation:** Backpropagation is a gradient descent-based supervised learning algorithm for artificial neural networks. The method calculates the gradient of the error function with respect to the neural network's weights given an artificial neural network and an error function. The gradient is determined in reverse order across the network, with the gradient of the final layer of weights calculated first and the gradient of the first layer of weights calculated last. Parts of the gradient computation from one layer are

reused in the gradient computation for the previous layer. This backwards flow of error information enables efficient gradient computation at each layer. To train a neural network with gradient descent the calculation of the gradient of the error function E(X,Θ) with respect to the weights w^k_{ij} and bias b^k_i. Then, according to the learning rate α each iteration of gradient descent updates the weights and biases (collectively denoted Θ) according to

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial E(X, \theta^t)}{\partial \theta}$$

where Θ^t denotes the parameters of the neural network at iteration *t* in gradient descent.

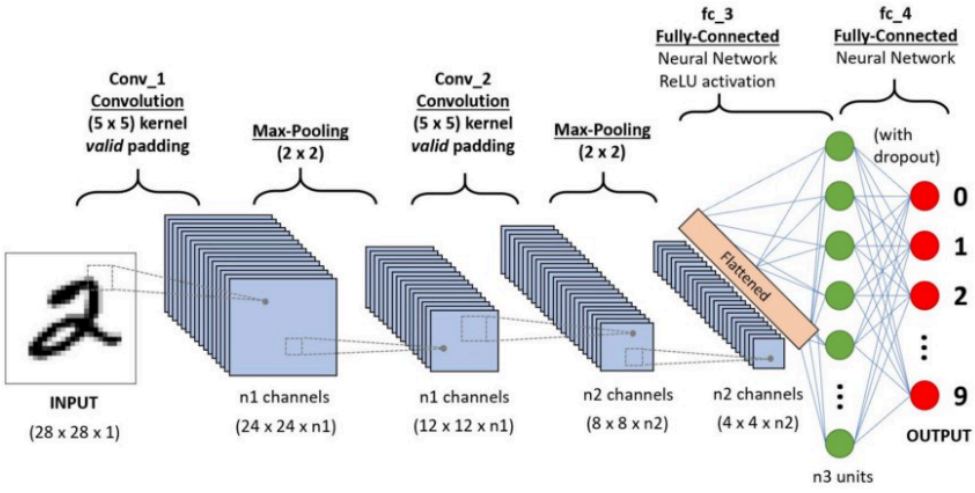
2.2 Image Classification

Image classification is the process of a computer analysing an image and determining which "class" it belongs to. (Or a chance that the picture belongs to a 'class.') A class is simply a name, such as "car," "animal," "building," and so on. The process of a computer analysing an image and telling you it's a sheep is known as image classification. Classifying pictures isn't a big deal for us. When it comes to robots, though, it's a great example of Moravec's paradox. (In other words, things that we find simple are difficult for AI.) Raw pixel data was used to classify images in the beginning. Computers can decompose images into individual pixels as a result of this. The issue is that two images of the same object may appear to be completely different. They may have a variety of backgrounds, angles, poses, and so on. Computers found it difficult to correctly 'see' and categorise images as a result of this.

Convolutional neural networks, or CNNs, are commonly used in deep learning image classification. The performance of the nodes in the hidden layers of CNNs is not always shared with any node in the next layer (known as convolutional layers). Machines can recognise and extract features from images using deep learning. This means they can learn what to look for in photographs by analysing a large number of them.

2.2.1 Convolutional Neural Network

CNN is a Deep Learning algorithm that can take an image as input, assign importance (weights and biases) to various aspects in the image, and distinguish between them. As compared to other classification algorithms, the amount of pre-processing needed by a ConvNet is significantly less. Although primitive methods require hand-engineering of filters, ConvNets can learn these filters/characteristics with enough preparation. The architecture of a ConvNet is inspired by the organisation of the Visual Cortex and is similar to the communication pattern of Neurons in the Human Brain.



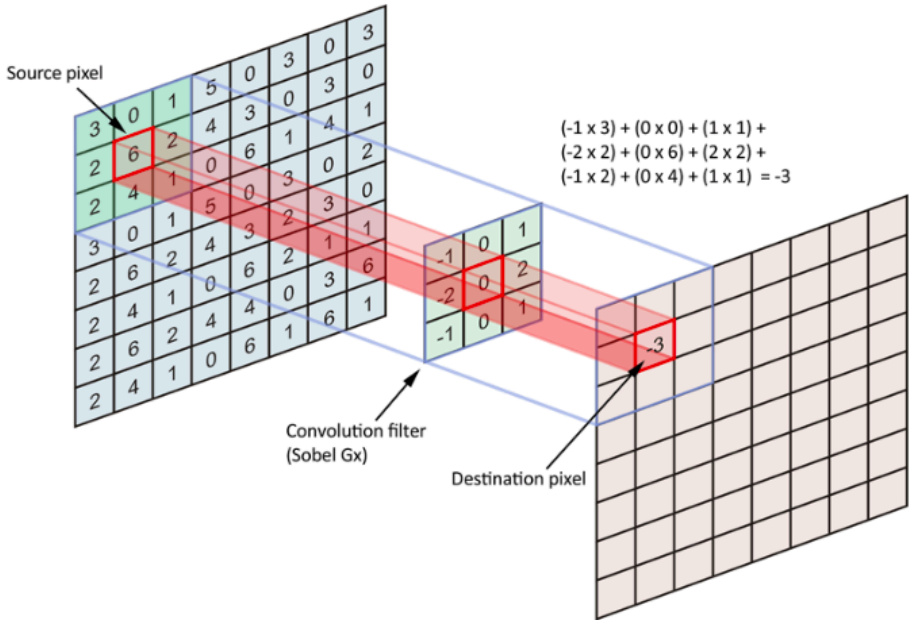
A CNN sequence to classify handwritten digits

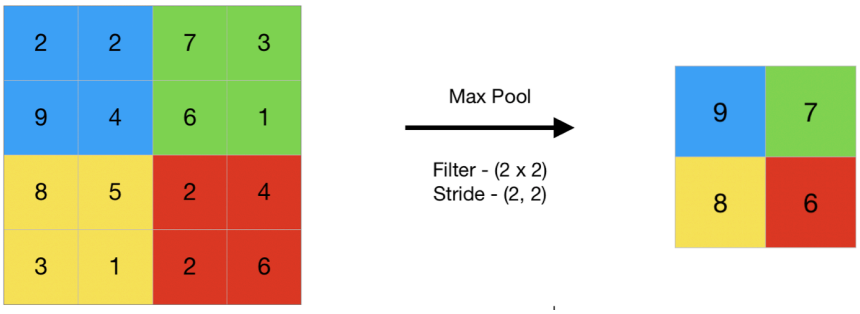
Through the application of relevant filters, a ConvNet can successfully capture the Spatial and Temporal dependencies in an image. Owing to the reduced number of parameters involved and the reusability of weights, the architecture performs better fitting to the image dataset. In other words, the network can be conditioned to better understand the image's sophistication.

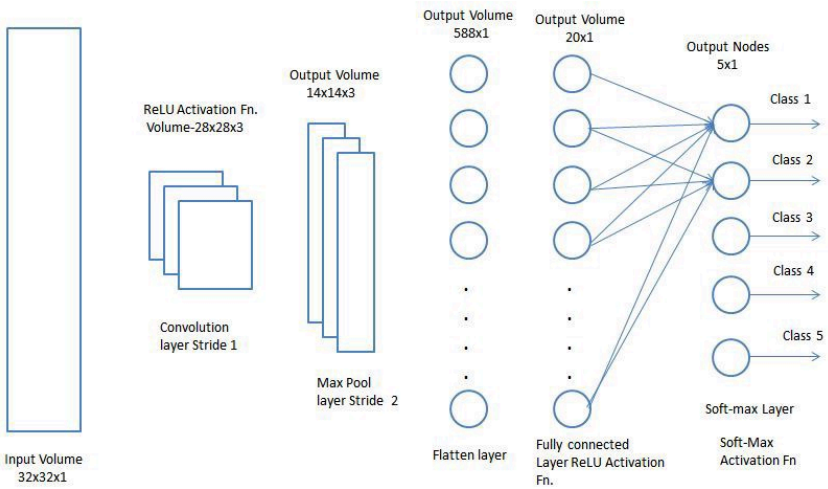
- **Convolution Layer - The Kernel:** Convolution is one of the main building blocks of a CNN. Convolution is performed on the input data using a filter or kernel to generate a feature map in the case of a CNN. By sliding the filter over the data, we perform a

convolution. A matrix multiplication is performed at each position, and the result is added to the function map. The Convolution Operation's goal is to extract high-level features from the input image, such as edges. There is no need to restrict ConvNets to only one Convolutional Layer. The first ConvLayer is traditionally responsible for capturing Low-Level features such as edges, colour, gradient orientation, and so on. The architecture works well with High-Level features as well as the added layers, giving us a network that has a complete understanding of the images in the dataset.

- **Pooling Layer:** The Pooling layer is in charge of shrinking the convolved feature's spatial size. Via dimensionality reduction, the computing power needed to process the data is reduced. It's also useful for extracting rotational and positional invariant dominant characteristics, which helps keep the model's training phase running smoothly. Pooling can be divided into two types: max pooling and average pooling. The max value from the portion of the image protected by the Kernel is returned by Max Pooling. Average Pooling, on the other hand, returns the average of all the values from the Kernel's portion of the image.
- **Fully Connected Layer (FC Layer):** Adding a Fully-Connected layer is a (typically) low-cost way of learning non-linear combinations of high-level features represented by the convolutional layer's performance. In that space, the Fully-Connected layer is learning a possibly non-linear function. We'll flatten the image into a column vector now that we've transformed it into a format appropriate for our Multi-Level Perceptron.

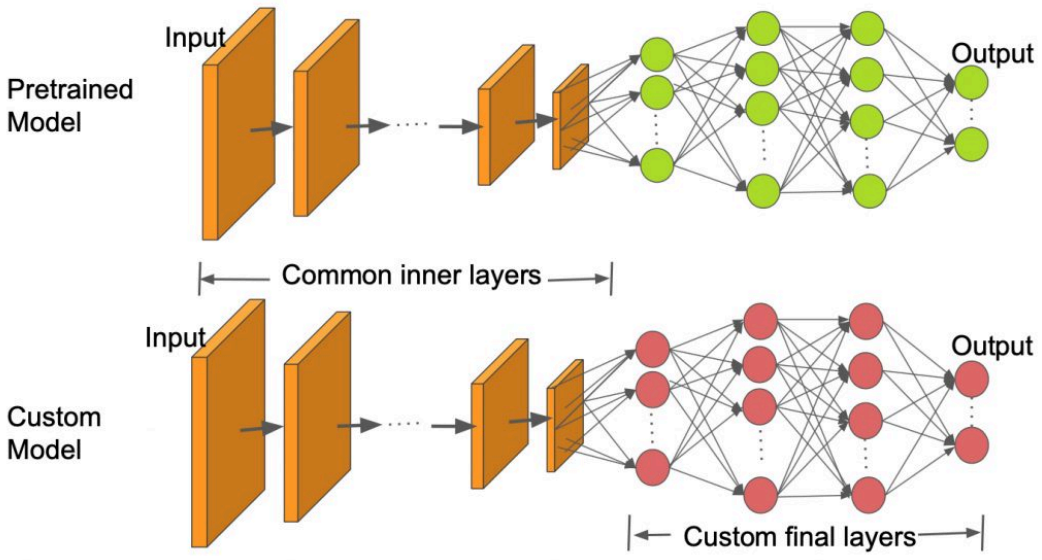

Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel


2 2 7 3
9 4 6 1
8 5 2 4
3 1 2 6
Max Pool
Filter - (2 x 2)
Stride - (2, 2)
9 7
8 6


Input Volume
32x32x1
ReLU Activation Fn.
Volume-28x28x3
Convolution
layer Stride 1
Output Volume
14x14x3
Max Pool
layer Stride 2
Output Volume
588x1
Flatten layer
Output Volume
20x1
Fully connected
Layer ReLU Activation
Fn.
Output Nodes
5x1
Class 1
Class 2
Class 3
Class 4
Class 5
Soft-max Layer
Soft-Max
Activation Fn

2.2.2 Transfer Learning

Transfer learning is a deep learning technique in which a model created for one task is used as the basis for a model on a different task. Given the vast compute and time resources needed to build neural network models on these problems, as well as the huge leaps in ability that they provide on related problems, it is a common approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks. The basic principle of transfer learning is straightforward: take a model that has been trained on a large dataset and apply it to a smaller dataset. The early convolutional layers of the network are freezed and the last few layers are trained which makes a prediction. The concept is that the convolutional layers extract general, low-level features that apply across images — such as edges, patterns, and gradients — while the later layers define unique features within a picture, such as eyes or wheels.



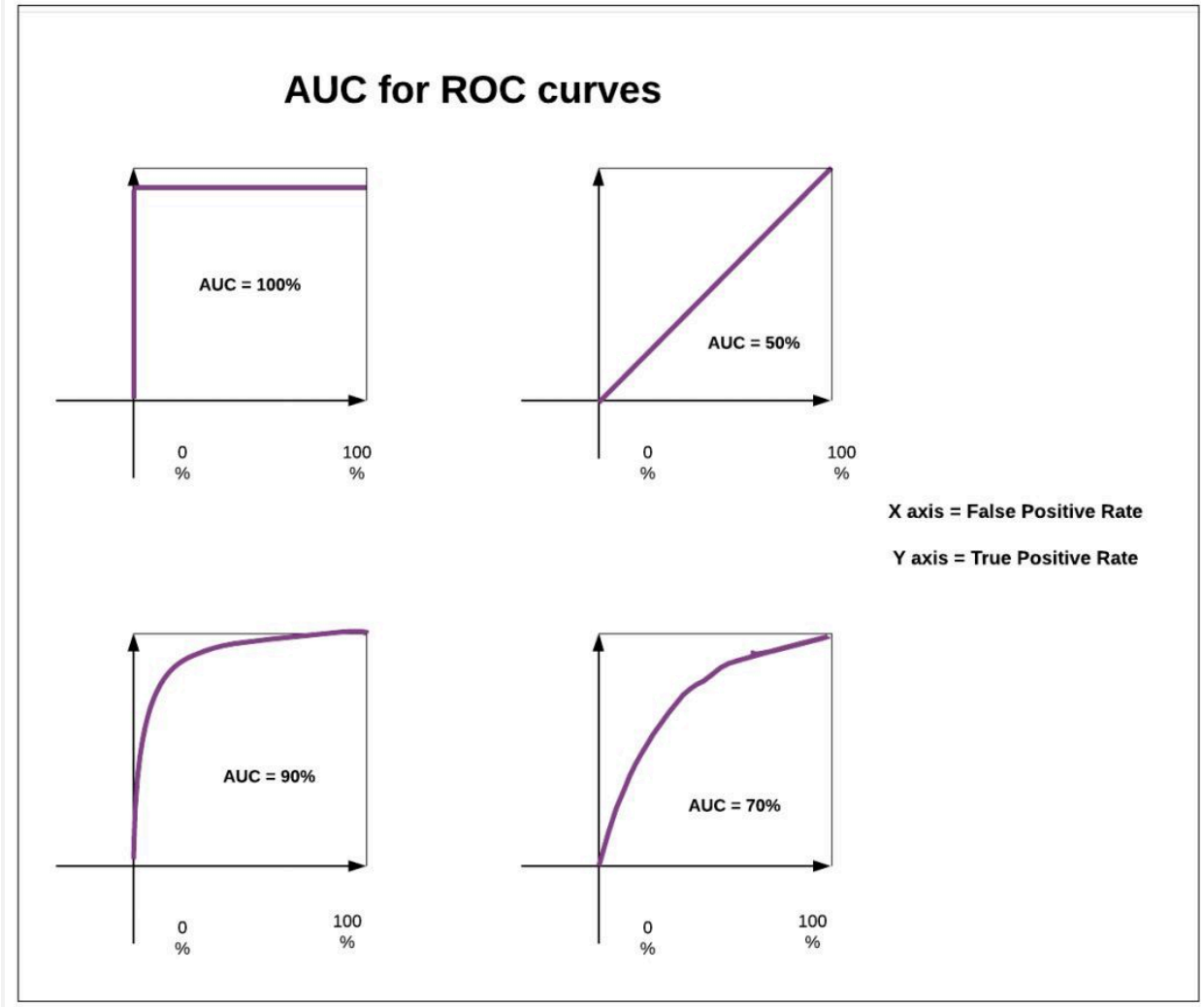
2.3 Evaluation of Image Classification

There are numerous ways of evaluating the performance of a given model. Models are trained on the training set and evaluated on the validation set. Some commonly used evaluation metrics are discussed below and their formulas for binary classification is discussed:

- **Accuracy** - Accuracy is the quintessential classification metric. The proportion of true findings among the total number of cases tested is known as accuracy. For classification problems that are well balanced and not biased or have no class imbalance, accuracy is a valid option of evaluation.
  `Accuracy = (TP+TN)/(TP+FP+FN+TN)`

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

- **Precision** - It calculates what proportion of predicted Positives is truly Positive. When we want to be absolutely certain of our forecast, precision is a good metric to use. `Precision = (TP)/(TP+FP)`
- **Recall** - It calculates what proportion of actual Positives is correctly classified. When we want to catch as many positives as possible, recall is a good choice of measurement measure. `Recall = (TP)/(TP+FN)`
- **F1 Score** - The F1 score is a number between 0 and 1 and is the harmonic mean of precision and recall. The F1 score strikes a balance between precision and recall.
- **AUC** - It is the area under the ROC curve. It indicates how the probabilities from the positive classes are separated from the negative classes. The ROC curve is the graph of TPR(True Positive Rate) and FPR(False Positive Rate). `TPR = TP/(TP+FN)` `FPR = FP/(TN+FP)`


AUC for ROC curves
AUC = 100%
0
%
100
%
AUC = 50%
0
%
100
%
X axis = False Positive Rate
Y axis = True Positive Rate
AUC = 90%
0
%
100
%
AUC = 70%
0
%
100
%

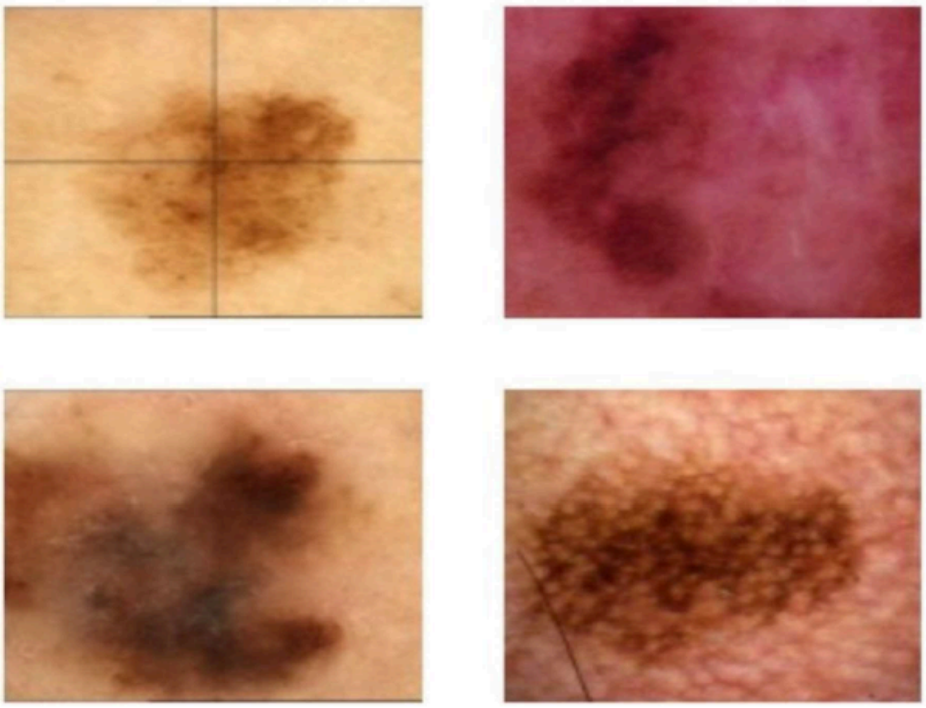
Chapter 3

Methodology

3.1 Dataset description

The images that are used for the given problem are provided in tfrecord format. These TFRecords contain both the image data and tabular data (meta data).The images are 256x256x3 jpegs. TFRecords is Tensorflow's own binary storage format. Since the dataset of images is huge, storing the data in binary format can have a significant impact on the performance of the import pipeline and as a consequence the training time of the model. Binary data takes up less space on disk, takes less time to copy and can be read much more efficiently from disk. This is especially true if your data is stored on spinning disks, due to the much lower read/write performance in comparison with SSDs. The dataset used is open source and is made available on the kaggle platform. The train tfrecord info is given as follows:-

```
feature = {
      'image': _bytes_feature,
      'image_name': _bytes_feature,
      'patient_id': _int64_feature,
      'sex': _int64_feature,
      'age_approx': _int64_feature,
      'anatom_site_general_challenge': _int64_feature,
      'diagnosis': _int64_feature,
      'target': _int64_feature,
      'width': _int64_feature,
      'height': _int64_feature }
```

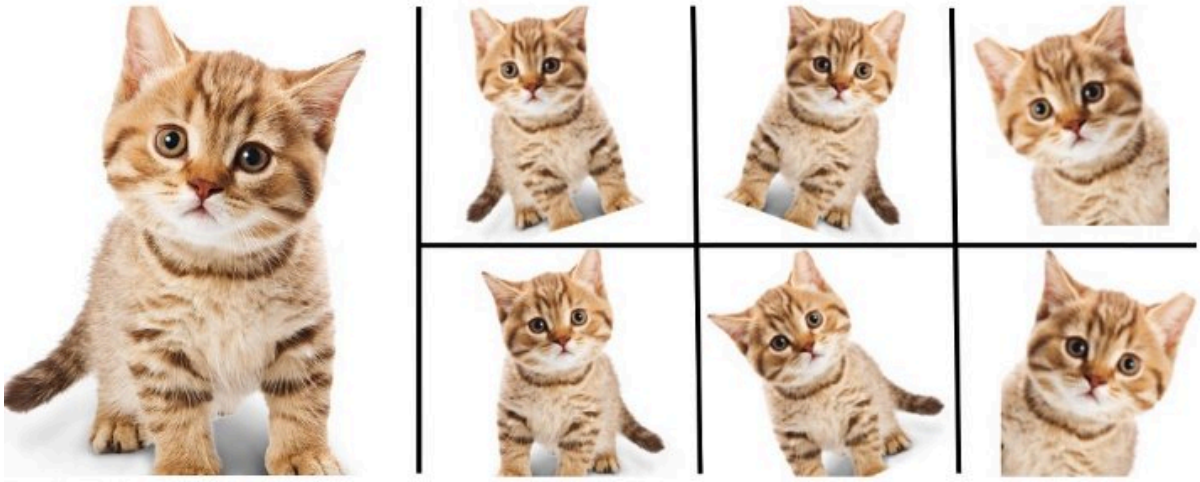


3.2 Data Augmentation

This is an important step which involves data preprocessing. Data augmentation has mainly two advantages. First it adds new images to the already existing dataset and hence increases data size which greatly helps the model in generalizing itself. Secondly it prevents the model from overfitting on the training dataset. The model memorizes the full dataset instead of only learning the main concepts underlying the problem. To summarize, if our model is overfitting, it will not know how to generalize and, therefore, will be less efficient. In this case the data augmentation techniques used are given below:

- **Flipping** : Flipping the horizontal axis is much more common than flipping the vertical axis.
- **Shearing** : like a parallelogram, moves one portion of the image
- **Cropping** : By cropping a central patch of each image, cropping images can be used as a processing step for image data with mixed height and width measurements.
- **Rotation** : The picture is rotated right or left on an axis between 1° and 359° for rotation augmentations. The rotation degree parameter has a significant impact on the protection of rotation augmentations.

- **Translation** : To avoid positional bias in data, shifting images left, right, up, or down can be a very useful transformation.



In our cases we do data augmentations in a random way so that the variance is removed as much as possible. So a particular range through which each image is rotated randomly, or flipping is done randomly on an image.

3.3 Image Classification Model Architecture - Densenet

There are a lot of good models for the purpose of image classification. In this case Densenet architecture has been used. It gives the best result out of all the others tested in the dataset. All previous layers' feature maps are used as inputs in this model, and its own feature maps are used as inputs into all subsequent layers. DenseNets have a number of compelling advantages, including the elimination of the vanishing-gradient problem, improved feature propagation, feature reuse, and a significant reduction in the number of parameters. The DenseNet architecture clearly distinguishes between data that is applied to the network and data that is saved. DenseNet layers are very narrow (e.g., 12 feature-maps per layer), adding only a limited number of feature-maps to the network's "collective information" and leaving the remaining feature-maps unchanged — and the final classifier makes a decision based on the entire network's feature-maps. Aside from improved parameter performance, DenseNets have a better flow of information and gradients across the network, making them easy to practise. Each layer has direct access to the loss function gradients and the original input signal, resulting in implicit deep supervision. This aids in the training of more complex network architectures. On tasks with

smaller training set sizes, dense connections have a regularising effect, which reduces overfitting.

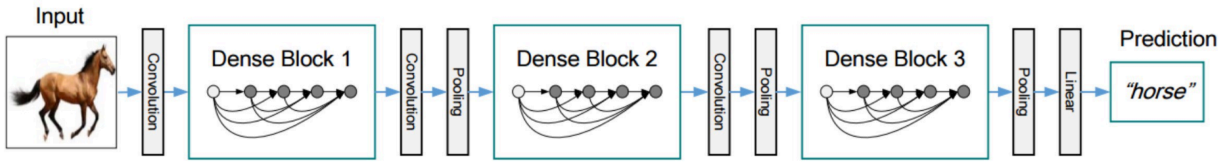


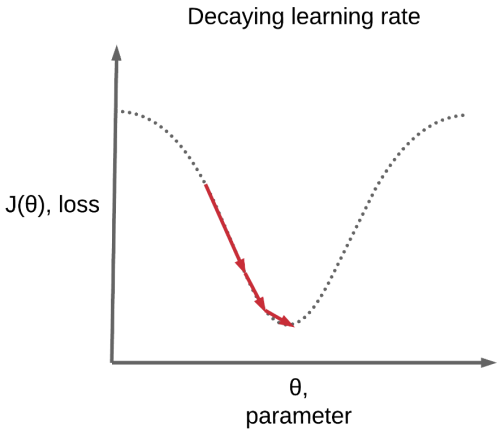
Figure 2. A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature map sizes via convolution and pooling.

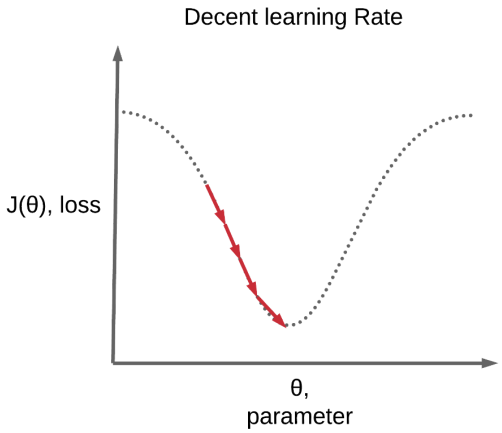
Layers	Output Size	DenseNet-121($k = 32$)	DenseNet-169($k = 32$)	DenseNet-201($k = 32$)	DenseNet-161($k = 48$)
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 36
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 24
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Table 1. DenseNet architectures for ImageNet. The growth rate for the first 3 networks is $k = 32$, and $k = 48$ for DenseNet-161. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.

3.5 Learning Rate Scheduler:

The learning rate schedule (or learning rate decay) describes how the learning rate varies over time (training epochs). The simplest scheduler is used in this dilemma, which reduces the learning rate as the number of training steps increases. This allows for large weight changes at the start of the learning process and minor weight changes or fine-tuning at the end.


Decaying learning rate
J(θ), loss
θ,
parameter

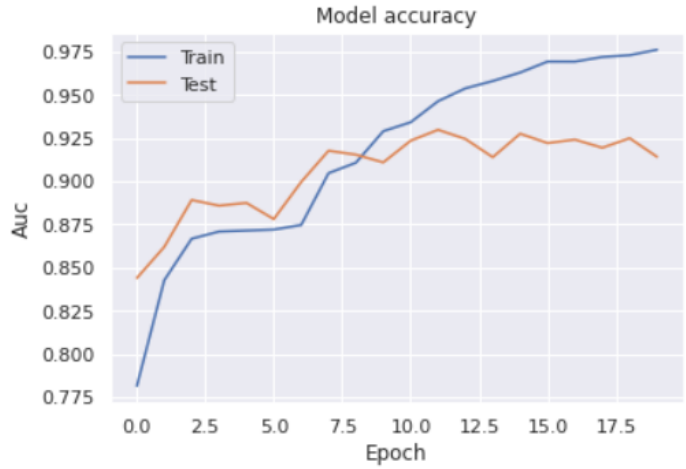

Decent learning Rate
J(θ), loss
θ,
parameter

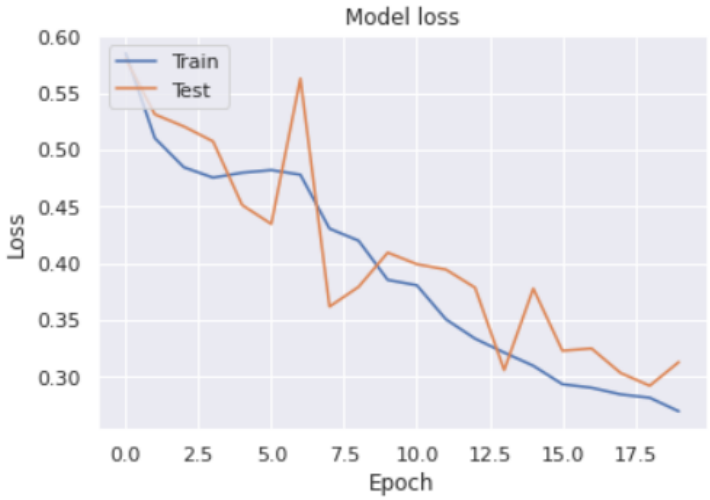
Chapter 4

Results and Conclusion

4.1 Model performance:-

There are plenty of good metrics for image classification. In this case AUC under the ROC curve has been used to evaluate the performance of the model in the validation dataset. AUC is preferred over accuracy for binary classification. The implicit goal of AUC is to deal with situations where you have a very skewed sample distribution, and don't want to overfit to a single class. The model was run for 20 epochs and the final AUC on the training set came out to be 0.976 while the AUC on the validation came out as 0.9141.



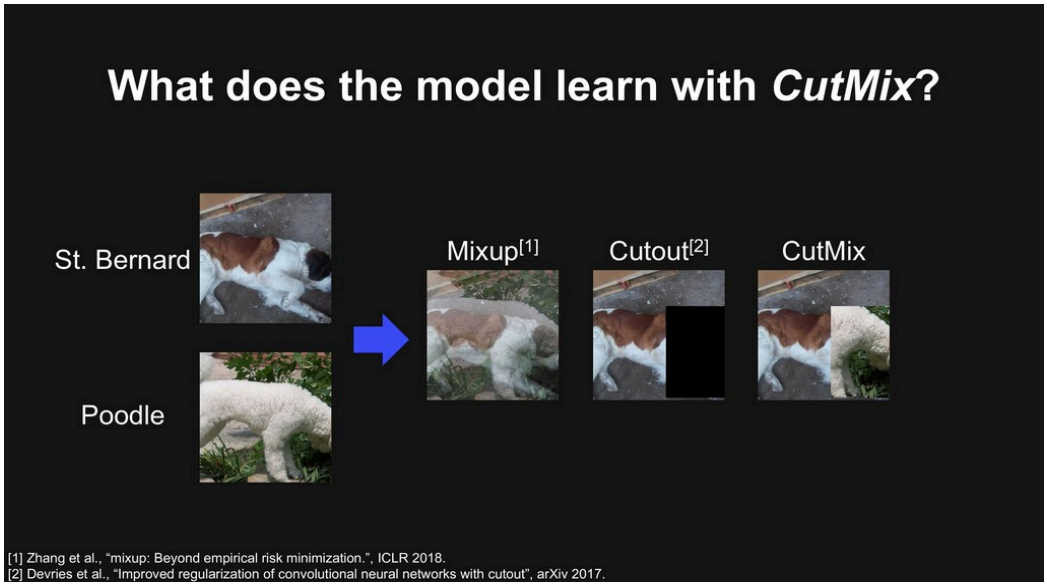


4.2 Conclusion:-

This project chose to research how to apply CNN-based classification to melanoma lesion image dataset and assess their efficiency due to the value of medical image classification and the specific challenge of the medical image-small dataset. Of all the approaches, CNN-based transfer learning is the most efficient. The best results come from the transfer learning of densenet with one retrained ConvLayer. CNN-based approaches are better than conventional methods because they can learn and pick features automatically and easily. The basic function will learn from the new dataset using the unfrozen ConvLayer. It's crucial to strike a balance between a model's expressive capacity and overfitting. A network that is too easy cannot learn enough from the data and therefore cannot achieve high accuracy. A very complex network, on the other hand, is difficult to train and easily overfits. As a consequence, accuracy remains an issue. Only a network model with the appropriate size and other successful overfit prevention strategies, such as a proper dropout rate and proper data augmentation, will produce the best results. Visualization must be introduced to increase the interpretation and clarification of the CNN-based system's effects, as these are needed for the system's implementation in real-world clinical applications.

4.3 Future Work:-

The CNN model can be improved in several ways. New innovative data augmentation techniques have come up like CutMix, MixUp, GridMask, CutOut augmentations which greatly improves the performance of the model with new images and reduces overfitting to a large extent. Also new CNN image classification models have come up like EfficientNet, Xception, NFnet which have been found to perform well in the imagenet dataset. Hence using this model can improve the accuracy of our model and give better results.



4.4 Reference:-

- M.D.Zeiler, R. Fergus, "Visualizing and understanding convolution neural network ", ECCV, 2014.
- J. Bergstra, R. Bardenet, Y. Bengio, B. Kegl, "Algorithms for hyperparameters optimization", Proc. Adv. Neural Inf. Process. Syst., pp. 2546-2554, 2011.
- M. A. Albahar, "Skin lesion classification using convolutional neural network with novel regularizer," IEEE Access, vol. 7, pp. 38306–38313, 2019
- S. Sabbaghi, M. Aldeen, and R. Garnavi, "A deep bag-of-features model for the classification of melanomas in dermoscopy images," in Proc. 38th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC), Aug. 2016, pp. 1369–1372.
- S. Mukherjee, A. Adhikari, and M. Roy, "Malignant melanoma classification using cross-platform dataset with deep learning CNN architecture," in Recent Trends in Signal and Image Processing. Singapore: Springer, 2019, pp. 31–41.
- T. J. Brinker, A. Hekler, J. S. Utikal, N. Grabe, D. Schadendorf, J. Klode, C. Berking, T. Steeb, A. H. Enk, and C. von Kalle, "Skin cancer classification using convolutional neural networks: Systematic review," J. Med. Internet Res., vol. 20, no. 10, Oct. 2018, Art. no. e11936
- D. S. Char, N. H. Shah, and D. Magnus, "Implementing machine learning in health care—Addressing ethical challenges," New England J. Med., vol. 378, no. 11, pp. 981–983, Mar. 2018