B. TECH.
PROJECT REPORT
On
Leukemia Classification of Microscopic Images Using Deep Learning Techniques

BY

Avinash Kumar

Pratik Ahuja

Preyaa Patel

DISCIPLINE OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

DECEMBER 2019

Leukemia Classification Of Microscopic Images Using Deep Learning Techniques

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degrees

of
BACHELOR OF TECHNOLOGY
in
ELECTRICAL ENGINEERING

Submitted by:
Avinash Kumar, Pratik Ahuja, Preyaa Patel

Guided by:
Professor Vimal Bhatia

INDIAN INSTITUTE OF TECHNOLOGY
INDORE
December 2019

CANDIDATE'S DECLARATION

We hereby declare that the project entitled "**Leukemia Detection of Microscopic Images Using Deep Learning Techniques"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in Electrical Engineering completed under the supervision of **Professor Vimal Bhatia,** IIT Indore is an authentic work.

Further, we declare that we have not submitted this work for the award of any other degree elsewhere.

Signature and name of the students with date

Avinash Kumar, Pratik Ahuja, Preyaa Patel

CERTIFICATE by BTP Guide

It is certified that the above statement made by the students is correct to the best of my/our knowledge.

Signature of BTP Guide(s) with dates and their designation

Prof. Vimal Bhatia, Professor, IIT Indore

Preface

This report on "Leukemia Detection of Microscopic Images Using Deep Learning Techniques" is prepared under the guidance of Professor Vimal Bhatia.

Through this thesis, we have tried to give a research-based approach to an existing problem that has been solved using a crude competitive approach by now in other works, ensembling many models. We have worked on making individual models strong enough to perform well in the task at hand.

Avinash Kumar
Pratik Ahuja
Preyaa Patel
B.Tech. IV Year
Discipline of Electrical Engineering
IIT Indore

Acknowledgments

We wish to thank Prof. Vimal Bhatia for providing his support and invaluable guidance over our work on the project. Prof. Vimal's optimism and cheerfulness kept us striving to work hard. His confidence in us was a driving force for our interest in research. We are immensely grateful for their direction and willingness to spend extra time with us over discussions and meetings.

Our work could not have been possible without the external support provided by Prof. Anubha Gupta, Professor, IIIT Delhi for aiding in the procurement of the dataset. We are profoundly grateful for her help.

I would like to wholeheartedly thank my project partners for the successful completion of the project on time.

We are really grateful to the Institute for providing this opportunity to be exposed to research in the healthcare domain and providing the necessary utilities to complete this project.

Sincere thanks to everyone who helped us complete this project.

Avinash Kumar
Pratik Ahuja
Preyaa Patel

B.Tech. IV Year
Discipline of Electrical Engineering
IIT Indore

Abstract

Acute lymphoblastic leukemia (ALL) is fast-growing leukemia of lymphatic tissues that forms the immune system. This makes it clear that it needs immediate attention. India witnesses 10 lakh new cancer cases every year. Of this, around 30000 are cases of acute leukemia, which is more common among children and young adults. Toxic deaths in ALL were also significant and ranged from 2%-13% during induction and 4%-24% during treatment. The 13% induction mortality was seen in ALL, is over ten times the rate in better-resourced countries. This calls for attention in the direction of leukemia detection in India. What is better than to use a new emerging tool, Deep Learning (DL) to bring about a solution to the problem. While we have reviewed other works in which the detection of ALL using DL has been done using ensembling by most people, we propose to improve the performance of stand-alone models using deep learning techniques which makes the solution more general, reliable and hence scalable. We used a skewed dataset of 10,000 images from IIIT Delhi, in which the number of cancerous data samples was double that of non-cancerous samples. This was to take care of the error cases. After experimenting with different combinations of architectures, optimizers and using various techniques, we found that resnet50 being a medium-deep network has given us the best results when loss penalization was done and optimum hyperparameters found using bayesian optimization algorithm. Our aim was to lower the false negatives to a minimum, (i.e. predicting healthy when cell is cancerous). Thus increasing its sensitivity/recall and F1 score and in return bettering the performance of the model.

TABLE OF CONTENTS

List Of Figures

List Of Tables

1

Introduction

1.1 Motivation

Out of the types of leukemia, we decided to work on Acute Lymphocytic/Lymphoblastic Leukemia because of two reasons as follows:

- Age group: Cancer has become the second leading cause of death globally and was responsible for an estimated 9.6 million deaths in 2018. India witnesses 10 lakh new cancer cases every year. Of this, around 30000 are cases of acute leukemia, more common in children.
- Deaths due to delay in diagnosis: Toxic deaths were significant, ranged from 2%-13% during induction, 4%-24% during treatment. This shows that by the time the disease was detected, cancer had reached a higher stage. The induction mortality, is over ten times the rate in better-resourced countries. This calls for attention in the direction of fast ALL detection in India.

1.2 Outline of the Report

Before the explanations, here is some information about what each chapter has. In chapter 2, we talk about the theory related to the project that we referred with the help of which we explain the results that were obtained during experiments. In chapter 3, preprocessing before training and the description of the dataset that we used is mentioned. In chapter 4, all the explanations of the results obtained, tools used, and the performance with different combinations of architectures, optimizers, etc. are mentioned which greatly sums up the project.

2

Literature Review

This chapter gives an overview of what is required to know before understanding the experimentation and results.

2.1 Acute Lymphoblastic Leukemia

The World Health Organisation defines cancer as the uncontrolled growth and spread of cells in the body. It can occur in any part of the body and can also be treated if detected well in advance. Leukemia is a cancer of the blood or bone marrow, one of the major types of cancer. Its symptoms are not very obviously out of the ordinary. Also, tumors are not formed in case of leukemia, unlike other types of cancers which make it even more difficult to detect. Leukemia is classified on the basis of two main factors:

- How fast it progresses:
    - Acute: Fast
    - Chronic: Slow
- Which White Blood Cell (WBC) it affects:
    - Lymphocytic: Attacks the lymphatic tissues that make up the immune system
    - Myelocytic: Attacks the RBC, WBC, and platelet forming tissues

Based on these broad classifications, there are four types of leukemia:

- Acute Lymphocytic/Lymphoblastic Leukemia (ALL)
- Acute Myelocytic Leukemia (AML)
- Chronic Lymphocytic Leukemia (CLL)
- Chronic Myelocytic Leukemia (CML)

2.2 Modern Solution in Healthcare

Conventionally, ALL is diagnosed via blood tests, bone marrow biopsy, etc. There are several other biological and chemical tests like cell flow cytometry that are conducted for detection. Of all those methods, detecting via microscopic images is the best option considering various factors like time delay, cost, etc. This can be looked at as a supervised learning problem where we feed our model, segmented microscopic images of cells from blood smear, as input and finally classify whether the patient has leukemia or not. And what is better than to deploy a deep learning model for the same which learns in a structural and sequential manner very similar to the human brain. Deep Learning solutions have proven to show very good performances in cases of medical imaging, not only limited to any particular disease.

2.3 Deep Learning

2.3.1 Advantages over traditional methods

Deep learning is a class of machine learning algorithms that uses multiple layers to progressively extract higher-level features from the raw input. So one advantage is that it learns in a sequential, structural and hierarchical manner. It also performs well when the dataset is large. This was necessary for us as this way we could make it more adaptable when we wish to train it on the data of new patients and keep it a continuously evolving process. Other than that, one big advantage is that deep learning models learn necessary features to differentiate between various classes so we have to worry less about feature engineering and can focus on bettering the performance using further tricks and techniques. This way we have a starting advantage in using it. Some benefits that deep learning offers are:

- Less time delay: Delay due to the time-consuming tests, waiting for results, traveling (if required), all these extra delays are cut down in our method.
- Easy access: This model can be deployed on any online system and then accessing it is not difficult.

- Reliable: With the techniques used, we have tried to make it as generic as we could, so that it performs well on unseen data samples too.
- Scalable: Anyone who has the microscopic image can use it. Also, because it is reliable, it can be practically used for testing many patients as the model would work well for unseen data as well.
- Cheap: It only requires online computation cost and an internet connection with decent bandwidth.

2.3.2 Artificial Neural Networks

Machine Learning is the study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. Deep Learning is a subfield of Machine Learning based on artificial neural networks. Artificial neural networks (ANN) are systems that are inspired by biological neural networks that constitute animal brains. In simple terms, a biological neuron picks up electrical signal through the dendrites, processes information in the soma and then passes it to the next neuron when activated via the axon terminals. Similarly, the artificial neural network consists of layers made of nodes/neurons each having different values that represent something and different weights (priority). A combination of those when passed through activation, gives the node values of the next layer. This activation is important also because it introduces non-linearities in the data interpretation which brings it close to real-world data.

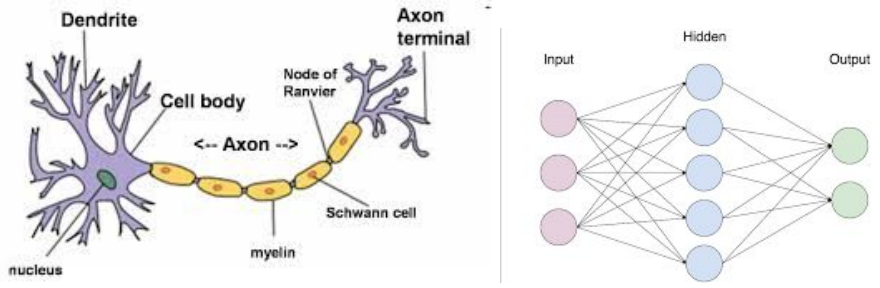

Fig 2.1. Biological Neuron structure; Artificial Neural Network

2.3.3 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a class of ANNs specifically designed for processing an image because of the high computational complexity involved. It performs mathematical convolution of filter matrix over the input data matrix which simplifies the matrix along with extracting the important information out of it. Typically a CNN consists of convolutional layers followed by other ANN layers. The initial layers learn low-level features like points, lines, edges as for example in the case of images. The deeper layers learn higher-level image features like contours, patterns, etc. So, to summarize, a convolutional network consists of convolutional blocks with activation followed by a block that will pick out the important information out of the convolved matrix (pooling layers). More such blocks make up the complete architecture of the network.

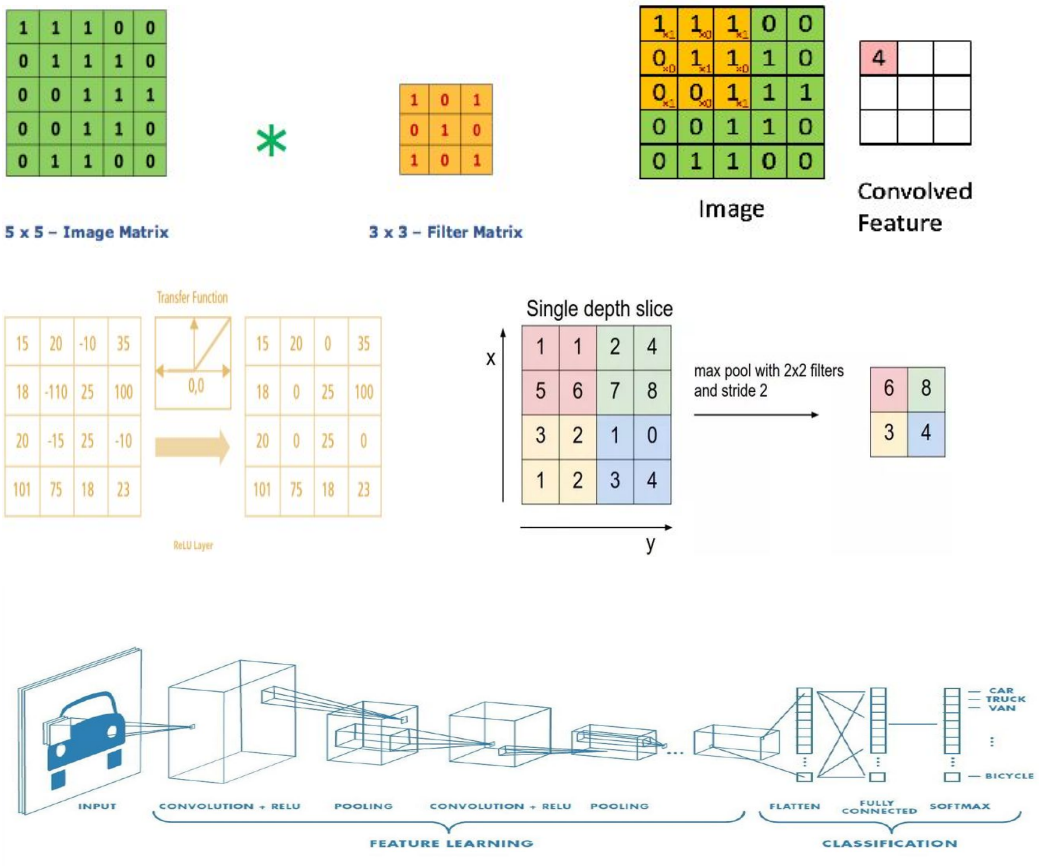

Fig 2.2. Input matrix and filter; their convolution; activation layer; max-pooling; CNN architecture example

There are many such combinations of blocks possible, but there are some standard architectures that have been developed over time by researchers that have proven to perform well in general for a majority of problems, that can mimic the learning of the brain for that kind of problems. Hence, the factors examined for the selection of the correct depth of the architectures were time complexity, memory complexity, and performance specific to the problem at hand (medical imaging). We noticed via experimentation, in addition, to theoretically that shallower architectures would not perform very well in case of imaging the contours as they contain a lot of information to be learned. They would underfit the training data itself so they would obviously not perform well for unseen data. Whereas, very deep architectures had a high chance of overfitting the data as the network had too many parameters to be learned with very fewer data available to do so. It would predict the given data very well but would fail to do so with test data. So, we decided to go ahead with medium architectures and focused on improving their performance using particular techniques.

2.3.3.1 AlexNet

AlexNet won the famous 2012 ImageNet LSVRC-2012 competition. The network featured Convolutional Layers stacked on top of each other. It consists of 8 layers in which 5 layers are convolutional layers and 3 fully connected layers. The model uses the ReLU activation function instead of Tanh and introduces non-linearity in the model. A CNN using ReLU was able to reach a 25% error on the CIFAR-10 dataset six times faster than a CNN using tanh. It also introduced the concept of Overlapping pooling. AlexNet has 60 million parameters and thus to reduce the problem of overfitting the concept of dropout with a determined probability of 50% was introduced.

2.3.3.2 Squeezenet

SqueezeNet is the name of a deep neural network for computer vision that was released in 2016. SqueezeNet was developed by researchers at DeepScale, University of California, Berkeley, and Stanford University. In designing

SqueezeNet, the authors' goal was to create a smaller neural network with fewer parameters that can more easily fit into computer memory. Squeezenet uses 1x1 filters to replace the 3x3 filters of AlexNet. It down-samples late so that convolutional layers have large activation maps.

The building block of squeezenet is called Fire Module which contains Squeeze Layer and an Expand Layer. The squeeze layer and the expand layer keep the same feature map size.

2.3.3.3 VGG19

The VGG network architecture was introduced by Simonyan and Zisserman in their 2014 paper, Very Deep Convolutional Networks for Large Scale Image Recognition. This network is characterized by its simplicity, using only 3×3 convolutional layers stacked on top of each other in increasing depth. Stack of three 3x3 Conv layers has the same effective receptive field as one 7x7 Conv layer. Reducing volume size is handled by max pooling. Two fully-connected layers, each with 4,096 nodes are then followed by a softmax classifier.

2.3.3.4 Resnet50

The specialty of this architecture is the residual block or the identity block. Even though there is a great development in the field of deep learning, its limitations still tend to create obstacles in learning like vanishing gradient descent, dimensionality problems, etc. To overcome these, some smart change in architecture is required that can change the way of learning. That is where the identity block steps in. In general, a layer directly feeds into the next one, but here, in addition to this, the layer also feeds directly into a block 2-3 blocks away. This feeds forward the information to be learned to the layer that is gradually learning it. This keeps a check on the process. As can be seen in the image, the learning by further layers is seen as previous learning plus some residue instead of new learning altogether. This way it just has to focus on the additional part. And if F(x) is made 0, then it directly learns the exact thing

that it is expected to learn. This is a very smart approach and such blocks have drastic effects on the results, learning faster and better.

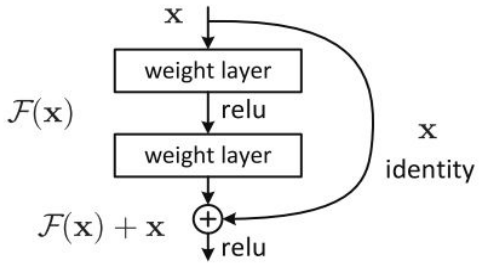

Fig 2.3. Identity block

2.3.3.5 Densenet121

Densenet is an extension of resnet in a way. While in resnet the have blocks in which a layer feeds forward to one other layer ahead of it, in densenet each layer feeds forward to all the layers ahead of it and takes input from all the layers preceding it in a block. Such dense blocks make up the architecture then. This increases the complexity greatly.

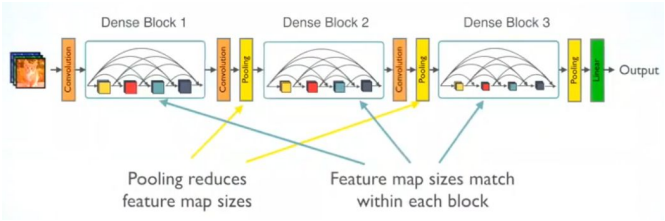

Fig 2.4. Dense block

2.3.4 Hyperparameters

In machine learning, there are two types of parameters, internal parameters, and hyperparameters. A hyperparameter is a parameter whose value is set before the learning process begins and it regulates the process. By contrast, the values of other parameters are derived via training. The main hyperparameters that we took into account in our project were learning rate, weight decay, and dropout rate. The learning rate is a tuning parameter in an optimization algorithm that determines the step size at each iteration while moving toward a minimum of a loss function. In weight decay, where after each update, the weights are multiplied by a factor slightly less than 1. The term dropout refers to dropping out units (both hidden and visible) in a neural network. Both

weight decay and dropout rate can be considered a regularization technique too as it prevents the model from overfitting. There are four major methods to select the correct hyperparameters:

- Trial and Error: Simply trying out arbitrary values
- Grid Search: Picking values in a regular interval from a decided range
- Random Search: Selecting random values lying on a statistically random distribution
- Bayesian Optimization: Considering the selection of hyperparameters to be another machine learning problem in itself and finding out the correct values

$$(LR * WD)/(TBS * (1 - \alpha)) \approx constant$$

where LR = learning rate, WD = weight decay coefficient, TBS = total batch size, and α = momentum

Fig 2.5. Relation between hyper-parameters

2.3.5 Metrics

A confusion matrix is the most helpful visual way to analyze the results and compare model performances. Some terms associated with it as true positives, true negatives, false positives, and false negatives. These are the terms when we are talking about binary classification. Considering that positive is the class that we want to predict and negative is the one other than that like in the case of leukemia, cancer is the class that we wish to predict so it is positive class and no cancer being the negative one. Then true positives are the ones that are cancerous and are also predicted cancerous correctly. Similarly, the other terms are shown in the following image.

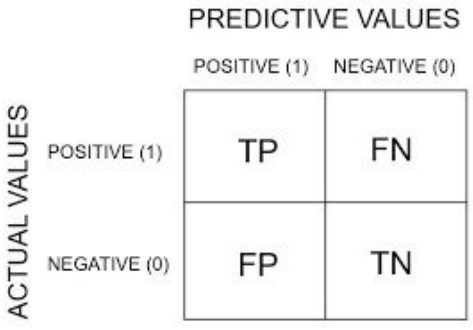

Fig 2.6. Confusion Matrix

Now building on these terms, we have some other metrics that help us evaluate the model's performance.

- Accuracy = (TP+TN) / (TN+TP+FN+FP): fraction of total correctly predicted classes
- Sensitivity/Recall = TP / (TP+FN): out of actual cancer class, number of predicted ones
- Precision = TP / (TP+FP): out of all predicted cancer class, how many are actual ones
- F1 Score= 2*(Precision*Recall) / (Precision+Recall):

Our focus was to lower the false negatives as we want to avoid the case when it is cancerous and predicts no cancer. As false negatives reduce, the sensitivity increases along with the accuracy of course. So, for this, we began with a skewed dataset that had more cancer class data as compared to the other one. Only accuracy is not very intuitive to evaluate a model's performance. To give an example of this, consider a data that consists of 90% class A data and 10% class B data, then predicting A always will also give 90% accuracy but that does not in any way suggest that we have a good model. So, to take into account all of that, we calculate precision, recall and the harmonic mean of the two, F1 score. Considering this along with accuracy is a more holistic approach to assessing and comparing the performances.

3

Dataset

This section provides a description of the dataset used for training and testing the model.

3.1 Description

We selected a dataset of cells with labels (normal vs cancerous) to train machine learning-based classifiers to identify normal cells from leukemic blasts (malignant/cancer cells). These cells have been segmented from the microscopic images by the data providers. The images are representative of real-world images because some of these contain staining noise and illumination errors, although most of these errors were fixed by the data providers via stain color normalization.

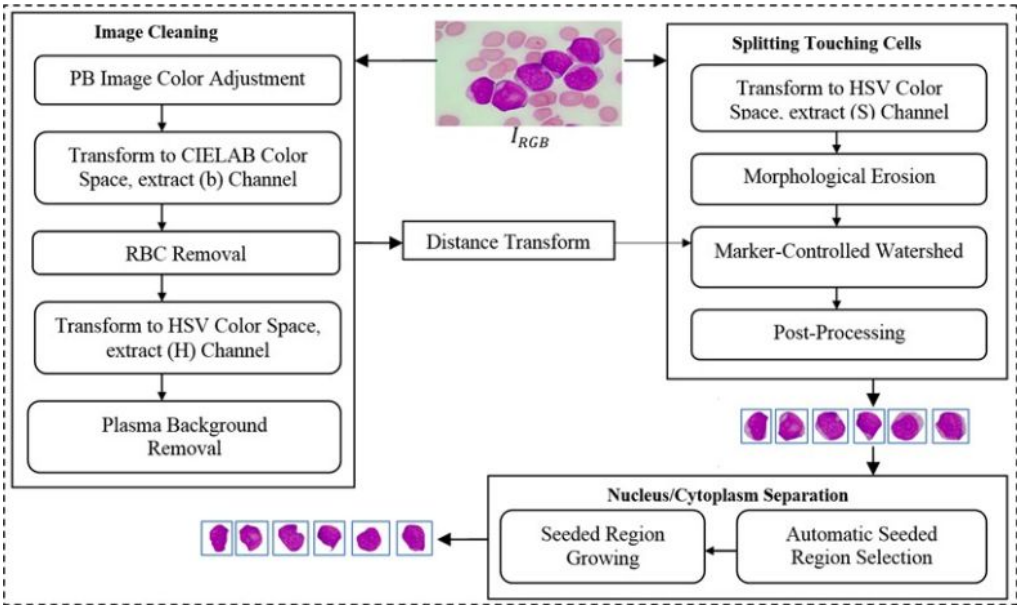

Fig 3.1. Flowchart of the segmentation pipeline

The training dataset consists of a total of 73 individual subjects, distributed as follows:

- ALL(cancer) subjects: 47
- Normal subjects: 26
- Total cells: 10,661
- ALL: 7272
- Normal: 3389

The ALL cells are almost double in number as compared to the normal cells. We preferred a skewed dataset over a balanced one because it automatically increases the average prediction accuracy of ALL cells over normal cells, which from the viewpoint of real clinical applications is more acceptable since mistakenly classified normal cells to ALL cells suffer much lower risk than reversed predictions.

The testing dataset consists of a total of 28 individual subjects, distributed as follows:

- ALL(cancer) subjects: 13
- Normal subjects: 15
- Total cells: 1867
- ALL: 1219
- Normal: 648

This dataset was provided by SBILAB- IIIT Delhi led by Prof. Anubha Gupta.

3.2 Pre-Processing

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

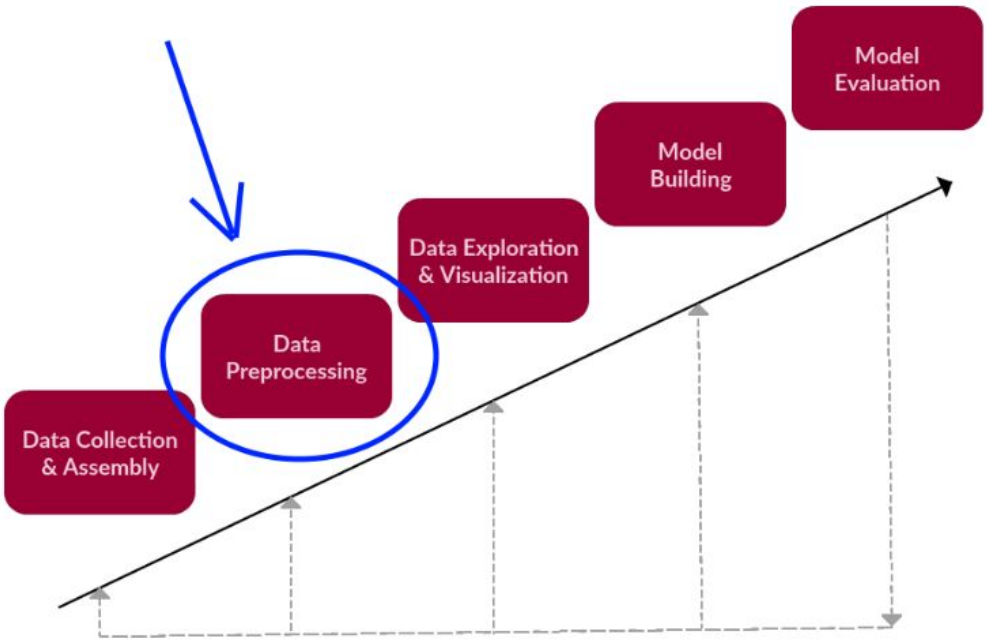

Fig 3.2. Pre-Processing in context of Machine learning Framework

3.2.1 Removing uninformative part of the data

The original images provided to us were of the size 300 x 300. A portion of these images didn't contain any significant information. So, we implemented an algorithm to extract only the informative part of the data. To solve our case, we looked for rows and columns that have at least one pixel along rows and columns that is greater than some lower limit or threshold as a pixel value. As the *black* areas are absolutely zeros, we set our threshold at '0'.

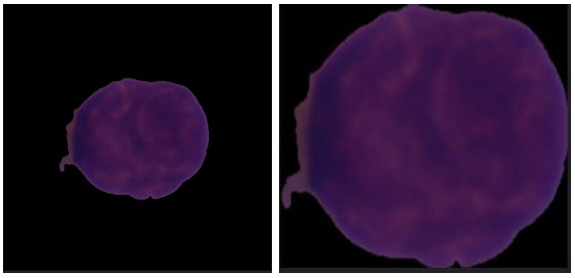

Fig 3.3. Image before preprocessing; Image after preprocessing

3.2.2 Contrast Enhancement using Histogram Equalisation

- *Histogram Equalization*: Histogram Equalization is a computer image processing technique used to improve contrast in images. It accomplishes this by effectively spreading out the most frequent intensity values, i.e. stretching out the intensity range of the image. This method usually increases the global contrast of images when its valuable data is represented by close contrast values. This allows for areas of lower local contrast to gain a higher contrast.

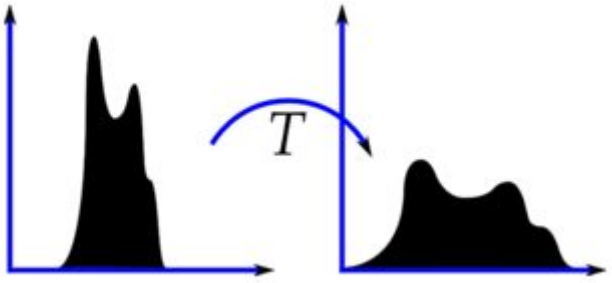

Fig 3.4. Histogram Equalization in Images

As observed in certain pieces of literature the contours in a nucleus play a significant role in distinguishing cancerous from non-cancerous cells. So, to make the contours more visible we proceeded forward with histogram equalization.

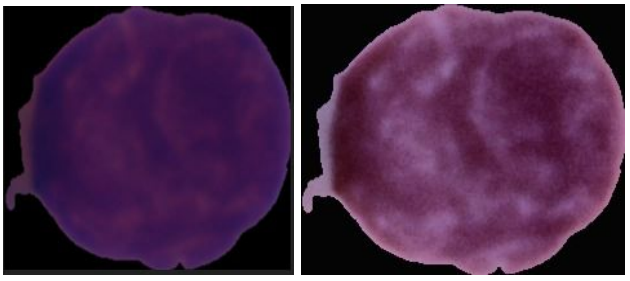

Fig 3.5. Original Image, Contrast-Enhanced Image

3.2.3 Augmentations

Data augmentation is perhaps the most important regularization technique when training a model for Computer Vision. Instead of feeding the model with the same images as input every time, we perform small random transformations/augmentations (eg. rotation, zoom, translation, etc.) that do not change what is inside the image (to the human eye) but do change its pixel values. Since the dataset is skewed with cancer cells being more in number (almost twice) as compared to the normal cells, models trained with augmented data generalize better.

Some of the augmentations that we did are as follows:

Flip: You can flip images horizontally and vertically. Below are examples of images that are flipped.

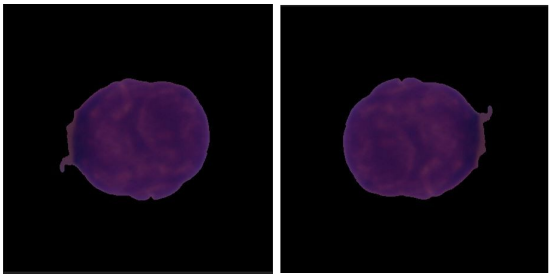

Fig 3.6. Original Image; Flipped Image

Rotate: One key thing to note about this operation is that image dimensions may not be preserved after rotation. If your image is a square, rotating it at right angles will preserve the image size. If it's a rectangle, rotating it by 180 degrees would preserve the size. Rotating the image by finer angles will also change the final image size.

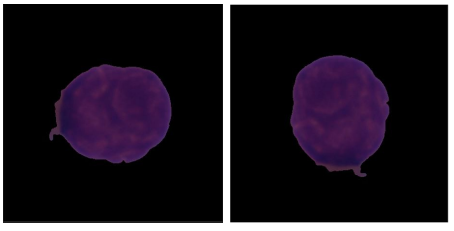

Fig 3.7. Original Image; Rotated Image

4

Analysis and Results

4.1 Architectures

In this chapter, we present to you Deep Learning Architectures that we used and trained for the detection and classification of nucleus image.

4.1.1 AlexNet

AlexNet won the famous 2012 ImageNet LSVRC-2012 competition. The network featured Convolutional Layers stacked on top of each other. ReLu activation was used instead of Tanh to add non-linearity. Dropout was used instead of Regularisation to deal with the overfitting.

Pretrained AlexNet model was used and fine-tuned so that it can be used for our application.
To find the appropriate learning rate for the training of AlexNet a plot of Loss vs Learning Rate was used. The following is the graph.

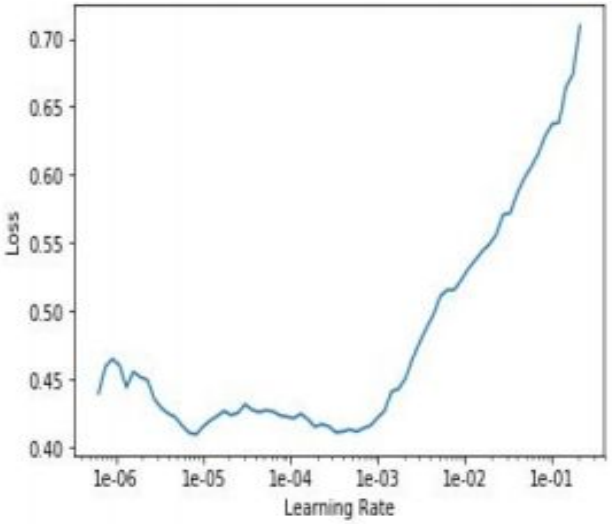

Fig 4.1. Loss vs Learning Rate Alexnet

The learning rate of 1e-5 was used where the slope of the graph was minimum.

The model was trained for 10 epochs for fine-tuning the last layer group (i.e. only the last layer of the model was trained) and for the next 10 epochs the entire model was trained. The table for the metrics for each epoch during the last training is as under.

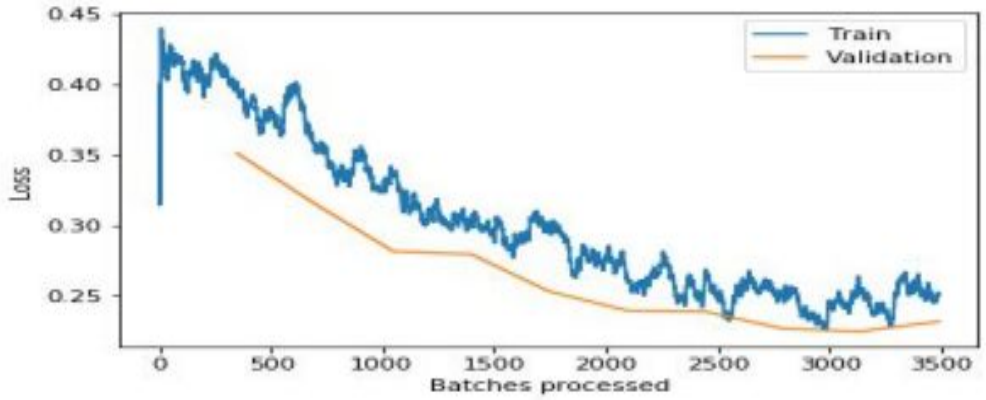

Fig 4.2. Loss vs Iterations

```
[ ]  learn.fit_one_cycle(10,max_lr=1e-5)
```

epoch	train_loss	valid_loss	accuracy	f_beta
0	0.398766	0.350675	0.847118	0.787851
1	0.352717	0.314839	0.870390	0.833581
2	0.337190	0.281185	0.885786	0.846840
3	0.303889	0.279138	0.886144	0.830078
4	0.293937	0.252794	0.896885	0.851747
5	0.263312	0.239237	0.906910	0.875777
6	0.260713	0.238530	0.904404	0.861786
7	0.252481	0.226321	0.915503	0.886572
8	0.254932	0.224500	0.914071	0.881263
9	0.250041	0.231406	0.911207	0.869598

Table 4.1. Alexnet Training

Evaluating the test set, AlexNet had an accuracy of 75.46% and did not perform well on unseen data points that were present in the test set.

4.1.2 SqueezeNet

SqueezeNet is the name of a deep neural network for computer vision that was released in 2016. SqueezeNet was developed by researchers at DeepScale, University of California, Berkeley, and Stanford University. In designing SqueezeNet, the authors' goal was to create a smaller neural network with fewer parameters that can more easily fit into computer memory

Squeezenet uses 1x1 filters to replace the 3x3 filters of AlexNet. It down-samples late so that convolutional layers have large activation maps.

The learning rate finder graph of Squeezenet.

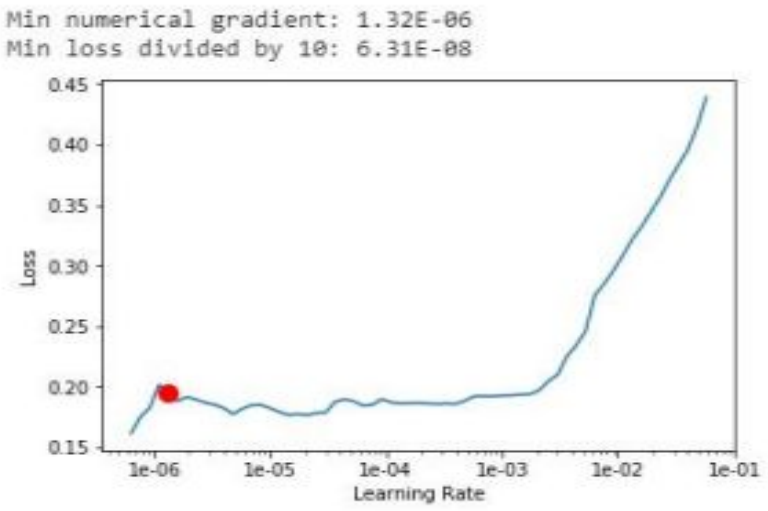

Fig 4.3. Loss vs Learning Rate Squeezenet

The pretrained model was trained on the learning rate 1.32e-6 for 12 epochs (Fine-tuning the last layer for 7 epochs and training the entire model for the other 5 epochs).

epoch	train_loss	valid_loss	accuracy	f_beta
0	0.545822	0.379732	0.841143	0.772496
1	0.441926	0.343372	0.863168	0.775986
2	0.407607	0.339096	0.866448	0.803977
3	0.380477	0.313627	0.878163	0.813588
4	0.366781	0.311211	0.884724	0.833097
5	0.352756	0.294872	0.898782	0.847820
6	0.345977	0.285536	0.900656	0.833573

Table 4.2: Squeezenet Training

epoch	train_loss	valid_loss	accuracy	f_beta
0	0.340316	0.626987	0.706186	0.821446
1	0.309950	0.452929	0.805530	0.858112
2	0.295913	0.578317	0.758669	0.859649
3	0.216614	0.214870	0.922212	0.915060
4	0.198527	0.146347	0.945173	0.910653

Table 4.3: Squeezenet Training

The confusion matrix for the validation set is as below.

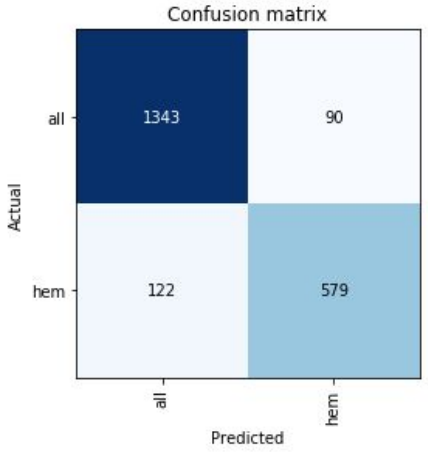

Fig 4.4. Squeezenet Confusion Matrix

Evaluating the test set, SqueezeNet had an accuracy of 80.23%.

4.1.3 VGG19

The VGG network architecture was introduced by Simonyan and Zisserman in their 2014 paper, Very Deep Convolutional Networks for Large Scale Image Recognition. This network is characterized by its simplicity, using only 3×3 convolutional layers stacked on top of each other in increasing depth. Reducing volume size is handled by max pooling. The Learning rate finder and the training graph of the VGG19 model are as shown below.

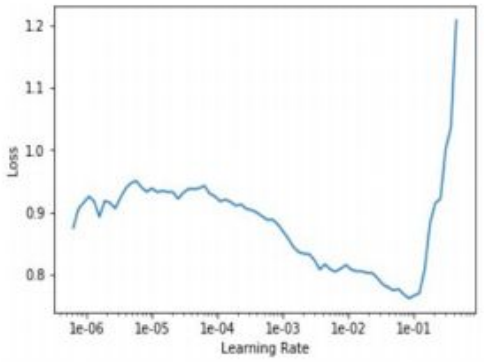

Fig 4.5. Loss vs Learning Rate VGG19

The optimum learning rate was found to be 5e-2. Metrics vs epochs for last stage training of VGG19 is shown below.

```
learn.fit_one_cycle(7,5e-2)
```

epoch	train_loss	valid_loss	accuracy	f_beta
0	0.397664	0.683696	0.764769	0.799878
1	0.458163	0.412378	0.823487	0.837562
2	0.388292	0.267063	0.887934	0.889125
3	0.261224	0.854885	0.586108	0.703893
4	0.268600	0.184562	0.933047	0.928489
5	0.166007	0.126086	0.955245	0.953721
6	0.158207	0.120816	0.957393	0.955942

Table 4.4 VGG19 Training

Evaluating the test set, SqueezeNet had an accuracy of 79.8%.

4.1.4 RESNET50

The Resnet50 model gave the best metrics and results for our model. Various techniques were deployed further on Resnet50 which improved the model's performance and accuracy.
The techniques and their results are summarised in the next section.

4.2 Techniques

4.2.1 Ensembling

In its simplest form, ensembling can consist of training a certain number of copies of a model with different initializations, and averaging the predictions of the copies to get the prediction of the ensemble. The downside of this approach is that you have to incur the cost of training 'n' different copies.

Ensembling gave us good accuracy on the test set. Various Ensembles were tried. Some of them are:

1. Resnet50 & VGG19
2. Resnet50, VGG19, SqueezeNet
3. Resnet50, Densenet121, VGG19

The best accuracy that we achieved using the ensembling method was 87% on the test set.
The problem with ensembling is that it is very much data specific and is generally preferred in competitive challenges. To make the model more scalable we, therefore, tried other approaches and techniques.

4.2.2 Progressive Image Resizing

It is the technique to sequentially resize all the images while training the CNNs on smaller to bigger image sizes. A great way to use this technique is to train a model with smaller image size say 16x16, then use the weights of this model to train another model on images of size 64x64 and so on. Each larger-scale model incorporates the previous smaller-scale

model layers and weights in its architecture. We trained the models in this fashion to 224x224 image size.

The model was trained in this fashion.

Image Size	Number of Epochs
8x8	8
16x16	16
24x24	24
32x32	28
64x64	40
128x128	36
224x224	14

Table 4.5 Image Size vs Epochs

The training was stopped when the losses suggested that the model is overfitting.

The table below shows the metrics for each epoch for the image size of 224x224.

epoch	train_loss	valid_loss	error_rate	accuracy
0	0.178092	0.114164	0.043159	0.956841
1	0.172704	0.113424	0.039930	0.960070
2	0.188210	0.125072	0.045214	0.954786
3	0.171467	0.126202	0.044334	0.955666
4	0.189348	0.412518	0.071345	0.928655
5	0.179394	0.142148	0.049618	0.950382
6	0.176880	0.142025	0.052261	0.947739
7	0.186361	0.133860	0.046976	0.953024
8	0.169429	0.136757	0.046389	0.953611
9	0.150861	0.146162	0.049912	0.950088
10	0.145812	0.136271	0.046095	0.953905
11	0.140660	0.137677	0.046682	0.953318
12	0.142327	0.147475	0.049031	0.950969
13	0.138379	0.139315	0.046976	0.953024

Table 4.6. Resnet50 Training for image size 224x224

This technique didn't give good results on the test set. Possibly the model was overfitting or giving more weights to less important features.

4.2.3 Loss Penalisation

Since the dataset was skewed dataset and thus modifying loss function was important. Thus modifying the loss function was important. Thus class weighting the loss function was important. Poor predictions of underweighted classes are penalized more heavily in the loss function.
In regular learning, we treat all misclassifications equally which causes issues in imbalanced classification problems, as there is no extra reward for predicting the minority class over the majority class. Cost-sensitive learning changes this. This allows us to penalize misclassifications of the majority class in hopes that this increases the true positive rate. A common scheme for this is to have the cost equal to the inverse of the proportion of the dataset that the class makes up. This increases the penalization as the class size decreases.
The following graphs are the training results when loss penalization was incorporated on the Resnet50 model.

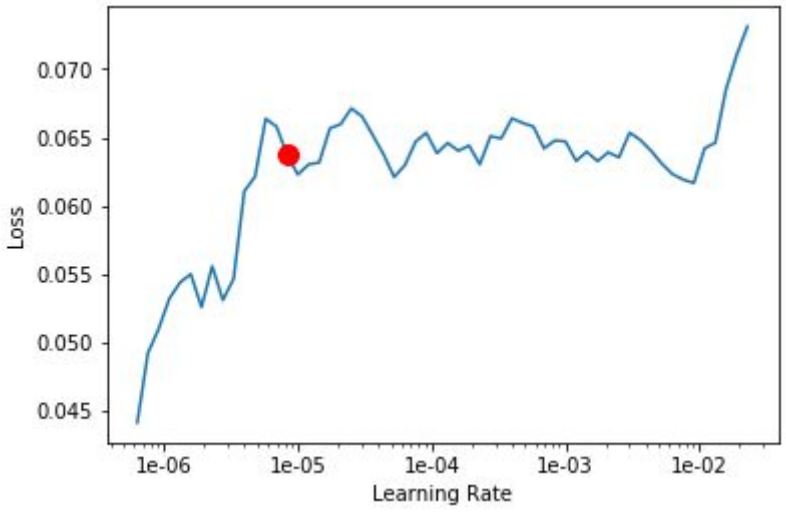

Fig 4.6. Loss vs Learning Rate (Resnet50, Loss Penalisation)

epoch	train_loss	valid_loss	accuracy	f_beta
0	0.319576	0.250306	0.896439	0.881616
1	0.271077	0.284902	0.883786	0.856863
2	0.230019	0.660709	0.706186	0.818965
3	0.197041	0.175465	0.927366	0.851343
4	0.159363	0.230026	0.910028	0.933661
5	0.097789	0.112483	0.962043	0.960237
6	0.067659	0.078514	0.970009	0.961375

Table 4.7. Resnet50 Training (Loss Penalisation)

epoch	train_loss	valid_loss	accuracy	f_beta
0	0.059107	0.086635	0.965323	0.958652
1	0.044624	0.072270	0.971415	0.960444
2	0.037252	0.065659	0.974227	0.960331

Table 4.8 Resnet50 Training (Loss Penalisation)

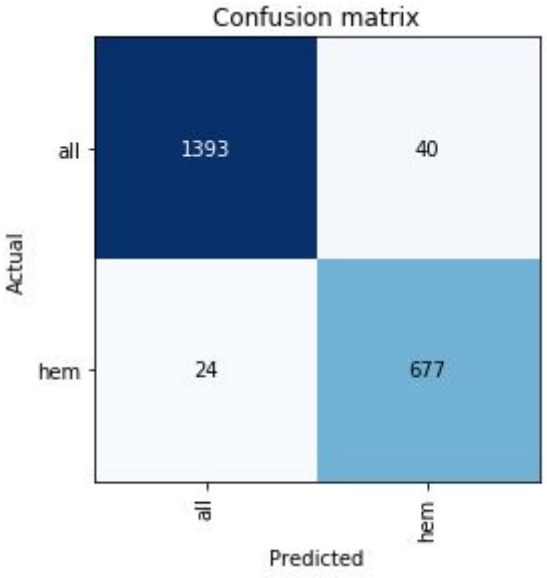

Fig 4.7. Resnet50 (Loss Penalisation) Confusion Matrix

4.2.4 K-Fold Cross-Validation

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

K-Fold is popular and easy to understand, it generally results in a less biased model compared to other methods. Because it ensures that every observation from the original dataset has the chance of appearing in training and test set.

K- Fold follows the below-mentioned steps:

- Split the entire data randomly into k folds (value of k shouldn't be too small or too high, ideally, we choose 5 to 10 depending on the data size). The higher value of K leads to a less biased model (but large variance might lead to overfitting), whereas the lower value of K is similar to the train-test split approach we saw before.
- Then fit the model using the K - 1 (K minus 1) folds and validate the model using the remaining Kth fold. Note down the scores/errors.
- Repeat this process until every K-fold serves as the test set. Then take the average of your recorded scores. That will be the performance metric for the model.

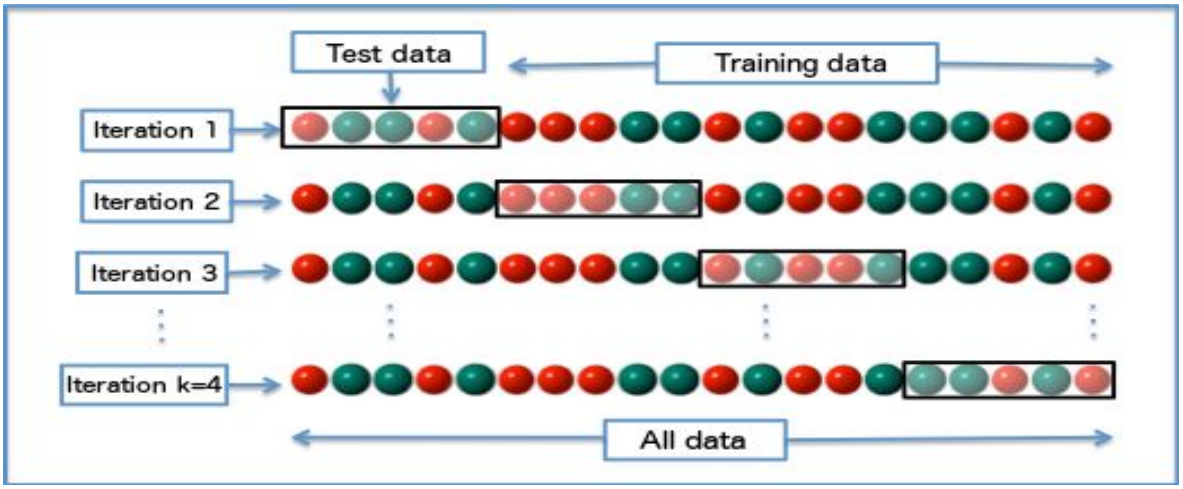

Fig 4.8. K-Fold Cross-Validation Procedure

If the estimator (model) is a classifier and ‘y’(target variable) is either binary/multiclass, then the ‘**StratifiedKfold**’ technique is used by default. In all other cases, the ‘**K_Fold**’ technique is used as a default to split and train the model.

The following are the code and graphs are the training results when K-Fold Cross- Validation was incorporated on the Resnet50 model:

```
skf=StratifiedKFold(n_splits=5,shuffle=True,random_state=1)
```

```
acc_val=[]
for train_idx,val_idx in skf.split(df.index,df['y']):
  data_fold=ImageList.from_df(df,path).split_by_idxs(train_idx,val_idx).label_from_df().transform(tfms,size=300).databunch(num_workers=0).normalize(imagenet_st
  learn=cnn_learner(data_fold,models.resnet34,metrics=[accuracy,FBeta(beta=1)])
  #loading model
  learn.data=data_fold
  learn.load('kfold')
  learn.fit_one_cycle(5)
  learn.unfreeze()
  learn.fit_one_cycle(5)
  #model saving
  learn.save('kfold')
  loss,acc=learn.validate()
  acc_val.append(acc.numpy())
```

Fig.4.9. K-Fold Cross-Validation Split Code

epoch	train_loss	valid_loss	accuracy	f_beta	time
0	0.490896	0.328332	0.868068	0.855605	1:23:49
1	0.342457	0.246638	0.894902	0.888361	07:32
2	0.281310	0.205754	0.917263	0.910063	07:17
3	0.261520	0.195918	0.924419	0.918711	07:17
4	0.225673	0.193243	0.928891	0.921944	07:18

Table.4.9. K-Fold Cross-Validation Iteration-1

epoch	train_loss	valid_loss	accuracy	f_beta	time
0	0.288378	0.312218	0.859571	0.865582	08:51
1	0.271129	0.319873	0.857782	0.827922	08:40
2	0.207290	0.659532	0.733900	0.779874	08:37
3	0.152347	0.344280	0.858676	0.867227	08:41
4	0.111221	0.084877	0.970483	0.968661	08:38

Table.4.10. K-Fold Cross-Validation Iteration-2

epoch	train_loss	valid_loss	accuracy	f_beta	time
0	0.166831	1.625004	0.612528	0.708417	08:48
1	0.207189	2.220948	0.580761	0.693691	08:40
2	0.182175	0.928382	0.668904	0.714065	08:41
3	0.125034	0.079684	0.965996	0.964880	08:42
4	0.084834	0.064129	0.974944	0.973807	08:40

Table.4.11. K-Fold Cross-Validation Iteration-3

epoch	train_loss	valid_loss	accuracy	f_beta	time
0	0.116740	0.855668	0.676063	0.687392	08:40
1	0.180892	0.337482	0.860403	0.868132	08:37
2	0.142518	0.379353	0.870246	0.878049	08:36
3	0.090688	0.104445	0.964206	0.963066	08:35
4	0.059379	0.059675	0.975392	0.974239	08:37

Table.4.12. K-Fold Cross-Validation Iteration-4

epoch	train_loss	valid_loss	accuracy	f_beta	time
0	0.096007	0.598316	0.771365	0.700293	08:41
1	0.143315	0.151556	0.939597	0.933921	08:41
2	0.140788	2.937513	0.588814	0.697995	08:46
3	0.093977	0.650893	0.815213	0.836176	08:50
4	0.054138	0.035359	0.988814	0.988168	08:50

Table.4.13. K-Fold Cross-Validation Iteration-5

4.2.5 Bayesian Optimisation for Selection of Hyper-parameters

Hyperparameters*:* In statistics, hyperparameter is a parameter from a prior distribution; it captures the prior belief before data is observed. In any machine learning algorithm, these parameters need to be initialized before training a model. In contrast, other parameters are learned during the training process.

Model Hyperparameters are the properties that govern the entire training process. The variables mentioned below are usually configured before training a model.

- Learning Rate
- Number of Epochs
- Weight Decay
- Dropout

Choosing good hyperparameters gives two benefits:

- Efficiently search the space of possible hyperparameters
- Easy to manage a large set of experiments for hyperparameter tuning.

Hyperparameters Optimisation Techniques: The process of finding the most optimal hyperparameters in machine learning is called hyperparameter optimization.
Some algorithms for Hyperparameter Optimisation are stated below:

- Grid Search
- Random Search
- Bayesian Optimisation

Grid Search: Grid search is a very traditional technique for implementing hyperparameters. It brute force all combinations. Grid search requires to create two sets of hyperparameters: Learning rate and Number of Layers

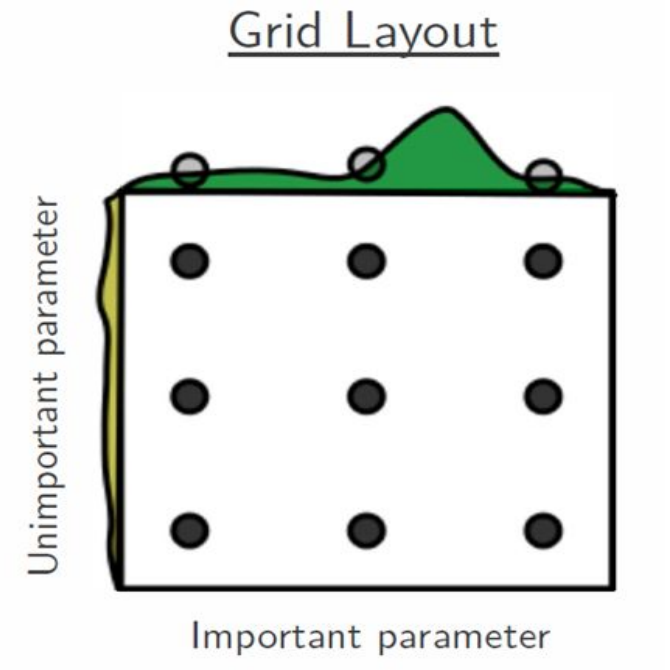

Fig.4.10. Grid Search

The Grid search method is a simpler algorithm to use but it suffers if data has a high dimensional space called the curse of dimensionality.

Random Search: Randomly samples the search space and evaluates sets from a specified probability distribution. For example, Instead of trying to check all 100,000 samples, we can check 1000 random parameters.

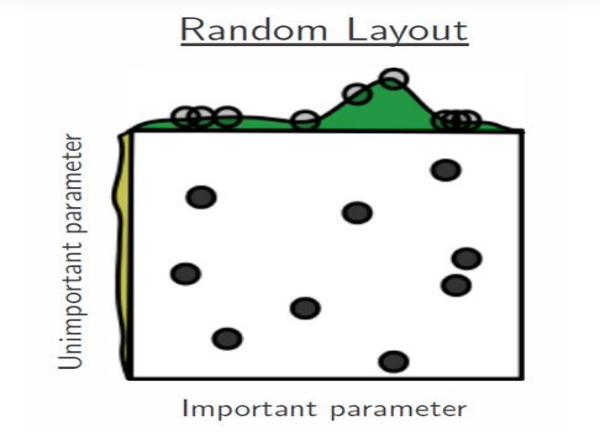

Fig 4.11. Random Search

The drawback of using the random search algorithm, however, it doesn't use information from prior experiments to select the next set and also it is very difficult to predict the next experiments.

Bayesian Optimisation:

The hyperparameter setting maximizes the performance of the model on a validation set. Machine learning algorithms frequently require to fine-tuning of model hyperparameters. Unfortunately, that tuning is often called as '*black function*' because it cannot be written into a formula since the derivates of the function are unknown.

Much more appealing way to optimize and fine-tune hyperparameters are enabling automated model tuning approach by using the Bayesian optimization algorithm. The model used for approximating the objective function is called the surrogate model. A popular surrogate model for Bayesian optimization is a Gaussian process (GP). Bayesian optimization typically works by assuming the unknown function was sampled from a Gaussian Process (GP) and maintains a posterior distribution for this function as observations are made.

There are two major choices must be made when performing Bayesian optimization.

1. Select prior over functions that will express assumptions about the function being optimized. For this, we choose the ***Gaussian Process*** prior
2. Next, we must choose an ***acquisition function*** that is used to construct a utility function from the model posterior, allowing us to determine the next point to evaluate.

The following are the results when Bayesian optimization was incorporated on the Resnet50 model:

```
Accuracy:94.41260695457458
|  11       |  0.9441   |  0.1      |  0.01     |  0.5595   |
=============================================================
Iteration 0:
        {'target': 0.9462750554084778, 'params': {'dp': 0.35021320282154444, 'lr': 0.0072035246099281395, 'wd': 0.01009035610570246}}
Iteration 1:
        {'target': 0.9426934123039246, 'params': {'dp': 0.28139954357910385, 'lr': 0.0014684121522803135, 'wd': 0.08294748986735026}}
Iteration 2:
        {'target': 0.9512894153594971, 'params': {'dp': 0.21175612682660255, 'lr': 0.003456261709703435, 'wd': 0.3234463046422293}}
Iteration 3:
        {'target': 0.9541547298431396, 'params': {'dp': 0.4232900404020141, 'lr': 0.004192525949518545, 'wd': 0.55132340531344}}
Iteration 4:
        {'target': 0.9462750554084778, 'params': {'dp': 0.22267134983891046, 'lr': 0.008781296246473063, 'wd': 0.03163619862636167}}
Iteration 5:
        {'target': 0.9419770836830139, 'params': {'dp': 0.5022805061070413, 'lr': 0.004173630718868903, 'wd': 0.45136496447214386}}
Iteration 6:
        {'target': 0.9527220726013184, 'params': {'dp': 0.1, 'lr': 0.01, 'wd': 0.8}}
Iteration 7:
        {'target': 0.9484240412712097, 'params': {'dp': 0.7, 'lr': 0.01, 'wd': 0.8}}
Iteration 8:
        {'target': 0.9462750554084778, 'params': {'dp': 0.3836654769237889, 'lr': 0.01, 'wd': 0.8}}
Iteration 9:
        {'target': 0.6160458326339722, 'params': {'dp': 0.7, 'lr': 1e-06, 'wd': 0.01}}
Iteration 10:
        {'target': 0.9441260695457458, 'params': {'dp': 0.1, 'lr': 0.01, 'wd': 0.5594508287555359}}
{'target': 0.9541547298431396, 'params': {'dp': 0.4232900404020141, 'lr': 0.004192525949518545, 'wd': 0.55132340531344}}
```

Fig.4.12. Iterations of Bayesian Optimization

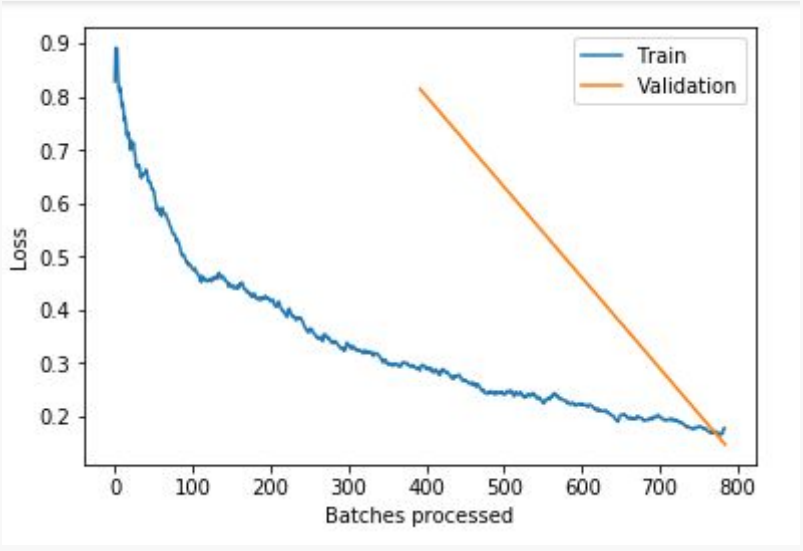

Fig.4.13. Loss vs Batches Processed Bayesian Optimization

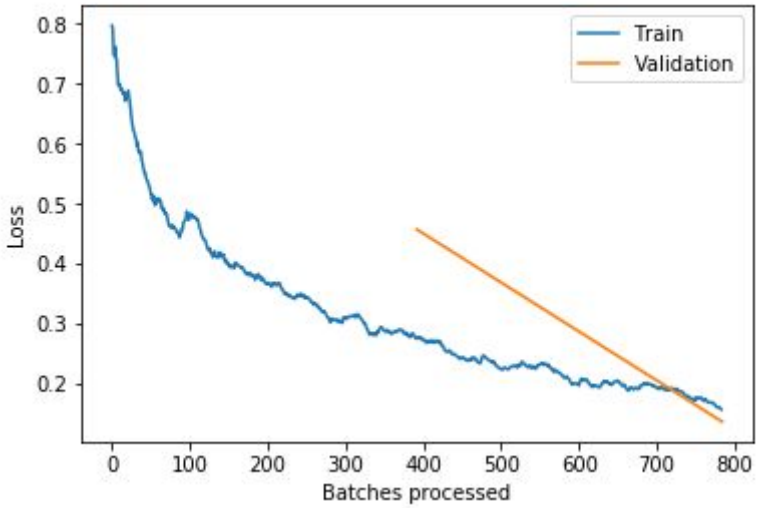

Fig.4.14. Loss vs Batches Processed Bayesian Optimization

epoch	train_loss	valid_loss	accuracy	f_beta	time
0	0.148517	0.137102	0.952006	0.950626	03:38
1	0.160374	0.137404	0.953438	0.951601	03:42
2	0.151868	0.137922	0.953438	0.951384	03:42

Table.4.14. Training Results (Bayesian Optimized Hyperparameter)

5

Conclusion

This report has studied the usage of Computer Vision as an alternative method for the detection of Acute Lymphoblastic Leukemia. The method employed makes use of the segmented images of the nucleus of Lymphoblasts to build a Neural Net model that can classify new images. All the CNNs used in this work are pre-trained architectures which then require only fine-tuning as results in the past have shown that pre-trained models gave better results compared to the model trained from scratch. This also saves computation time.

Using convergence algorithms to obtain the Hyperparameters is probably a step in the right direction as models trained on hyperparameters obtained using Bayesian Optimisation Algorithms gave better accuracies than the ones trained without using it. We also employed novel techniques like Loss Penalisation while training the model, considering that the dataset was skewed. Augmenting the images helped to generate multiple images from a single image and increased the training dataset.

Other important conclusions that can be drawn from the work are:

- CNN's are reliable and better for image classification especially in the view of the increasing complexity of the images.
- It is very essential to investigate and analyze the possible neural network architectures and also learning techniques that need to be deployed.
- CNN's trained using Bayesian Optimized Hyperparameters perform better than randomly chosen parameters.

Samsung Innovation Merit Award

The aforementioned idea of "Leukemia Classification from microscopic images using Deep Learning Techniques" won the merit award at the Samsung Innovation Award 2019 held at IIT Indore on 8th November 2019.

References

1. Library for implementing architectures: docs.fast.ai/
2. Acute lymphoblastic leukemia cells image analysis with deep bagging ensemble learning: doi.org/10.1101/580852
3. Neural Networks: towardsdatascience.com/understanding-neural-networks-19020b758230
4. Jonas Prellberg and Oliver Kramer: Acute Lymphoblastic Leukemia Classification from Microscopic Images using Convolutional Neural Networks:arxiv.org/pdf/1906.09020.pdf
5. Hyperparameter-Optimization:towardsdatascience.com/understanding-hyperparameters-and-its-optimisation-techniques-f0debba07568
6. Histogram Equalization:towardsdatascience.com/histogram-equalization-5d1013626e64
7. About ALL: www.mayoclinic.org/diseases-conditions/leukemia/symptoms-causes/syc-20374373
8. Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional
9. neural networks. In Advances in neural information processing systems (2012), p. 1097–1105.
10. Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014).
11. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385,2015.
12. Shafique, S., Tehsin, S.: Acute lymphoblastic leukemia detection and classification of its subtypes using pre-trained deep convolutional neural networks. Technology in Cancer Research & Treatment 17, 1533033818802789 (2018). https://doi.org/10.1177/1533033818802789, pMID: 30261827

13. SQUEEZENET: ALEXNET-LEVEL ACCURACY WITH 50X FEWER PARAMETERS AND <0.5MB MODEL SIZE: https://arxiv.org/pdf/1602.07360.pdf
14. Chiaretti, S., Zini, G., Bassan, R.: Diagnosis and subclassification of acute lymphoblastic leukemia. Mediterranean journal of hematology and infectious diseases 6(1), e2014073–e2014073 (Nov 2014), https://www.ncbi.nlm.nih.gov/pubmed/ 25408859
15. Duggal, R., Gupta, A., Gupta, R.: Segmentation of overlapping/touching white blood cell nuclei using artificial neural networks. In: CME Series on HematoOncopathology. All India Institute of Medical Sciences (AIIMS), New Delhi, India (2016)
16. Duggal, R., Gupta, A., Gupta, R., Mallick, P.: Sd-layer: Stain deconvolutional layer for cnns in medical microscopic imaging. In: Descoteaux, M., Maier-Hein, L., Franz, A., Jannin, P., Collins, D.L., Duchesne, S. (eds.) Medical Image Computing and Computer Assisted Intervention (MICCAI 2017). pp. 435–443. Springer International Publishing, Cham (2017)
17. Duggal, R., Gupta, A., Gupta, R., Wadhwa, M., Ahuja, C.: Overlapping cell nuclei segmentation in microscopic images using deep belief networks. In: Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing. pp. 82:1–82:8. ICVGIP '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/3009977.3010043, http://doi.acm.org/10.1145/
18. . Gupta, A., Duggal, R., Gupta, R., Kumar, L., Thakkar, N., Satpathy, D.: GCTI-SN: Geometry-inspired chemical and tissue invariant stain normalization of microscopic medical images. Under review
19. Gupta, R., Mallick, P., Duggal, R., Gupta, A., Sharma, O.: Stain color normalization and segmentation of plasma cells in microscopic images as a prelude to development of computer assisted automated disease diagnostic tool in multiple myeloma. Clinical Lymphoma, Myeloma and Leukemia 17(1), e99 (2019/03/18 2017)