



e-ISSN: 2319-8753 | p-ISSN: 2347-6710

IJIRSET

International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 12, Issue 8, August 2023

ISSN INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA

Impact Factor: 8.423

 9940 572 462

 6381 907 438

 ijirset@gmail.com

 www.ijirset.com

Acute Lymphoblastic Leukemia(ALL) Detection using Deep Learning Model

Insha Rashid Shah, Vivek Kumar

PG Student, Department of Computer Science and Engineering, Swami Vivekanand Institute of Engineering and Technology, I.K Gujral Punjab Technical University, Kapurthala Jalandhar, India

Department of Computer Science and Engineering, Swami Vivekanand Institute of Engineering and Technology, I.K Gujral Punjab Technical University, Kapurthala Jalandhar, India

ABSTRACT: Acute Lymphoblastic Leukaemia(ALL) is a cancer of the lymphoid line of blood cells characterized by the development of large numbers of immature lymphocytes. Being an acute cancer, early detection plays a vital role in the successful treatment of the subjects. Extensive research has been done over the application of machine learning algorithms for detection of ALL but use of Deep Learning (DL) models is relatively scarce. For our project we study and compare the applicability of available DL architectures and also developed our own model for detection of ALL. Our model was able to achieve the best accuracy of 93.0.

I. INTRODUCTION

Cancer is a group of diseases involving abnormal cell growth with the potential to invade or spread to other parts of the body [1][2]. Cancer has a major effect on societies worldwide as it is one of the leading causes of deaths. In 2015, about 90.5 million people were known to be cancer affected with about 14.1 million new cases occurring each year (not including skin cancer and other melanomas), and led to deaths of 8.8 million people [3][4][5]. Leukemia is one of the most common cases of cancers. Acute Lymphoblastic Leukemia(ALL) is a cancer of the lymphoid line of blood cells characterized by the development of large numbers of immature lymphocytes [6]. As an acute leukemia, ALL progresses rapidly and is typically fatal within weeks or months if left untreated [7]. Table 1 shows types of ALL, along with the type of cells affected as well as characteristics of the affected cells. In 2015 there were 876,000 cases of ALL which resulted in 111,000 deaths [3][5]. In 2019 there have been 5930 cases of ALL and 1500 deaths so far [8]. Diagnosing ALL begins with a thorough medical history, physical examination, complete blood count, and blood smears. Bone Marrow Biopsy. Lumbar Puncture & Medical Imaging(Metastasis). Pathological examination and Cytogenetics [9]. The method through microscopic images to detect and classify various cells (not limited to ALL cells) can be viewed as a kind of well-defined and investigated computer vision (CV) problems. Recently, with developing of deep learning (DL) based techniques, researches have investigated the potential use of applying DL methods in cell image classification problems.

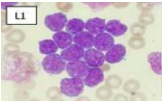
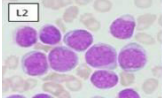
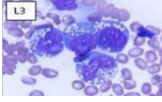
	<u>FAB Subtype</u>	<u>Cell Type</u>	<u>Characteristics</u>
	ALL - L1	T cell or pre-B cell	Small and homogeneous (uniform) cells
	ALL - L2	T cell or pre-B cell	Large and heterogeneous (varied) cells
	ALL - L3	B cell	Large and varied cells with vacuoles

Table 1: ALL types with affected cell types and cell characteristics.

1.1 Problem Statement

As an acute leukaemia, ALL progresses rapidly and is typically fatal within weeks or months if left untreated [7]. Diagnosing ALL begins with a thorough medical history, physical examination, complete blood count, and blood smears. Bone Marrow Biopsy. Lumbar Puncture & Medical Imaging (Metastasis). Pathological examination and Cytogenetics [9] and are subject to human bias. The two main challenges that we faced are:

- Due to low variance in our dataset classification proves to be a challenging task.
- Transfer learning renders to be ineffective for our specific dataset.

1.2 Aim

Our aim is to study the applicability of different available DL architectures and to devise a new, better model/architecture for ALL detection from microscopic ALL histopathological images. These images are provided as input to the DL models which classify the images as ALL cells or normal cells.

1.3 Main Objectives

Following are the objectives we strived to achieve during our project work:

1. Choosing the appropriate dataset.
2. Research current available architectures for image classification.
3. Applying the various architectures for classification.
4. Comparing the performance of various architectures.
5. Devising a new model for improved classification performance.

II. BACKGROUND VERIFICATION

Work on Computer Vision dates back to the 1960's with the research in Artificial Intelligence. While understanding the human perception researchers tried to build similar systems for computers. Computer Vision really came into the limelight after the arrival of large scale computing power along with easy access to massive amounts of data. GPUs were instrumental in development of accurate Object Recognition Model's as they worked in parallel and could speed up large complex calculations. Object classification and detection are two fundamental problems in computer vision and pattern recognition. They play fundamental and crucial roles in many applications, e.g., intelligent visual surveillance, image and video retrieval and web content analysis.



Figure 1: Object Detection, Localization and Classification/Recognition.

In the last few years, the field of machine learning has made tremendous progress on addressing these difficult problems. In particular, we've found that a kind of model called a deep convolutional neural network can achieve reasonable performance on hard visual recognition tasks -- matching or exceeding human performance in some domains. Researchers have demonstrated steady progress in computer vision by validating their work against ImageNet -- an academic benchmark for computer vision.

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in visual object recognition, object detection and many other domains. Deep learning discovers intricate structure in large datasets. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio [10].



Figure 2: Example of Classification/Recognition on IMAGENET.

III. WORK CARRIED OUT

The idea of the project came out of the intention to make a difference in people's lives. Having already delved in the python programming language and machine learning, we already had the resources that we needed to carry out the work. But, the problem to work upon still remained a question. Since our purpose was to make human life better, medicinal domain attracted our attention. Diagnosing a disease at a treatable moment would lead to saving of many lives. With this motivation we researched about different types of cancers for months and at last came to ALL. ALL having unfavorable prognosis if detected late makes it necessary to have early diagnosis. Having the tools, problem and the motivation, we embarked on a journey to reach our set target. In this section we will document all our efforts and the work we undertook to go from the beginning to the end of our project.

We divided the project work into six phases:

1. Research and analysis.
2. Finding/Selecting an image dataset.
3. Image pre-processing.
4. Build image classification models.
5. Network Training.
6. Compare model accuracies on test set.

3.1 Research and analysis

The research work progressed chronologically in the following order:

1. Research in to the various domains of Computer Vision, get a broad perspective of the field and get familiar with the technical language of machine learning specifically deep learning.
2. Perform background survey on object recognition.
3. Understand the state-of-the-art techniques used for image classification.
4. Know the common tools, languages, libraries and frameworks used to build Image Recognition Models.

Since this field is rapidly evolving and active research is being published every other day, we found open published research papers to be our best guide along with helpful blog posts by experts.

3.1.1 Gradient Descent

Gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks. Gradient descent is a way to minimize an objective function (θ) parameterized by a model's parameters $\theta \in R$ by updating the parameters in the opposite direction of the gradient of the objective function $\nabla \theta(\theta)$ w.r.t. to the parameters. The learning rate η determines the size of the steps we take to reach a (local) minimum. In other words, we follow the direction of the slope of the surface created by the objective function downhill until we reach a valley.

1. Batch gradient descent

Batch gradient descent computes the gradient for the entire training dataset and hence can be very slow and intractable for datasets that do not fit in memory. It also does not allow us to update our model online, i.e. with new examples on-the-fly.

```
for i in range(nb_epochs):
    params_grad = evaluate_gradient(loss_function , data, params)
    params = params - learning_rate * params_grad
```

2. Stochastic gradient descent

Stochastic gradient descent (often shortened to SGD), also known as incremental gradient descent, is an iterative method for optimizing a differentiable objective function, a stochastic approximation of gradient descent optimization.

```
for i in range(nb_epochs):
    np.random.shuffle(data)
    for example in data:
        params_grad = evaluate_gradient(loss_function , example , params)
        params = params - learning_rate * params_grad
```

3. Mini batch gradient descent

Mini-batch gradient descent takes the best of both worlds and performs an update for every mini-batch of n training examples.

```
for i in range(nb_epochs):
    np.random.shuffle(data)
    for batch in get_batches(data, batch_size=50):
        params_grad = evaluate_gradient(loss_function , batch , params)
        params = params - learning_rate * params_grad
```

3.1.2 Backpropagation

Multilayer Neural Network architectures can be trained by simple stochastic gradient descent. As long as the modules are relatively smooth functions of their inputs and of their internal weights, one can compute gradients using the backpropagation procedure. The backpropagation procedure to compute the gradient of an objective function with respect to the weights of a multilayer stack of modules is nothing more than a practical application of the chain rule for derivatives.

The key insight is that the derivative (or gradient) of the objective with respect to the input of a module can be computed by working backwards from the gradient with respect to the output of that module (or the input of the subsequent module). The backpropagation equation can be applied repeatedly to propagate gradients through all modules, starting from the output at the top (where the network produces its prediction) all the way to the bottom (where the external input is fed). Once these gradients have been computed, it is straightforward to compute the gradients with respect to the weights of each module." [15].

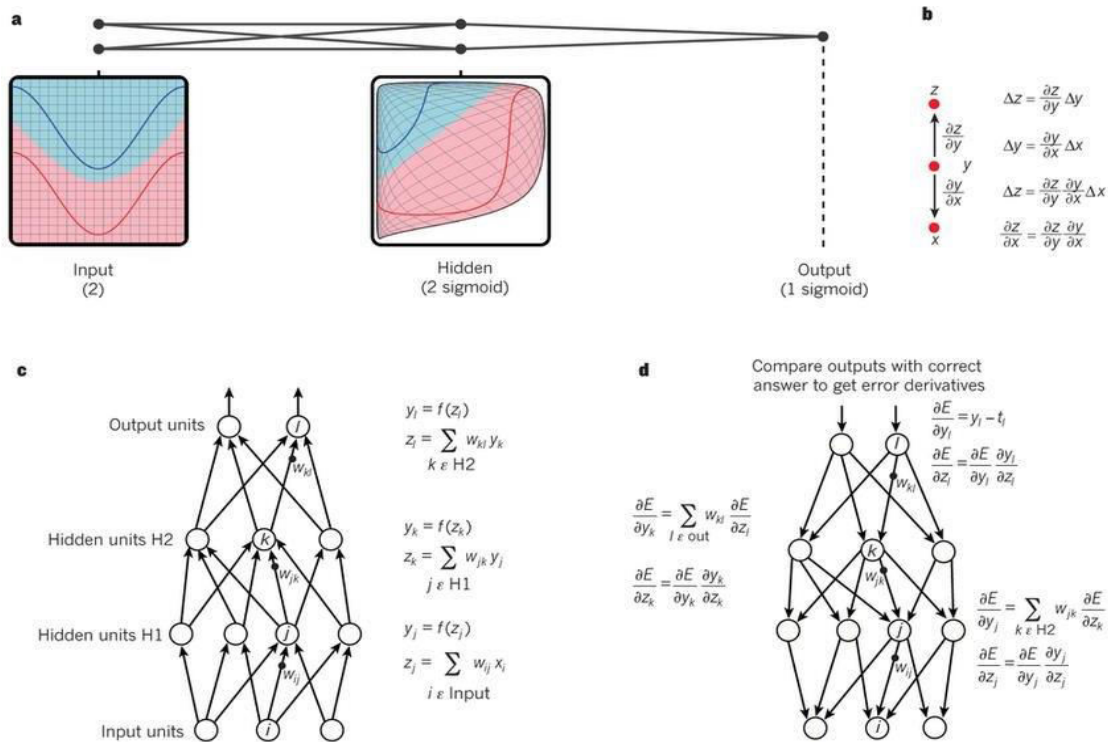


Figure3:Backpropagation

3.1.3 Activation Function

Activation functions are really important for an Artificial Neural Network They introduce non-linear properties to our Network. Their main purpose is to convert an input signal of a node in a NN to an output signal. That output signal now is used as a input in the next layer in the stack. Specifically in a NN we do the sum of products of inputs (X) and their corresponding Weights (W) and apply a Activation function (X) to it to get the output of that layer and feed it as an input to the next layer.

ReLU

ReLU has become very popular in the past couple of years. It was recently proved that it had 6 times improvement in convergence from Tanh function. It avoids and rectifies the vanishing gradient problem in Sigmoid and Tanh but can suffer from dead neurons during training. To fix the problem of dead neurons Leaky ReLU was introduced. Almost all deep learning Models use ReLU nowadays. It's used in hidden layers only and not in the output layer.

Mathematical form of ReLU is simple and efficient:

$$f(x) = \max(0, x)$$

Almost all deep learning Models use ReLU nowadays. It's used in hidden layers only and not in the output layer.

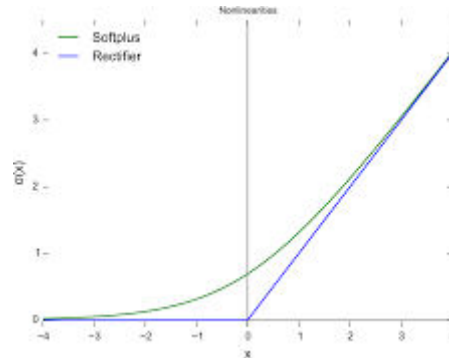


Figure 4: ReLU.

Softmax

For output layers use a Softmax function for a classification problem to compute the probabilities for the classes. Softmax assigns decimal probabilities to each class in a multi-class problem. Those decimal probabilities must add up to 1.0.

3.1.4 Loss functions

In a supervised learning problem, we have to find the error between the actual values and the predicted value. There can be different metrics which can be used to evaluate this error. This metric is often called loss function or cost function or objective function. Loss functions are helpful to train a neural network. Given an input and a target, they calculate the loss, i.e. difference between output and target variable.

3.1.5 Convolutional Neural Network

A convolutional neural network (CNN) is used to progressively extract higher- and higher-level representations of the image content. Instead of pre-processing the data to derive features like textures and shapes, a CNN takes just the image's raw pixel data as input and "learns" how to extract these features, and ultimately infer what object they constitute. To start, the CNN receives an input feature map: a three-dimensional matrix where the size of the first two dimensions corresponds to the length and width of the images in pixels. The size of the third dimension is 3 (corresponding to the 3 channels of a color image: red, green, and blue). The CNN comprises a stack of modules, each of which performs three operations.

1. Convolution
2. Activation (usually ReLU)
3. Pooling

Convolution: A convolution extracts tiles of the input feature map, and applies filters to them to compute new features, producing an output feature map, or convolved feature (which may have a different size and depth than the input feature map). Convolutions are defined by two parameters: Size of the tiles that are extracted (typically 3x3 or 5x5 pixels). The depth of the output feature map, which corresponds to the number of filters that are applied. During a convolution, the filters (matrices the same size as the tile size) effectively slide over the input feature maps grid horizontally and vertically, one pixel at a time, extracting each corresponding tile. For each filter-tile pair, the CNN performs element-wise multiplication of the filter matrix and the tile matrix, and then sums all the elements of the resulting matrix to get a single value. Each of these resulting values for every filter-tile pair is then output in the convolved feature matrix.

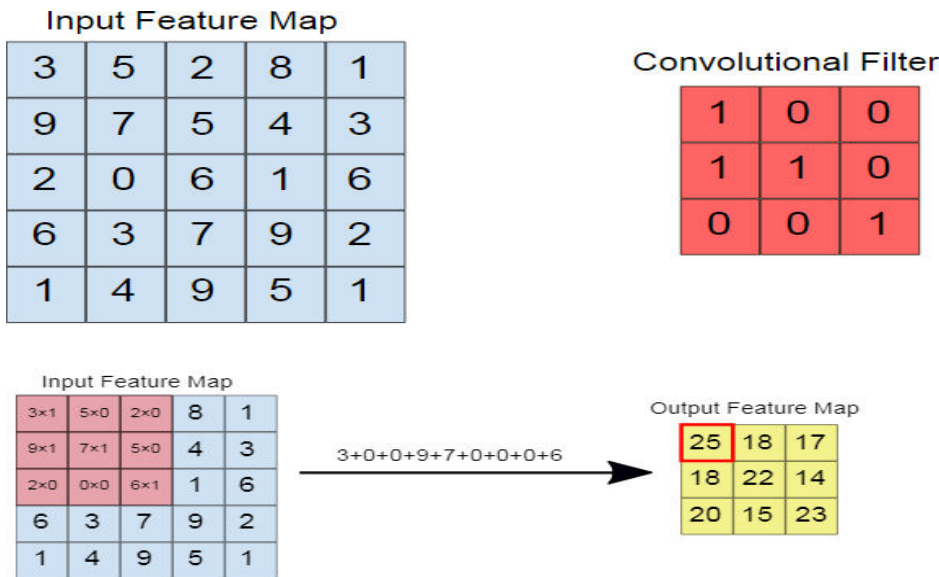


Figure 5: Convolution operation.

Activation – ReLU

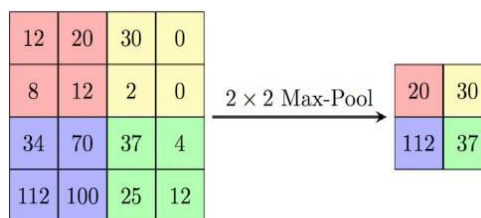
Following each convolution operation, the CNN applies a Rectified Linear Unit (ReLU) transformation to the convolved feature, in order to introduce nonlinearity into the model.

Pooling :After ReLU comes a pooling step, in which the CNN down samples the convolved feature (to save on processing time), reducing the number of dimensions of the feature map, while still preserving the most critical feature information. A common algorithm used for this process is called max pooling.

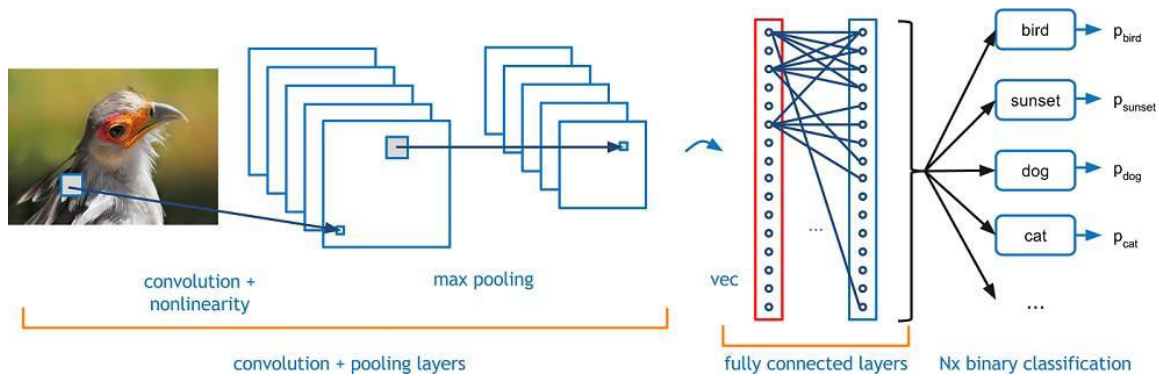
Max pooling operates in a similar fashion to convolution. We slide over the feature map and extract tiles of a specified size. For each tile, the maximum value is output to a new feature map, and all other values are discarded. Max pooling operations take two parameters:

- Size of the max-pooling filter (typically 2x2 pixels).
- Stride: the distance, in pixels, separating each extracted tile. Unlike with convolution, where filters slide over the feature map pixel by pixel, in max pooling, the stride determines the locations where each tile is extracted. For a 2x2 filter, a stride of 2 specifies that the max pooling operation will extract all non-overlapping 2x2 tiles from the feature map.

Figure 6: MaxPooling



Fully Connected: At the end of a convolutional neural network are one or more fully connected layers (when two layers are "fully connected," every node in the first layer is connected to every node in the second layer). Their job is to perform classification based on the features extracted by the convolutions. Typically, the final fully connected layer contains a Softmax activation function, which outputs a probability value from 0 to 1 for each of the classification labels the model is trying to predict.



2. Finding/Selecting an image dataset

We decided not to work on the All-IDB dataset since it's a fairly older dataset and consists of only a few 100 data points. Instead we decided to work on the ALL dataset that was released as part of an ISBI challenge on codalab's website. The dataset consists of 10,600 WBC nuclei microscopic images with an image size of 450 * 450. The main challenge faced with the classification of normal cells from malignant cells in this dataset is the close visual similarity between the images from the two classes.

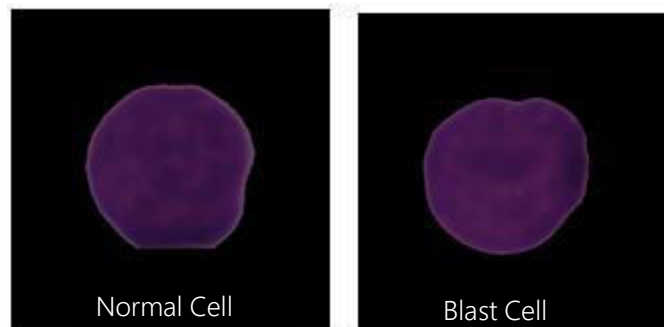


Figure 14: Dataset images with close visual similarities.

3.1 Data preparation and pre-processing

We organized our data in two folders with the folder names serving as the label for the images in them. insert data folder snippet here. Now we split our data into two sets with a split ratio of 8:2

Training set - 8,480 images & Testing set - 2,120 images. To get a good split, it is important to pick images randomly without any bias. Instead of doing this manually we built a custom python script which takes as input the dataset path, the output path, split ratio and optionally a random seed value. The script builds the train and test image datasets for all image categories given a split ratio randomly by first shuffling all the images randomly then splitting them into two halves. After splitting the dataset, we proceeded to pre-process the images for better and improved training all the while trying to avoid model over-fitting.

3.1.1 Augmentation

To increase the amount of training data for better classification and also to avoid overfitting, so that at training time, model will never see the exact same picture twice. We performed augmentation on our training dataset by applying random rotations and flips to them. This was done by using the "Augmentor" python library available on github. Using augmentor we increased the number of training images from 8,480 to 20,000.

3.1.2 Crop

The borders from the images were cropped out to the maximum possible amount while making sure not to remove any part from the nucleus image from any of the images. The images were cropped from their original size of 450 * 450 to a new size of 400 * 400. This was done so as to retain as much information as possible during the image resize, which

was necessary step owing to the resource constraints of our system.

3.2 Build image classification models

We decided to use 4 of the most popular available CNN models due to their popularity and good results for image classification tasks. The models we used are:

1. AlexNet.
2. VGG16.
3. InceptionV3.
4. ResNet50.

We also decided to design our own model due to unsatisfactory results of the aforementioned architectures.

3.3 Proposed Model

Owing to the unsatisfactory results from the available DL models, we decided to develop our own model for the classification task. Our model consists of 17 layers, of which 14 are convolutional layers - divided into 6 blocks.

Architecture

- 14 {convolution + relu} modules
- 5 {Max pooling} modules
- 3 {fully connected + fully connected} modules
- 2 {dropout} module
- 1 {softmax} module

Convolutions operate on 3x3 windows and max pooling layers operate on 2x2 windows.

First convolution block extracts 32 filters, second convolution block extracts 64 filters, third convolution block extracts 128 filters, fourth convolution block extracts 256 filters, fifth convolution block extracts 384 filters and the last block extracts 512 filters.

The model is built using the following layers:

- Conv2D - 2D Convolutional Network
- Activation – Relu
- Batch Normalization
- Max pooling
- Dense - Fully Connected Neural Network
- Softmax Layer as output

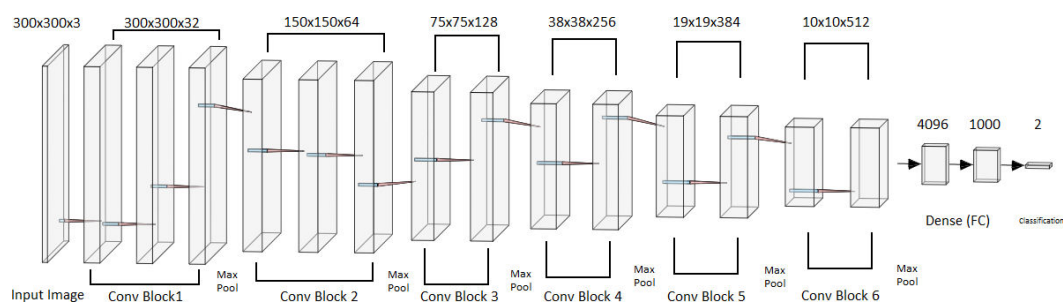


Figure 23: Proposed Architecture



3.1 Network Training

The model is compiled with training parameters set. The Activation function used for hidden layers is Non-Linear activation function “Relu”. The activation function used for the output layer is Probabilistic activation function “Softmax”. During training, SGD optimizer performed the best at a learning rate of 0.0002. The model ran for 100 epochs with the batch size of 30.

3.1.1 Visualize Network Layers

To see what kind of features the convnet has learned, we visualized how an input gets transformed as it goes through the convnet. The images go from the raw pixels to increasingly abstract and compact representations. The representations downstream start highlighting what the network pays attention to, and they show fewer and fewer features being “activated”; most are set to zero.

IV. RESULTS

The performance metric selected for model evaluation was accuracy. There are four possible outcomes of a classifier: true positives (when cancerous cells are correctly classified), true negatives (when non-cancerous cells are correctly identified), false positives (when non-cancerous cells are classified as cancerous) and false negatives (when cancerous cells are classified as non-cancerous).

$$Accuracy = \frac{\text{No. of correct Predictions}}{\text{Total no. of predictions}}$$

Callbacks were chosen to monitor the validation accuracy and save the model at best possible parametric weights. Considering the simplicity of the model architecture, an accuracy rate of 93.02% was achieved. Table 2 shows the performance comparison of the various different models and our model

Architecture	Accuracy(%)
AlexNet	81.62
VGG_16	84.8
ResNet50	84.77
InceptionV3	83.67
Proposed Model	93.02

V. CONCLUSION

Our results conclude that a simple architecture can be devised which classifies our data with more precision and accuracy. The classical architectures fail mainly due to high invariance in the dataset and thus being incapable of effective feature selection and classification. The main advantage of this model is that it is computationally efficient compared to the much complex architectures. The model also continues to grow effectively as the epochs are increased. We also found out that the of all the optimizers used, “SGD” gave the best classification result in all the configurations at a learning rate of 0.0002. “ReLU”, “Sigmoid” function was used as the activation functions for the convolution network layers and “Softmax” function was used as the activation function for the last Fully Connected (FC) layer for classification.



VI. FUTURE WORK

The results can further be improved by combining various similar models in an ensemble technique. Also, auto-encoders can be applied for weight matrix initialization. Implementing the same model with greater number of epochs in a faster GPU system might also enhance the models functionality. Further, additional image processing techniques may also be explored for data preparation. Standardizing the image acquisition and pre-processing standards can, in effect, make the model universally applicable.

REFERENCES

1. "Cancer". World Health Organization. 12 September 2018. Retrieved 19 December 2018.
2. "Defining Cancer". National Cancer Institute. 17 September 2007. Retrieved 28 March 2018.
3. Vos, Theo, et al. "Global, regional, and national incidence, prevalence, and years lived with disability for 310 diseases and injuries, 1990–2015: a systematic analysis for the Global Burden of Disease Study 2015." *The Lancet* 388.10053 (2016): 1545-1602.
4. Bernard, W. "Stewart and Christopher P. Wild.: World cancer report, ISBN 978-92-832-0429-9." (2014).
5. Wang, Haidong, et al. "Global, regional, and national life expectancy, all-cause mortality, and cause-specific mortality for 249 causes of death, 1980–2015: a systematic analysis for the Global Burden of Disease Study 2015." *The Lancet* 388.10053 (2016): 1459-1544.
6. "Childhood Acute Lymphoblastic Leukemia Treatment". National Cancer Institute. 8 December 2017. Retrieved 20 December 2017.
7. Marino, Bradley S.; Fine, Katie S. (2013). *Blueprints Pediatrics*. Lippincott Williams & Wilkins. p. 205. ISBN 9781451116045
8. "Key Statistics for Acute Lymphocytic Leukemia (ALL)" American Cancer Society.
9. Ferri, Fred F. *Ferri's Clinical Advisor 2016 E-Book: 5 Books in 1*. Elsevier HealthSciences, 2015.
10. LeCun, Y., Y. Bengio, and G. Hinton. "Deep learning. *nature* 521." (2015).
11. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.
12. Vogado, Luis HS, et al. "Unsupervised leukemia cells segmentation based on multi-space color channels." 2016 IEEE International Symposium on Multimedia (ISM). IEEE, 2016.
13. Laosai, Jakkrich, and Kosin Chamnongthai. "Classification of acute leukemia using CD markers." 2016 8th International Conference on Knowledge and Smart Technology (KST). IEEE, 2016.
14. Laosai, Jakkrich, and Kosin Chamnongthai. "Acute leukemia classification by using SVM and K-Means clustering." 2014 International Electrical Engineering Congress (iEECON). IEEE, 2014.
15. LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *nature* 521.7553 (2015): 436.
16. <https://scikit-learn.org/stable/documentation.html>
17. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
18. He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
19. Srivastava, Rupesh Kumar, Klaus Greff, and Jürgen Schmidhuber. "Highway networks." arXiv preprint arXiv:1505.00



INNO SPACE
SJIF Scientific Journal Impact Factor
Impact Factor: 8.423



ISSN INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 **6381 907 438** **ijirset@gmail.com**



www.ijirset.com

Scan to save the contact details