# A

# PROJECT REPORT

# On

# CLASSIFICATION OF BLOOD CELL USING MACHINE LEARNING

# Submitted in partial fulfillment of the requirement for the award of degree of

# BACHELOR OF TECHNOLOGY

# In

# BIOMEDICAL ENGINEERING

# Submitted by

**K. MANASA - 17211A1112**

**B. TEJA - 17211A1102**

**CH. PREMSAI – 17211A1110**

**DEPARTMENT OF BIOMEDICAL ENGINEERING**

**B.V. RAJU INSTITUTE OF TECHNOLOGY Vishnupur, Narsapur, Medak District – 502313.**

**2020 - 2021**

# CERTIFICATE

This is to certify that the project titled “CLASSIFICATION OF BLOOD CELL USING MACHINE LEARNING”is a bonafide work submitted by K. Manasa (17211A1112) of **B. Tech IV year II Semester, Biomedical Engineering** in partial fulfilment of the requirements for the award of the degree by Jawaharlal Nehru Technological University, Hyderabad.

Head of the Department

Dr. H. Sujana

INTERNAL GUIDE

Mr. V. Sripathi Raja

Dr. Ibrahim Patel

External Examiner

EXTERNAL GUIDE

**DECLARATION BY THE CANDIDATE**

I, **KONDAMEEDI MANASA, BIYYALA TEJA, CHAKILAM PREM SAI,** bearing roll number **17211a112,17211A1102,17211A1110** hereby declare that the thesis entitled “**CLASSIFICATION OF BLOOD CELL USING MACHINE LEARNIG**” submitted to the Department of Biomedical Engineering, B. V. Raju Institute of Technology, in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Biomedical Engineering** is a record of the original work done by me and has not been submitted to any other institute or published elsewhere.

**Kondameedi Manasa**

**BIYYALA TEJA**

**CHAKILAM PREM SAI**

**Place:** Hyderabad

**Date:**

**ACKNOWLEDGEMENT**

We deem a distant pleasure to acknowledge our indebtedness to the following people who have helped us in completing our project.

Our profound thanks to Dr. K. Lakshmi Reddy, Principal, B. V. Raju Institute of Technology, Vishnupur, Narsapur, Medak district for his moral support and excellent facilities provided.

We express our sincere thanks to Dr.H.Sujana, Head of the Department of Biomedical Engineering, B. V. Raju Institute of Technology, for his bound co- operation, inspiring guidance,critical suggestions, advices and encouragements during entire period of the study.

Our sincere thanks to our guides, Mr. V. Sripathi Raja assistant professor , Dr. Ibrahim Patel, professor, Biomedical engineering, for consistently providing us the required knowledge and helped us in successfully completing our project.

It gives us great pleasure to express our grateful thanks to our counselor Dr.H.Sujana.

Head of the Department for providing moral support.

It gives us great pleasure to express our grateful thanks to our project coordinator Mr. M. Prem Kumar, Assistant professor for providing technical support.

We express a deep scene of gratitude to Mr. Prem Kumar, Dr. H. Sujana, Mrs.P. Jahnavi, Mrs. V. Jothsna, Dr. Ibrahim Patel, Mrs. A. Srilatha, Mrs. A. Sandhya, Mr. J. Manikyam, Mr. K. Kumar of B. V. Raju Institute of Technology, Vishnupur, Narsapur, Medak District.

Finally, it would be incomplete without thanking all our friends who helped us sharing their knowledge. Also, we are thankful to them who contributed for the successful completion of our project.

Thank you

|  |  |
| --- | --- |
| **Abstract** |  |
| **1. Introduction** |  |
| **1.1 Objective of project** |  |
| **2.literature survey** |  |
| |  | | --- | | 2.1 Existing System | | 2.2 Proposed System | |  |
| |  | | --- | | **3. Requirements** | | 3.1 Functional Requirements | | 3.2 Non Functional Requirements | |  |
| |  | | --- | | **4. Algorithm** | | 4.1 CNN Algorithm | |  |
| |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | **5. Software Requirements** | | 5.1 Hardware Requirements | | 5.2 Software Requirements | | |  |
| |  | | --- | | **6. Design** | | 6.1 Introduction | | 6.2 Architectural Design | |  |
| **7. Pseudo Code** |  |
| 8. implementation  Methods of implement |  |
| 9.Testing |  |
| **10.output screen**  **Result analysis** |  |
| **11.Conclusion** |  |
| **13.Referance** |  |
|  |  |

**LIST OF FIGURE**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |

**ABSTRACT**

In medical diagnosis blood cell count plays very important role. Increment or decrement in the count of blood cell causes many diseases to occur in the human body. There are different techniques of blood cell counting which involves conventional as well as automatic techniques. The conventional method of manual counting under microscope is time consuming and yields inaccurate results. Although there are hardware solutions such as the Automated Hematology Counter, developing countries are not capable of organizing such unaffordable expensive machines in every hospital laboratory in the country. As a solution to this problem, to provide a software-based cost effective and an efficient alternative in recognizing and analyzing blood cells. We have come up with a solution of using Machine Learning these tests. The number of blood cell count that is RBC & WBC count is then may be use to diagnose the patient as well as detection of abnormalities like leukemia. For this purpose, few pre- processing and post-processing techniques have been implemented on blood cells image in order to provide a much clearer and cleaner image.

**1.Introduction**

A complete blood cell (CBC) count is an important test often requested by medical professionals to evaluate health condition. The main three types of cells that constitute blood are red blood cells (RBCs), white blood cells (WBCs), and platelets. RBCs also known as erythrocytes are the most common type of blood cell, which consists of 40–45% of blood cells. Platelets also known as thrombocytes are also in huge number in blood. WBCs also known as leukocytes, are just 1% of total blood cells. RBCs carry oxygen to our body tissues and the amount of oxygen tissues receives is affected by the number of RBCs. WBCs fight against infections and platelets help with blood clotting. As these blood cells are huge in number, traditional manual blood cell counting system using haemocytometer is highly time consuming and erroneous and most of the cases accuracy vastly depends on the skills of a clinical laboratory analyst. Therefore, an automated process to count different blood cells from a smear image will greatly facilitate the entire counting process.

With the development of machine learning techniques, image classification and object detection applications are becoming more robust and more accurate. As a result, machine learning based methods are being applied in different fields. Particularly, deep learning methods are being applied in different medical applications such as abnormality detection and localisation in chest X-rays, automatic segmentation of the left ventricle in cardiac MRI and detection of diabetic retinopathy in retinal fundus photographs. Thus, it is worth to look into deep learning based methods that can be applied to identify and count the blood cells in the smear images.

We have come up with a deep learning based blood cell counting method has been proposed. We employ a deep learning based object detection method to detect different blood cells. Among the state-of-the-arts object detection algorithms such as regions with convolutional neural network

The detection of blood cell counting by the microscopic image of the human blood we can detect RBC,WBC and PLATELETS by using machine learning. From a microscopic image we will recognize the cells. We can also know the count of cells from the image. By using an image as input by using image detection process (Single Shot Multi Detector ). The bounding boxes identify the cells that present in the given input image. The Convolution neural network (CNN) helps to get high accuracy in object detection process.

The CNN is mainly used for the classification of images and detect the object from the given input. And also very effective for computer vision applications.

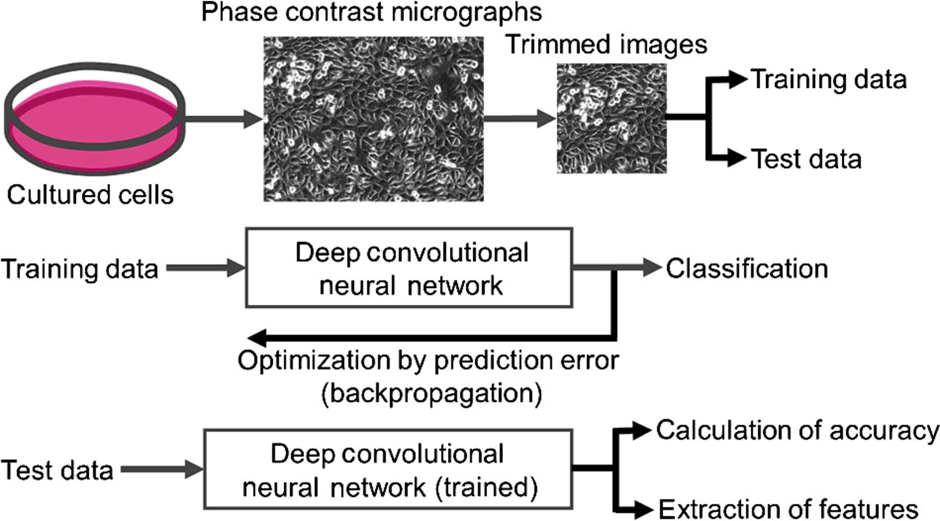


Fig ;- 1

**Objective of project:-**

The aim of this thesis is to detection of number of blood cells by using a microscopic blood image of human by getting the accuracy of 98% in less span of time. We hope this model used here and very useful for blood cell count and also it reduces the time

# 2. LITERATURE SURVEY

Blood cell detection using machine learning, obtains an accuracy of 98%. Using single shot multi detector box which is mainly used for the object detection process. The bounding boxes are used to detect the cells that present in the input image . In each input image the image is divided into bounding boxes and those boxes represent the cells. We get to know that the number of cells that are present in the each image. By using this model we get the accuracy of 98% in less span of time by using a microscopic human blood image as input. The Convolution neural network (CNN) helps to get high accuracy in object detection process. The CNN is mainly used for the classification of images and detect the object from the given input. And also very effective for computer vision applications. The implementation of machine learning algorithms, especially Convolutional Neural Networks (CNN), brings huge benefits to the medical field, where a huge number of images are to be processed and analyzed. A CNN-based framework is built to automatically classify the blood cell images into subtypes of the cells. Our method will bypass the stringent demand of constructing a good feature proﬁle by studying and analyzing the pixels of the microscopic images directly. As the procedure of hand-engineered features could be tricky but also critical, the expertise in diﬀerent domains is required. How-ever, sometimes the suggestion from the domain- based specialist may also have their own biases or subjective preferences. Our method takes the pixels of images. As these blood cells are huge in number traditional manual blood cell counting system using hemocytometer is highly time consuming and erroneous and most of the cases accuracy vastly depends on the skills of clinical laboratory analyst , an automated process to count different blood cells from a microscopic image will greatly facilitate the entire counting process. To adopt it for blood cells identification, we modify it for three classes consisting of WBC, RBC, and platelets. Due to modifying the class number, the number of filters in the final convolutional layer in the CNN architecture is needed to be changed as well. The main three types of cells that constitute blood are red blood cells (RBCs), white blood cells (WBCs), and platelets. RBCs also known as erythrocytes are the most common type of blood cell, which consists of 40–45% of blood cells of Haematology. Platelets also known as thrombocytes are also in huge number in blood. WBCs also known as leukocytes, are just 1% of total blood cells. RBCs carry oxygen to our body tissues and the amount of oxygen tissues receives is affected by the number of RBCs. Convolutional neural network architectures considering architecture complexity, reported accuracy, and running time with this framework and compare the accuracy of the models for blood cells detection. They also tested the trained model on smear images from a different dataset and found that the learned models are generalised. Overall the computer-aided system of detection and counting enables us to count blood cells from microscopic images .The variables may be morphological correlated to the colour , but the algorithm will take good care of them .

Machine Learning is the method paradigm that makes inferences from the available data using mathematical and statistical methods and makes predictions about the unknown with these inferences. Some academic research in the past have shown that after a certain stage, the machines must be learned the data. As a result of this, researchers carried out their studies in order to approach various problems by using various symbolic methods . A significant number of these approaches have ability to estimation, prediction and classification. In this section, the properties of machine learning algorithms that used in this study for classification of RBC,WBC and PLATELETS are mentioned.During training in each step, we record loss and moving average loss. Our proposed method is a machine learning approach where we use CNN algorithm for automatic identification and counting of blood cells. It includes a training model with a modified configuration where we change the final convolution layer for three outputs, identification of blood cells with an appropriate threshold, and count them from their labels

## **Python:**

The Python programming language is an Open Source, cross-platform, high level, dynamic, interpreted language.

The Python 'philosophy' emphasizes readability, clarity and simplicity, whilst maximizing the power and expressiveness available to the programmer. The ultimate compliment to a Python programmer is not that his code is clever, but that it is elegant. For these reasons Python is an excellent 'first language', while still being a powerful tool in the hands of the seasoned and cynical programmer.

Python is a very flexible language. It is widely used for many different purposes. Typical uses include:

* Web application programming with frameworks like Zope, Django and Turbogears
* System administration tasks via simple scripts
* Desktop applications using GUI toolkits like Tkinter or wxPython (and recently Windows Forms and IronPython)
* Creating windows applications, using the Pywin32 extension for full windows integration and possibly Py2exe to create standalone programs
* Scientific research using packages like Scipy and Matplotlib

#### **Good to know**

* The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

#### **Python Syntax compared to other programming languages**

* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpos

#### **What is Colaboratory?**

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.

#### **Is it really free to use?**

**Yes. Colab is free to use.**

**Users who are interested in more reliable access to better resources may be interested in Colab** [**Pro**](http://colab.research.google.com/signup?utm_source=faq&utm_medium=link&utm_campaign=seems_too_good).

#### **What is the difference between Jupyter and Colab?**

[**Jupyter**](https://jupyter.org/) **is the open source project on which Colab is based. Colab allows you to use and share Jupyter notebooks with others without having to download, install, or run anything.**

## **Existing System:**

* A.I. Shahin et al,(was author) in published year was 2019
* He title of journal was White Blood Cells Identification System Based on Convolutional Deep Neural Learning Networks
* Convolutional Neural Network (CNN) The overall system accuracy achieved by the proposed WBCs Net is (96.1%) and classified 5 healthy subtypes.it was advantage of it
* Clarity and size of the microscopic images affect the accuracy. The varied shapes of WBCs subtypes are also challenging it was disadvantage of it

## **Proposed system:**

we are using single shot detector (ssd) method to detect the total blood count in human beings through micro scope image of blood by implementing bounding boxes to identify the total cell present in given image after that we convert that image into pascol voc form after it changes to the code and we add that into the main code

# 

# 3. Requirements

#### **Functional Requirements:**

Accuracy of model should be high then only we can get perfect results

Need to be analyze the data and remove the unwanted data, if there is any missing values there need to remove those missing values or else has to put suitable for it

Feature selection is the major part of the data analysis get the perfect feature to build model.

## **Non Functional Requirements:**

The major non-functional Requirements of the system are as follows

#### **Usability:**

The system is designed with completely automated process hence there is no or less user intervention.

#### **Reliability:**

The system is more reliable because of the qualities that are inherited from the chosen platform java. The code built by using java is more reliable.

#### **Performance:**

This system is developing in the high level languages and using the advanced front-end and back- end technologies it will give response to the end user on client system with in very less time.

#### **Supportability:**

The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is having JVM, built into the system.

#### **Implementation:**

The system is implemented in web environment using struts framework. The apache tomcat is used as the web server and windows xp professional is used as the platform.

# 

# 4.ALGORITHM

**Convolution neural network;-**

**What is convolutional neural network?**

A Convolutional Neural Network, also known as CNN or ConvNet, is a class of neural networks that specializes in processing data that has a grid-like topology, such as an image. ... Each neuron works in its own receptive field and is connected to other neurons in a way that they cover the entire visual field.

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

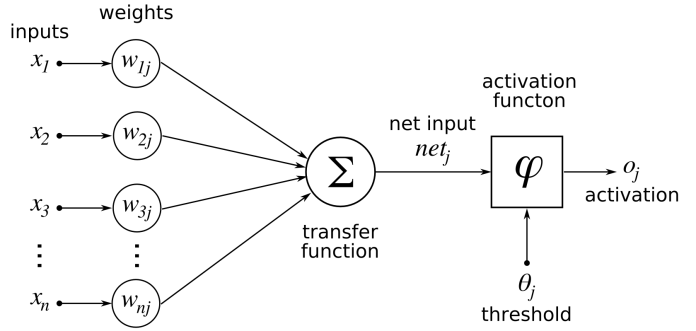
A Convolutional neural network (CNN) is a neural network that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and also for other auto correlated data. A convolution is essentially sliding a filter over the input.

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural network, most commonly applied to analyze visual imagery. ... CNNs are regularized versions of multilayer perceptrons**.**

**What are convolutional neural networks used for?**

**A Convolutional neural network (CNN) is a neural network that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and also for other auto correlated data. A convolution is essentially sliding a filter over the input**.

One of the main parts of Neural Networks is Convolutional neural networks (CNN). ... They are made up of neurons with learnable weights and biases. Each specific neuron receives numerous inputs and then takes a weighted sum over them, where it passes it through an activation function and responds back with an output

****

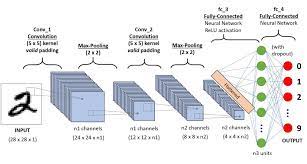
**Fig;-4.1**

The structure of an artificial neuron, the basic component of artificial neural networks

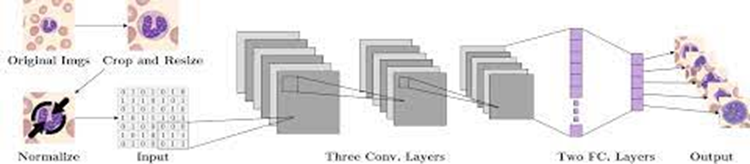
The behavior of each neuron is defined by its weights. When fed with the pixel values, the artificial neurons of a CNN pick out various visual features.

When you input an image into a ConvNet, each of its layers generates several activation maps. Activation maps highlight the relevant features of the image. Each of the neurons takes a patch of pixels as input, multiplies their color values by its weights, sums them up, and runs them through the activation function.

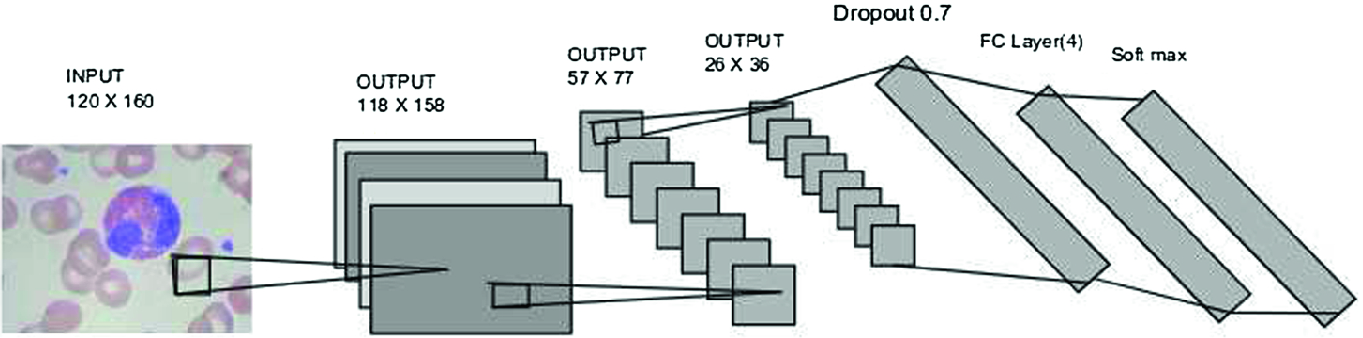
The first (or bottom) layer of the CNN usually detects basic features such as horizontal, vertical, and diagonal edges. The output of the first layer is fed as input of the next layer, which extracts more complex features, such as corners and combinations of edges. As you move deeper into the convolutional neural network, the layers start detecting higher-level features such as objects, faces, and more.

****

**Fig ;-4.2**

****

**Fig;-4.3**



**Fig;- 4.4**

**How does Python implement CNN?**

We have 4 steps for convolution:

1. Line up the feature and the image.
2. Multiply each image pixel by corresponding feature pixel.
3. Add the values and find the sum.
4. Divide the sum by the total number of pixels in the feature.

**How do you implement SSD object detection?**

SSD Object Detection extracts feature map using a base deep learning network, which are CNN based classifiers, and applies convolution filters to finally detect objects. Our implementation uses MobileNet as the base network (others might include- VGGNet, ResNet, DenseNet)

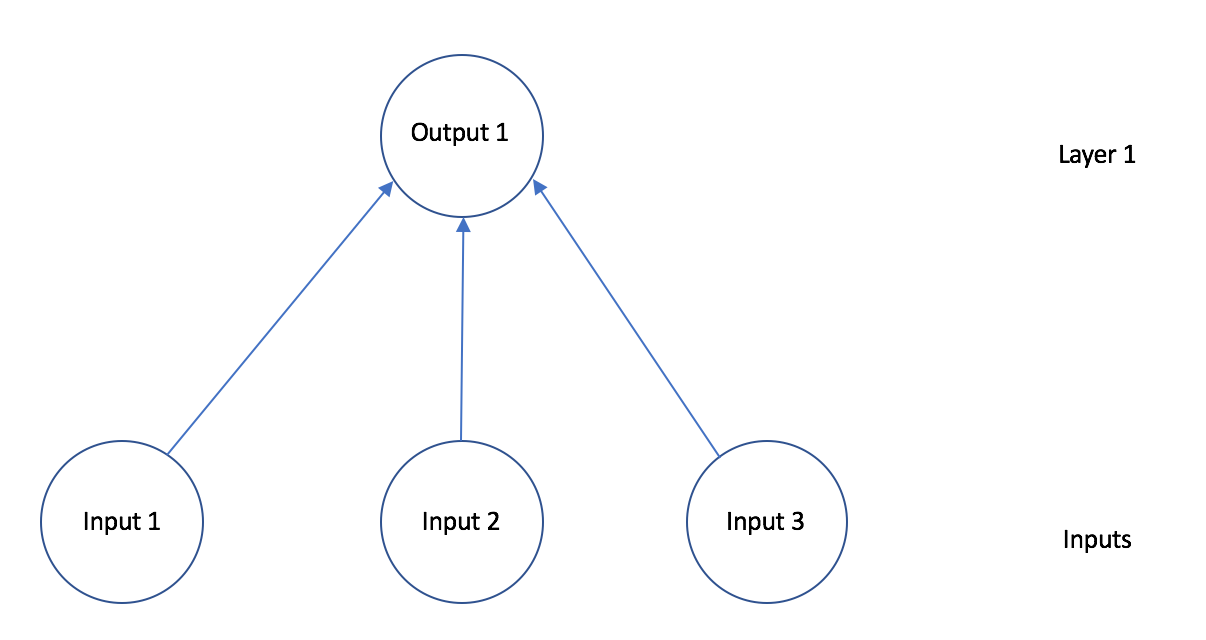
**How does CNN work?**

**One of the main parts of Neural Networks is Convolutional neural networks (CNN). ... They are made up of neurons with learnable weights and biases. Each specific neuron receives numerous inputs and then takes a weighted sum over them, where it passes it through an activation function and responds back with an output**.

**If you’ve heard about Artificial Intelligence, Machine Learning, or Deep Learning recently, then you might have heard of a Neural Network.**

**Neural Networks are a key piece of some of the most successful machine learning algorithms. The development of neural networks have been key to teaching computers to think and understand the world in the way that humans do. Essentially, a neural network emulates the human brain. Brains cells, or neurons, are connected via synapses. This is abstracted as a graph of nodes (neurons) connected by weighted edges (synapses).**

**So let’s dive in. What is a neural network? The human brain consists of 100 billion cells called neurons, connected together by synapses. If sufficient synaptic inputs fire to a neuron, that neuron will also fire. We call this process “thinking”. We can model this process by creating a neural network on a computer. A neural network has input and output neurons, which are connected by weighted synapses. The weights affect how much of the forward propagation goes through the neural network. The weights can then be changed during the back propagation — this is the part where the neural network is now learning. This process of forward propagation and backward propagation is conducted iteratively on every piece of data in a training data set. The greater the size of the data set and the greater the variety of data set that there is, the more that the neural network will learn, and the better that the neural network will get at predicting outputs.**

****

**Fig;- 4.5**

This neural network has one layer, three inputs, and one output. Any neural network can have any number of layers, inputs, or outputs.

Simply put, a neural network is a connected graph with input neurons, output neurons, and weighted edges. Let’s go into detail about some of these components:

1) Neurons. A neural network is a graph of neurons. A neuron has inputs and outputs. Similarly, a neural network has inputs and outputs. The inputs and outputs of a neural network are represented by input neurons and output neurons. Input neurons have no predecessor neurons, but do have an output. Similarly, an output neuron has no successor neuron, but does have inputs.

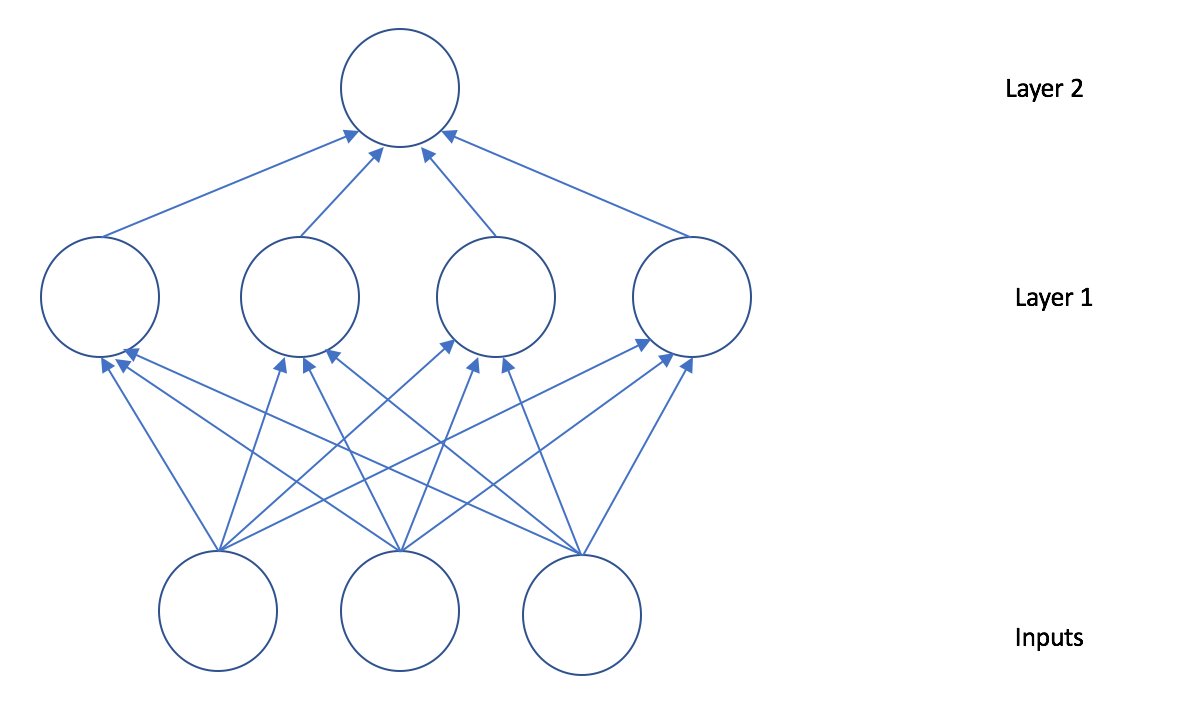
2) Connections and Weights. A neural network consists of connections, each connection transferring the output of a neuron to the input of another neuron. Each connection is assigned a weight.

3) Propagation Function. The propagation function computes the input of a neuron from the outputs of predecessor neurons. The propagation function is leveraged during the forward propagation stage of training.

4) Learning Rule. The learning rule is a function that modifies the weights of the connections. This serves to produce a favored output for a given input for the neural network. The learning rule is leveraged during the backward propagation stage of training.

**Deep Neural Networks**

So now that we know what a Neural Network is. What is a Deep Neural Network? A Deep Neural Network simply has more layers than smaller Neural Networks. A smaller Neural Network might have 1–3 layers of neurons. However, a Deep Neural Network (DNN) has more than a few layers of neurons. A DNN might have 20 or 1,000 layers of neurons.

****

**Fig;- 4.6**

This neural network has two layers, three inputs, and one output. Any neural network can have any number of layers, inputs, or outputs. The layers between the input neurons and the final layer of output neurons are hidden layers of a deep neural network.

That’s basically it for a Neural Network. A neural network is just a core architecture. There are different types of neural networks. For example, Convolutional Neural Networks have been very effective for Computer Vision applications. Recurrent Neural Networks are also very popular

**Artificial Neural Networks Are Inspired By Biological Neural Networks**

Just like biological neural network, artificial neural network is constantly learning and updating its knowledge and understanding of the environment based on experiences that it encountered.

An artificial neural network is simply a set of mathematical algorithms that work together to perform operations on the input. These operations then produce an output.

**Why Should We Use Neural Networks?**

Neural networks can help us understand relationships between complex data structures. The neural networks can use the trained knowledge to make predictions on the behavior of the complex structures.

Neural networks can be utilised to predict linear and non-linear relationships in data.

Neural networks can process images and even make complex decisions such as on how to drive a car, or which financial trade to execute next.

**What Are The Shortcomings?**

Although this is subjective but people have had hard time convincing business how neural networks have produced the answers. Hence the business users are slightly reluctant to trust its reasoning when compared to simple models like random forests and regression.

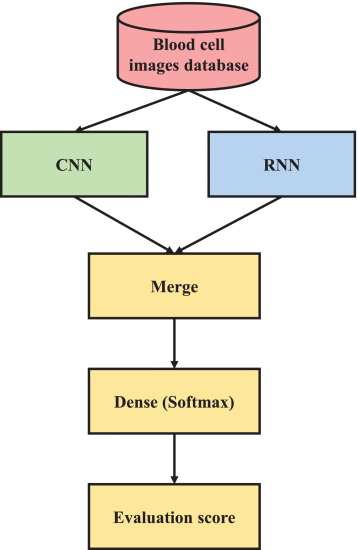
Although, neural networks can be sophisticated and can solve complex problems, they are slower than most machine algorithms. They can also end up over-fitting the training data.

**How do you implement SSD object detection?**

SSD Object Detection extracts feature map using a base deep learning network, which are CNN based classifiers, and applies convolution filters to finally detect objects. Our implementation uses MobileNet as the base network (others might include- VGGNet, ResNet, DenseNet)

**How does SSD algorithm work?**

SSD uses a matching phase while training, to match the appropriate anchor box with the bounding boxes of each ground truth object within an image. Essentially, the anchor box with the highest degree of overlap with an object is responsible for predicting that object's class and its location.

****

**We are using cnn**

**Fig;- 4.7**

# 

# 5.SOFTWARE AND HARDWARE REQUIREMENTS

## **Software Requirements:**

* + - Platform: python 3.7
    - IDE: Colaboratory

## **Hardware Requirements:**

* + - RAM: 4GB and Higher Processor: Intel i3 and above
    - Hard Disk: 500GB: Minimum

# 

# 

# 6.DESIGN

## **Introduction:**

Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation.

## **convolution neural work ;-**

In the last few years of the IT industry, there has been a huge demand for once particular skill set known as Deep Learning. Deep Learning a subset of Machine Learning which consists of algorithms that are inspired by the functioning of the human brain or the neural networks.

These structures are called as Neural Networks. It teaches the computer to do what naturally comes to humans. Deep learning, there are several types of models such as the Artificial Neural Networks (ANN), Autoencoders, Recurrent Neural Networks (RNN) and Reinforcement Learning. But there has been one particular model that has contributed a lot in the field of computer vision and image analysis which is the Convolutional Neural Networks (CNN) or the ConvNets.

CNNs are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analyzing visual images. Their applications range from image and video recognition, image classification, medical image analysis, computer vision and natural language processing.

The term ‘Convolution” in CNN denotes the mathematical function of convolution which is a special kind of linear operation wherein two functions are multiplied to produce a third function which expresses how the shape of one function is modified by the other. In simple terms, two images which can be represented as matrices are multiplied to give an output that is used to extract features from the image.

## **Basic Architecture**

**There are two main parts to a CNN architecture**

* A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction
* A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.

****

**Fig;- 6.1**

[**Dreaming to Study Abroad? Here is the Right program for you**](https://www.upgrad.com/machine-learning-ai-pgd-iiitb/?utm_source=BLOG&utm_medium=TEXTCTA&utm_campaign=DV_ML_PGD_BLOG_TEXTCTA_91673)

### **Convolution Layers**

There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers. When these layers are stacked, a CNN architecture will be formed. In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function which are defined below.

### **1. Convolutional Layer**

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).

The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

### **2. Pooling Layer**

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations.

In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer

### **3. Fully Connected Layer**

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

In this, the input image from the previous layers are flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place.

### **4. Dropout**

Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model’s performance when used on a new data.

To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network.

### **5. Activation Functions**

Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network.

It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. Each of these functions have a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred an for a multi-class classification, generally softmax us used.

**7.PSUEDO CODE**

**import numpy as np**

**import matplotlib.pyplot as plt import pandas as pd**

**import chainer**

**import cupy**

**import chainercv**

**import matplotlib**

**chainer.print\_runtime\_info()**

**print('ChainerCV:', chainercv.\_\_version\_\_)**

**print('matplotlib:', matplotlib.\_\_version\_\_)**

**import os**

**import xml.etree.ElementTree as ET**

**import numpy as np**

**from chainercv.datasets import VOCBboxDataset**

**bccd\_labels = ('rbc', 'wbc', 'platelets')**

**Requirement already satisfied: chainer>=6.0 in /usr/local/lib/python3.7/dist-packages (from chainercv) (7.4.0)**

**Requirement already satisfied: Pillow in /usr/local/lib/python3.7/dist-packages (from chainercv) (7.1.2)**

**Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from chainer>=6.0->chainercv) (3.7.4.3)**

**Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.7/dist-packages (from chainer>=6.0->chainercv) (1.19.5)**

**Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-packages (from chainer>=6.0->chainercv) (1.15.0)**

**Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from chainer>=6.0->chainercv) (3.0.12)**

**Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from chainer>=6.0->chainercv) (54.2.0)**

**Requirement already satisfied: protobuf>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from chainer>=6.0->chainercv) (3.12.4)**

**Building wheels for collected packages: chainercv**

**Building wheel for chainercv (setup.py) ... done**

**Created wheel for chainercv: filename=chainercv-0.13.1-cp37-cp37m-linux\_x86\_64.whl size=538294 sha256=7c7ae429027389d415bf98e8b7e59bc0ef178602f6025effd09bf05b36031e3c**

**Stored in directory: /root/.cache/pip/wheels/ea/10/01/e221beaa4b3d8341aa819a39ab8d4677457c79c81f521f3a94**

**Successfully built chainercv**

**Installing collected packages: chainercv**

**Successfully installed chainercv-0.13.1**

# 8.IMPLEMENTATION AND TESTING

## 

## **Method of Implementation:**

Machine learning (ML) may be defined as a subset of Artificial Intelligence that inculcates the ability of learning into a system on the basis of a data set used for the purpose of training in contrast to the normal approach of coding all possible outcomes beforehand. Multiple approaches and techniques are present to making systems which can learn. Some of them are neural networks, decision trees and clustering.

#### **ML Types:**

ML is to be broadly categorized under three categories namely – reinforcement learning, supervised learning and unsupervised learning and.

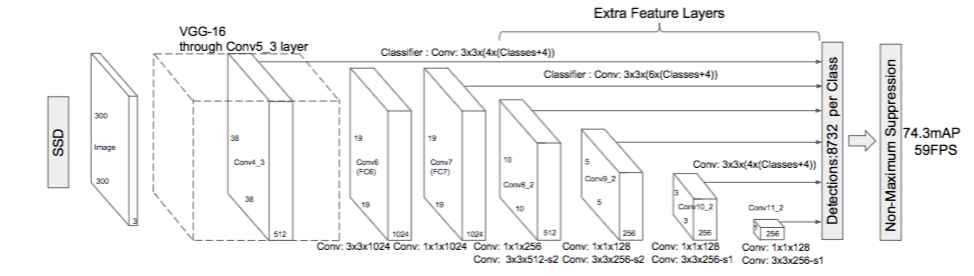
1. Supervised Learning: generates a function predicting outputs based on input observations. The function is generated from the training data and guides the system to produce useful epiphanies for new data sets introduced to the system.
2. Unsupervised Learning: Learning in this technique, the machine is forced to train from an unlabeled dataset and then differentiating it on the basis of some characters and allowing the algorithm to act on that information without external guidance.
3. Reinforcement Learning: The learning process continues from the environment in an iterative fashion. All possible system states are eventually learned by the system over a prolonged period of time.

**ssd**

The paper about SSD: Single Shot MultiBox Detector (by C. Szegedy et al.) was released at the end of November 2016 and reached new records in terms of performance and precision for object detection tasks, scoring over 74% mAP (mean Average Precision) at 59 frames per second on standard datasets such as PascalVOC and COCO. To better understand SSD, let’s start by explaining where the name of this architecture comes from:

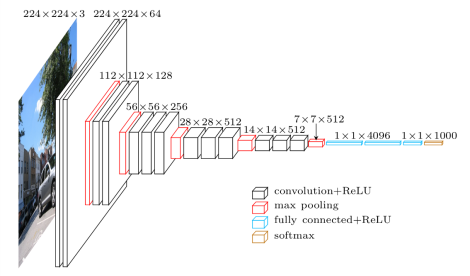
* **Single Shot:**this means that the tasks of object localization and classification are done in a single forward pass of the network
* **MultiBox:**this is the name of a technique for bounding box regression developed by Szegedy et al. (we will briefly cover it shortly)
* **Detector:**The network is an object detector that also classifies those detected objects

## **Architecture**

****

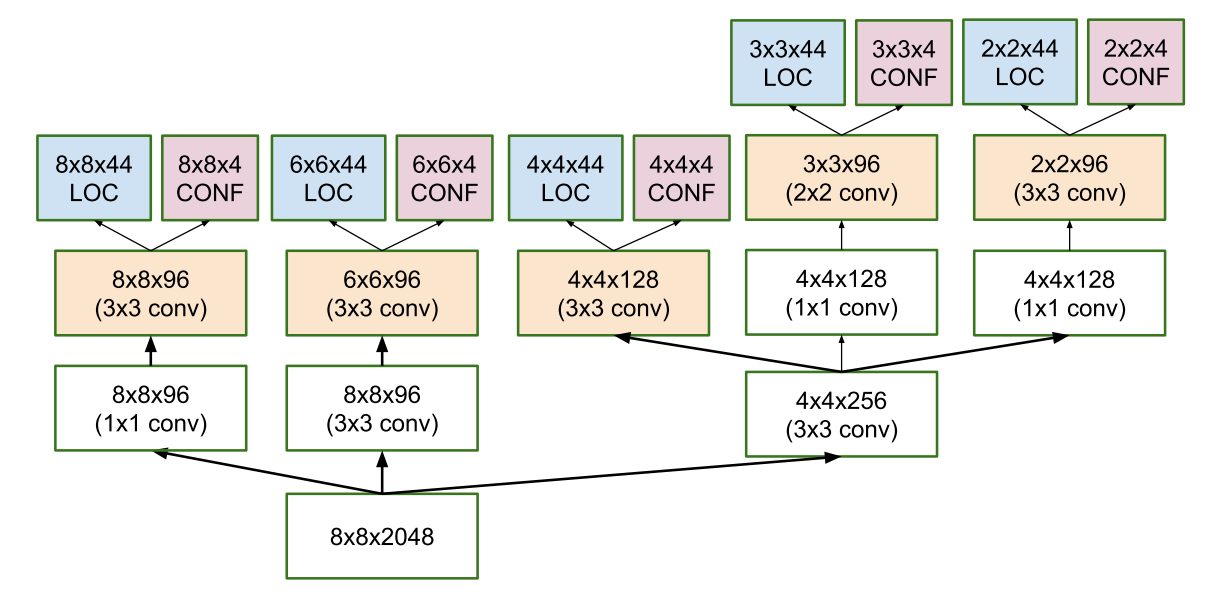
**Fig 8.1;-Architecture of Single Shot MultiBox detector (input is 300x300x3)**

As you can see from the diagram above, SSD’s architecture builds on the venerable VGG-16 architecture, but discards the fully connected layers. The reason VGG-16 was used as the base network is because of its strong performance in high quality image classification tasks and its popularity for problems where transfer learning helps in improving results. Instead of the original VGG fully connected layers, a set of auxiliary convolutional layers (from conv6 onwards) were added, thus enabling to extract features at multiple scales and progressively decrease the size of the input to each subsequent layer.

**   
Fig 8.2;-VGG architecture (input is 224x224x3)**

## **MultiBox**

The bounding box regression technique of SSD is inspired by Szegedy’s work on MultiBox, a method for fast class-agnostic bounding box coordinate proposals. Interestingly, in the work done on MultiBox an Inception-style convolutional network is used. The 1x1 convolutions that you see below help in dimensionality reduction since the number of dimensions will go down (but “width” and “height” will remain the same).

****

**Fig 8.3 ;- Architecture of multi-scale convolutional prediction of the location and confidences of multibox**

MultiBox’s loss function also combined two critical components that made their way into SSD:

* **Confidence Loss**: this measures how confident the network is of the objectness of the computed bounding box. Categorical cross-entropy is used to compute this loss.
* **Location Loss:**this measures how far away the network’s predicted bounding boxes are from the ground truth ones from the training set. L2-Norm is used here.

Without delving too deep into the math (read the paper if you are curious and want a more rigorous notation), the expression for the loss, which measures how far off our prediction “landed”, is thus:

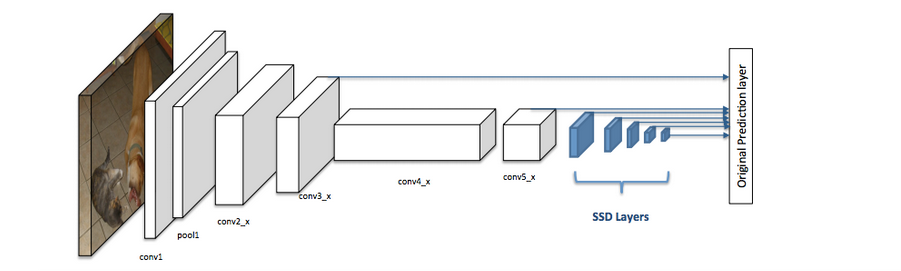
multibox\_loss = confidence\_loss + alpha \* location\_loss

The alpha term helps us in balancing the contribution of the location loss. As usual in deep learning, the goal is to find the parameter values that most optimally reduce the loss function, thereby bringing our predictions closer to the ground truth.

## **Single-Shot Detector (SSD) working ;-**

SSD has two components: a **backbone** model and **SSD head**. Backbone model usually is a pre-trained image classification network as a feature extractor. This is typically a network like ResNet trained on ImageNet from which the final fully connected classification layer has been removed. We are thus left with a deep neural network that is able to extract semantic meaning from the input image while preserving the spatial structure of the image albeit at a lower resolution. For ResNet34, the backbone results in a 256 7x7 feature maps for an input image. We will explain what feature and feature map are later on. The SSD head is just one or more convolutional layers added to this backbone and the outputs are interpreted as the bounding boxes and classes of objects in the spatial location of the final layers activations.

In the figure below, the first few layers (white boxes) are the backbone, the last few layers (blue boxes) represent the SSD head.

****

**Figure 8.4;- Architecture of a convolutional neural network with a SSD detector [2]**

**Next, let's go through the important concepts/parameters in SSD.**

### **Grid cell**

Instead of using sliding window, SSD divides the image using a grid and have each grid cell be responsible for detecting objects in that region of the image. Detection objects simply means predicting the class and location of an object within that region. If no object is present, we consider it as the background class and the location is ignored. For instance, we could use a 4x4 grid in the example below. Each grid cell is able to output the position and shape of the object it contains.

Now you might be wondering what if there are multiple objects in one grid cell or we need to detect multiple objects of different shapes. There is where anchor box and receptive field come into play.

### **Anchor box**

Each grid cell in SSD can be assigned with multiple anchor/prior boxes. These anchor boxes are pre-defined and each one is responsible for a size and shape within a grid cell. For example, the swimming pool in the image below corresponds to the taller anchor box while the building corresponds to the wider box**.**

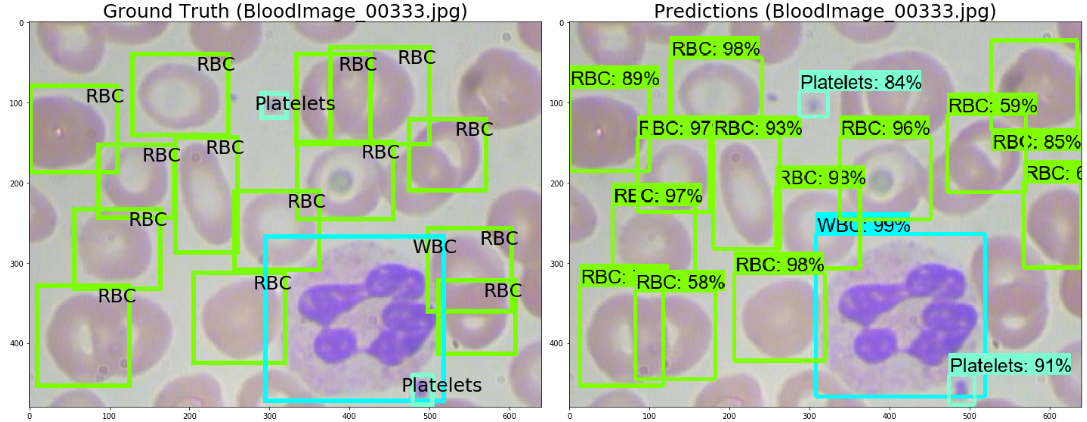
****

**Figure 8.5;- Example of two anchor boxes**

SSD uses a matching phase while training, to match the appropriate anchor box with the bounding boxes of each ground truth object within an image. Essentially, the anchor box with the highest degree of overlap with an object is responsible for predicting that object’s class and its location. This property is used for training the network and for predicting the detected objects and their locations once the network has been trained. In practice, each anchor box is specified by an aspect ratio and a zoom level.

#### **Aspect ratio**

Not all objects are square in shape. Some are longer and some are wider, by varying degrees. The SSD architecture allows pre-defined aspect ratios of the anchor boxes to account for this. The ratios parameter can be used to specify the different aspect ratios of the anchor boxes associates with each grid cell at each zoom/scale level.



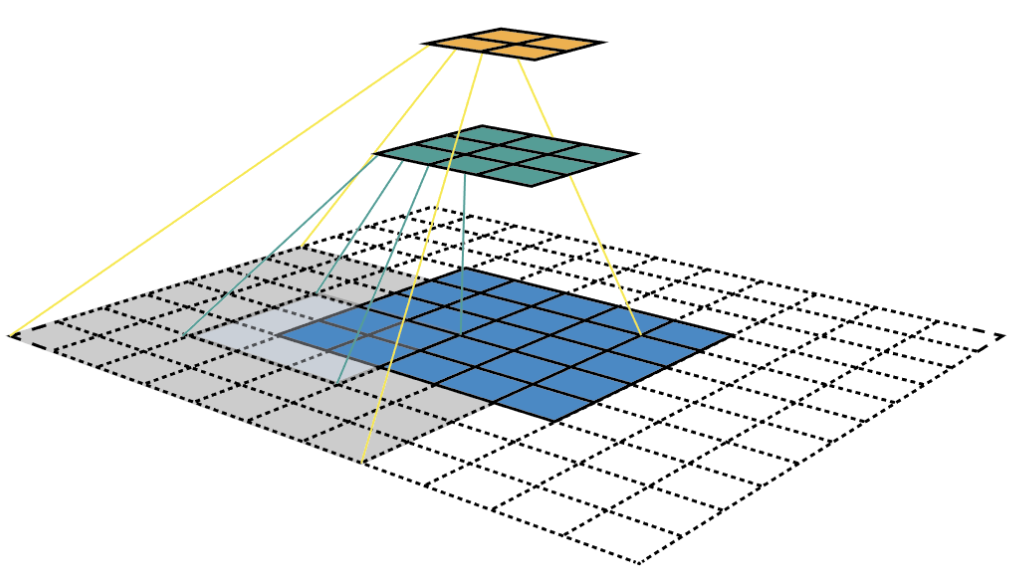
**Figure 8.6 ;- . The bounding box of building 1 is higher, while the bouding box for building 2 is wider**

#### **Zoom level**

It is not necessary for the anchor boxes to have the same size as the grid cell. We might be interested in finding smaller or larger objects within a grid cell. The zooms parameter is used to specify how much the anchor boxes need to be scaled up or down with respect to each grid cell. Just like what we have seen in the anchor box example, the size of building is generally larger than swimming pool.

### **Receptive Field**

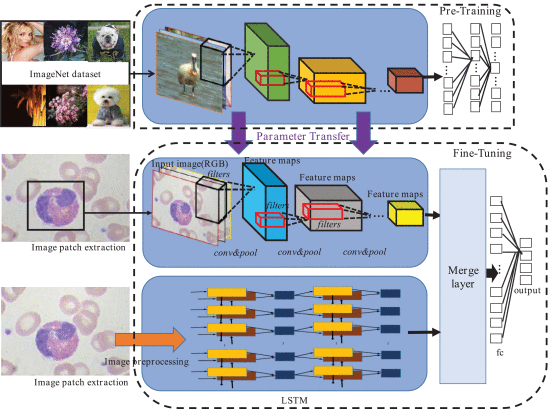
Receptive field is defined as **the region in the input space that a particular CNN’s feature is looking at (i.e. be affected by)**. We will use "feature" and "activation" interchangeably here and treat them as the linear combination (sometimes applying an activation function after that to increase non-linearity) of the previous layer at the corresponding location [3]. Because of the the convolution operation, features at different layers represent different sizes of region in the input image. As it goes deeper, the size represented by a feature gets larger. In this example below, we start with the bottom layer (5x5) and then apply a convolution that results in the middle layer (3x3) where one feature (green pixel) represents a 3x3 region of the input layer (bottom layer). And then apply the convolution to middle layer and get the top layer (2x2) where each feature corresponds to a 7x7 region on the input image. These kind of green and orange 2D array are also called **feature maps** which refer to a set of features created by applying the same feature extractor at different locations of the input map in a sliding window fastion. Features in the same feature map have the same receptive field and look for the same pattern but at different locations. This creates the spatial invariance of ConvNet.

****

**Figure 8.7;- Visualizing CNN feature maps and receptive field**

Receptive field is the central premise of the SSD architecture as it enables us to detect objects at different scales and output a tighter bounding box. Why? As you might still remember, the ResNet34 backbone outputs a 256 7x7 feature maps for an input image. If we specify a 4x4 grid, the simplest approach is just to apply a convolution to this feature map and convert it to 4x4. This approach can actually work to some extent and is exatcly the idea of YOLO (You Only Look Once). The extra step taken by SSD is that it applies more convolutional layers to the backbone feature map and has each of these convolution layers output a object detection results. **As earlier layers bearing smaller receptive field can represent smaller sized objects, predictions from earlier layers help in dealing with smaller sized objects**.

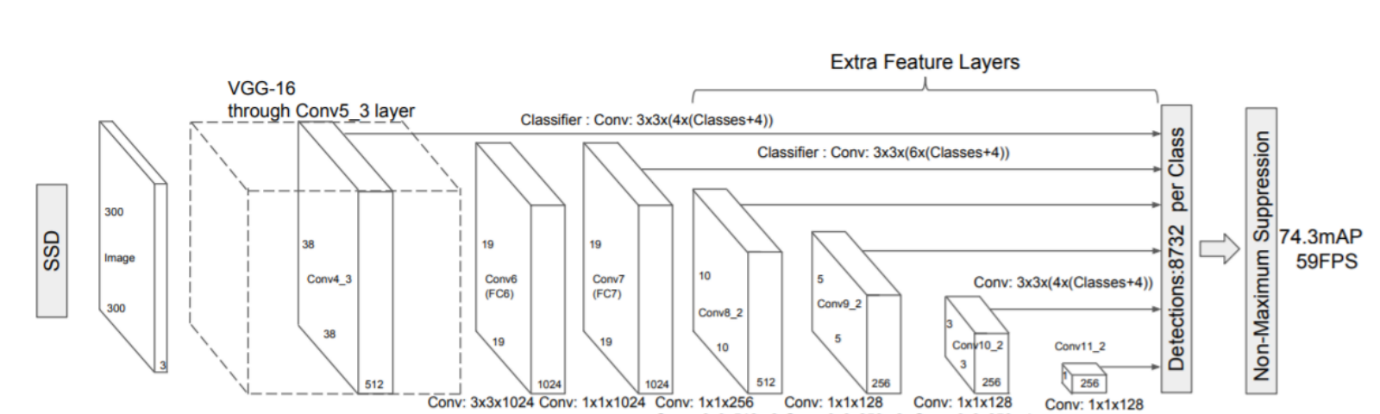
Because of this, SSD allows us to define **a hierarchy of grid cells** at different layers. For example, we could use a 4x4 grid to find smaller objects, a 2x2 grid to find mid sized objects and a 1x1 grid to find objects that cover the entire image.

****

**Fig;-8.4**

# ****What is Single Shot MultiBox Detector?****

Single Shot MultiBox Detector is a deep learning model used to detect objects in an image or from a video source. Single Shot Detector is a simple approach to solve the problem but it is very effective till now. SSD has two components and they are the **Backbone Model** and the **SSD Head**. Backbone Model is a pre-trained image classification network as a feature extractor. Usually, the fully connected classification layer is removed from the model. SSD Head is another set of convolutional layers added to this backbone and the outputs are interpreted as the bounding boxes and classes of objects in the spatial location of the final layer's activations.

****

**Fig -;8.5**

Instead of using a traditional sliding window algorithm, SSD divides the image as grids, and each grid cell responsible for detecting objects in that region of the image. If there is no object detected then we output it as nothing or to be more precise we will put a “0” indicating that there is no object found.

What if there are many objects of the same instance in a single image. This is where Anchor Box comes into play. Anchor Boxes are simple boxes that are assigned with multiple anchors/prior boxes, which are predefined and have fixed size and shape within the grid cell. Based on this we are able to detect multiple objects in an image.

# 

# 9.TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and attendant costs associated with a software failure are motivating factors for we planned, through testing. Testing is the process of executing a program with the intent of finding an error. The design of tests for software and other engineered products can be as challenging as the initial design of the product itself.

## **Training the convolutional neural network**

One of the great challenges of developing CNNs is adjusting the weights of the individual neurons to extract the right features from images. The process of adjusting these weights is called “training” the neural network.

In the beginning, the CNN starts off with random weights. During training, the developers provide the neural network with a large dataset of images annotated with their corresponding classes (cat, dog, horse, etc.). The ConvNet processes each image with its random values and then compares its output with the image’s correct label. If the network’s output does not match the label—which is likely the case at the beginning of the training process—it makes a small adjustment to the weights of its neurons so that the next time it sees the same image, its output will be a bit closer to the correct answer.

The corrections are made through a technique called backpropagation (or backprop). Essentially, backpropagation optimizes the tuning process and makes it easier for the network to decide which units to adjust instead of making random corrections.

Every run of the entire training dataset is called an “epoch.” The ConvNet goes through several epochs during training, adjusting its weights in small amounts. After each epoch, the neural network becomes a bit better at classifying the training images. As the CNN improves, the adjustments it makes to the weights become smaller and smaller. At some point, the network “converges,” which means it essentially becomes as good as it can.

After training the CNN, the developers use a test dataset to verify its accuracy. The test dataset is a set of labeled images that are were not part of the training process. Each image is run through the ConvNet, and the output is compared to the actual label of the image

. Essentially, the test dataset evaluates how good the neural network has become at classifying images it has not seen before.

If a CNN scores good on its training data but scores bad on the test data, it is said to have been “overfitted.” This usually happens when there’s not enough variety in the training data or when the ConvNet goes through too many epochs on the training dataset.

The success of convolutional neural networks is largely due to the availability of huge image datasets developed in the past decade. ImageNet, the contest mentioned at the beginning of this article, got its title from a namesake dataset with more than 14 million labeled images. There are other more specialized datasets, such as the MNIST, a database of 70,000 images of handwritten digits.

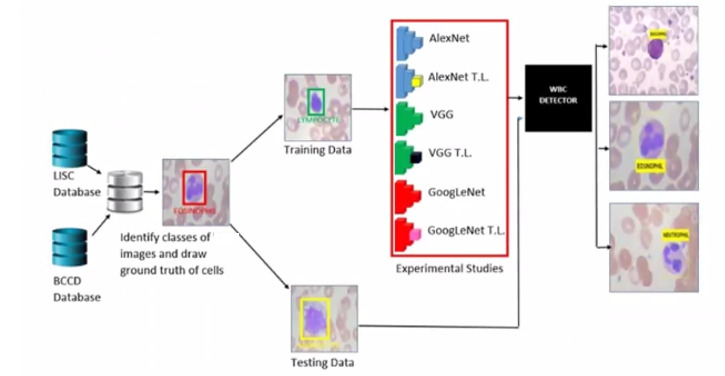
You don’t, however, need to train every convolutional neural network on millions of images. In many cases, you can use a pretrained model, such as the AlexNet or Microsoft’s ResNet, and finetune it for another more specialized application. This process is called transfer learning, in which a trained neural network is retrained a smaller set of new examples.

# Training & Running SSD

## **Datasets**

You will need training and test datasets with ground truth bounding boxes and assigned class labels (only one per bounding box). The Pascal VOC and COCO datasets are a good starting point.

**Images from Pascal VOC dataset**



**Fig;- 9.1**

blood

## **Default Bounding Boxes**

It is recommended to configure a varied set of default bounding boxes, of different scales and aspect ratios to ensure most objects could be captured. The SSD paper has around 6 bounding boxes per feature map cell**.**

****

**Fig;- 9.2**

blood

## **Feature Maps**

Features maps (i.e. the results of the convolutional blocks) are a representation of the dominant features of the image at different scales, therefore running MultiBox on multiple feature maps increases the likelihood of any object (large and small) to be eventually detected, localized and appropriately classified. The image below shows how the network “sees” a given image across its feature maps**:**

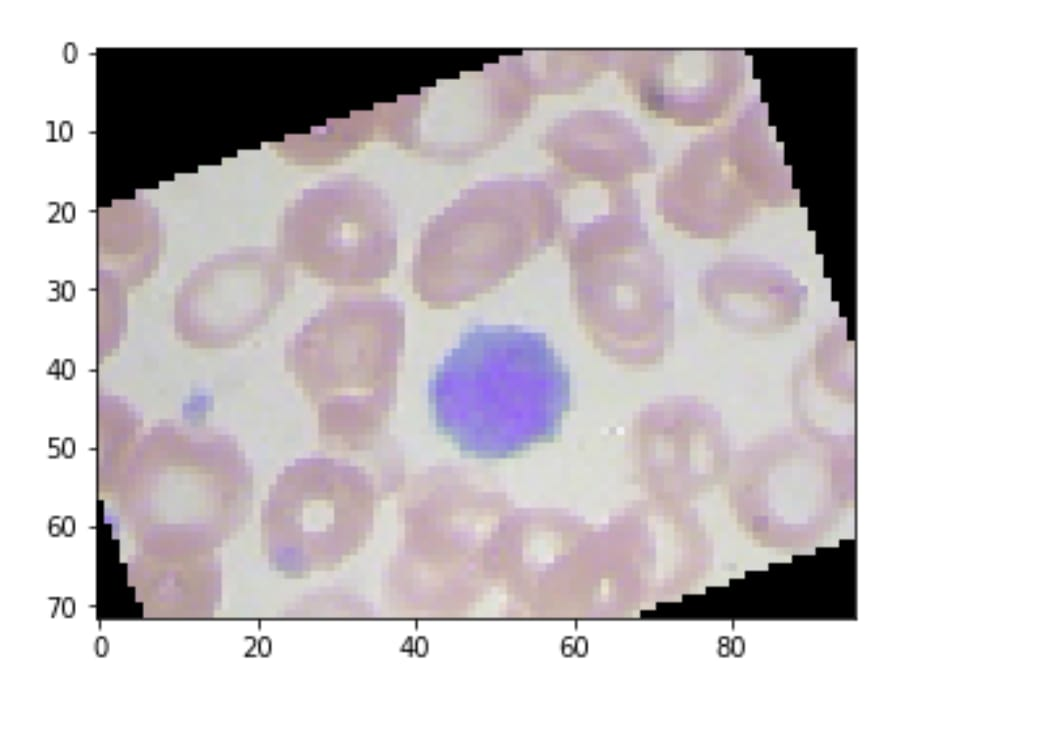
## **Hard Negative Mining**

During training, as most of the bounding boxes will have low IoU and therefore be interpreted as negative training examples, we may end up with a disproportionate amount of negative examples in our training set. Therefore, instead of using all negative predictions, it is advised to keep a ratio of negative to positive examples of around 3:1.

The reason why you need to keep negative samples is because the network also needs to learn and be explicitly told what constitutes an incorrect detection**.**

## **Data Augmentation**

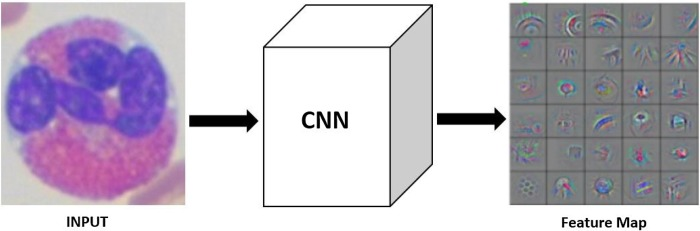
The authors of SSD stated that data augmentation, like in many other deep learning applications, has been crucial to teach the network to become more robust to various object sizes in the input. To this end, they generated additional training examples with patches of the original image at different IoU ratios (e.g. 0.1, 0.3, 0.5, etc.) and random patches as well. Moreover, each image is also randomly horizontally flipped with a probability of 0.5, thereby making sure potential objects appear on left and right with similar likelihood**.**



**Fig;- 9.3**

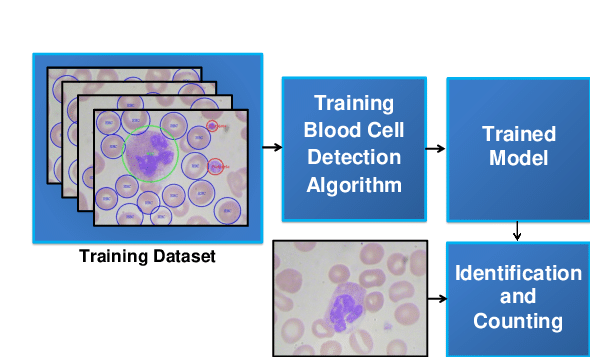
**Non-Maximum Suppression (NMS)**

Given the large number of boxes generated during a forward pass of SSD at inference time , it is essential to prune most of the bounding box by applying a technique known as non-maximum suppression: boxes with a confidence loss threshold less than ct (e.g. 0.01) and IoU less than lt (e.g. 0.45) are discarded, and only the top N predictions are kept. This ensures only the most likely predictions are retained by the network, while the more noisier ones are removed.

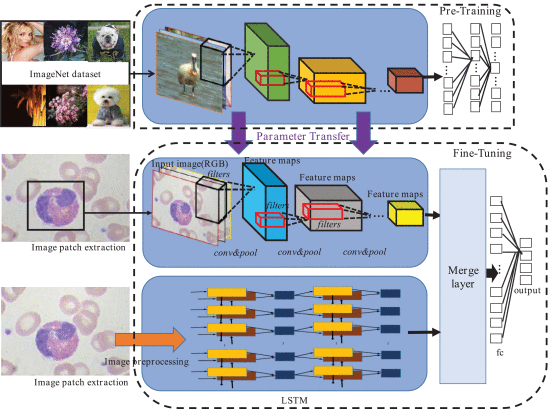


**Fig;- 9.4**

blood



**Fig;- 9.5**

****

**FIGURE 9.6**

**Combinatorial model that is fine-tuned based on human blood cell dataset with parameters transferred from another CNN pre-trained on the ImageNet dataset. The weight parameters of the blue area in the pre-trained model are migrated to the same position of another CNN model for fine-tuning on blood cell dataset.**

**Dataset Download of blood ;-**

* The first step is to prepare a dataset of blood microscopy images called [BCCD Dataset](https://github.com/Shenggan/BCCD_Dataset). This dataset contains 364 images and an XML file with a file name corresponding to each image. In each image, the coordinates of the bounding boxes surrounding the RBC, WBC, and platelet cells are stored. Since a single image may contain multiple cells, the XML file may contain descriptions of multiple cells.
* The BCCD Dataset is very small compared to other benchmark datasets that are widely used in object detection research, and is distributed on Github. Run the following cell to download the dataset first.
* After the download is complete, take a look at the file structure under the BCCD\_Dataset directory. This dataset is distributed with the following file structure.
* This structure follows the format of the Pascal VOC dataset, which has been used as a standard benchmark dataset for object detection for many years. Therefore, it is possible to use almost all of the classes that ChainerCV provides to easily handle the Pascal VOC dataset.
* There are actually other directories, but we will use only the ones included in the file tree above. The following is an explanation of what is included in each directory
* .\**Annotations directory: Contains the correct answer information for each cell image in the same format as the \**Pascal VOC dataset. The correct answer information is stored as an XML file, with the same file name except for the extension so that it can be easily matched to the image file.
* \**ImageSets directory: \**This directory contains a text file containing a list of images to be used for the training dataset (train), validation dataset (val), and testing dataset (test). According to these lists, the dataset is divided into three parts: the images listed in train.txt are used for training, the images listed in val.txt are used for validation (a dataset split used to roughly check the generalization performance during training), and the images listed in test.txt are used for training. The images listed in test.txt are used for validation (dataset splitting to roughly check generalization performance during training), and the images listed in test.txt are used for final performance evaluation after training.
* \**JPEGImages directory: \** Contains all the image data in this dataset.

Now, train\_dataset was an object of the BCCDDataset class, which inherits from VOCBboxDataset. Therefore, it should inherit the functions provided by the VOCBboxDataset class, except for the \_get\_annotations method overridden above. Let's check the documentation of the VOCBboxDataset class to see what features are provided: [VOCBboxDataset](<https://chainercv.readthedocs.io/en/stable/reference/> datasets.html?highlight=VOCBboxDataset#vocbboxdataset**)**

It contains a table like the one below. This dataset looks like a list, with each element having the following.

| Name | shape | dtype | format |
| --- | --- | --- | --- |
| Img | (3,H,W) | float32 | RGB, [0,255] |
| Bbox | (R,4) | float32 | (ymin,xmin,ymax,xmax) |
| Label | (R,) | int32 | [0,#fg\_class−1] |
| difficult (optional)\* | (R,) | bool | – |

* #fg\_class is the number of classes in the foreground
* difficult is valid only if return\_difficult = True.

However, since the return\_difficult option was not explicitly set to True when creating the dataset object, the default value of False is used. Therefore, the difficult element in the last row of the table above is not returned.

All three dataset objects we have created are three arrays, each of which contains (image data, list of correct bboxes, class for each bbox).

There are 19 bboxes in a row, and each bbox is represented by four numbers, (y\_min, x\_min, y\_max, x\_max). These four numbers represent the image coordinates (position in the image plane) of the upper left and lower right of the bbox.

One of the purposes of object detection is to output these four numbers for each object in the image. However, it is also necessary to output the class to which each bbox belongs (the type of object inside the bbox). The correct answer for this is in the last element. Let's display it.

There were 19 numbers in the box. Each of them corresponds to the bbox shown above (first\_datum[1]) in turn, and indicates to which class each bbox belongs (0: RBC, 1: WBC, 2: Platelet).

At the end of this section, let's try to visualize and check a dataset that is grouped by these three elements. We will take an image from the train dataset, its corresponding bbox, and the class labels for each of them, and use the convenient visualization functions provided by ChainerCV to display the image with the bounding box and the corresponding class name superimposed on it.

**Data augmentation implementation;-**

In deep learning, the ability to prepare a large amount of data has a significant impact on the generalization performance of the model. The technique of applying various transformations to images and their associated labels without changing the meaning of the data (data augmentation) is a method that can be used to augment the training data.

Below, we define a class that describes the transformations we want to apply to each data point in the training dataset. The transformations to be performed are the five described in the \_\_call\_\_ method. For example, you can change the color of the image, flip it horizontally, or scale it up or down without changing its meaning. Note that the correct answer labels must also be converted appropriately when doing so. For example, if the image is flipped horizontally, the correct answer label should also be flipped horizontally. Another useful technique is to mask or hide a part of the image. This allows recognition to be based on a variety of information instead of relying on only one piece of information.

**Evaluation metrics;-**

In object detection, the bbox that the model judges to be "detected" (given a certain level of confidence) is considered to be "True Positive" when it is actually equal to or greater than the correct bbox and IoU > 0.5. \*\*The average precision (AP) is generally used for evaluation. IoU is explained in the previous section on Semantic Segmentation, and IoU

in object detection is the same. The IoU in object detection is the size of the area enclosed by either or both the predicted rectangle and the correct rectangle divided by the size of the area enclosed by the common rectangle.

ChainerCV provides an extension called DetectionVOCEvaluator that calculates the APs for each class and the overall mAPs during training using the passed iterators (in this case, the iterator val\_iter created for the validation dataset). It calculates the AP for each class and the overall mAP during training. This extension is also used here.

**Inference with training results ;-**

The parameters of the model obtained by training are saved in a file by the Trainer extension named extensions.snapshot(). By default, the file is saved in the directory specified by the out argument passed when initializing the Trainer object. In this case, it should be under results

After executing the shell commands you should have found a file named snapshot\_epoch\_300.npz. This file contains all the parameters that were stored in Trainer during the training process and are needed to restart the training. Therefore, it also contains parameters other than those of the model itself, such as those of the Optimizer. In this tutorial, we will try to extract only the model parameters necessary for inference from this file.

One way to extract the model parameters is to use chainer.serializers.load\_npz to load the .npz file into the model object, and then specify that the keys in the .npz file should only be viewed below a certain level. If you take a snapshot of the whole Trainer object, you will see that it contains more than just the parameters of the model, such as the iteration count of the Optimizer, but if you pass the prefix updater/model:main/model, you will see only the parameters of the model. However, by passing the updater/model:main/model prefix, we can extract only the parameters of the model.

Now, let's create a new model object and load the trained parameters into it, assuming that we are given only this snapshot and the model definition code in a different location than the one used for training.

Now, let's try to detect cells in one of the test images using the model loaded with the trained weights. In the following code, we use ChainerCV to load the image, perform the inference, and visualize the results in sequence.

**Evaluate the Trained Model ;-**

Once training is complete, the resulting model can be evaluated on a test dataset. The validation dataset is not directly used to calculate the amount of parameter updates during training, but it is used to adjust hyperparameters such as the learning rate and the ratio and timing of learning rate decay. So, strictly speaking, we cannot call it data that is not used during training. \**Therefore, in order to get an idea of the generalization performance of the final model, it is necessary to evaluate it using a third dataset that is not included in either the training or validation datasets. \**.

The Evaluator, one of Chainer's Trainer Extensions, can be used independently of the Trainer, and the DetectionVOCEvaluator provided by ChainerCV also inherits from Chainer's The DetectionVOCEvaluator provided by TrainerCV is also an extended version of Evaluator, which inherits from Trainer's Evaluator, and can be used independently of Trainer for evaluation purposes.

Let's create an iterator using the test\_dataset that we prepared at the beginning, and pass it to the DetectionVOCEvaluator with the trained model that we used earlier to perform the final performance evaluation\*\* using the test dataset.

The results displayed here show that the prediction for white blood cells is the most accurate, followed by red blood cells, while the prediction for platelets is much lower than the other two. In such cases, we need to check whether platelets, red blood cells, and white blood cells appear in the dataset at the same frequency. If the frequency varies greatly from class to class, then the model is probably observing the less frequent class less often than the more frequent class. It is not optimal to train them completely in the same way (without distinguishing them).

When training an object detector for real-world applications, it is important to first train the detector on a well-known model and produce results, and then compare the results with the data to examine the prediction tendencies of the model and the characteristics of the data set itself.

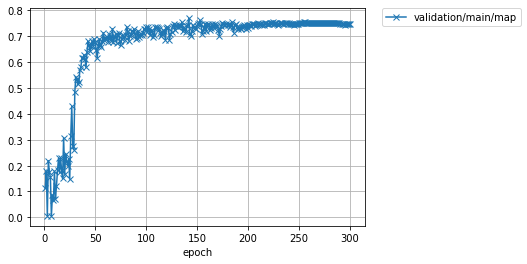
# 

# 10.OUTPUT SCREEN

333333

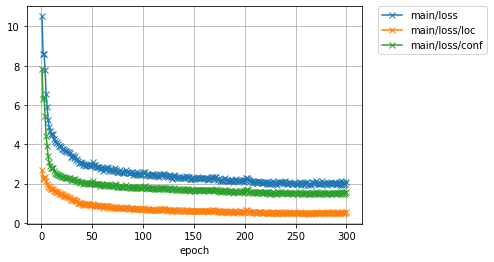
**Fig;- 10.1**

blood



**Fig;- 10.2**

blood



**Fig;- 10.3**

**Result Analysis;-**

the proposed methodology results in RBC, WBC and Platelets identiﬁcation, segmentation, and counting.To train and evaluate our model, we use a real dataset of microscopic blood smear images which includes 150 images. The loss function joins classiﬁcation, segmentation, and the generated mask loss functions. We notice that, during the training and testing,the loss functions are continuously decreasing and converging to near-zero values. This proves that the developed deeplearning model is well-trained without overﬁtting problem. CNN succeeds not only in cells identiﬁcation and segmentation but also incells classiﬁcation. Results indicate that ourproposed model efﬁciently detects 92% of the RBCs and 96%of the WBCs.

# 11.CONCLUSION

The blood cell disorders and the alteration in hematological parameters and indices arise by the reason of many conditions such as extreme physical stress or emotional stress, smoking, pregnancy, or even extreme exercise. Machine learning techniques are stand out in classification, segmentation, detection, and prediction of blood components and parameters. The references made known the different types of machine learning techniques implemented in blood components to identify the types and subtypes and the classifications and segmentation of blood cells from the peripheral blood smear images. In future enhancement, it will be necessary to consider the quality of PBS images and complex- overlapped cells.

# 

# 12. REFERENCES

[1] M. I. JordanandT. M. Mitchell, Machine learning: Trends, perspectives, and prospect, Science 17 Jul 2015, Vol. 349, Issue 6245, pp. 255-260

[2] Gao Huang et al, Semi-Supervised and Unsupervised Extreme Learning Machines,IEEE TRANSACTIONS ON CYBERNETICS,2168-2267 , 2014 IEEE.

[3] DaniloBzdok et al, Machine learning: supervised methods, Nature Methods volume15, pages5–6 (2018)

[4] Dmitry Krotov and John J. Hopfield, Unsupervised learning by competing hidden units,PNAS | April 16, 2019 | vol. 116 | no. 16 | 7723–7731

[5] Avital Oliver, Realistic Evaluation of Deep Semi-Supervised Learning Algorithms, 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada

[6] SATINDER SINGH et al, Machine Learning, 39, 287–308, 2000. Convergence Results for Single-Step On-Policy ReinforcementLearning Algorithms

[7] VolodymyrMnih et al, Human-level control through deep reinforcement learning,26 FEBRUARY 2015 | VOL 518 | NATURE | 529

[8] Joesph R. Wiencek et al, Detection of Nicotine and Nicotine Metabolites in Units of Banked Blood, American Journal Clinical Pathology 2019;XX:1-6

[9] S Asgary et al,Effects of cigarette smoke, nicotine and cotinine on red blood cell hemolysis and their -SH capacity,ExpClinCardiolVol 10 No 2 2005

[10] EKBERG-JANSSON, The expression of lymphocyte surface antigens in bronchial biopsies, bronchoalveolar lavage cells and blood cells in healthy smoking and never-smoking men, 60 years old,RESPIRATORY MEDICINE (2000) 94, 264-272

[11] RazaliTomari, Computer Aided System for Red Blood Cell Classification in Blood Smear Image,Procedia Computer Science 42 ( 2014 ) 206 – 213

[12] NiranjanChatap, SiniShibu, Analysis of blood samples for counting leukemia cells using Support vector machine and nearest neighbour,IOSR Journal of Computer Engineering (IOSR-JCE)

[13] SedatNazlibilek et al, Automatic Segmentation, Counting, Size Determination and Classification of White Blood Cells,Measurement 55,58–65(2014)

[14] R Shouval et al, 2014, Application of machine learning algorithms for clinical predictive modeling: a data-mining approach in SCT, Bone marrow Transplantation (2014) 49, 332–33

[15] Charles B. Delahunt et al, 2015, Automated Microscopy and Machine Learning for Expert-Level Malaria Field Diagnosis, IEEE 2015 Global Humanitarian Technology Conference

[16] RomelBhattacharjee, et al, 2015, Robust Technique for the Detection of Acute Lymphoblastic Leukemia, 2015 IEEE Power, Communication and Information Technology Conference (PCITC) Siksha ‘O’ Anusandhan University, Bhubaneswar, India.

[17] Syed HamadShirazi, Accurate Microscopic Red Blood Cell Image Enhancement and Segmentation,IWBBIO 2015, Part I, LNCS 9043, pp. 183–192, 2015.

[18] John A. Quinn et al,Deep Convolutional Neural Networks for Microscopy-Based Point of Care Diagnostics,Proceedings of International Conference on Machine Learning for Health Care 2016 JMLR W&C Track Volume 56

[19] Hany A Elsalamony, Healthy and unhealthy red blood cellDetection in human blood smears using neural networks,Micron 2016

[20] HanSangPark,Automated Detection of P.falciparum Using Machine Learning Algorithms with Quantitative Phase Images of Unstained Cells,PLOSONE,September16,2016