A Project Report on

GASTROINTESTINAL CARCINOMA CLASSIFICATION USING CNN

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

Bachelor of Technology

In

Computer Science and Engineering

Submitted by

CH SAHITHI	**(18P31A0516)**
W VENU GOPAAL	**(18P31A0559)**
V KRISHNA PRAKASH	**(18P31A0555)**
J N S R S SRINIVAS	**(18P31A0527)**

Under the esteemed supervision of

Mrs. J L SARWANI THEEPARTHI, M. Tech, [Ph.D]

Assistant Professor

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to JNTUK, Kakinada)

Surampalem, East Godavari District

Andhra Pradesh - 533 437

2018-2022

Aditya College of Engineering & Technology

Aditya Nagar, ADB Road, Surampalem - 533437

VISION

To induce higher planes of learning by imparting technical education with

- ✓ International standards
- ✓ Applied research
- ✓ Creative Ability
- ✓ Value based instruction and to emerge as a premiere institute.

MISSION

Achieving academic excellence by providing globally acceptable technical education by forecasting technology through

- ✓ Innovative Research And development
- ✓ Industry Institute Interaction
- ✓ Empowered Manpower

Principal

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY SURAMPALEM

PRINCIPAL
Aditya College of
Engineering & Technology
SURAMPALEM- 533 437

Department of Computer Science and Engineering

VISION

- ✓ To become a center for excellence in Computer Science and Engineering education and innovation.

MISSION

- ✓ Provide state of art infrastructure.
- ✓ Adapt Skill based learner centric teaching methodology.
- ✓ Organize socio-cultural events for better society.
- ✓ Undertake collaborative works with academia and industry.
- ✓ Encourage students and staff self-motivated, problem-solving individuals using Artificial Intelligence.
- ✓ Encourage entrepreneurship in young minds.

Head of the Department

Head of the Department
Dept.of CSE
Aditya College of Engineering
& Technology
SURAMPALEM-533437

Department of Computer Science and Engineering

Program Educational Objectives

PEO 1	Capability to design and develop new software products as per requirements of the various domains and eligible to take the roles in various government, research organizations and industry
PEO 2	More enthusiastic to adopt new technologies and to improve existing solutions by reducing complexity which serves society requirements as per timeline changes
PEO 3	With good hands-on basic knowledge and ready improve academic qualifications in India or abroad.
PEO 4	Ability to build and lead the team to achieve organizational goals

Head of the Department

Head of the Department
Dept.of CSE
Aditya College of Engineering
& Technology
SURAMPALEM-533437

Department of Computer Science and Engineering

PROGRAM SPECIFIC OUTCOMES

PSO 1: The ability to design and develop computer programs for analyzing the data.

PSO 2: The ability to analyze data & develop Innovative ideas and provide solution by adopting emerging technologies for real time problems of software industry.

PSO 3: To encourage the research in software field that contribute to enhance the standards of human life style and maintain ethical values.

Head of the Department

Head of the Department
Dept.of CSE
Aditya College of Engineering
& Technology
SURAMPALEM-533437

ADITYA COLLEGE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to JNTUK, Kakinada)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project work entitled, **"GASTROINTESTINAL CARCINOMA CLASSIFICATION USING CNN",** is a bonafide work carried out by **CH SAHITHI (18P31A0516), W VENU GOPAAL (18P31A0559), V KRISHNA PRAKASH (18P31A0555), J N S R S SRINIVAS (18P31A0527)**, in partial fulfillment of the requirements for the award **of the degree of Bachelor of Technology** in **COMPUTER SCIENCE AND ENGINEERING** from Aditya College of Engineering and Technology, Surampalem, during the academic year 2021-2022.

This project work has not been submitted in full or part to any other University or educational institute for the award of any degree or diploma.

PROJECT SUPERVISOR
Mrs. J L Sarwani Theeparthi,
M.Tech., [Ph.D.,]
Assistant Professor

HEAD OF THE DEPARTMENT
Dr.M.Anil Kumar, M.Tech., Ph.D.,
Professor

EXTERNAL EXAMINER

DECLARATION

We hereby declare that this project entitled "**GASTROINTESTINAL CARCINOMA CLASSIFICATION USING CNN** " has been undertaken by us and this work has been submitted to Department of Computer Science & Engineering, **ADITYA COLLEGE OF ENGINEERING AND TECHONOLOGY,** Surampalem affiliated to JNTUK, Kakinada, in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **COMPUTER SCIENCE AND ENGINEERING.**

We further declare that this project work has not been submitted in full or part to any other University or educational institute for the award of any degree or diploma.

PROJECT ASSOCIATES

CH SAHITHI	**(18P31A0516)**
W VENU GOPAAL	**(18P31A0559)**
V KRISHNA PRAKASH	**(18P31A0555)**
J N S R S SRINIVAS	**(18P31A0527)**

ACKNOWLEDGEMENT

It is with immense pleasure that we would like to express our indebted gratitude to my **project supervisor**, **Mrs. J L SARWANI THEEPARTHI, M.Tech.,[Ph.D]** who has guided us a lot and encouraged us in every step of project work, his valuable moral support and guidance has been helpful in successful completion of this Project.

We wish to express our sincere thanks to **Dr. M. ANIL KUMAR M.Tech.,Ph.D., Head of the Department of CSE**, for his valuable guidance given to us throughout the period of the project work.

We feel elated to thank **Principal**, **Dr. T. K. RAMA KRISHNA RAO M.Tech.,Ph.D.**, of Aditya College of Engineering and Technology for his cooperation in completion of our project and throughout our course.

We feel elated to thank **Dr. A. RAMA KRISHNA M.Tech.,Ph.D.**, **Dean (Academics & Administration)** of Aditya College of Engineering and Technology for his cooperation in completion of our project work.

We wish to express our sincere thanks to all faculty members, and lab programmers for their valuable guidance given to us throughout the period of the project.

We avail this opportunity to express our deep sense and heart full thanks to the **Management** of **Aditya College of Engineering & Technology** for providing a great support for us by arranging the trainees, and facilities needed to complete our project and for giving us the opportunity for doing this work.

PROJECT ASSOCIATES

CH SAHITHI	**(18P31A0516)**
W VENU GOPAAL	**(18P31A0559)**
V KRISHNA PRAKASH	**(18P31A0555)**
J N S R S SRINIVAS	**(18P31A0527)**

ABSTRACT

Gastrointestinal cancers account for approximately 20% of all cancer diagnoses and are responsible for 22.5% of cancer deaths worldwide. Artificial intelligence based diagnostic support systems, in particular convolutional neural network (CNN) based image analysis tools, have shown great potential in medical computer vision. In this research, we are proposing the more accurate and fast recognition of Gastrointestinal cancers based on computer vision and Deep learning technology. We extract the feature from the histological images for MSI vs MSS classification in gastrointestinal cancer. Then we train the model on the train set and test the model on the test set we have prepared. And we expecting the good accuracy than the existing models out there.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER – 1

INTRODUCTION

1.1 Introduction

Gastrointestinal cancers comprise esophageal, gastric, colon and rectal tumors. As reported by the WHO, approximately 3.5 million new gastrointestinal cancer cases worldwide have been recorded in 2018. Whereas the incidence of esophageal cancer is comparatively low, gastric cancer (GC) is the fifth most frequent type of cancer and the third leading cause of cancer death. Colorectal cancer (CRC) is the third most common cancer worldwide after lung and breast cancer but the second leading cause of cancer death. Although various predictive and prognostic biomarkers exist, high mortality rates for gastrointestinal cancer patients show that there is still potential to improve diagnostics to pave the way for more personalized therapy strategies leading to a better prognosis and/or fewer side effects.

Gastric cancer is the fourth most commonly diagnosed cancer and the second most common cause of cancer-related death worldwide. Although the incidence of gastric cancer has gradually decreased over the last half century, cancer at proximal stomach is on the rise. Today, gastric cancer is still the seventh most common cause of cancer-related death in the United States and the prognosis of advanced gastric cancer remains poor. Gastric carcinogenesis is a multistep and multifactorial process. While the intestinal type of gastric cancer is often related to environmental factors such as Helicobacter pylori infection, diet, and life style, the diffuse type is more often associated with genetic abnormalities. Recent advances in molecular medicine have not only shed light on the carcinogenesis of gastric cancer, but also offered novel approaches regarding prevention, diagnosis and therapeutic intervention.

During the “Genesis of Cancer” the word “Cancer” was rarely heard and we never thought that we would be hearing it so often. As per IARC (International Agency for Research on Cancer) 1 in 5 people develop cancer. Among all cancer related deaths Gastrointestinal Cancer constitutes to 35% of global cancer related deaths. Computer Vision was used to detect cancer tumors through histological

images which drastically cut down both the time and money to carry out conventional testing methods. Microsatellite is defined as the rudimentary repetitive sequence of the Deoxyribonucleic acid (DNA). DNA comprises of many microsatellites.

DNA Mismatch Repair (MMR) is a system which monitors the replication process of microsatellites and DNA, if it finds any error in the DNA recombination and replication it performs repair with the help of MMR proteins. Failure of MMR leads to unstable microsatellites/DNA which is the genesis of cancer. Based on global genomic status cancer tumor is classified into 'Micro satellite instable'(MSI) and 'Microsatellite Stable' (MSS) tumor.

High amount of instability in tumor classifies it as MSI-H and it can be inherited, in which the immune cells are shut off from fully doing their job. By using 'Immunotherapy' MSI-H can be cured. In MSS the DNA in tumor cell has the same number of microsatellites that of a healthy cell, this can be cured by 'radiation' and 'chemotherapy'-treatments which are opposite to immunotherapy. 26.4% of gastrointestinal cancer patients are classified as MSI-H and the rest i.e., 73.6% as MSS. Therefore, detection of MSI or MSS of cancer has the same significance as detection of cancer to give appropriate treatment.

Gastric carcinoma is clinically classified as early or advanced stage to help determine appropriate intervention, and histologically into subtypes based on major morphologic component. For the classification based on anatomic location, difficulty often arises when the tumor is located at proximal stomach or cardia, especially when the tumor also involves gastroesophageal junction (GEJ). It is not only because there are shared histologic features and immunophenotypes between the inflamed gastric cardiac mucosa due to Helicobacter infection and the metaplastic columnar epithelium-lined distal esophageal mucosa secondary to reflux disease, but also because there is no universal consensus regarding the anatomic definition of gastric cardiac. Several classifications were proposed in order to address this issue. The scheme endorsed by the International Gastric Cancer Association separates gastric cancers into type I, type II and type III, to represent the tumors at distal esophagus, at cardia and at the stomach distal to cardia, respectively. This classification, however,

has not clearly defined the criteria for each of these anatomic locations. Most recently, the 7th Edition of the TNM classification by American Joint Committee on Cancer (AJCC) has simplified the classification of the carcinoma at proximal stomach based on the location of tumor epicenter and the presence or absence of GEJ involvement.

The tumor is to be stage grouped as esophageal carcinoma if its epicenter is in the lower thoracic esophagus or GEJ, or within the proximal 5 cm of stomach (i.e., cardia) with the tumor mass extending into GEJ or distal esophagus. If the epicenter is >5 cm distal to the GEJ, or within 5 cm of GEJ but does not extend into GEJ or esophagus, it is stage grouped as gastric carcinoma. This classification, although easy for pathologists to follow, could still face some challenges. For example, a bulky gastric cardiac cancer with its epicenter 4 cm below GEJ will still be diagnosed and classified as an esophageal tumor if the proximal end of tumor extends into GEJ by only 0.5 cm (even if the distal end of tumor is 4 cm from the epicenter extending into the stomach). For the operating surgeon who sees the tumor in situ, it may be difficult for him or her to accept this tumor as an esophageal cancer. In addition, a recent retrospective study by Huang et al. shows that cardiac carcinoma involving GEJ or distal esophagus is more appropriately classified and staged as gastric rather than esophageal cancers, at least in the Chinese population. In that study, cardiac carcinomas were staged according to the depth of invasion, status of positive lymph nodes and distant metastasis, as both gastric and esophageal tumors.

1.1.1 The gastrointestinal cancer mainly classified into two types, MSI and MSS

1.1.1.1 Microsatellite Instability (MSI)

MSI means instability in cancer cells which are the repeated sequences of DNA. MSI is a condition of impaired DNA Mismatch Repair (MMR). MMR consists of a family of proteins that detect DNA replication error. The important genes which are responsible for the MMR factors are MLH1, MSH2, MLH3, MSH6 and PMS2. Any mutation that occurs to these genes will lead to non-functional MMR which leads to increase or decrease of microsatellite which is the basis for MSI. It is associated with colon cancer, gastric cancer, ovarian cancer and endometrium cancer. But it is most predominant in affiliation with colon cancer. But it is most predominant in affiliation

with colon cancer. In MSI, MMR is deficient which increases the mutation rate and is an alteration in the DNA sequence that makes up a gene. Due to deficient MMR, DNA replication go unrepaired leading to high mutation tumor. MSI can be detected with the help of automatic techniques like machine learning and deep learning very easily. By using histological images of tissue slide we can detect MSI status easily

1.1.1.2 Microsatellite Stability (MSS)

Microsatellite Stability means, there is no instability in tumor. It is just the opposite of MSI and tissues are found same as normal tissue which does not confirm any instability in biomarkers. In MSS, MMR is proficient which leads to low mutation rate. In the presence of functional MMR system, the replication error occurs at a very low mutation rate which slows down the process of growth of cancer cell replication. The cancer detection measures are required to differentiate between MSI and MSS gastrointestinal cancer. The procedure used for cancer detection is elaborated in the upcoming section.

1.1.2 Gastrointestinal Cancer Detection Process by Pathologist

The process used for detection of cancer is represented by Figure. The slide scanner as shown in Figure is used to scan the tissue slides images like MSI and MSS, then after scanning, images are used for tissue mapping where different magnification tissues are converted into same size and are mapped with each other as represented by Figure. After mapping tissues are analyzed according to their structure as shown in Figure. Finally, the tissue area is divided into different rectangles according to region of interest (ROI) where structural and nuclear features are analyzed as shown in Figure and according to the given slides results are predicted as shown in Figure.

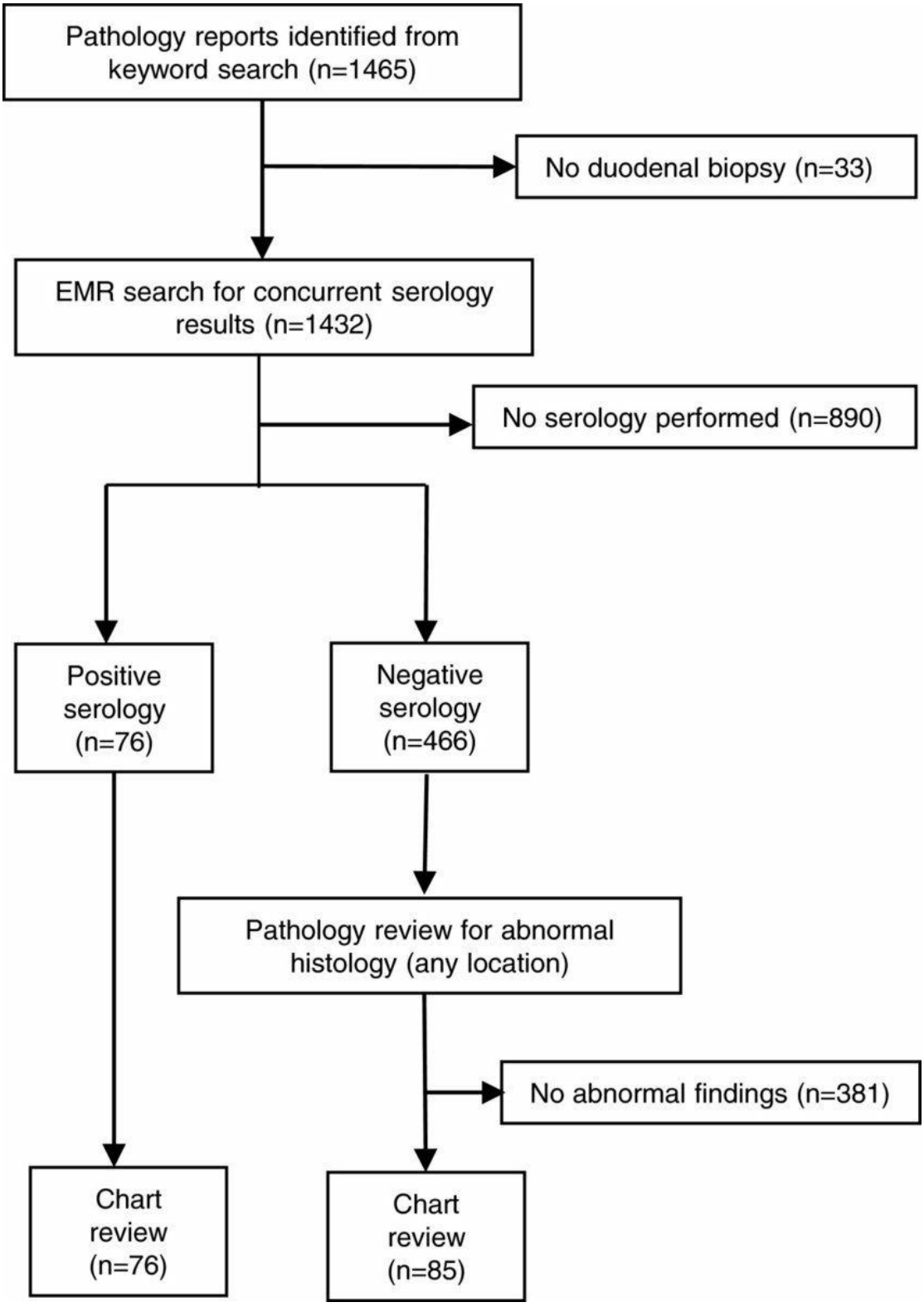


Fig. 1.1. Analysis flow of Pathologist

1.2 Literature Survey

In 2020 Lizuka et al. [1] Proposed an AI based computational technique for classification of gastric and colon cancer. CNN is used for feature extraction and RNN model is used to classify data under time constraints into Adenoma, Nonneoplastic and Adenocarcinoma.

In 2019 Fu et al. [2] Proposed a prediction model for MSI status of right sided Colon Cancer (RCC) based on the qualitative transcriptional signature. RCC samples is used for the relative expression orderings of gene pairs and based on the feature selection with RCCs authors achieved F-scores is near to 1, 0.9630, 0.9412 and 0.8798, respectively.

In 2019 Wan et al. [3] Proposed an expert system is developed to detect cancer at an early stage. Various machine learning algorithm like SVM, KNN, EB, RF are compared with deep learning algorithm like CNN, RNN1 and RNN2. It is found that machine learning algorithm worked well because the dataset was very small in size.

In 2019 Nakahira et al. [4] Proposed a deep neural network for gastric cancer detection. Computerized system is developed to analyse the images of cancer prediction. In this the system detected gastric cancer risk in three different groups low, moderate and high. CNN algorithm is used for detection purpose with gradient booster.

In 2019 Siegel et al. [5] Presented an analysis of gastric cancer according to the American Society of Cancer, the number of new cases of cancer instances and the deaths rates resulting from cancer and the latest cancer information collected.

In 2019 Muhammad et al. [6] Proposed Artificial Neural Network (ANN) has been implemented to tackle early detection of pancreatic cancer. Two data sources are used for this purpose. The results can be further improved by using another technique.

In 2018 Yoshida et al. [7] Proposed an automated image analysis in the field of surgical pathology to achieve the desired results. In this research 3062 gastric

specimen were used. The main focus of the research to perform classification of pathologist software of images. Histopathological study of images can be done further.

In 2018 Kim et al. [8] Collected cases with slide level and region-level labels and trained deep neural network for tissue classification. To improve the tissue classification performance, they exploited slide-level weak label for training the model with patches without region-level label. For whole slide classification, they extracted features representative for whole slide characteristics.

In 2017 Sharma et al. Proposed an automatic Classification of gastric carcinoma using whole slide images in digital histopathology. Deep learning is used to detect gastric cancer with the help of CNN Architecture. The classification results are compared with the traditional analysis methods used for histopathological images.

In 2017 Liu et al. Proposed the various parameters of histopathological images of gastric cancer with the help of CT texture analysis. The main focus is to analyse the correlation between these two by using t-sample test.

In 2015 Goto et al. Proposed a new technology names as hyperspectral imaging. The study is done to differentiate gastric tumour and normal mucosa with the help of hyperspectral camera.

In 2014 Tao et al. Proposed different methods like magnifying endoscopy and chromo endoscopy are used for enhancement of gastric cancer.643 specimens are used as sample for analysis. Data is collected from Peking Union Medical College Hospital of two years and then study is performed.

In 2014 Chen et al. Presented an innovative genetic selection method using swarm optimization in conjunction with a classifier known as decision tree. Mathematical analysis demonstrates that the proposed method performs better than other common optimization algorithms by conducting research on 11 datasets of cancer expression of genes.

1.3 Problem Statement

Gastrointestinal cancer is one of the most common malignant tumors in the world and the leading cause of death. Early detection and diagnosis of Gastrointestinal cancer is of critical importance. At present, biopsy serves as the gold standard for diagnosing gastrointestinal tumors. As an AI algorithm that automatically learns features from the data, CNN has been utilized mainly for image recognition. With the development of new technologies such as magnifying endoscopy with narrow band imaging, endoscopists achieved better accuracy for diagnosis of gastrointestinal cancer using different algorithms. Therefore, this study explored the research status and development trends of deep learning on Gastrointestinal cancer image classification using different CNN networks.

1.4 Objectives of the Research

Our main objective of this research is classifying the Gastrointestinal cancer images into two different classes MSIMUT, MSS. We have collected the large dataset from the open-source medical repository and we want the classification was more-accurate and fast when compared to existing methods and development of the application was also should be economically feasible.

1.5 Databases Description

We have collected the dataset from the Kaggle resource: https://www.kaggle.com/linjustin/train-val-test-tcga-coad-msi-mss. The total dataset consists of 5.84 GB of the data. And we sampled the data set and trained CNN models on 2 classes of the images. Our sampled dataset consists of total 1,92,312 histological images.

Sample Images from Dataset

MSIMUT

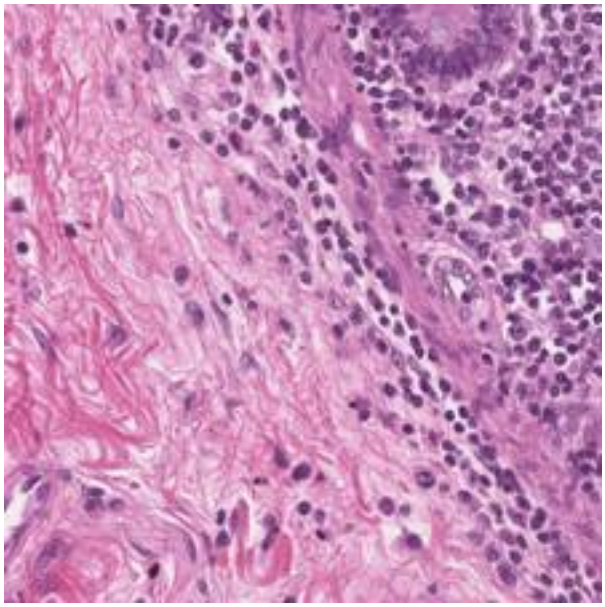


Fig. 1.2. MSIMUT Histological Image - 1

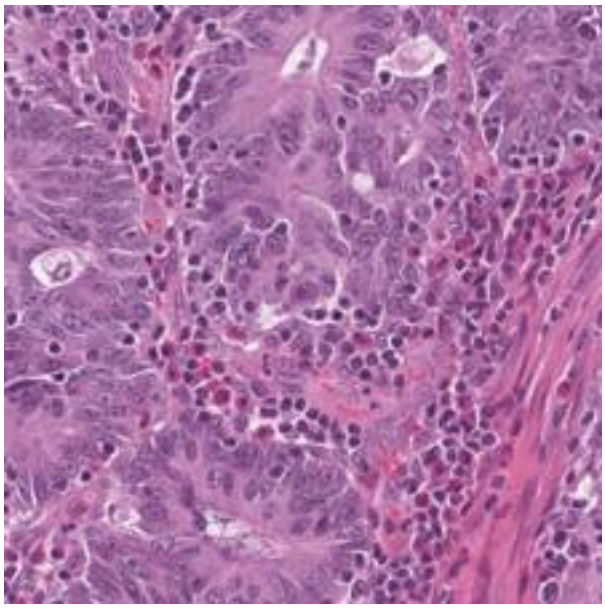


Fig. 1.3. MSIMUT Histological Image - 2

MSS

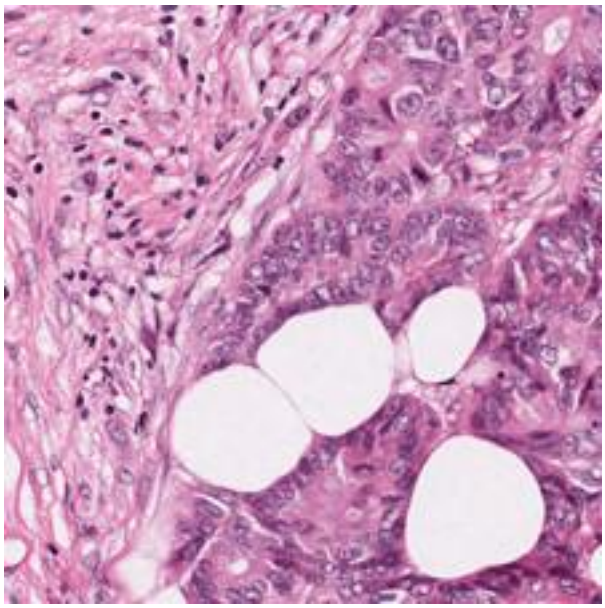


Fig. 1.4. MSS Histological Image – 1

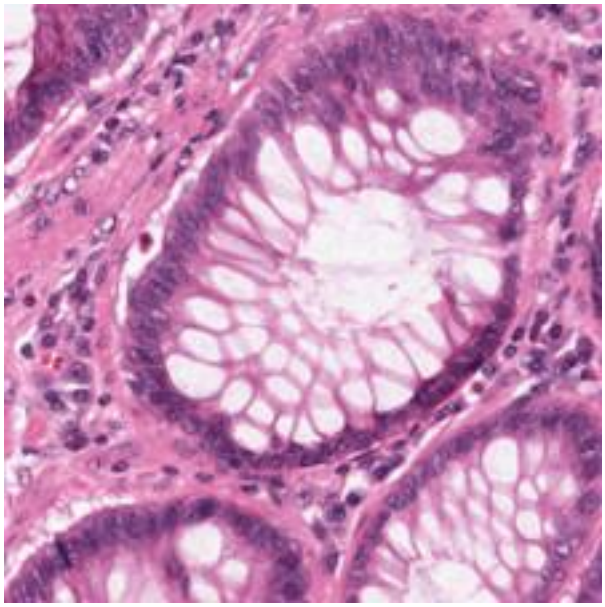


Fig. 1.5. MSS Histological Image -2

1.6 Performance Evaluation Measures

Micro, Macro and weighted precision, recall and F1-Scores are considered as the performance metrics for the gastrointestinal cancer classification models. A macro-average will compute the metric independently for each class and then take the average, whereas a micro-average will aggregate the contributions of all classes to compute the average metric and the weights are the number of instances in each class. The Precision is taken for fraction of true positive among all the positive's recalled. The Recall score taken for fraction of true positives among all the correct events. F1-Score is for to calculate harmonic mean of the precision and recall.

Micro-Precision:

$$= \sum_{i=1}^{n} \frac{True\ Positives(i)}{True\ Positives(i) + False\ Positives(i)}$$

Micro-Recall:

$$= \sum_{i=1}^{n} \frac{True\ Positives(i)}{True\ Positives(i) + False\ Positives(i) + True\ Negative(i) + False\ Negative(i)}$$

Micro F1-Score:

$$= 2.\frac{Micro - Precison * Micro - Recall}{Micro - Precison + Micro - Recall}$$

Micro-Precision:

$$= \frac{Micro\ Precison}{2}$$

Micro-Recall:

$$= \frac{Micro\ Recall}{2}$$

Micro F1-Score:$= 2.\frac{Macro-Precison*Macro-Recall}{Macro-Precison+Macro-Recall}$

Accuracy:

Accuracy in classification problems is the number of correct predictions made by the model over all kinds predictions made. In the Numerator, are our correct predictions (True positives and True Negative) and in the denominator, are the kind of all predictions made by the algorithm (Right as well as wrong ones).

CHAPTER – 2

GASTROINTESTINAL CARCINOMA CLASSIFICATION USING CNN

2.1 BREIF OUTLINE OF THE CHAPTER

2.1.1 Convolutional Neural Networks

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. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

CNN is a deep learning algorithm which takes an input image, assign weights to the object and classify them according to features. It is also known as ConvNet, and is type of artificial neural network. CNN can be used in image processing, Natural Language Processing etc. It is basically four layered concept which consists of Convolutional layer, pooling layer, Flattening layer and Fully Connected layer. The description of each layer is given below.

The architecture of a ConvNet is analogous to that of the connectivity pattern of ortex. Individual neurons respond to stimuli only in a restricted region of the vNeurons in the Human Brain and was inspired by the organization of the Visual Cisual field known as the Receptive Field. A collection of such fields overlaps to cover the entire visual area.

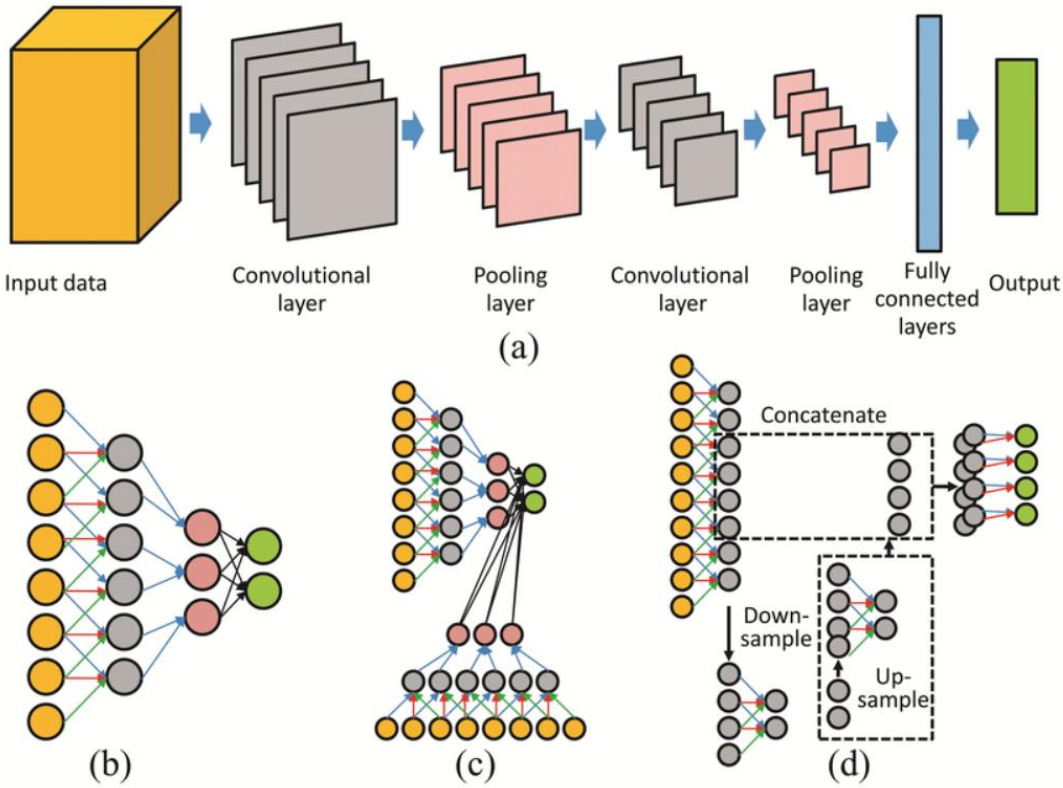


Fig. 2.1. Convolution Neural Network Architecture

2.1.2 Convolutional Neural Networks Architecture

2.1.2.1 Convolution Layer

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

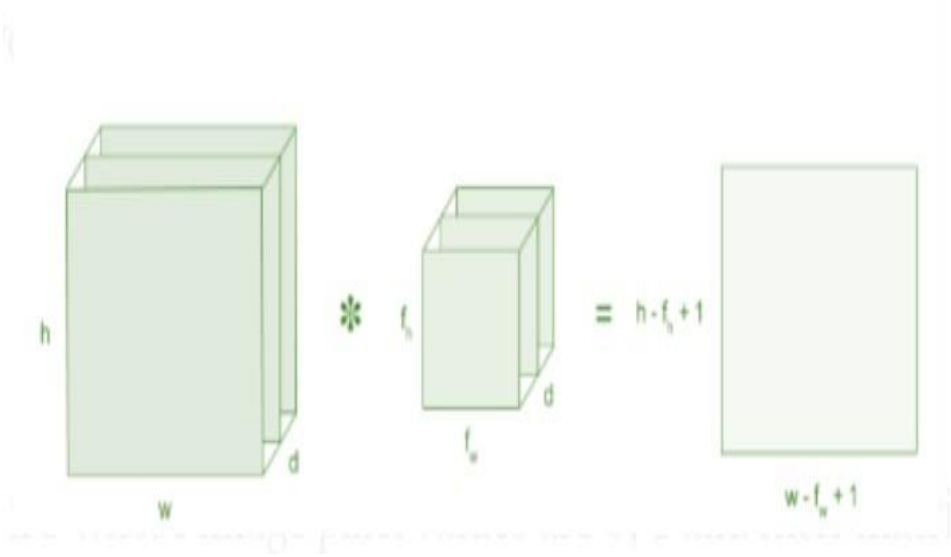


Fig. 2.2. Image matrix multiplies kernel or filter matrix

- An image matrix (volume) of dimension (h x w x d)
- A filter (f_h x f_w x d)
- Outputs a volume dimension (h - f_h + 1) x (w – f_w + 1) x 1

Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below

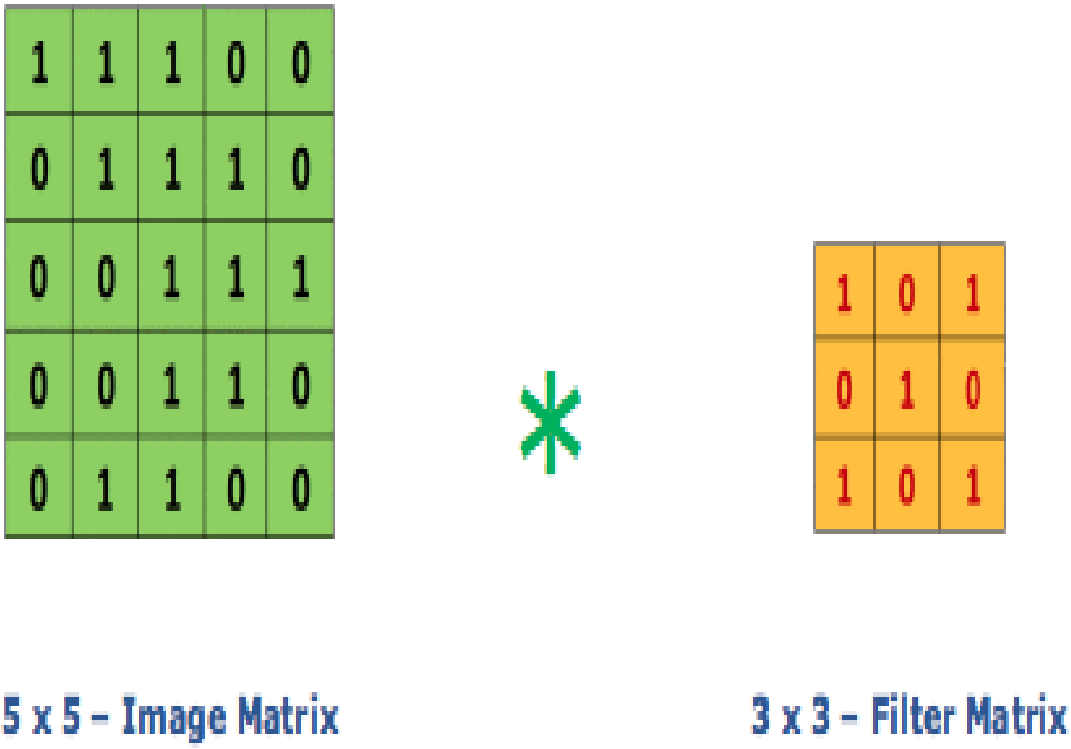


Fig. 2.3 Image matrix multiplies kernel

Then the convolution of 5 x 5 image matrix multiplies with 3 x 3 filter matrix which is called “Feature Map” as output shown in below. Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters.

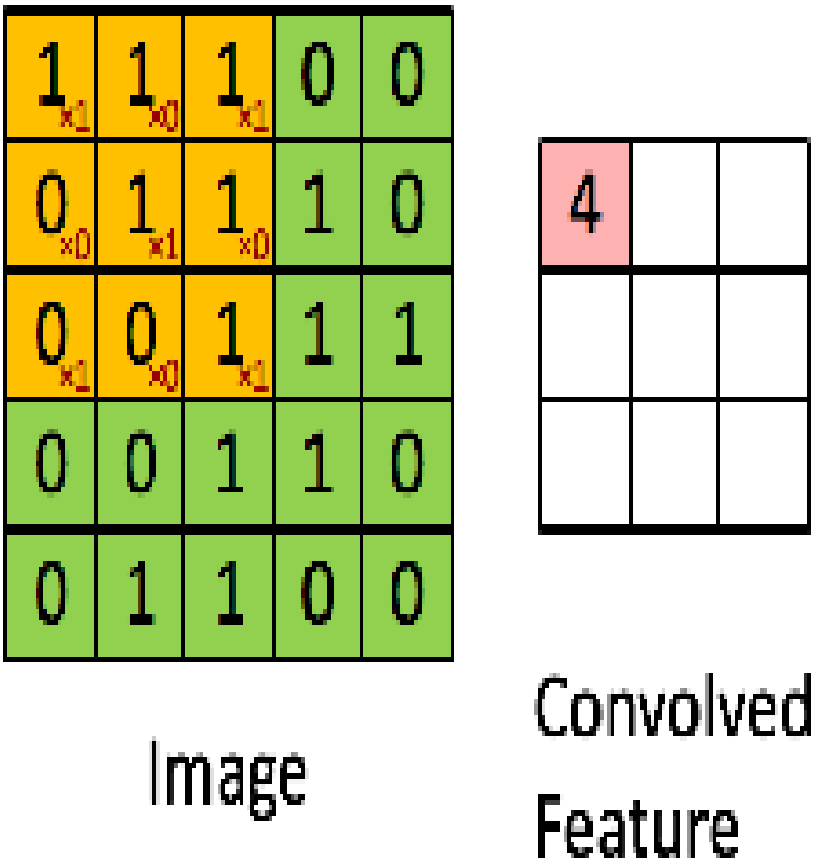


Fig. 2.4. 3 x 3 Output Matrix

2.1.2.2 Strides

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

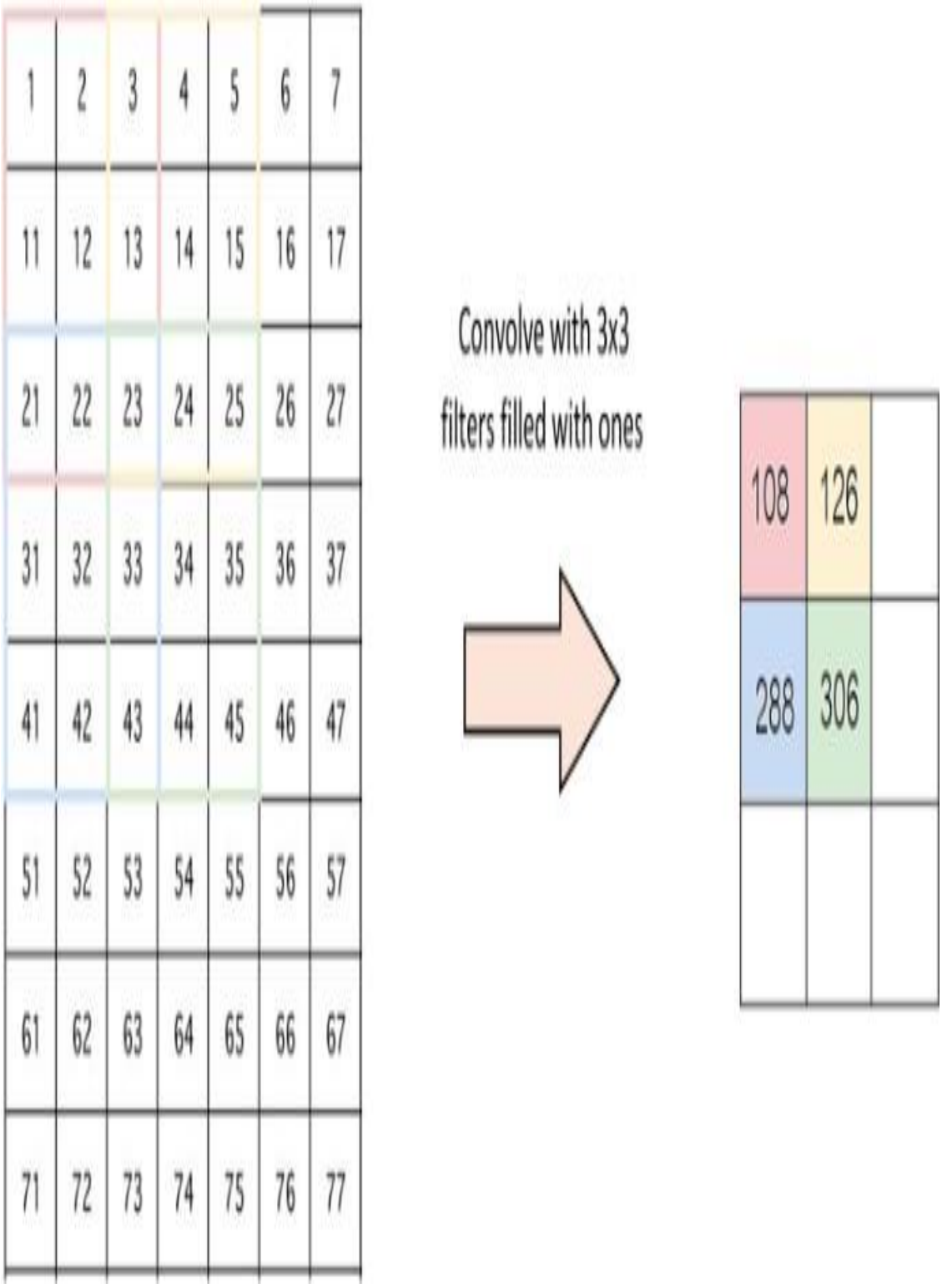


Fig. 2.5. Stride of 2 pixels

2.1.2.3 Padding

Sometimes filter does not fit perfectly fit the input image. We have two options:

- Pad the picture with zeros (zero-padding) so that it fits.
- Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

2.1.2.4 Non-Linearity (ReLU)

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is $f(x) = \max(0, x)$. ReLU's purpose is to introduce non-linearity in our ConvNet. Since, the real-world data would want our ConvNet to learn would be non-negative linear values.

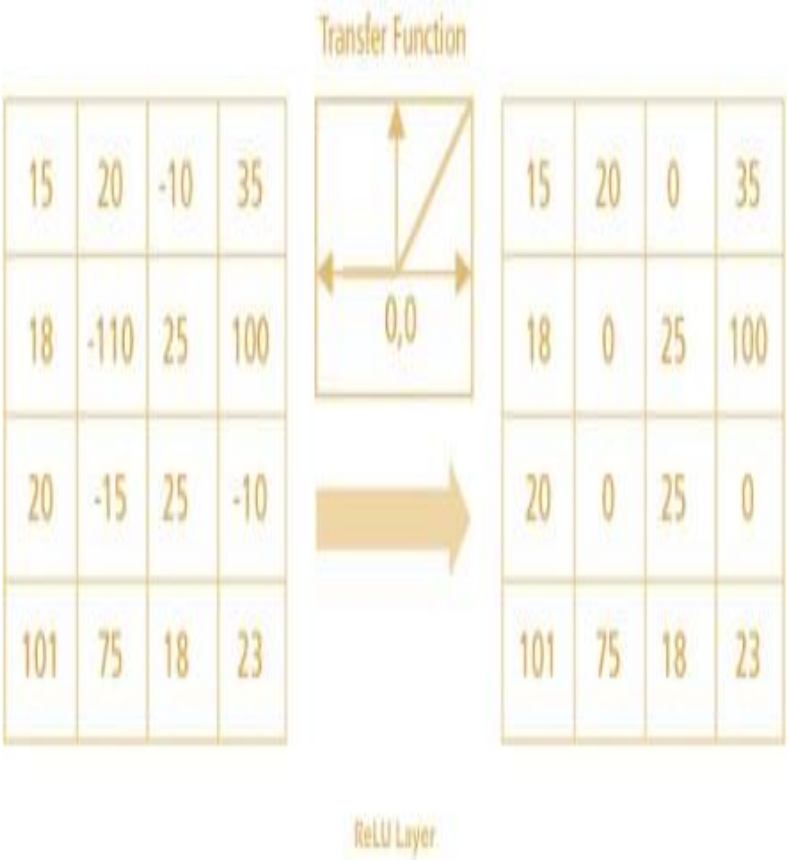


Fig. 2.6. ReLU Operation

There are other non-linear functions such as tanh or sigmoid can also be used instead of ReLU. Most of the data scientists uses ReLU since performance wise ReLU is better than other two.

2.1.2.5 Pooling Layer

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or down sampling which reduces the dimensionality of each map but retains the important information. Spatial pooling can be of different types:

- Max Pooling
- Average Pooling
- Sum Pooling

Max pooling takes the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

The main function of this layer is to reduce the spatial size of the matrix. In this, filter is passed over the results of convolutional layer. There is max, min and average pooling approach which can be applied to the matrix. The most common approach used is max pooling because this allows network to train faster as shown by Figure 5. Stride of 2*2 is used on the image to access the maximum features from image and formed a new matrix known as pooled feature map which is then passed to Flattening layer.

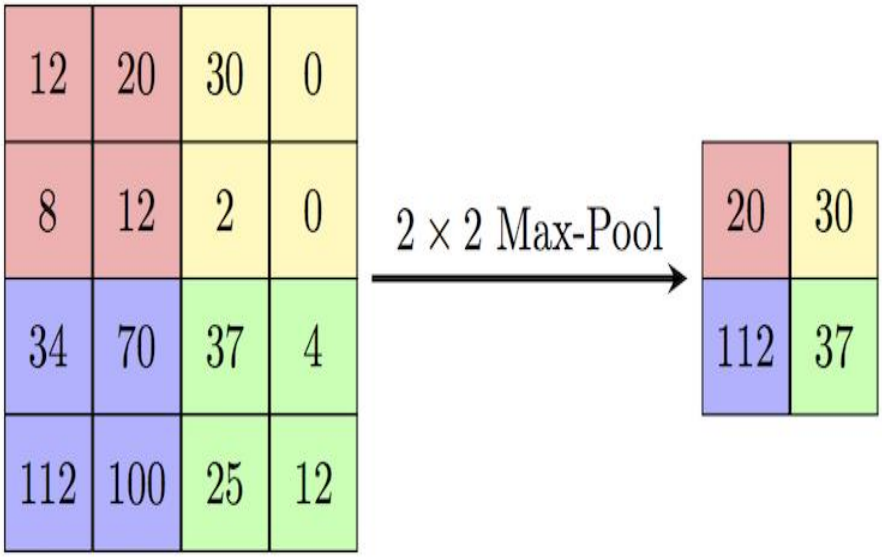


Fig. 2.7. Max Pooling

2.1.2.6 Flattening Layer

It is used to convert the pooled feature map into a single column which is used as an input for the next layer as shown by Figure. In this, output is flattened into a single long feature vector which is connected to Fully Connected layer.

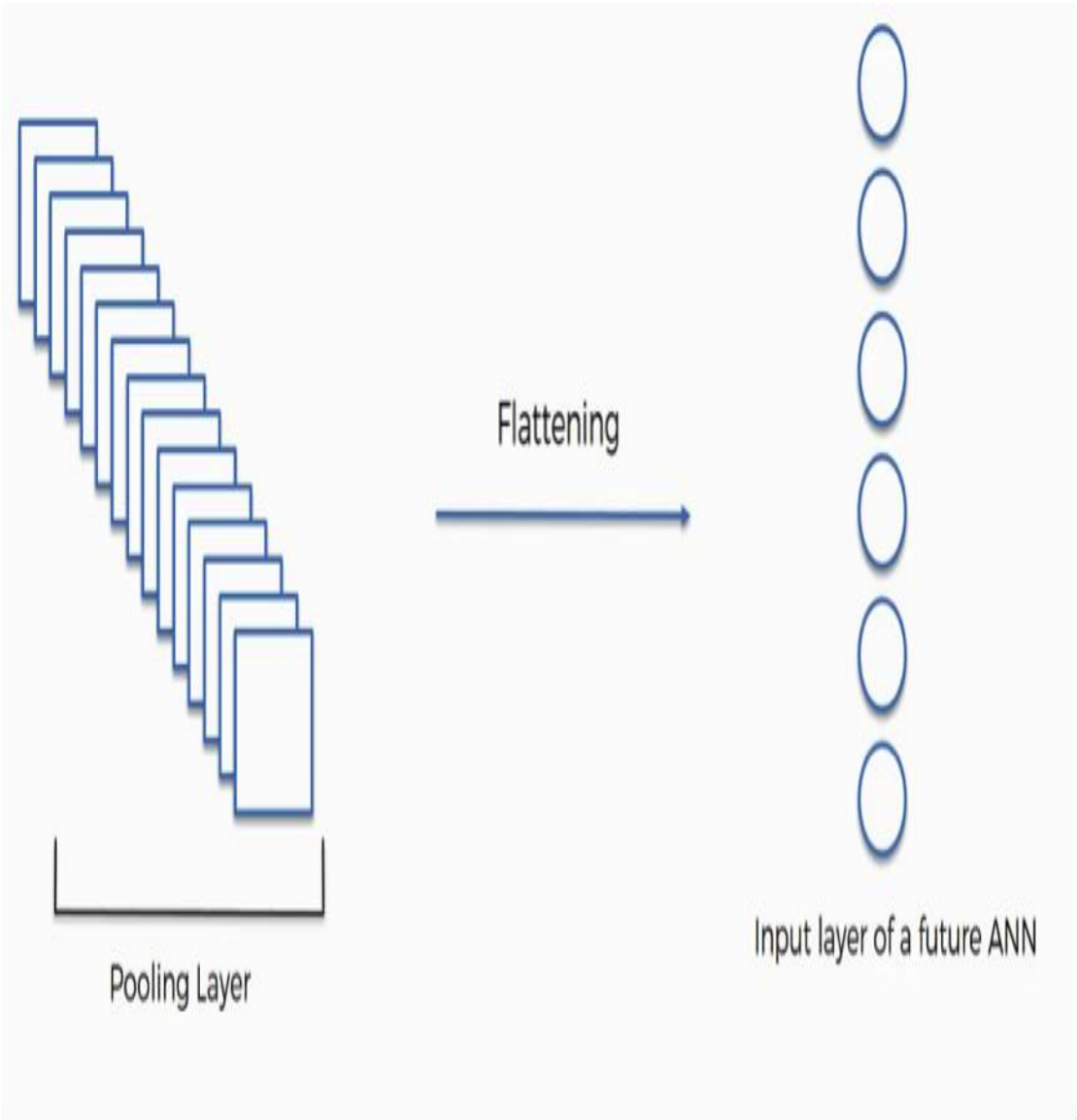


Fig. 2.8. Flattening layer

2.1.2.7 Fully Connected Layer

The flattened feature map is passed to neural network for processing. This layer consists of input layer, fully connected layer and output layer. In this, every node in the first layer is connected to second layer of every node and gives the final probability to classify the images.

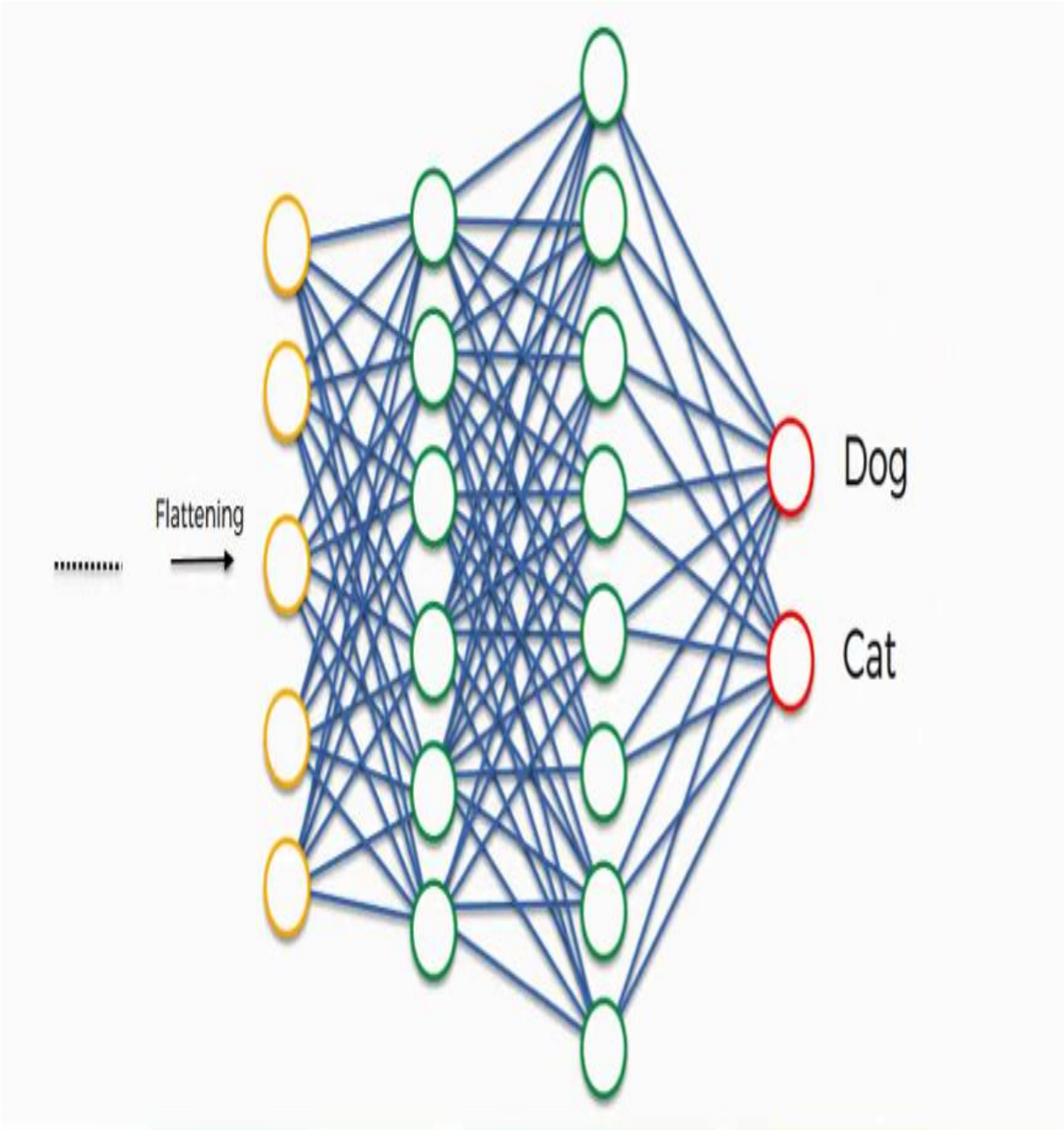


Fig. 2.9 After Pooling Layer, Flattened as FC Layer

In the above diagram, feature map matrix will be converted as vector (x1, x2, x3, …). With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as soft-max or sigmoid to classify the outputs as cat, dog, car, truck etc.

Structure to evaluate clinical applicability of a classifier.

Level	Characteristics of CNN-based classifiers in cancer diagnosis
1	Training and testing with only one usually comparatively small dataset
2	Testing with an external dataset to demonstrate its generalization independent from varying sample conditions
3	Comparison with the performance of pathologists to reveal its additional value
4	Testing in a clinical setting as a supportive system in combination with diagnosis of pathologists (At this point, the classifier is confronted with acute patient data for the first time and the classification result has influence on the patient's therapy. This application represents a change from a retrospective to a prospective analysis.)
5	Implementation in clinical routine to actively support cancer diagnosis

Fig. 2.10. Characteristics of CNN in Cancer Diagnosis

2.2 Related Work

In 2020 Lizuka et al. [9] Proposed an AI based computational technique for classification of gastric and colon cancer. CNN is used for feature extraction and RNN model is used to classify data under time constraints into Adenoma, Nonneoplastic and Adenocarcinoma.

In 2019 Fu et al. [10] Proposed a prediction model for MSI status of right sided Colon Cancer (RCC) based on the qualitative transcriptional signature. RCC samples is used for the relative expression orderings of gene pairs and based on the feature selection with RCCs authors achieved F-scores is near to 1, 0.9630, 0.9412 and 0.8798, respectively.

In 2019 Wan et al. [13] Proposed an expert system is developed to detect cancer at an early stage. Various machine learning algorithm like SVM, KNN, EB, RF are compared with deep learning algorithm like CNN, RNN1 and RNN2. It is found that machine learning algorithm worked well because the dataset was very small in size.

Deep learning can predict microsatellite instability directly from histology in gastrointestinal cancer by Jakob Nikolas Kather 1,2,3,4,5*, Alexander T. Pearson4, Niels Halama 2,5,6, Dirk Jäger2,3,5, Jeremias Krause 1 , Sven H. Loosen1 , Alexander Marx7 , Peter Boor 8, Frank Tacke9 , Ulf Peter Neumann10, Heike I. Grabsch 11,12, Takaki Yoshikawa13,14, Hermann Brenner2,15,16, Jenny Chang-Claude17,18, Michael Hoffmeister15, Christian Trautwein1 and Tom Luedde 1 used resnet 18 architecture of cnn to detect one of the gastro intestinal cancer MSI is present or not.

Modified ResNet Model for MSI and MSS Classification of Gastrointestinal Cancer by CH Sai Venkatesh*, Caleb Meriga, M.G.V.L Geethika, T Lakshmi Gayatri, V.B.K.L Aruna used modified resnet architecture in cnn is proposed for the classification of Microsatellite instability (MSI) and Microsatellite stability (MSS) of gastrointestinal cancer.

2.3 Proposed Method

In this research, we classify gastrointestinal cancers MSI AND MSS, using inceptionV3 algorithmic architecture of Convolutional Neural Network (CNN).

2.3.1 Inception Model

The main idea of the Inception architecture is to consider how an optimal local sparse structure of a convolutional vision network can be approximated and covered by readily available dense components. Note that assuming translation invariance means that our network will be built from convolutional building blocks. All we need is to find the optimal local construction and to repeat it spatially.

Arora et al. suggests a layer-by layer construction where one should analyze the correlation statistics of the last layer and cluster them into groups of units with high correlation. These clusters form the units of the next layer and are connected to the units in the previous layer. We assume that each unit from an earlier layer corresponds to some region of the input image and these units are grouped into filter banks. In the lower layers (the ones close to the input) correlated units would concentrate in local regions. Thus, we would end up with a lot of clusters concentrated in a single region and they can be covered by a layer of 1×1 convolutions in the next layer. However, one can also expect that there will be a smaller number of more spatially spread-out clusters that can be covered by convolutions over larger patches, and there will be a decreasing number of patches over larger and larger regions.

In order to avoid patch-alignment issues, current incarnations of the Inception architecture are restricted to filter sizes 1×1, 3×3 and 5×5; this decision was based more on convenience rather than necessity. It also means that the suggested architecture is a combination of all those layers with their output filter banks concatenated into a single output vector forming the input of the next stage. Additionally, since pooling operations have been essential for the success of current convolutional networks, it suggests that adding an alternative parallel pooling path in each such stage should have additional beneficial effect, too.

As these “Inception modules” are stacked on top of each other, their output correlation statistics are bound to vary: as features of higher abstraction are captured by higher layers, their spatial concentration is expected to decrease. This suggests that the ratio of 3×3 and 5×5 convolutions should increase as we move to higher layers. One big problem with the above modules, at least in this naïve form, is that even a modest number of 5×5 convolutions can be prohibitively expensive on top of a convolutional layer with a large number of filters. This problem becomes even more pronounced once pooling units are added to the mix: the number of output filters equals to the number of filters in the previous stage. The merging of output of the pooling layer with outputs of the convolutional layers would lead to an inevitable increase in the number of outputs from stage to stage. While this architecture might

cover the optimal sparse structure, it would do it very inefficiently, leading to a computational blow up within a few stages. This leads to the second idea of the Inception architecture: judiciously reducing dimension wherever the computational requirements would increase too much otherwise. This is based on the success of embeddings: even low dimensional embeddings might contain a lot of information about a relatively large image patch. However, embeddings represent information in a dense, compressed form and compressed information is harder to process.

The representation should be kept sparse at most places and compress the signals only whenever they have to be aggregated en masse. That is, 1×1 convolutions are used to compute reductions before the expensive 3×3 and 5×5 convolutions. Besides being used as reductions, they also include the use of rectified linear activation making them dual-purpose. The final result is depicted in Figure 2(b). In general, an Inception network is a network consisting of modules of the above type stacked upon each other, with occasional max-pooling layers with stride 2 to halve the resolution of the grid. For technical reasons (memory efficiency during training), it seemed beneficial to start using Inception modules only at higher layers while keeping the lower layers in traditional convolutional fashion. This is not strictly necessary, simply reflecting some infrastructural inefficiencies in our current implementation. A useful aspect of this architecture is that it allows for increasing the number of units at each stage significantly without an uncontrolled blow-up in computational complexity at later stages. This is achieved by the ubiquitous use of dimensionality reduction prior to expensive convolutions with larger patch sizes.

Furthermore, the design follows the practical intuition that visual information should be processed at various scales and then aggregated so that the next stage can abstract features from the different scales simultaneously. The improved use of computational resources allows for increasing both the width of each stage as well as the number of stages without getting into computational difficulties. One can utilize the Inception architecture to create slightly inferior, but computationally cheaper versions of it. We have found that all the available knobs and levers allow for a controlled balancing of computational resources resulting in networks that are $3 - 10\times$

faster than similarly performing networks with non-Inception architecture, however this requires careful manual design at this point.

Building a powerful deep neural network is possible by increasing the number of layers in a network Two problems with the above approach are that increasing the number of layers of a neural network may lead to overfitting especially if you have limited labelled training data and there is an increase in the computational requirement.

Inception networks were created with the idea of increasing the capability of a deep neural network while efficiently using computational resources.

Inception networks are released in versions, each version having some improvement over the previous ones. Let’s start our discussion with Inception Version 1 aka Inception V1.

Transfer learning allows you to retrain the final layer of an existing model, resulting in a significant decrease in not only training time, but also the size of the dataset required. One of the most famous models that can be used for transfer learning is Inception V3. As mentioned above, this model was originally trained on over a million images from 1,000 classes on some very powerful machines. Being able to retrain the final layer means that you can maintain the knowledge that the model had learned during its original training and apply it to your smaller dataset, resulting in highly accurate classifications without the need for extensive training and computational power.

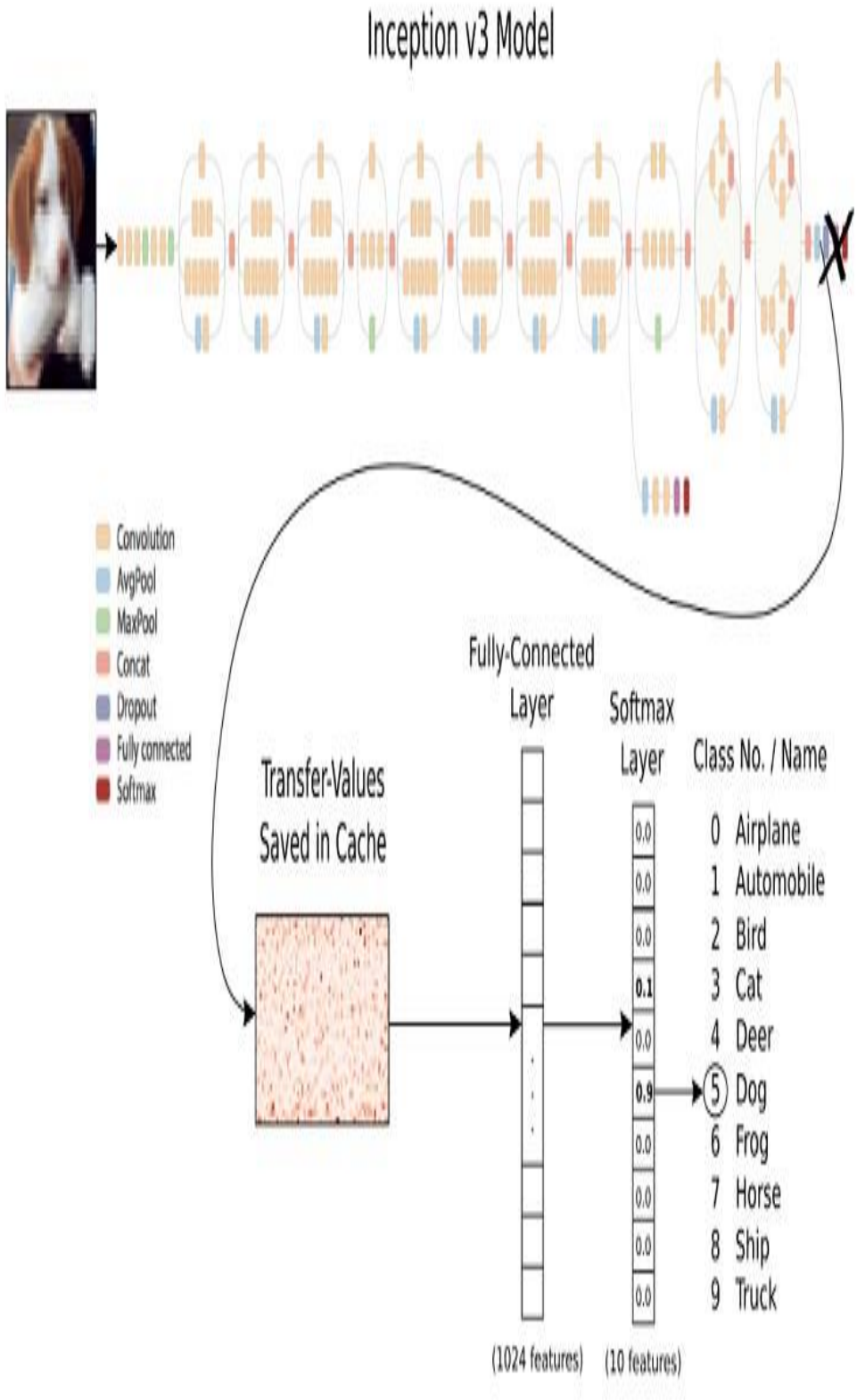


Fig. 2.11. Inception V3 Transfer Learning

2.3.2 The Architecture of Inception V1

Consider the below images of peacocks. The area of the image occupied by the peacock varies in both images, selecting the right kernel size thus becomes a difficult choice. A large kernel size is used to capture a global distribution of the image while a small kernel size is used to capture more local information.

Inception network architecture makes it possible to use filters of multiple sizes without increasing the depth of the network. The different filters are added parallelly instead of being fully connected one after the other.

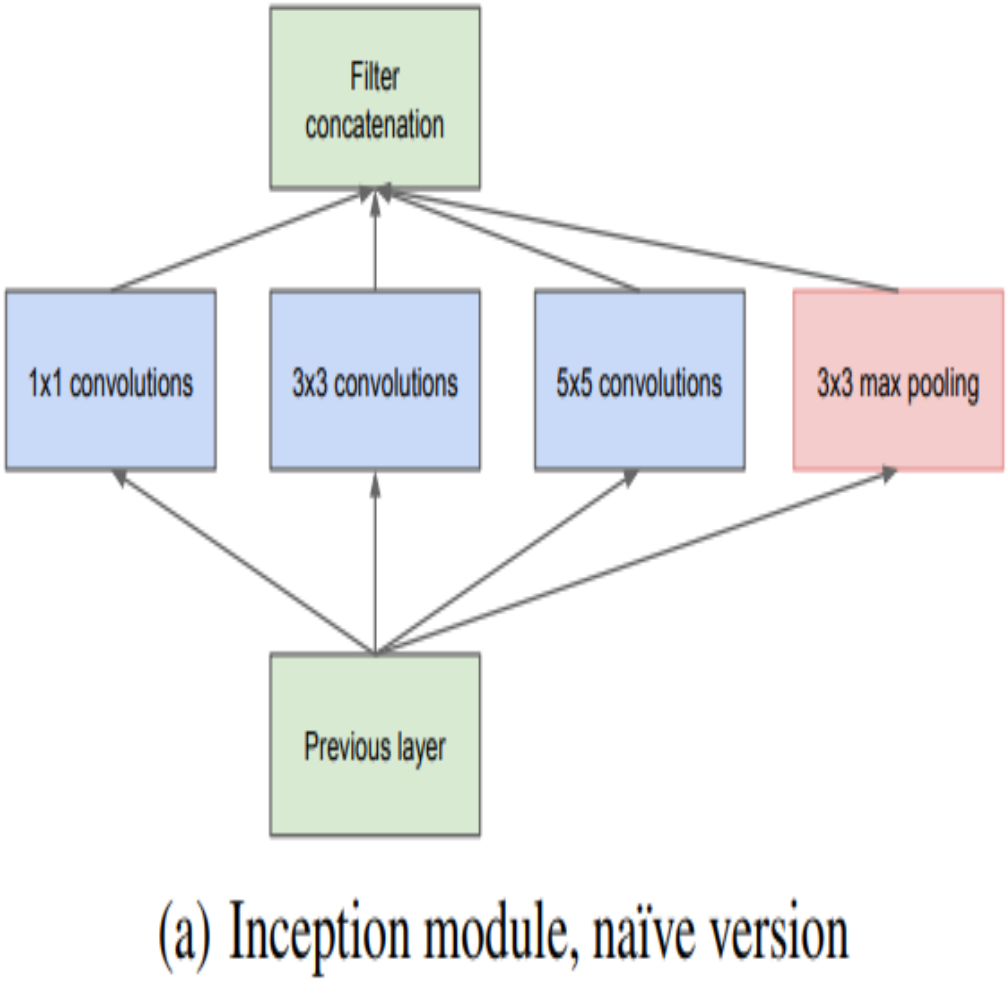
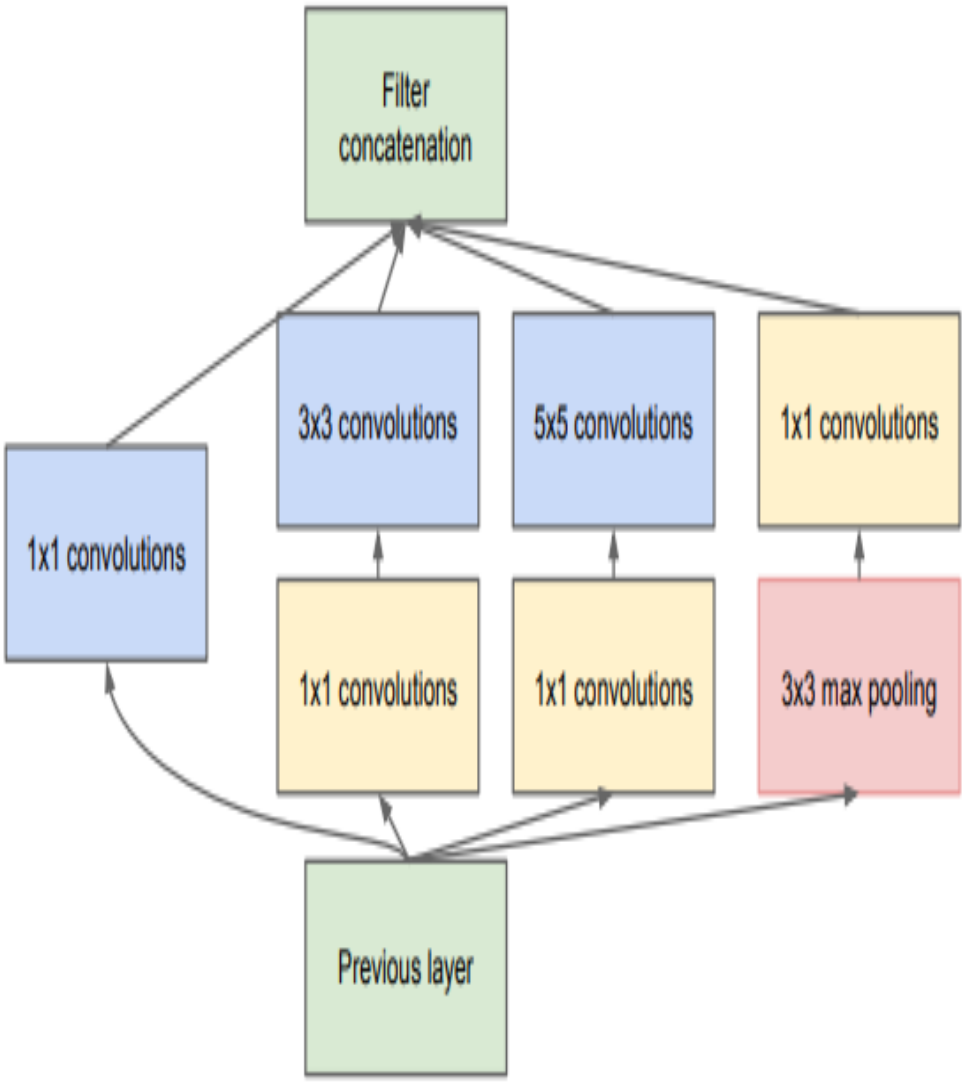


Fig. 2.12. Inception Module Naïve Version

This is known as the naive version of the inception model. The problem with this model was the huge number of parameters. To mitigate the same, they came up with the below architecture.

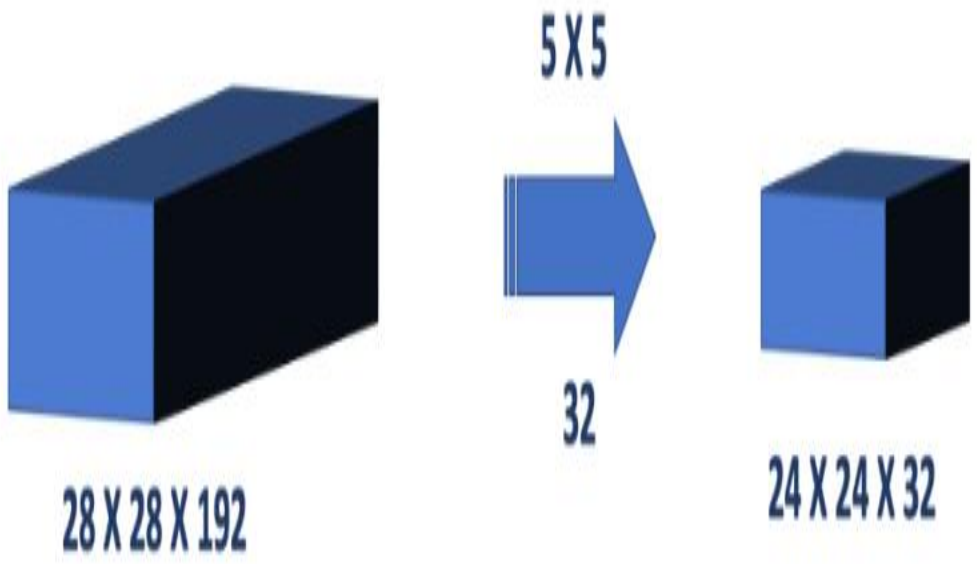


(b) Inception module with dimension reductions

Fig. 2.13. Inception Module with Dimension Reductions

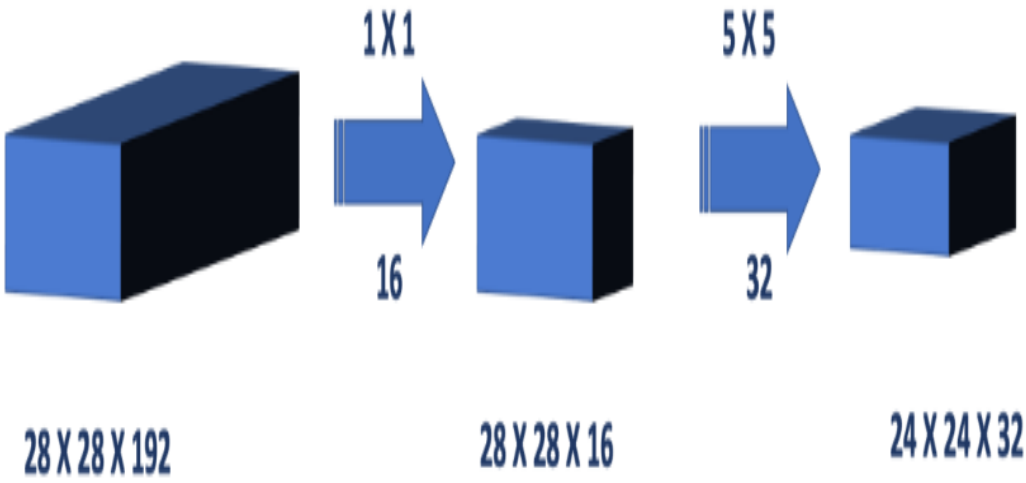
2.3.3 Architecture Dimensionality Reduction

Adding a 1X1 convolution before a 5X5 convolution would reduce the number of channels of the image when it is provided as an input to the 5X5 convolution, in turn reducing the number of parameters and the computational requirement.



Number of Operations: (28x28x32)x(5x5x192)=120.422 Million Ops

Fig. 2.14. Dimensionality Reduction 1



Number of Operations 1x1 Conv Step:(28x28x16)x(1x1x192)=2.4 Million Ops

Number of Operations 5x5 Conv Step:(28x28x32)x(5x5x16)=10 Million Ops

Total Number of Operations=12.4 Million Ops

Fig. 2.15. Dimensionality Reduction 2

2.3.4 Difference between Inception V3 and Inception V1

Inception V3 is an extension of the V1 module, it uses techniques like factorizing larger convolutions to smaller convolutions (say a 5X5 convolution is factorized into two 3X3 convolutions) and asymmetric factorizations (example: factorizing a 3X3 filter into a 1X3 and 3X1 filter).

These factorizations are done with the aim of reducing the number of parameters being used at every inception module. Below is an image of the inception V3 module.

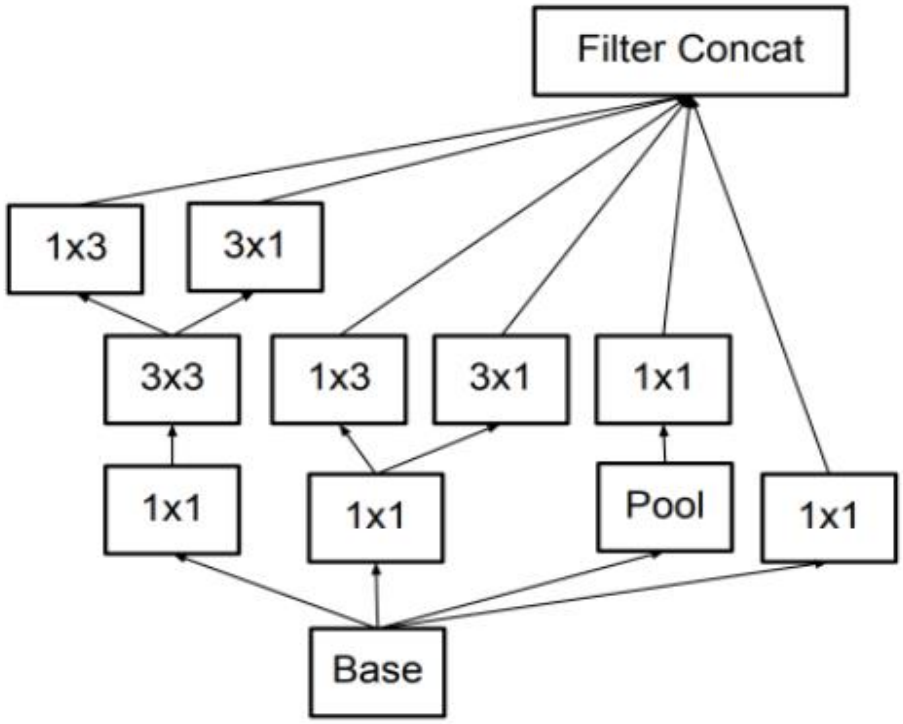


Figure 7. Inception modules with expanded the filter bank outputs. This architecture is used on the coarsest (8×8) grids to promote high dimensional representations, as suggested by principle 2 of Section 2. We are using this solution only on the coarsest grid, since that is the place where producing high dimensional sparse representation is the most critical as the ratio of local processing (by 1×1 convolutions) is increased compared to the spatial aggregation.

Fig. 2.16. Inception Modules with Filter

2.4 Results and Discussion

In the previous research papers, the author used modified resnet to classify msi and mss and achieved an accuracy of 87%. We have used inception v3 to classify the msi and mss and got an accuracy of 92%.

Inception v3 graphs:

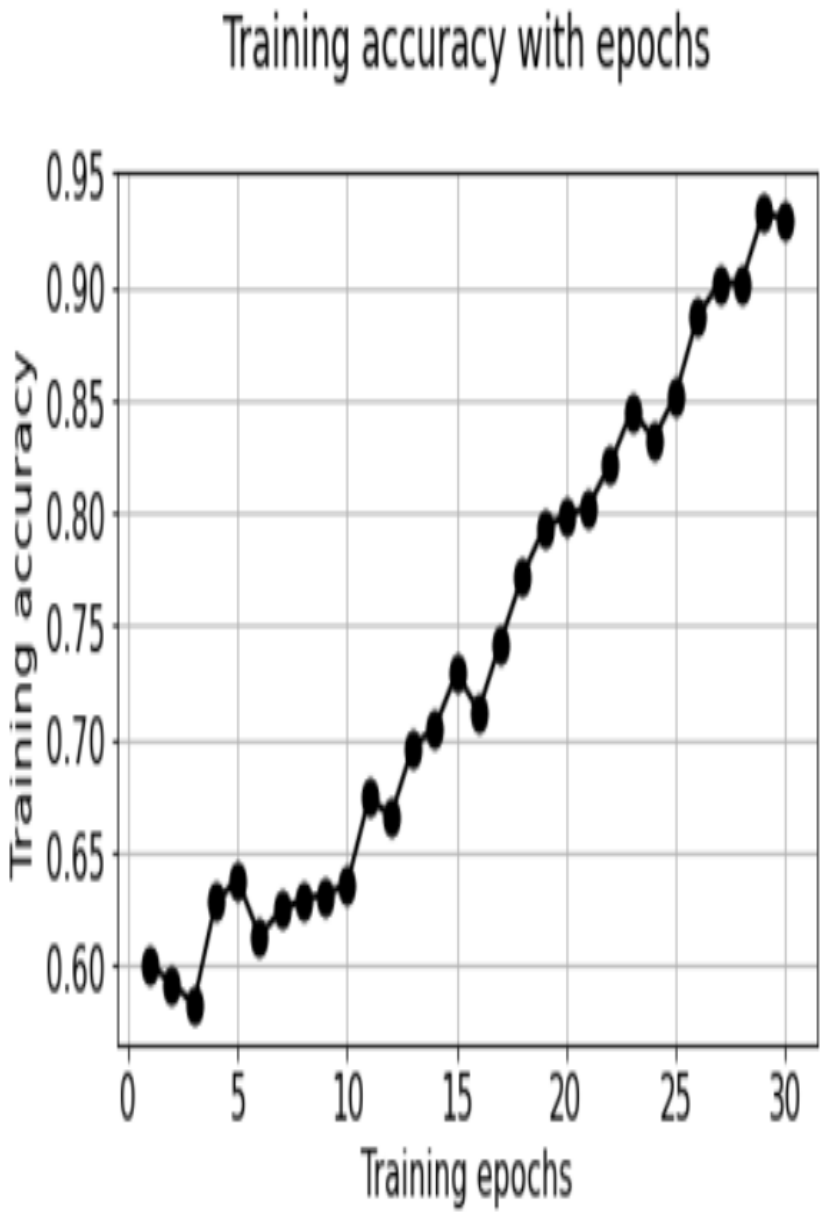


Fig. 2.17. Training Accuracy with Epochs Graph

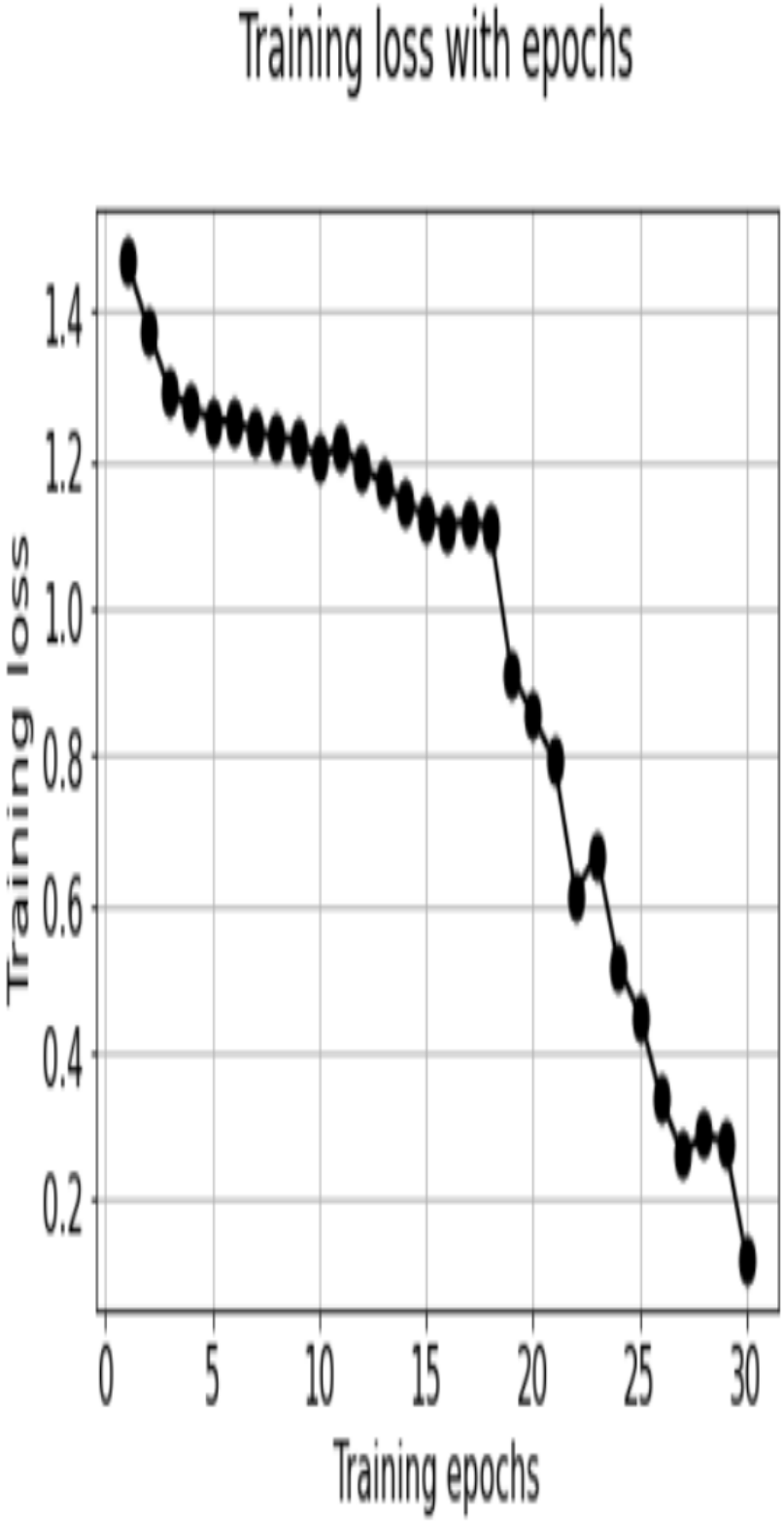


Fig. 2.18. Training Loss with Epochs Graph

Modified resnet:

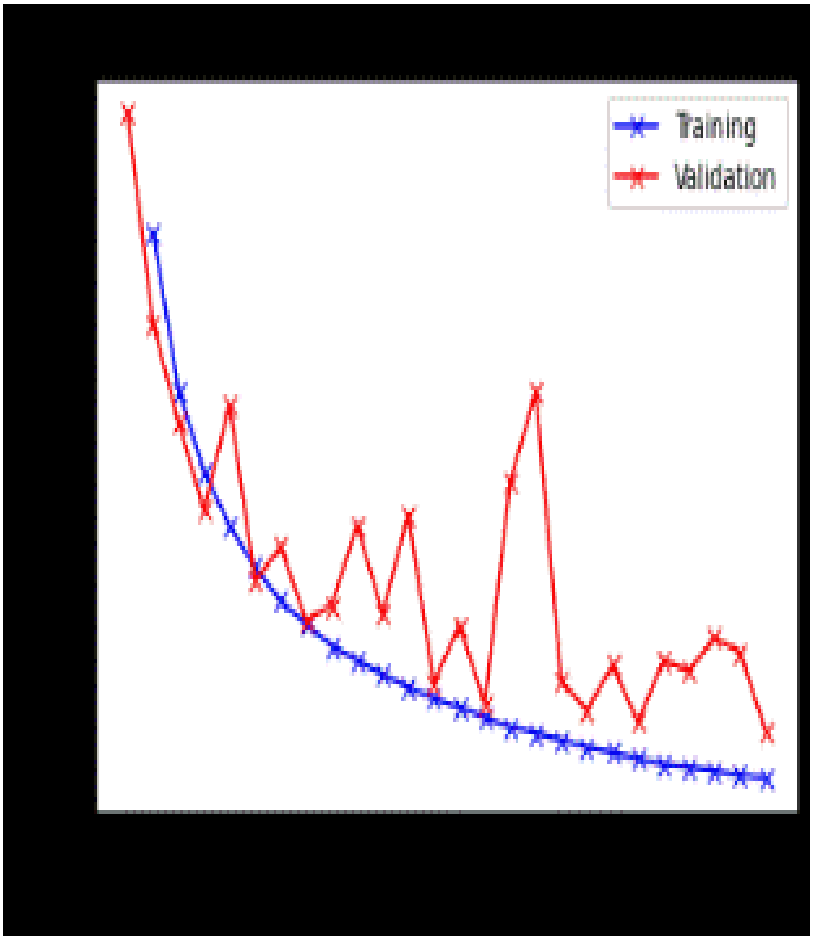


Fig. 2.19. Loss Vs number of epochs of Modified ResNet Model

Table 1. Accuracy Comparison of different Architectures

	Accuracy Values	
Architecture	**Original Dataset (1.5 Lakh Images)**	**Sample Dataset (50 Images)**
Resnet18 (Research Paper)	87%	NA
Resnet50	90%	52%
InceptionV3	92%	60%

Outputs using Inception V3 Architecture:

```
Epoch 1/30
200/200 [==============================] - 54m 7s/step - loss: 1.4677 - accuracy: 0.6001
Epoch 2/30
200/200 [==============================] - 57m 9s/step - loss: 1.3759 - accuracy: 0.5921
Epoch 3/30
200/200 [==============================] - 53m 7s/step - loss: 1.2911 - accuracy: 0.5827
Epoch 4/30
200/200 [==============================] - 61m 9s/step - loss: 1.2689 - accuracy: 0.6278
Epoch 5/30
200/200 [==============================] - 58m 8s/step - loss: 1.2547 - accuracy: 0.6377
Epoch 6/30
200/200 [==============================] - 65m 9s/step - loss: 1.2492 - accuracy: 0.6128
Epoch 7/30
200/200 [==============================] - 55m 7s/step - loss: 1.2387 - accuracy: 0.6242
Epoch 8/30
200/200 [==============================] - 63m 9s/step - loss: 1.2296 - accuracy: 0.6289
Epoch 9/30
200/200 [==============================] - 64m 9s/step - loss: 1.2253 - accuracy: 0.6312
Epoch 10/30
200/200 [==============================] - 58m 7s/step - loss: 1.2007 - accuracy: 0.6354
Epoch 11/30
200/200 [==============================] - 57m 9s/step - loss: 1.2192 - accuracy: 0.6739
Epoch 12/30
200/200 [==============================] - 61m 9s/step - loss: 1.1892 - accuracy: 0.6658
Epoch 13/30
200/200 [==============================] - 65m 9s/step - loss: 1.1679 - accuracy: 0.6966
Epoch 14/30
200/200 [==============================] - 54m 7s/step - loss: 1.1418 - accuracy: 0.7047
Epoch 15/30
200/200 [==============================] - 59m 8s/step - loss: 1.1237 - accuracy: 0.7292
```

```
Epoch 16/30
200/200 [==============================] - 60m 9s/step - loss: 1.1110 - accuracy: 0.7122
Epoch 17/30
200/200 [==============================] - 64m 9s/step - loss: 1.1149 - accuracy: 0.7427
Epoch 18/30
200/200 [==============================] - 59m 9s/step - loss: 1.1093 - accuracy: 0.7719
Epoch 19/30
200/200 [==============================] - 62m 9s/step - loss: 0.9137 - accuracy: 0.7935
Epoch 20/30
200/200 [==============================] - 57m 7s/step - loss: 0.8542 - accuracy: 0.7986
Epoch 21/30
200/200 [==============================] - 53m 7s/step - loss: 0.7939 - accuracy: 0.8025
Epoch 22/30
200/200 [==============================] - 56m 6s/step - loss: 0.6145 - accuracy: 0.8212
Epoch 23/30
200/200 [==============================] - 62m 8s/step - loss: 0.6691 - accuracy: 0.8454
Epoch 24/30
200/200 [==============================] - 55m 8s/step - loss: 0.5179 - accuracy: 0.8325
Epoch 25/30
200/200 [==============================] - 57m 8s/step - loss: 0.4517 - accuracy: 0.8522
Epoch 26/30
200/200 [==============================] - 59m 8s/step - loss: 0.3375 - accuracy: 0.8869
Epoch 27/30
200/200 [==============================] - 59m 8s/step - loss: 0.2626 - accuracy: 0.9015
Epoch 28/30
200/200 [==============================] - 61m 9s/step - loss: 0.2885 - accuracy: 0.9017
Epoch 29/30
200/200 [==============================] - 58m 7s/step - loss: 0.2792 - accuracy: 0.9326
Epoch 30/30
200/200 [==============================] - 63m 9s/step - loss: 0.1222 - accuracy: 0.9297
```

Fig. 2.20. Training Accuracy Run History

2.4.1 SAMPLE CODE

MODULES

1. Collecting the dataset

The dataset collected from the Kaggle; dataset consists of the histological images of gastro intestinal tract. There are 1,92,312 images which are of size around 224 x 224.

2. Data Pre-processing

Preprocessing refers to all the transformations on the raw data before it is fed to the machine learning or deep learning algorithm. For instance, training a convolutional neural network on raw images will probably lead to bad classification performances. The preprocessing is also important to speed up training (for instance, centering and scaling techniques, etc.).

```
In [9]: from tensorflow.keras.preprocessing.image import ImageDataGenerator

        # All images will be rescaled by 1./255
        data_generator = ImageDataGenerator(rescale=1/255,validation_split=0.2)
```

Fig. a. Data Pre-processing Code Fragment

The basics of some preprocessing techniques that can be applied to any kind of data — mean normalization, standardization, and whitening.

A. Mean normalization

Mean normalization is just removing the mean from each observation.

B. Standardization or normalization

Standardization is used to put all features on the same scale. Each zero-centered dimension is divided by its standard deviation.

C. Whitening

Whitening, or sphering, data means that we want to transform it to have a covariance matrix that is the identity matrix — 1 in the diagonal and 0 for the other cells. It is called whitening in reference to white noise.

3. Rescale the data

The next step is to scale the uncorrelated matrix in order to obtain a covariance matrix corresponding to the identity matrix. To do that, we scale our decorrelated data by dividing each dimension by the square-root of its corresponding eigenvalue.

Note: we add a small value (here 10^-5) to avoid division by 0.

Rescaling of all images is only the pre-processing technique. In real world business problems, the raw data that was collected from hospital or an organization should be pre-processed. But the dataset that we have considered was already a pre-processed data, so there are no other pre-processing techniques are added.

4. CNN Model Building

We have developed a model using CNN inception v3 architecture. For this we have used TensorFlow and Keras modules.

Difference between TensorFlow and Keras

Both TensorFlow and Keras are famous machine learning modules used in the field of data science. In this article, we will look at the advantages, disadvantages and the difference between these libraries.

TensorFlow

TensorFlow is an open-source platform for machine learning and a symbolic math library that is used for machine learning applications.

Advantages of TensorFlow

Tensor flow has a better graph representation for a given data rather than any other top platform out there.

- Tensor flow has the advantage that it does support and uses many backend software like GUI and ASIC.
- When it comes to community support tensor flow has the best.
- Tensor flow also helps in debugging the sub-part of the graphs.
- Tensor flow has shown a better performance when compared with other platforms.
- Easy to extend as it gives freedom to add custom blocks to build on new ideas.

Disadvantages of TensorFlow

- Tensor flow not specifically designed for the Windows operating systems but it is designed for other OS like Linux but tensor flow can be installed in windows with the help of a python package installer(pip).
- The speed of the tensor flow is less when it is compared to other platforms of the same type.
- For a better understanding of tensor flow, the user must have the fundamentals of calculus.
- Tensor flow does not support OpenCL.

Keras

It is an Open-Source Neural Network library that runs on top of Theano or TensorFlow. It is designed to be fast and easy for the user to use. It is a useful library to construct any deep learning algorithm of whatever choice we want.

Advantages of Keras:

- Keras is the best platform out there to work on neural network models.
- The API that Keras has a user-friendly where a beginner can easily understand.
- Keras has the advantage that it can choose any libraries which support it for its backend support.
- Keras provides various pre-trained models which help the user in further improving the models the user is designing.
- When it comes to community support Keras has the best like stack overflow.

Disadvantages of Keras:

- The major drawback of Keras is it is a low-level application programming interface.
- Few of the pre-trained models that the Keras has been not much supportive when it comes to designing of some models.
- The errors given by the Keras library were not much helpful for the user.

Table. 2. Difference between TensorFlow and Keras

S.No	TensorFlow	Keras
1.	Tensor high-performance Flow is written in C++, CUDA, Python.	Keras is written in Python.

S.No	TensorFlow	Keras
2.	TensorFlow is used for large datasets and high-performance models.	Keras is usually used for small datasets.
3.	TensorFlow is a framework that offers both high and low-level APIs.	Keras is a high-Level API.
4.	TensorFlow is used for high-performance models.	Keras is used for low-performance models.
5.	In TensorFlow performing debugging leads to complexities.	In Keras framework, there is only minimal requirement for debugging the simple networks.
6.	TensorFlow has a complex architecture and not easy to use.	Keras has a simple architecture and easy to use.
7.	TensorFlow was developed by the Google Brain team.	Keras was developed by François Chollet while he was working on the part of the research effort of project ONEIROS.

```
In [7]: from tensorflow.keras.applications.inception_v3 import InceptionV3
```

```
In [8]: batch_size=64
```

```
In [10]: base_model = InceptionV3(weights="imagenet", include_top=False, input_shape=(224,224,3))
         base_model.trainable = True
```

```
In [11]: base_model.summary()
```

```
Model: "inception_v3"
```

```
In [11]: base_model.summary()
```

```
Model: "inception_v3"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 111, 111, 32) 864         input_1[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 111, 111, 32) 96          conv2d[0][0]
__________________________________________________________________________________________________
activation (Activation)         (None, 111, 111, 32) 0           batch_normalization[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 109, 109, 32) 9216        activation[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 109, 109, 32) 96          conv2d_1[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 109, 109, 32) 0           batch_normalization_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 109, 109, 64) 18432       activation_1[0][0]
```

```
In [12]: from tensorflow.keras import layers, models

         flatten_layer = layers.Flatten()
         dense_layer_1 = layers.Dense(50, activation='relu')
         dense_layer_2 = layers.Dense(20, activation='relu')
         prediction_layer = layers.Dense(1, activation='softmax')


         model = models.Sequential([
             base_model,
             flatten_layer,
             dense_layer_1,
             dense_layer_2,
             prediction_layer
         ])
```

```
In [13]: model.summary()

         Model: "sequential"
         _________________________________________________________________
         Layer (type)                 Output Shape              Param #
         =================================================================
         inception_v3 (Functional)    (None, 5, 5, 2048)        21802784
         _________________________________________________________________
         flatten (Flatten)            (None, 51200)             0
         _________________________________________________________________
         dense (Dense)                (None, 50)                2560050
         _________________________________________________________________
         dense_1 (Dense)              (None, 20)                1020
         _________________________________________________________________
         dense_2 (Dense)              (None, 1)                 21
         =================================================================
         Total params: 24,363,875
         Trainable params: 24,329,443
         Non-trainable params: 34,432
```

Fig. b. CNN Model Building Code Fragment

5. Train and Test Split

One of the golden rules in machine learning is to split your dataset into train, validation, and test set. Learn how to bypass the most common caveats!

The reason we do that is very simple. If we would not split the data into different sets the model would be evaluated on the same data it has seen during training. We therefore could run into problems such as overfitting without even knowing it.

Back before using deep learning models, we often used three different sets.

- A train set is used for training the model
- A validation set that is used to evaluate the model during the training process
- A test set that is used to evaluate the final model accuracy before deployment

How do we use the train, validation, and test set?

Usually, we use the different sets as follows:

1. We split the dataset *randomly* into three subsets called the train**,** validation, and test set. Splits could be 60/20/20 or 70/20/10 or any other ratio you desire.
2. We train a model using the train set.
3. During the training process, we evaluate the model on the validation set.
4. If we are not happy with the results, we can change the hyperparameters or pick another model and go again to step 2
5. Finally, once we're happy with the results on the validation set, we can evaluate our model on the test set.
6. If we're happy with the results we can now again train our model on the train and validation set combined using last the hyperparameters we derived.
7. We can again evaluate the model accuracy on the test set and if we're happy deploy the model.

Most ML frameworks provide built-in methods for random train/ test splits of a dataset. The most well-known example is the train_test_split function of scikit-learn.

Illustration from Wikipedia showing how k-fold cross-validation works. We iteratively shuffle the data that is used for training and testing and evaluate the overall statistics.

This approach is barely used in recent deep learning methods as it’s super expensive to train a model k times.

With the rise of deep learning and the massive increase in dataset sizes, the need for techniques such as cross-validation or having a separate validation set has diminished. One reason for this is that experiments are very expensive and take a long time. Another one is that due to the large datasets and nature of most deep learning methods the models got less affected by overfitting.

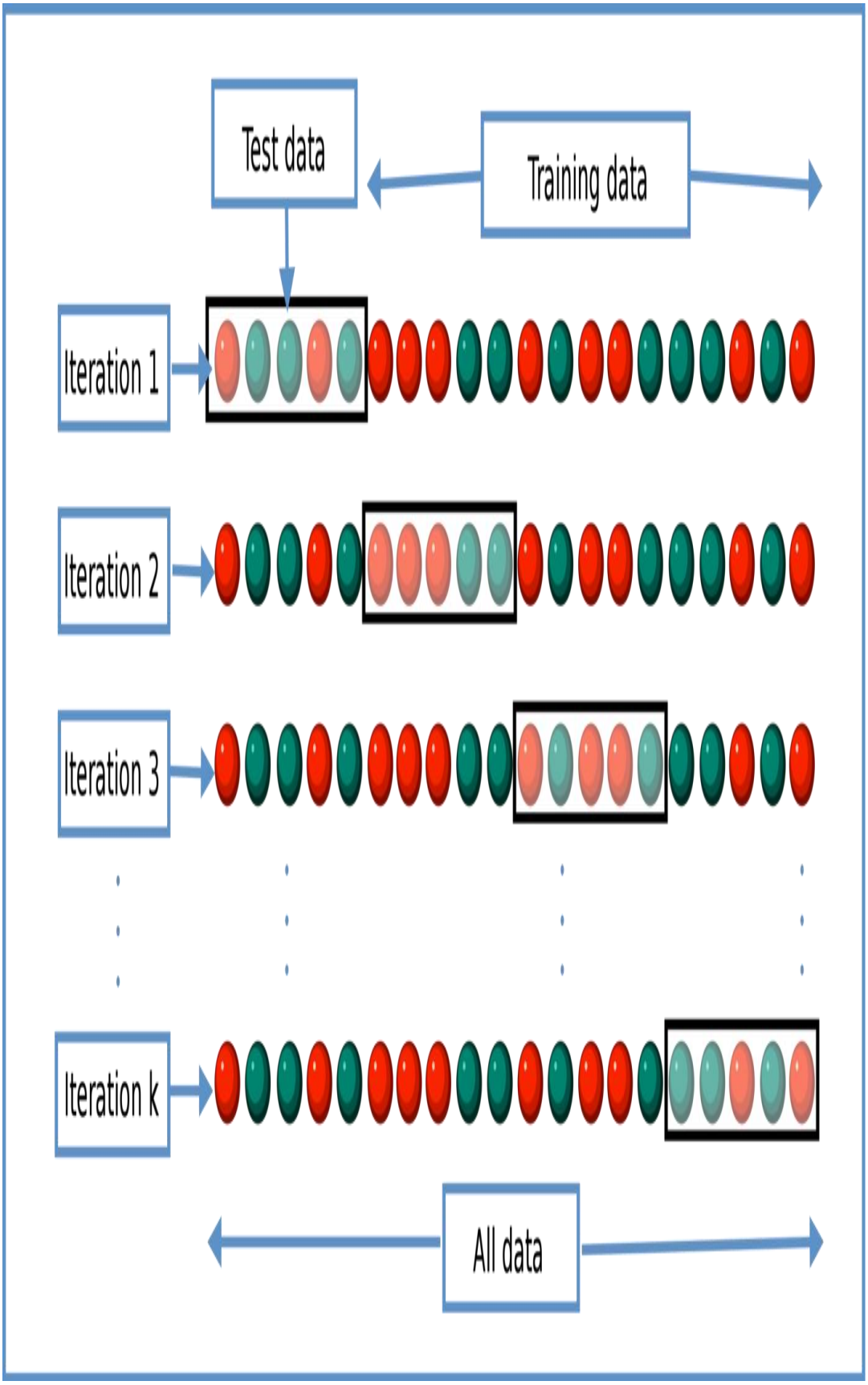


Fig. c. Splitting Train and Test Data

Overfitting is still a problem in deep learning. But overfitting to 50 samples with 10 features happens faster than overfitting to 100k images with millions of pixels

One could argue that researchers and practitioners got lazy/ sloppy. It would be interesting to see any recent paper investigating such effects again. For example, it could be that researchers in the past years have heavily overfitted their models to the test set of ImageNet as there has been an ongoing struggle to improve it and become state-of-the-art.

How should I pick my train, validation, and test set?

Naively, one could just manually split the dataset into three chunks. The problem with this approach is that we humans are very biased and this bias would get introduced into the three sets.

In academia, we learn that we should pick them randomly. A random split into the three sets guarantees that all three sets follow the same statistical distribution. And that's what we want since ML is all about statistics.

Deriving the three sets from completely different distributions would yield some unwanted results. There is not much value in training a model on pictures of cats if we want to use it to classify flowers.

How should I pick my train, validation, and test set?

However, the underlying assumption of a random split is that the initial dataset already matches the statistical distribution of the problem we want to solve. That would mean that for problems such as autonomous driving the assumption is that our dataset covers all sorts of cities, weather conditions, vehicles, seasons of the year, special situations, etc.

As you might think this assumption is actually not valid for most practical deep learning applications. Whenever we collect data using sensors in an uncontrolled environment, we might not have the desired data distribution.

But that's bad. What am I supposed to do if I'm not able to collect a representative dataset of the problem I try to solve?

What you're looking for is the research area around finding and dealing with domain gaps**,** distributional shifts, or data drift. All these terms have their own specific definition. I'm listing them here so you can search for the relevant problems easily.

With a domain, we refer to the data domain, as the source and type of the data we use. There are three ways to move forward:

- Solve the data gap by collecting more representative data
- Use data curation methods to make the data already collected more representative
- Focus on building a robust enough model to handle such domain gaps

The latter approach is focusing on building models for out-of-distribution tasks.

Picking a train test split for out-of-distribution tasks

In machine learning, we refer to out-of-distribution whenever our model has to perform well in a situation where the new input data is from a different distribution than the training data. Going back to our autonomous driving example from before, we could say that for a model that has only been trained on sunny California weather, doing predictions in Europe is out of distribution.

Now, how should we do the split of the dataset for such a task?

Since we collected the data using different sensors, we also might have additional information about the source for each of the samples (a sample could be an image, lidar frame, video, etc.).

We can solve this problem by splitting the dataset in the following way:

- we train on a set of data from cities in list A
- and evaluate the model on a set of data from cities in list B

The dataset1 should be split into 2 parts train and test. We have 2 folders separated as train and test with both MSI and MSS images.

```
# Flow training images in batches of 128 using train_datagen generator
train_generator = data_generator.flow_from_directory(
        'Dataset/train/',  # This is the source directory for training images
        target_size=(224, 224),  # All images will be resized to 224 x 224
        batch_size=batch_size,
        # Specify the classes explicitly
        classes = ['MSIMUT','MSS'],
        # Since we use categorical_crossentropy loss, we need categorical labels
        class_mode='binary',subset="training")

test_generator = data_generator.flow_from_directory(
        'Dataset/train/',  # This is the source directory for training images
        target_size=(224, 224),  # All images will be resized to 224 x 224
        batch_size=batch_size,
        # Specify the classes explicitly
        classes = ['MSIMUT','MSS'],
        # Since we use categorical_crossentropy loss, we need categorical labels
        class_mode='binary',subset="validation")
```

```
Found 123080 images belonging to 2 classes.
Found 30769 images belonging to 2 classes.
```

Fig. d. Train and Test Split Code Fragment

6. Training and Testing the Model

Two techniques, normalization and standardization, both have the objective of transforming the data by putting all the data points on the same scale in preparation for training.

The normalization process usually consists of scaling the numerical data down to a scale from zero to one. Standardization, on the other hand, usually consists of subtracting the mean of the dataset from each data point and then dividing the difference by the standard deviation of the datasets. That forces the standardized data to take on a mean of zero and a standard deviation of one. Standardization is often referred to as normalization; both boils down to putting data on some known or standard scale

Once a model has been trained, performance is gauged according to a confusion matrix and precision/accuracy metrics.

The algorithm will be trained with the images in the dataset and test the algorithm with sample test data. The performance of the algorithm will be measured using the accuracy.

```
In [16]: model.compile(
             optimizer=RMSprop(lr=0.0001),
             loss='binary_crossentropy',
             metrics=['accuracy'],
         )
```

```
In [22]: total_sample=train_generator.n
         n_epochs=30
```

```
In [18]: history = model.fit_generator(
                 train_generator,
                 steps_per_epoch=200,
                 epochs=n_epochs,
                 verbose=1)
```

```
Epoch 1/30
200/200 [==============================] - 54m 7s/step - loss: 1.4677 - accuracy: 0.6001
Epoch 2/30
200/200 [==============================] - 57m 9s/step - loss: 1.3759 - accuracy: 0.5921
Epoch 3/30
200/200 [==============================] - 53m 7s/step - loss: 1.2911 - accuracy: 0.5827
Epoch 4/30
200/200 [==============================] - 61m 9s/step - loss: 1.2689 - accuracy: 0.6278
Epoch 5/30
200/200 [==============================] - 58m 8s/step - loss: 1.2547 - accuracy: 0.6377
Epoch 6/30
200/200 [==============================] - 65m 9s/step - loss: 1.2492 - accuracy: 0.6128
Epoch 7/30
200/200 [==============================] - 55m 7s/step - loss: 1.2387 - accuracy: 0.6242
Epoch 8/30
200/200 [==============================] - 63m 9s/step - loss: 1.2296 - accuracy: 0.6289
Epoch 9/30
200/200 [==============================] - 64m 9s/step - loss: 1.2253 - accuracy: 0.6312
Epoch 10/30
200/200 [==============================] - 58m 7s/step - loss: 1.2007 - accuracy: 0.6354
Epoch 11/30
200/200 [==============================] - 57m 9s/step - loss: 1.2192 - accuracy: 0.6739
Epoch 12/30
200/200 [==============================] - 61m 9s/step - loss: 1.1892 - accuracy: 0.6658
Epoch 13/30
200/200 [==============================] - 65m 9s/step - loss: 1.1679 - accuracy: 0.6966
Epoch 14/30
200/200 [==============================] - 54m 7s/step - loss: 1.1418 - accuracy: 0.7047
Epoch 15/30
200/200 [==============================] - 59m 8s/step - loss: 1.1237 - accuracy: 0.7292
```

```
Epoch 16/30
200/200 [==============================] - 60m 9s/step - loss: 1.1110 - accuracy: 0.7122
Epoch 17/30
200/200 [==============================] - 64m 9s/step - loss: 1.1149 - accuracy: 0.7427
Epoch 18/30
200/200 [==============================] - 59m 9s/step - loss: 1.1093 - accuracy: 0.7719
Epoch 19/30
200/200 [==============================] - 62m 9s/step - loss: 0.9137 - accuracy: 0.7935
Epoch 20/30
200/200 [==============================] - 57m 7s/step - loss: 0.8542 - accuracy: 0.7986
Epoch 21/30
200/200 [==============================] - 53m 7s/step - loss: 0.7939 - accuracy: 0.8025
Epoch 22/30
200/200 [==============================] - 56m 6s/step - loss: 0.6145 - accuracy: 0.8212
Epoch 23/30
200/200 [==============================] - 62m 8s/step - loss: 0.6691 - accuracy: 0.8454
Epoch 24/30
200/200 [==============================] - 55m 8s/step - loss: 0.5179 - accuracy: 0.8325
Epoch 25/30
200/200 [==============================] - 57m 8s/step - loss: 0.4517 - accuracy: 0.8522
Epoch 26/30
200/200 [==============================] - 59m 8s/step - loss: 0.3375 - accuracy: 0.8869
Epoch 27/30
200/200 [==============================] - 59m 8s/step - loss: 0.2626 - accuracy: 0.9015
Epoch 28/30
200/200 [==============================] - 61m 9s/step - loss: 0.2885 - accuracy: 0.9017
Epoch 29/30
200/200 [==============================] - 58m 7s/step - loss: 0.2792 - accuracy: 0.9326
Epoch 30/30
200/200 [==============================] - 63m 9s/step - loss: 0.1222 - accuracy: 0.9297
```

Saving the model weights

```
In [ ]: from sklearn.metrics import accuracy_score
```

```
In [ ]: predict_x=model.predict(test_generator, verbose=1)

        predicted_class=np.argmax(predict_x,axis=1)
        original_class=test_generator.labels
        mobilenet=accuracy_score(original_class,predicted_class)*100
        print("Test set accuracy "+str(mobilenet))
```

Fig. e. Training and Testing the Model Code Fragment

7. Prediction Using Sample Image

A saved model can be used in multiple places, such as to continue training, to fine tune the model, and for prediction.

The algorithm is given a sample image to classify whether it is MSI or MSS.

```
In [34]: result=loaded_model.predict(my_image)
         import numpy as np
         class_value=np.argmax(result)
```

```
In [35]: if(class_value==0):
             print("The given image was MSIMUT class")
         elif(class_value==1):
             print("The given image was MSS class")
```

Fig. f. Prediction Using Sample Image Code Fragment

2.5 The Main Contribution of the Chapter

As the health is an important aspect, the concept of AI as deep learner can be helpful in finding the MSI and MSS in gastrointestinal cancer detection at an early stage. The main objective of this paper is to find out the best technique for gastrointestinal cancer detection than previous researches. Deep Learning has proven boon to this aspect, it gives accurate and generalized results on the large datasets. In this paper, various techniques like resnet18 and inception v3 are described for gastrointestinal tract cancer detection. After the comparative analysis of accuracy and training loss graphs, it has been analysed that inception v3 architecture has been proved best technique till now.

2.6 Conclusion

Many studies have already been published as stepping-stones toward the application of AI in diagnosing EGC. Some systems even showed high accuracy which could be compared with those of experts. In this paper, the high potential and shortcomings in deep learning research studies applied to gastrointestinal cancer. Our results demonstrate the effectiveness of gastrointestinal cancer tissue analysis by deep learning applications using inception v3 in cnn and with an accuracy of 92%. Moreover, we have taken a dataset having more images around 1.9 lakh , but we got more accuracy than the previous researches. The CNN detected more early gastric cancer cases in a shorter time than the endoscopists. The CNN needs further training to achieve higher diagnostic accuracy. However, a diagnostic support tool for gastric cancer using a CNN will be realized in the near future.

CHAPTER – 3

CONCLUSIONS AND FUTURE SCOPE

Many studies have already been published as stepping-stones toward the application of AI in diagnosing EGC. Some systems even showed high accuracy which could be compared with those of experts. However, before AI systems were verified by multicenter RCTs, it seems reasonable to only use AI for auxiliary diagnosis, like determining whether there is a blind spot during the process of EGD. Early detection of the Gastrointestinal Cancer was very important in diagnosis stage, so we can treat the patients early and prevent the damage to the near-by cells So we have developed a computer vision application which can be used on the histological images of gastrointestinal area and get the good results than the existing methods, so we have trained an CNN algorithm called Inception v3 model and get the accuracy of 92%. In future we can use the more image dataset and get the above the 95% accuracy.

BIBLIOGRAPHY

[1] Jiménez-Gaona, Y., Rodríguez-Álvarez, M.J. and Lakshminarayanan, V., 2020. Deep-Learning-Based Computer-Aided Systems for Breast Cancer Imaging: A Critical Review. Applied Sciences, 10(22), p.8298.

[2] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[3] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[4] Dhanasekaran, S.M., Barrette, T.R., Ghosh, D., Shah, R., Varambally, S., Kurachi, K., Pienta, K.J., Rubin, M.A. and Chinnaiyan, A.M., 2001. Delineation of prognostic biomarkers in prostate cancer. Nature, 412(6849), pp.822-826.

[5] Brinker, T.J., Hekler, A., Utikal, J.S., Grabe, N., Schadendorf, D., Klode, J., Berking, C., Steeb, T., Enk, A.H. and Von Kalle, C., 2018. Skin cancer classification using convolutional neural networks: systematic review. Journal of medical Internet research, 20(10), p.e11936.

[6] Iizuka, O., Kanavati, F., Kato, K., Rambeau, M., Arihiro, K. and Tsuneki, M., 2020. Deep learning models for histopathological classification of gastric and colonic epithelial tumours. Scientific reports, 10(1), pp.1-11.

[7] Kather, J.N., Pearson, A.T., Halama, N., Jäger, D., Krause, J., Loosen, S.H., Marx, A., Boor, P., Tacke, F., Neumann, U.P. and Grabsch, H.I., 2019. Deep learning can predict microsatellite instability directly from histology in gastrointestinal cancer. Nature medicine, 25(7), pp.1054-1056.

[8] Ilhan, U., Uyar, K. and Iseri, E.I., 2020, August. Breast Cancer Classification Using Deep Learning. In International Conference on Theory and Applications of Fuzzy Systems and Soft Computing (pp. 709-714). Springer, Cham.

[9] Yamada, M., Saito, Y., Imaoka, H., Saiko, M., Yamada, S., Kondo, H., Takamaru, H., Sakamoto, T., Sese, J., Kuchiba, A. and Shibata, T., 2019. Development of a real-time endoscopic image diagnosis support system using deep learning technology in colonoscopy. Scientific reports, 9(1), pp.1-9.

[10] Basha, S.S., Ghosh, S., Babu, K.K., Dubey, S.R., Pulabaigari, V. and Mukherjee, S., 2018, November. Rccnet: An efficient convolutional neural network for histological routine colon cancer nuclei classification. In 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV) (pp. 1222-1227). IEEE.

[11] Lu, D., Polomac, N., Gacheva, I., Hattingen, E. and Triesch, J., 2021, June. Human-expert-level brain tumor detection using deep learning with data distillation and augmentation. In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 3975-3979). IEEE.

[12] Khairnar, P., Thiagarajan, P. and Ghosh, S., 2020. A modified Bayesian Convolutional Neural Network for Breast Histopathology Image Classification and Uncertainty Quantification. arXiv preprint arXiv:2010.12575.

[13] Hasan, M.R. and Kabir, M.A., 2019. Lung Cancer Detection and Classification based on Image Processing and Statistical Learning. arXiv preprint arXiv:1911.10654.

[14] Darvin, P., Toor, S. M., Sasidharan Nair, V. & Elkord, E. Exp. Mol. Med. 50, 165 (2018).

[15] Le, D. T. et al. N. Engl. J. Med. 372, 2509–2520 (2015).

[16] Bonneville, R. et al. JCO Precis. Oncol. 2017, 1–15 (2017).

[17] Le, D. T. et al. Science 357, 409–413 (2017).

[18] Kather, J. N., Halama, N. & Jaeger, D. Semin. Cancer Biol. 52, 189–197 (2018).

[19] Franke, A. J. et al. J. Clin. Oncol. 36, 796 (2018).

[20] 25. Liu, Y. et al. Cancer Cell 33, 721–735 (2018)

REFERENCES

[1] Deep learning can predict microsatellite instability directly from histology in gastrointestinal cancer, Jakob Nikolas Kather, Alexander T. Pearson, Niels Halama, Dirk Jäger, Jeremias Krause, Sven H. Loosen, Alexander Marx, Peter Boor, Frank Tacke, Ulf Peter Neumann, Heike I. Grabsch, Takaki Yoshikawa, Hermann Brenner, Jenny Chang-Claude, Michael Hoffmeister, Christian Trautwein, Tom Luedde, https://www.nature.com/articles/s41591-019-0462-y

[2] Modified ResNet Model for MSI and MSS Classification of Gastrointestinal Cancer, CH Sai Venkatesh, Caleb Meriga, M.G.V.L Geethika, T Lakshmi Gayatri, V.B.K.L Aruna, https://arxiv.org/abs/2202.01905