Educate Elevate Enlighten

JSS Mahavidyapeetha

JSS Science And Technology University

(Established Under JSS Science and Technology University Act No. 43 of 2013)

(Formerly Known as SJCE)

JSS SCIENCE AND TECHNOLOGY UNIVERSITY MYSURU

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

REPORT FOR THE FINAL YEAR PROJECT

"Deep Learning approach for subtype classification and localization from non-small cell lung cancer histopathology images"

Project Report submitted in Partial Fulfillment of curriculum prescribed
for awarding the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE & ENGINEERING

By,

Project Batch No: B19

Sl-No	USN	Student Name
1.	01JST16CS035	Hymavathi B U
2.	01JST16CS132	Santosh Umesh Shet
3.	01JST17CS421	Tejas H R

Under the Guidance of

Dr. M P Pushpalatha

Professor & Head of Department of Computer Science and Engineering

JSS S&TU – Mysuru.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Year: **2019-2020**

JSS Mahavidyapeetha
JSS Science And Technology University
(Formerly Known as SJCE)

CERTIFICATE

This is to certify that the work entitled ***"Deep Learning approach for subtype classification and localization from non small cell lung cancer histopathology images"*** is a bonafied work carried out by ***Hymavathi B U, Santosh Umesh Shet, Tejas H R*** in partial fulfillment of the award of the degree of Bachelor of Engineering in Computer Science and Engineering of JSS Science And Technology University during the year 2019-20. It is certified that all corrections / suggestions indicated during CIE have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the Bachelor of Engineering degree.

Guide and HOD

Dr.M P Pushpalatha

HOD, Department of Computer Science & Engineering,

SJCE, Mysuru

Examiners: 1. .....

Place: Mysuru 2. .....

Date: 3. .....

Declaration

We hereby declare that the project work entitled ***" Deep Learning approach for subtype classification and localization from non small cell lung cancer histopathology images"*** submitted to the JSS STU, Mysuru, is a record of an original work done by us under the guidance of ***Dr. M P Pushpalatha***, HOD, Department of Computer Science and Engineering from JSS STU, Mysore in the partial fulfillment of the requirements for the award of the degree of Bachelors of Engineering in Computer Science Engineering during the year 2019-20. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree.

Hymavathi B U	**Santosh Umesh Shet**	**Tejas H R**
(01JST16CS035)	**(01JST16CS132)**	**(01JST17CS421)**

Acknowledgement

First and foremost, We Would like to express our sincere gratitude to ***Dr. M P Pushpalatha***, HOD of Dept. of Computer Science and Engineering, JSS STU for providing an excellent environment for our education and its encouragement throughout our stay in college.

We extend our heartfelt gratitude to our guide ***Dr. M P Pushpalatha***, HOD of Dept. of Computer Science and Engineering, JSS STU who has supported us throughout our project with his patience and knowledge whilst allowing us the room to work in our own way.

Abstract

Non-small cell lung carcinoma is a widely prevalent disease in which malignant cells form in the tissues of the lung. Adenocarcinoma (LUAD) and squamous cell carcinoma (LUSC) are two most predominant subtypes of non-small cell lung carcinoma. The analysis of the histopathology images of lung tissue is done to identify the type, subtype and the stage of cancer. In this project, we have trained deep learning Convolutional Neural Networks (CNN) models (Inception v3 and a custom model) on histopathology images obtained from The Cancer Genome Atlas (TCGA) dataset to accurately classify whole-slide pathology images into adenocarcinoma, squamous cell carcinoma or normal lung tissue. Our models also highlight the region of interest which predominantly influences the outcome of the model. The inception v3 model has an accuracy of 84% and our custom model provides the prediction with 98% accuracy.

Contents

List of Figures

Chapter 1

Introduction

According to the National Cancer Institute[1], the US federal government's principal agency for cancer research and training, Lung cancer accounts for more deaths than any other cancer in both men and women which amounts to about 28% of all cancer deaths. In the US alone, approximately 228,820 new cases of lung cancer and approximately 135,720 deaths from lung cancer are estimated for the year 2020[2]. Non small cell lung carcinoma(NSCLC) is a predominant type of lung cancer which amounts to 84% of all lung cancer cases. Adenocarcinoma (LUAD) and squamous cell carcinoma (LUSC) are two most common subtypes of non-small cell lung carcinoma, while there are several other types that occur less frequently. The cancer cells of each type grow and spread in different ways and visual analysis of a histopathology slide of lung tissue is used by pathologists to identify the subtype. Histopathological assessment is indispensable for the diagnosis of NSCLC and further on for the identification of the subtype. But, this poses a challenge, as assessment of the histopathological slide even by a trained pathologist is time intensive and sometimes prone to error. In order to overcome this hurdle, many approaches have been presented to provide an automated analysis of histopathological slide images which assists the pathologists and in some cases has provided better accuracies than a pathologist. Virtual microscopy of stained images of tissues are typically acquired at magnifications of x20 to x40, generating very large two-dimensional images (10,000 to over 100,000 pixels in each dimension)

that can be tricky to visually analyze in an exhaustive way. Previous approaches have utilized conventional thresholding, image processing techniques with Naïve Bayes, SVM and random forest classifiers to attain a Area Under the Curve (AUC) of 0.85 in distinguishing normal from tumor slides, and 0.75 in distinguishing LUAD from LUSC slides[3]. The healthcare sector is witnessing several advances lately due to the advent of deep learning approaches[4]. This problem domain of automated histopathological slide image analysis can greatly benefit from solutions using deep learning methodologies. Here, we have utilized two deep learning approaches for the automated histopathological slide image analysis. In our approach, we obtained haematoxylin and eosin stained whole-slide images from the Genomic Data Commons database which consisted of tumor tissues and normal tissues.These images were used to train two Convolution Network Network models, Inception v3 and a custom model developed specifically for this problem statement. Translation invariance of CNNs makes them ideal for identification of cancerous cells in a slide image. Inception v3 is a widely-used model that has been shown to attain greater than 78.1% accuracy on the over 1 million training images containing ImageNet dataset. The model is the culmination of many ideas developed by multiple researchers over the years. Inception v3 has already been successfully adapted for skin cancer classification, cervical cell classification, breast cancer classification and diabetic retinopathy detection among many other applications in the healthcare domain. Inception v3 architecture of using multiple kernels in each convolution layer and its ability to handle multiple resolution images makes it suitable for problems involving pathological analysis. Our custom designed model is robust and specifically equipped for the present purpose.

1.1 Aim/Statement of the problem

To develop a deep learning Convolutional Neural Network (CNN) model trained on histopathology images to accurately classify whole-slide pathology images into adenocarcinoma, squamous cell carcinoma or normal lung tissue to assist pathologists by providing a fast, accurate and inexpensive detection. Further, implementing an explainable model which provides localization of the region of interest and visualizes the model's results to make it easily understandable for both specialists and patients.

1.2 Objective of the project

- Data collection using GDC Tool - LUAD, LUSC and Normal Cells.
- Classifying whole-slide histopathology images into normal tissue slide or Non-Small Cell Lung Cancer(NSCLC) subtypes viz adenocarcinoma, squamous cell carcinoma.
- Compare the results and performance of Inception v3 and our custom designed CNN model with relevant graphs.
- Implementing Explainable model to highlight key features, that provides both specialists and patients a fast, accurate, easily understandable and inexpensive analysis of the whole-slide histopathology image.

1.3 Applications

- Accurate interpretation of whole slide tissues can be difficult even for trained pathologists and the distinction between LUAD and LUSC is not always clear, particularly in poorly-differentiated tumors, where ancillary studies are recommended for accurate classification. Our model provides accurate predictions to assist the experts.

- The model provides the predictions and the explainable output image in a matter of seconds, thus it is very time effective and pathologists can use the model without any time constraints.
- It is a cost effective solution that can be used extensively in places where healthcare services are hindered.
- Our explainable model highlights the regions of the tissue slide which contributed highest to the model's decision. Thus, it localizes the region of interest to aid pathologists in their analysis.

1.4 Existing solution methods

Studies have been performed using conventional digital image processing techniques along with machine learning methods such as Support Vector Machines, Naive Bayes Classiffier. These methods have accuracy at 75% levels in classifying LUAD and LUSC which leaves a lot to chance and makes the situation precarious. Deep learning methods have an edge over the conventional solutions. Convolutional Neural Networks based approach will outperform these methodologies.

1.5 Proposed solution methods

Our proposed solution method is the use of Deep Learning methodologies. Convolutional Neural Networks are being used extensively in medical image analysis and studies have shown their high accuracies. Usage of CNNs in our problem domain is limited and thus leaves a range of opportunities to explore. Our proposed methodology involves training inception v3 architecture and a custom model on the TCGA dataset and the comparison of their performances. Our solution further provides visualization of the output by generating a heatmap highlighting the areas that contributed most towards the classification as a particular class.

1.6 Time schedule for completion of the project work (Gantt Chart)

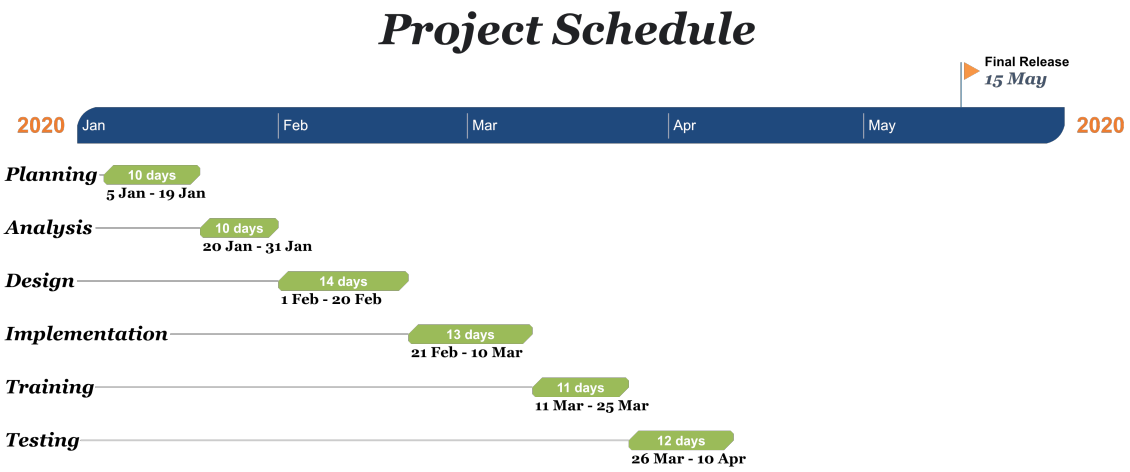


Figure 1.1: Gantt Chart

Chapter 2

Literature Survey

Deep Learning is emerging as the leading machine-learning tool in the general imaging and computer vision domains. Among various deep learning techniques, convolutional neural networks (CNNs) have proven to be powerful tools for a wide range of computer vision applications. Deep CNNs automatically learn mid-level and high-level abstractions obtained from raw data (e.g., images). Recent results indicate that the generic descriptors extracted from CNNs are extremely effective in object recognition and localization in natural images. Medical image analysis groups across the world are quickly entering the field and applying CNNs and other deep learning methodologies to a wide variety of applications[5]. One of the most widely used CNN model is Inception v3.

Inception v3 has been used in many automated medical image analysis tasks due to its robust architecture and for the fact that it has been shown to attain greater than 78.1% accuracy on the ImageNet dataset which consists of over 1 million training images.

Inception v3 has been used to solve a plethora of problems in the medical domain. The following studies highlight some of the applications:

In a study by A.Esteva et al.[6], inception v3 was trained end-to-end with 129,450 clinical images to achieve two critical binary classification use cases: keratinocyte carcinomas versus benign seborrheic keratoses; and malignant melanomas versus benign

nevi with AUC for each case over 91%.

Inception v3 has been used in the diagnosis of breast cancer. In a study by Y.Mednikov et al.[7], inception v3 has been trained on the INBreast database to detect breast cancer with a maximal area under the receiver operating characteristics curve of 0.91.

In a study by C.Wang et al.[8] ,a fine-tuned Inception-v3 model based on transfer learning has been trained on JSRT dataset to extract features automatically from chest x-ray images and then different classifiers (Softmax, Logistic, SVM) have been used to classify the pulmonary images with the highest accuracy being 86.4%.

Guan, Qing et al.[9] have exploited the Inception-v3 deep convolutional neural network model to differentiate cervical lymphadenopathy using cytological images. A dataset of 80 cases was collected through the fine-needle aspiration (FNA) of enlarged cervical lymph nodes, which consisted of 20 cases of reactive lymphoid hyperplasia, 24 cases of non-Hodgkin's lymphoma (NHL), 16 cases of squamous cell carcinoma (SCC), and 20 cases of adenocarcinoma. The classification accuracies for the original images of reactive lymphoid hyperplasia, NHL, SCC and adenocarcinoma were 88.46%, 80.77%, 89.29% and 100%, respectively. The total accuracy achieved on the test dataset was 89.62%.

In a study by Jin Li et al.[10], transfer learning of inception v3 model has been applied for Colorectal Cancer Lymph Node Metastasis Classification. The dataset from Harbin Medical University Cancer Hospital consisting of 619 samples, including 312 benign and 307 malignant were trained with an accuracy of 94.4%.

Yiming Ding et al.[11] have used inception v3 to predict diagnosis of Alzheimer Disease by using F-FDG PET images of the brain. Dataset of prospective F-FDG PET brain images from the Alzheimer's Disease Neuroimaging Initiative (ADNI) were used to train inception v3 to obtain an area under the ROC curve of 0.98.

Gulshan et al.[12] have used inception v3 architecture for identifying diabetic retinopathy in retinal fundus photographs. The datasets used: EyePACS-1 dataset consisting of 9963 images from 4997 patients ; the Messidor-2 dataset with 1748 images from 874 patients.For detecting referable diabetic retinopathy , the approach had an area

under the receiver operating curve of 0.991 for EyePACS-1 and 0.990 for Messidor-2. Considering the NSCLC problem domain, previous approaches towards automated pathological tissue slide analysis have used conventional image segmentation and machine learning methods to achieve considerable results.

Yu, Kun-Hsing et al.[13] in their study have extracted objective morphological information from thousands of whole slide non-small cell carcinoma images and then built a fully automated image-segmentation pipeline to identify the tumour nuclei and tumour cytoplasm from the histopathology images using the Otsu thresholding method. They extracted 9,879 quantitative image features and used regularized machine-learning methods to select the top features and to distinguish shorter-term survivors from longer-term survivors with an AUC of 0.85.

Chapter 3

System Requirements and Analysis

3.1 Hardware Requirements

- Processor – Intel Core i7-9750H processor, turbo up to 4.10 GHz
- Memory - 16GB DDR4 Ram
- Storage - 2TB HDD + 256GB SSD
- Graphics - NVIDIA GeForce GTX 1660ti with 6 GB of Dedicated GDDR5 VRAM Graphics

3.2 Software Requirements

- python 3.6.5
- tensorflow-gpu 1.9.0
- numpy 1.14.3
- matplotlib 2.1.2
- sklearn

- scipy 1.1.0
- openslide-python 1.1.1
- Pillow 5.1.0
- Pytorch 1.4.0
- Flask

3.3 Functional Requirements

1. The system should process the input given by the user only if it is an image.
2. System shall show the error message to the user when the input given is not in the required format.
3. System should detect features present in the image.
4. System should retrieve features present in the image and display them to the user.

3.4 Non-Functional Requirements

Non-functional requirements as the name suggests, are those requirements, which are not directly concerned with the specific functions delivered by the system. They may relate to the emergent system properties such as reliability and usability. Many non-functional requirements relate to the system as a whole rather than to the individual system features. They are sometimes more critical than individual functional requirements while failure to meet a non-functional system requirement then the whole system may be unusable. Therefore, non-functional are as equally important as the functional requirements and hence proper care should be taken to ensure that the below-mentioned non- functional requirements are satisfied to the full set.

The key non-functional requirements are as follows:

- **Reliability:** The system should be reliable and must not degrade the performance of the existing system and should not lead to the hanging of the system.
- **Usability:** A reasonably good Interface has to be provided. The users of the system should feel comfortable to use the application. The application should not confuse the users with too many options in a single screen.
- **Scalability:** The application is designed such that it is scalable.

Chapter 4

Tools and Technologies used

4.1 Tensorflow

TensorFlow is an open-source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) that flow between them. This flexible architecture lets you deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device without rewriting code.

TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well.

4.2 GDC Tool

The GDC Data Transfer Tool Client provides a command-line interface supporting both GDC data downloads and submissions.

The GDC Data Transfer Tool:

An Overview Raw sequence data, stored as BAM files, make up the bulk of data stored at the NCI Genomic Data Commons (GDC). The size of a single file can vary

greatly. Most BAM files stored in the GDC are in the 50 MB - 40 GB size range, with some of the whole genome BAM files reaching sizes of 200-300 GB. The GDC Data Transfer Tool, a command-line driven application, provides an optimized method of transferring data to and from the GDC and enables resumption of interrupted transfers.

Preparing for Data Downloads:

The GDC Data Transfer Tool is intended to be used in conjunction with the GDC Data Portal and the GDC Data Submission Portal to transfer data to or from the GDC. First, the GDC Data Portal's interface is used to generate a manifest file or obtain UUID(s) and (for Controlled-Access Data) an authentication token. The GDC Data Transfer Tool is then used to transfer the data files listed in the manifest file or identified by UUID(s).

Obtaining a Manifest File for Data Download:

The GDC Data Transfer Tool supports downloading multiple files listed in a GDC manifest file. Manifest files can be generated and downloaded directly from the GDC Data Portal

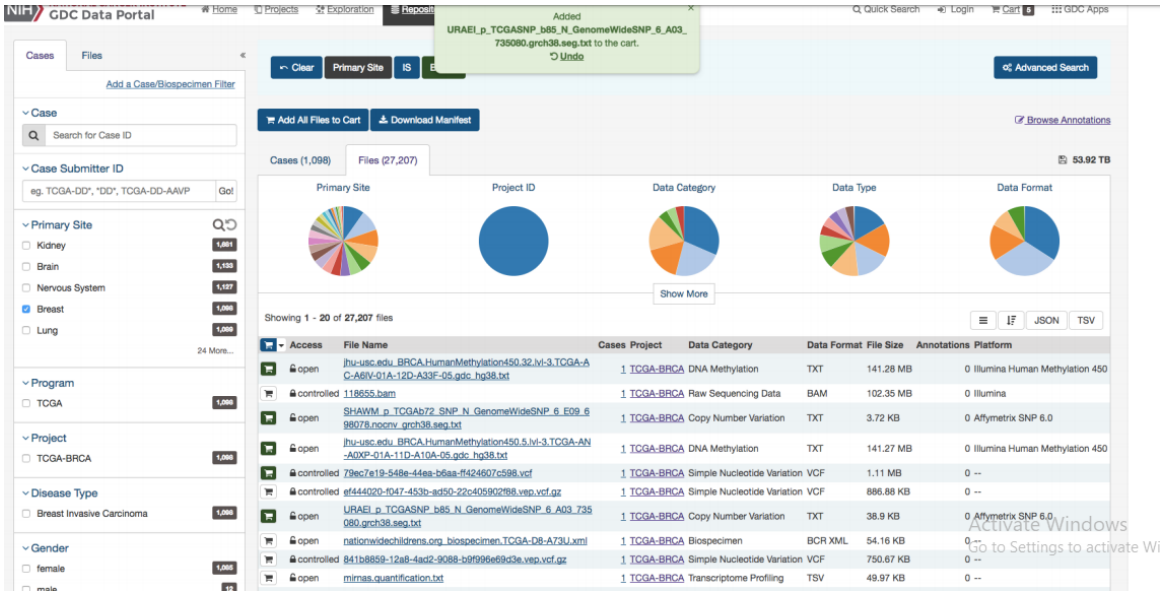


Figure 4.1: GDC Data Portal: Selecting Files of Interest

Obtaining UUIDs for Data Download:

A manifest file is not required to download files from GDC. The GDC Data Transfer Tool will accept file UUID(s) instead of a manifest file for downloading individual data files.

To download the dataset a manifest file is required which is obtained by manually selecting the whole slide tissue images from TCGA website and using generate option. The manifest file consists of UUID which is used to identify each case of cancer in TCGA dataset.

The size of each whole slide tissue image (.svs) varies from few MBs to GBs as each image resolution can be extremely large (+100,000 pixel wide). Reading these images using standard image tools or libraries is a challenge because these tools are typically designed for images that can comfortably be uncompressed into RAM or a swap file. Whole-slide images routinely exceed RAM sizes, often occupying tens of gigabytes when uncompressed.

gdc-client download –m gdc-manifest.txt

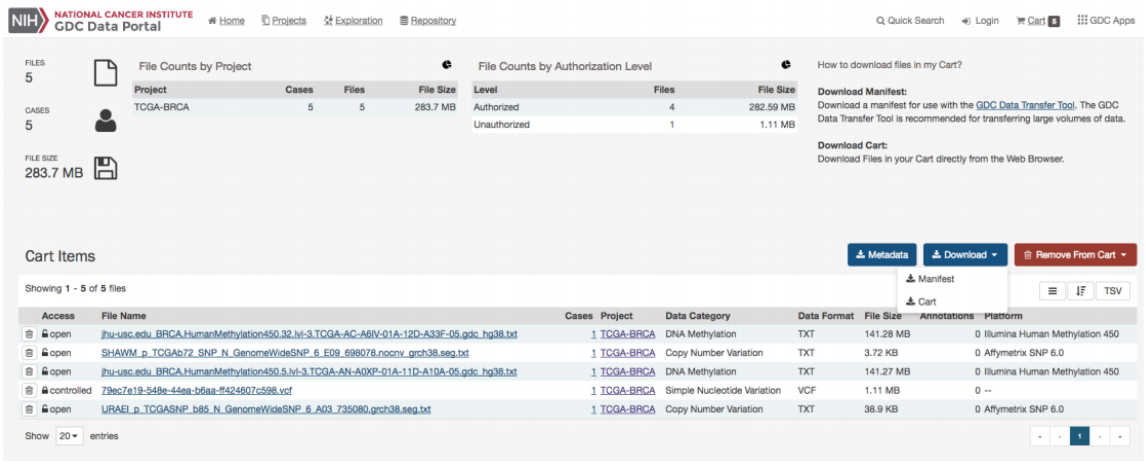


Figure 4.2: GDC Data Portal: Cart Page

Downloading Data Using a Manifest File

A convenient way to download multiple files from the GDC is to use a manifest file generated by the GDC Data Portal. After generating a manifest file (see Preparing for

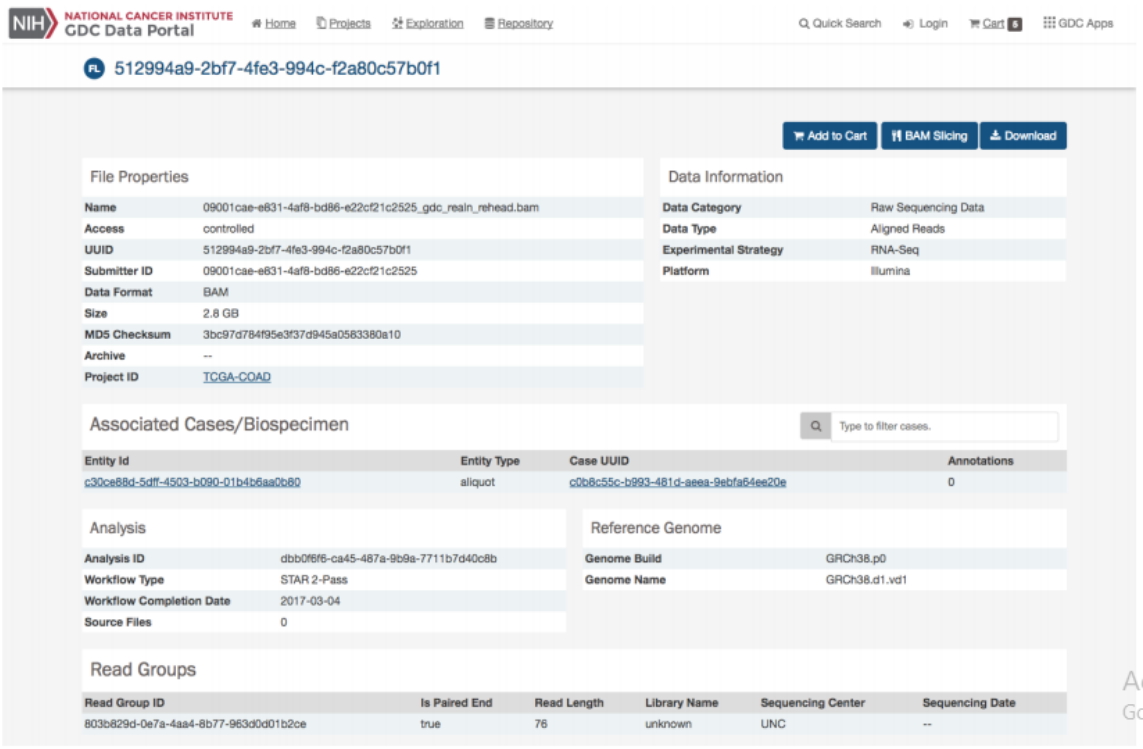


Figure 4.3: GDC Data Portal: Detailed File Page

Data Download and Upload for instructions), initiate the download using the GDC Data Transfer Tool by supplying the -m or –manifest option, followed by the location and name of the manifest file. OS X users can drag and drop the manifest file into Terminal to provide its location. The following is an example of a command for downloading files from GDC using a manifest file: gdc-client download –m gdc-manifest.txt

Downloading Data Using GDC File UUIDs

The GDC Data Transfer Tool also supports downloading of one or more individual files using UUID(s) instead of a manifest file. To do this, enter the UUID(s) after the download command: gdc-client download 22a29915-6712-4f7a-8dba-985ae9a1f005 Multiple UUIDs can be specified, separated by a space: gdc-client download e5976406-473a-4fbd-8c97-e95187cdc1bd fb3e261b-92ac-4027-b4d9-eb971a92a4c3

4.3 Numpy 1.14.3

NumPy is a Python package which stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n-dimensional array object, provide tools for integrating C, C++ etc. It is a general-purpose array-processing package designed to efficiently manipulate large multi-dimensional arrays of arbitrary records without sacrificing too much speed for small multi-dimensional arrays. NumPy is built on the Numeric code base and adds features introduced by numarray as well as an extended C-API and the ability to create arrays of arbitrary type which also makes NumPy suitable for interfacing with general-purpose data-base applications.

4.4 Matplotlib 2.1.2

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It produces publication-quality figures in a variety of hard-copy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, web application servers, and various graphical user interface toolkits.

4.5 Sklearn

Scikit-learn (formerly scikits.learn and also known as sklearn) is a software machine learning library for the Python programming language.It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

4.6 Scipy 1.1.0

SciPy is open-source software for mathematics, science, and engineering. The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. The SciPy library is built to work with NumPy arrays, and provides many user-friendly and efficient numerical routines such as routines for numerical integration and optimization. Together, they run on all popular operating systems, are quick to install, and are free of charge. NumPy and SciPy are easy to use, but powerful enough to be depended upon by some of the world's leading scientists and engineers.

4.7 Openslide-python 1.1.1

OpenSlide Python is a Python interface to the OpenSlide library. It is a C library that provides a simple interface for reading whole-slide images, also known as virtual slides, which are high-resolution images used in digital pathology. These images can occupy tens of gigabytes when uncompressed, and so cannot be easily read using standard tools or libraries, which are designed for images that can be comfortably uncompressed into RAM. Whole-slide images are typically multi-resolution; OpenSlide allows reading a small amount of image data at the resolution closest to a desired zoom level.
OpenSlide can read virtual slides in several formats:

- Aperio (.svs, .tif)
- Hamamatsu (.ndpi, .vms, .vmu)
- Leica (.scn)
- MIRAX (.mrxs)
- Philips (.tiff)

- Sakura (.svslide)
- Trestle (.tif)
- Ventana (.bif, .tif)
- Generic tiled TIFF (.tif)

4.8 Pillow 5.1.0

Pillow is the friendly PIL fork. PIL is the Python Imaging Library.The Python Imaging Library adds image processing capabilities to your Python interpreter.
This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

4.9 Pytorch 1.4.0

PyTorch is an optimized tensor library for deep learning using GPUs and CPUs.PyTorch is a Python package that provides two high-level features like Tensor computation (like NumPy) with strong GPU acceleration and Deep neural networks built on a tape-based autograd system.

4.10 Flask

Flask is a lightweight Web Server Gateway Interface(WSGI) web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug(a WSGI web application library) and Jinja(a web template engine for python) and has become one of the most popular Python web application frameworks. Flask has no database

abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more frequently than the core Flask program.

4.11 FPDF

PyFPDF is a library for PDF document generation under Python, ported from PHP (see FPDF: "Free"-PDF, a well-known PDFlib-extension replacement with many examples, scripts and derivatives).

4.12 Network Drawing - NN-SVG

NN-SVG is a tool for creating Neural Network (NN) architecture drawings parametrically rather than manually. It also provides the ability to export those drawings to Scalable Vector Graphics (SVG) files.

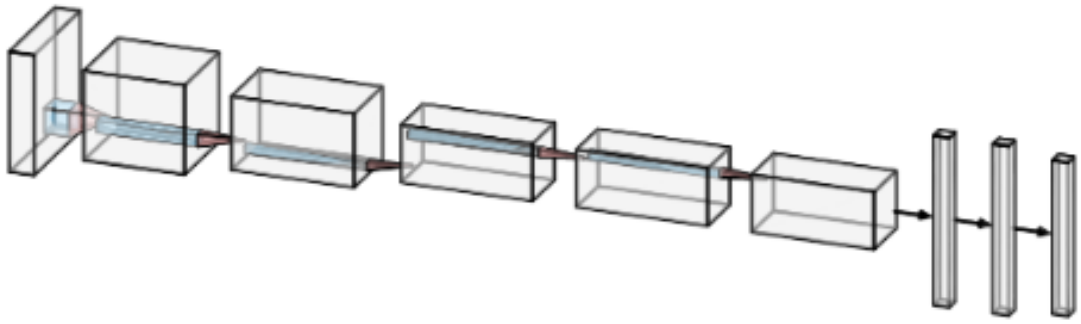


Figure 4.4: Example of Neural Network Architecture Schematics

The tool provides the ability to generate figures of three kinds: classic Fully-Connected Neural Network (FCNN) figures, Convolutional Neural Network (CNN) figures of the sort introduced in the LeNet paper, and Deep Neural Network figures following the style introduced in the AlexNet paper. The former two are accomplished using the D3 javascript library and the latter with the javascript library Three.js. NN-SVG[14] provides the ability to style the figure to the user's liking via many size, color, and layout parameters.

4.13 Docker

Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. As an open source containerization platform, Docker enables developers to package applications into containers—standardized executable components that combine application source code with all the operating system (OS) libraries and dependencies required to run the code in any environment.

Chapter 5

System Design

5.1 Inception V3

Inception-v3 is a convolutional neural network that is 48 layers deep. Inception V3 by Google is the 3rd version in a series of Deep Learning Convolutional Architectures. Inception V3 was trained using a dataset of 1,000 classes from the original ImageNet dataset which was trained with over 1 million training images, the Tensorflow version has 1,001 classes which is due to an additional 'background' class not used in the original ImageNet. Inception V3 was trained for the ImageNet Large Visual Recognition Challenge.

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 149, 149]	864
BatchNorm2d-2	[-1, 32, 149, 149]	64
BasicConv2d-3	[-1, 32, 149, 149]	0
Conv2d-4	[-1, 32, 147, 147]	9,216
BatchNorm2d-5	[-1, 32, 147, 147]	64
BasicConv2d-6	[-1, 32, 147, 147]	0
Conv2d-7	[-1, 64, 147, 147]	18,432
BatchNorm2d-8	[-1, 64, 147, 147]	128
BasicConv2d-9	[-1, 64, 147, 147]	0
Conv2d-10	[-1, 80, 73, 73]	5,120

[illegible]	[illegible]	[illegible]
BatchNorm2d-11	[-1, 80, 73, 73]	160
BasicConv2d-12	[-1, 80, 73, 73]	0
Conv2d-13	[-1, 192, 71, 71]	138,240
BatchNorm2d-14	[-1, 192, 71, 71]	384
BasicConv2d-15	[-1, 192, 71, 71]	0
Conv2d-16	[-1, 64, 35, 35]	12,288
BatchNorm2d-17	[-1, 64, 35, 35]	128
BasicConv2d-18	[-1, 64, 35, 35]	0
Conv2d-19	[-1, 48, 35, 35]	9,216
BatchNorm2d-20	[-1, 48, 35, 35]	96
BasicConv2d-21	[-1, 48, 35, 35]	0
Conv2d-22	[-1, 64, 35, 35]	76,800
BatchNorm2d-23	[-1, 64, 35, 35]	128
BasicConv2d-24	[-1, 64, 35, 35]	0
Conv2d-25	[-1, 64, 35, 35]	12,288
BatchNorm2d-26	[-1, 64, 35, 35]	128
BasicConv2d-27	[-1, 64, 35, 35]	0
Conv2d-28	[-1, 96, 35, 35]	55,296
BatchNorm2d-29	[-1, 96, 35, 35]	192
BasicConv2d-30	[-1, 96, 35, 35]	0
Conv2d-31	[-1, 96, 35, 35]	82,944
BatchNorm2d-32	[-1, 96, 35, 35]	192
BasicConv2d-33	[-1, 96, 35, 35]	0
Conv2d-34	[-1, 32, 35, 35]	6,144
BatchNorm2d-35	[-1, 32, 35, 35]	64
BasicConv2d-36	[-1, 32, 35, 35]	0
InceptionA-37	[-1, 256, 35, 35]	0
Conv2d-38	[-1, 64, 35, 35]	16,384
BatchNorm2d-39	[-1, 64, 35, 35]	128
BasicConv2d-40	[-1, 64, 35, 35]	0
Conv2d-41	[-1, 48, 35, 35]	12,288
BatchNorm2d-42	[-1, 48, 35, 35]	96
BasicConv2d-43	[-1, 48, 35, 35]	0
Conv2d-44	[-1, 64, 35, 35]	76,800
BatchNorm2d-45	[-1, 64, 35, 35]	128
BasicConv2d-46	[-1, 64, 35, 35]	0
Conv2d-47	[-1, 64, 35, 35]	16,384
BatchNorm2d-48	[-1, 64, 35, 35]	128
BasicConv2d-49	[-1, 64, 35, 35]	0
Conv2d-50	[-1, 96, 35, 35]	55,296
BatchNorm2d-51	[-1, 96, 35, 35]	192
BasicConv2d-52	[-1, 96, 35, 35]	0
Conv2d-53	[-1, 96, 35, 35]	82,944
BatchNorm2d-54	[-1, 96, 35, 35]	192
BasicConv2d-55	[-1, 96, 35, 35]	0
Conv2d-56	[-1, 64, 35, 35]	16,384
BatchNorm2d-57	[-1, 64, 35, 35]	128
BasicConv2d-58	[-1, 64, 35, 35]	0

```
      BatchNorm2d-251          [-1, 448, 8, 8]             896
      BasicConv2d-252          [-1, 448, 8, 8]               0
           Conv2d-253          [-1, 384, 8, 8]       1,548,288
      BatchNorm2d-254          [-1, 384, 8, 8]             768
      BasicConv2d-255          [-1, 384, 8, 8]               0
           Conv2d-256          [-1, 384, 8, 8]         442,368
      BatchNorm2d-257          [-1, 384, 8, 8]             768
      BasicConv2d-258          [-1, 384, 8, 8]               0
           Conv2d-259          [-1, 384, 8, 8]         442,368
      BatchNorm2d-260          [-1, 384, 8, 8]             768
      BasicConv2d-261          [-1, 384, 8, 8]               0
           Conv2d-262          [-1, 192, 8, 8]         245,760
      BatchNorm2d-263          [-1, 192, 8, 8]             384
      BasicConv2d-264          [-1, 192, 8, 8]               0
       InceptionE-265         [-1, 2048, 8, 8]               0
           Conv2d-266          [-1, 320, 8, 8]         655,360
      BatchNorm2d-267          [-1, 320, 8, 8]             640
      BasicConv2d-268          [-1, 320, 8, 8]               0
           Conv2d-269          [-1, 384, 8, 8]         786,432
      BatchNorm2d-270          [-1, 384, 8, 8]             768
      BasicConv2d-271          [-1, 384, 8, 8]               0
           Conv2d-272          [-1, 384, 8, 8]         442,368
      BatchNorm2d-273          [-1, 384, 8, 8]             768
      BasicConv2d-274          [-1, 384, 8, 8]               0
           Conv2d-275          [-1, 384, 8, 8]         442,368
      BatchNorm2d-276          [-1, 384, 8, 8]             768
      BasicConv2d-277          [-1, 384, 8, 8]               0
           Conv2d-278          [-1, 448, 8, 8]         917,504
      BatchNorm2d-279          [-1, 448, 8, 8]             896
      BasicConv2d-280          [-1, 448, 8, 8]               0
           Conv2d-281          [-1, 384, 8, 8]       1,548,288
      BatchNorm2d-282          [-1, 384, 8, 8]             768
      BasicConv2d-283          [-1, 384, 8, 8]               0
           Conv2d-284          [-1, 384, 8, 8]         442,368
      BatchNorm2d-285          [-1, 384, 8, 8]             768
      BasicConv2d-286          [-1, 384, 8, 8]               0
           Conv2d-287          [-1, 384, 8, 8]         442,368
      BatchNorm2d-288          [-1, 384, 8, 8]             768
      BasicConv2d-289          [-1, 384, 8, 8]               0
           Conv2d-290          [-1, 192, 8, 8]         393,216
      BatchNorm2d-291          [-1, 192, 8, 8]             384
      BasicConv2d-292          [-1, 192, 8, 8]               0
       InceptionE-293         [-1, 2048, 8, 8]               0
           Linear-294                [-1, 1024]       2,098,176
      BatchNorm1d-295                [-1, 1024]           2,048
             ReLU-296                [-1, 1024]               0
          Dropout-297                [-1, 1024]               0
           Linear-298                 [-1, 512]         524,800
      BatchNorm1d-299                 [-1, 512]           1,024
             ReLU-300                 [-1, 512]               0
          Dropout-301                 [-1, 512]               0
           Linear-302                   [-1, 3]           1,539
================================================================
```

```
================================================================
Total params: 24,413,155
Trainable params: 2,627,587
Non-trainable params: 21,785,568
----------------------------------------------------------------
Input size (MB): 1.02
Forward/backward pass size (MB): 224.16
Params size (MB): 93.13
Estimated Total Size (MB): 318.32
----------------------------------------------------------------
```

5.1.1 Factorizing Convolutions

The aim of factorizing Convolutions is to reduce the number of connections/parameters without decreasing the network efficiency.

Factorization Into Smaller Convolutions

Two 33 convolutions replaces one 5X5 convolution as follows:

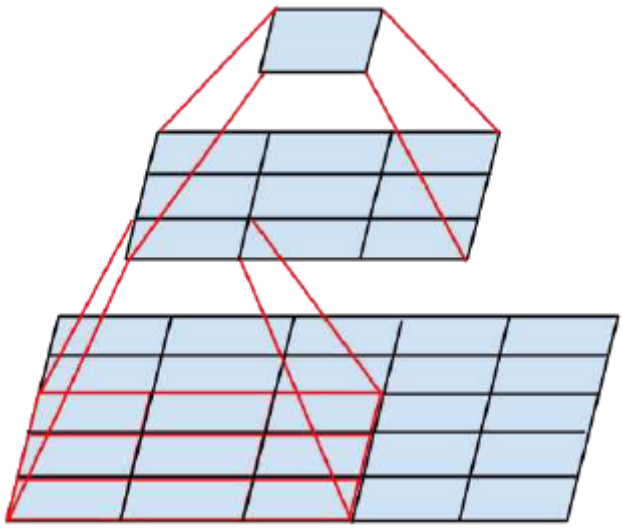


Figure 5.1: Two 3X3 convolutions replacing one 5X5 convolution

By using 1 layer of 5X5 filter, number of parameters = 5X5=25

By using 2 layers of 3X3 filters, number of parameters = 3X3+3X3=18

Number of parameters is reduced by 28%.

With this technique, one of the new Inception modules (I call it Inception Module A here) becomes:

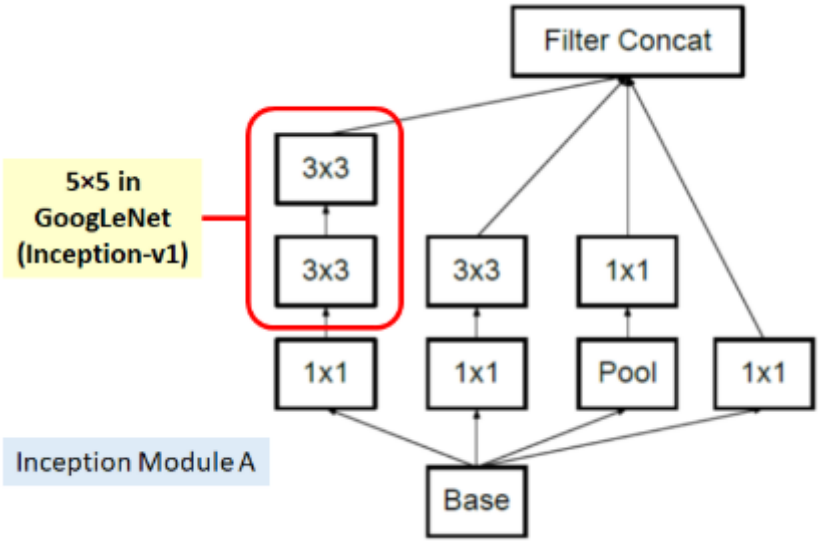


Figure 5.2: Inception Module A using factorization

Factorization Into Asymmetric Convolutions

One 3X1 convolution followed by one 1X3 convolution replaces one 3X3 convolution as follows:

By using 3x3 filter, number of parameters = 3x3=9

By using 3x1 and 1x3 filters, number of parameters = 3x1+1x3=6

Number of parameters is reduced by 33%.

With this technique, one of the new Inception modules (I call it Inception Module B here)becomes:

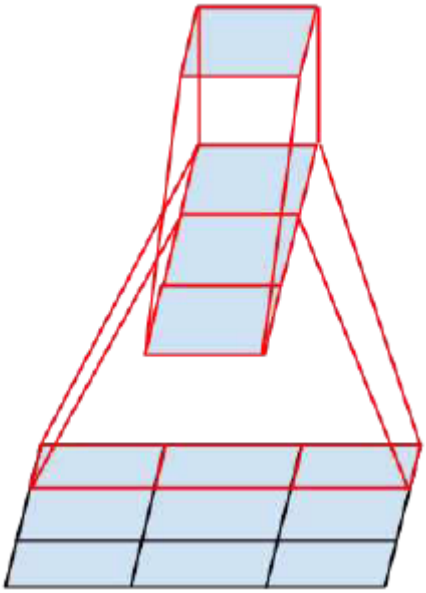


Figure 5.3: One 3X1 convolution followed by one 1X3 convolution replaces one 3X3 convolution

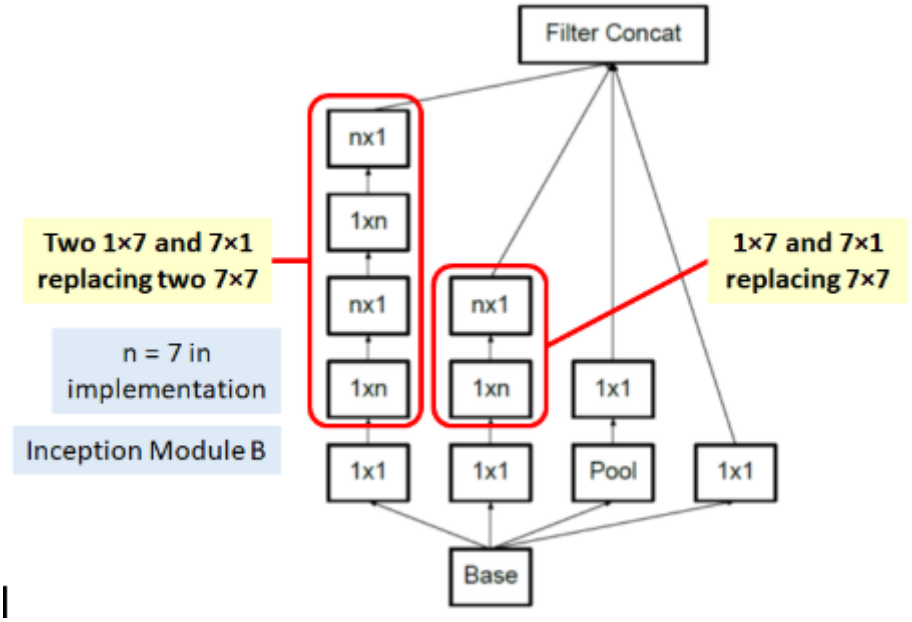


Figure 5.4: Inception Module B using asymmetric factorization

5.1.2 Auxiliary Classifier

Auxiliary Classifiers were already suggested in GoogLeNet / Inception-v1 [4]. There are some modifications in Inception-v3.Only 1 auxiliary classifier is used on the top of

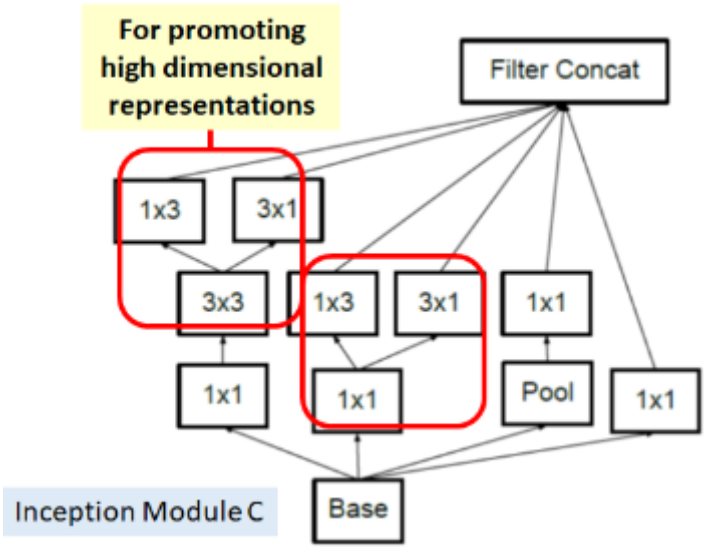


Figure 5.5: Inception Module C using asymmetric factorization

the last 17x17 layer, instead of using 2 auxiliary classifiers. (The overall architecture would be shown later.) The purpose is also different. In GoogLeNet / Inception-v1,

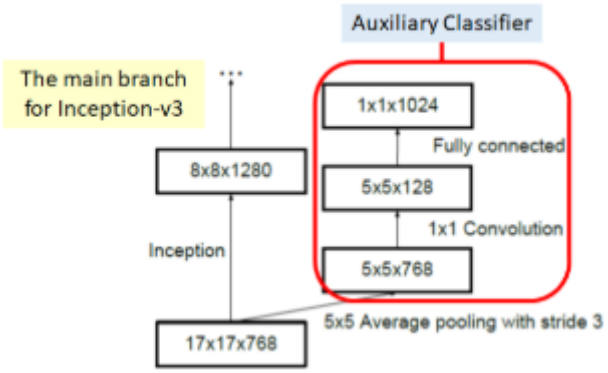


Figure 5.6: Auxiliary Classifier act as a regularization

auxiliary classifiers are used for having deeper network. In Inception-v3, auxiliary classifier is used as regularizer. So, actually, in deep learning, the modules are still quite intuitive. Batch normalization, suggested in Inception-v2, is also used in the auxiliary classifier.

5.1.3 Efficient Grid Size Reduction

Conventionally, such as AlexNet and VGGNet, the feature map downsizing is done by max pooling. But the drawback is either too greedy by max pooling followed by conv layer, or too expensive by conv layer followed by max pooling. Here, an efficient grid size reduction is proposed as follows: With the efficient grid size reduction, 320

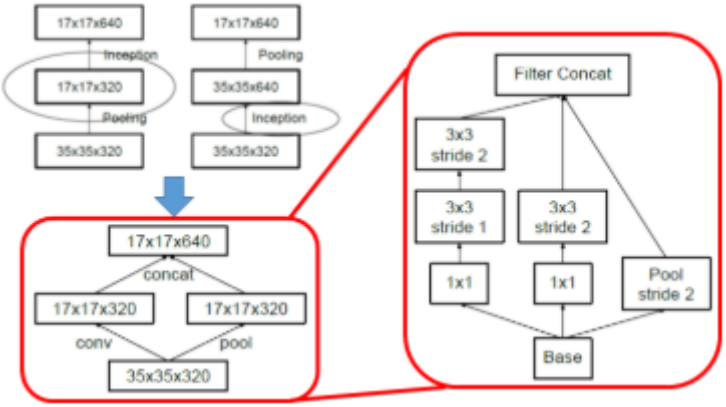


Figure 5.7: Conventional downsizing (Top Left), Efficient Grid Size Reduction (Bottom Left), Detailed Architecture of Efficient Grid Size Reduction (Right)

feature maps are done by conv with stride 2. 320 feature maps are obtained by max pooling. And these 2 sets of feature maps are concatenated as 640 feature maps and go to the next level of inception module. Less expensive and still efficient network is achieved by this efficient grid size reduction.

5.1.4 Overview of Inception-v3 Architecture

The inception v3 architecture was trained by transfer learning using our training datasets and following the procedure described. We used back-propagation, cross entropy loss, and Adam optimization method along with the hyperparameters of the transfer learning case for training. In this approach we optimized the weights of the fully connected layer. This strategy was tested on the classification task of Normal vs LUAD vs LUSC. The training jobs were run for 25 epochs. We computed the cross-entropy loss function on the train and validation dataset, and used the model with the best validation score as our final model. We did not tune the number of layers or hyper-parameters of the inception network such as size of filters. Inception-v3 consists

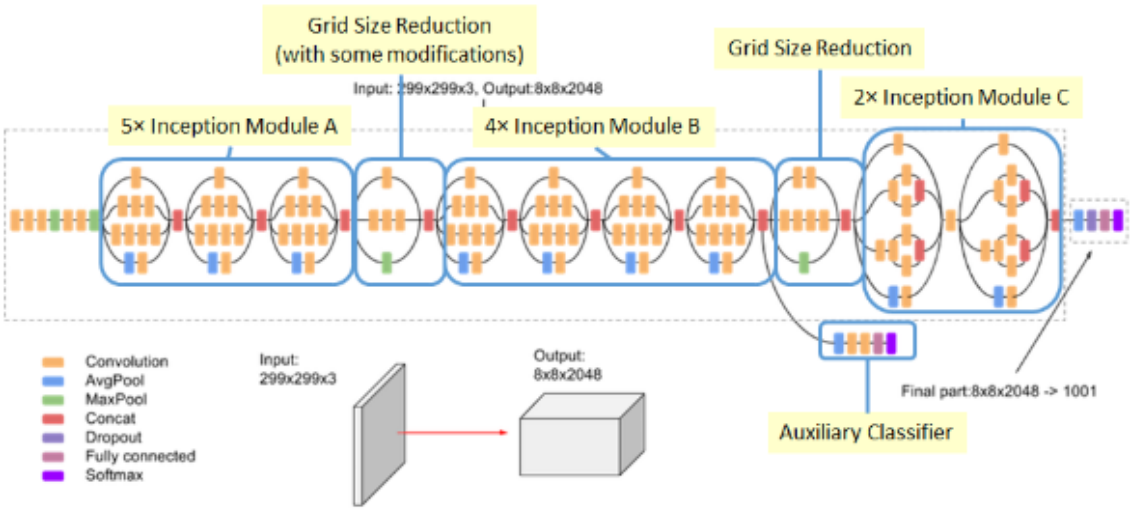


Figure 5.8: Inception-v3 Architecture (Batch Norm and ReLU are used after Conv)

of two parts:

- Feature extraction part with a convolutional neural network.
- Classification part with fully-connected and softmax layers.

The pre-trained Inception-v3 model achieves state-of-the-art accuracy for recognizing general objects with 1000 classes, like "Zebra", "Dalmatian", and "Dishwasher". The

model extracts general features from input images in the first part and classifies them based on those features in the second part. With 42 layers deep, the computation cost is only about 2.5 higher than that of GoogLeNet, and much more efficient than that of VGGNet .

5.2 Custom model architecture

Building a convolutional neural network (CNN) to classify images without relying on pre-trained models. There are a number of popular pre-trained Inception v3 models out there that are helpful for overcoming sampling deficiencies, they have already been trained on many images and can recognize a variety of features. These models typically have complex architectures that are necessary when deciphering the difference between hundreds or thousands of classes. The complexity that offers predictive capacity for a variety of objects can be a hindrance for more simplistic tasks, as the pre-trained model can overfit the data. Additionally, the architecture can be difficult for a beginner to conceptualize.

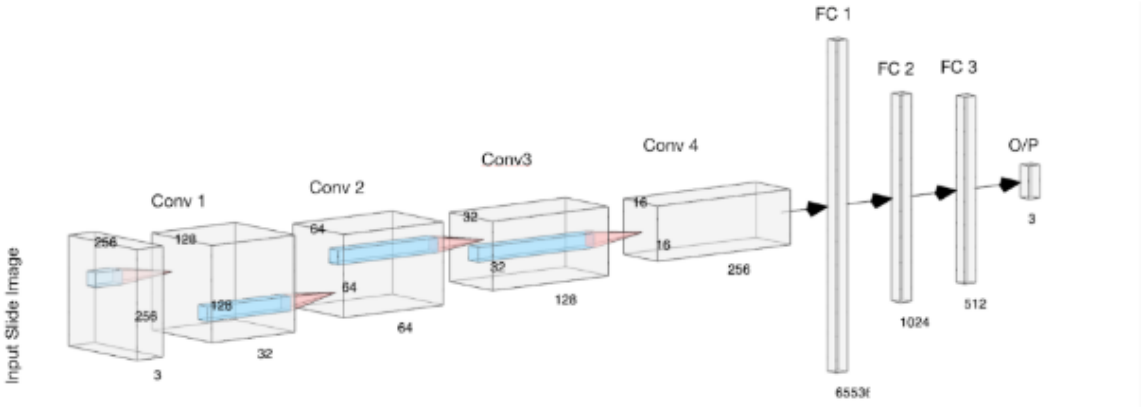


Figure 5.9: Custom designed neural network

The above fig shows the custom network that we have designed. The network takes in an already tiled whole slide image of shape (256,256,3) as input. The first convolution layer extracts multiple features of shape (128,128,32) with kernel shape

(3,3) with stride and padding (1). The Conv2 layer takes input of shape (128,128,32) and outputs (64,64,64) where number of features extracted is 64 with kernel shape (3,3) with stride and padding (1).The Conv3 layer takes input of shape (64,64,64) and outputs (32,32,128) where number of features extracted is 128 with kernel shape (3,3) with stride and padding (1).The Conv4 layer takes input of shape (32,32,128) and outputs (16,16,256) where number of features extracted is 256 with kernel shape (3,3) with stride and padding (1). After each convolution layer we normalize the image pixels using 2D Batch Norm followed by Maxpool (2,2), which normalizes the pixels and reduces the features set size by a factor of 2. Then the tensor is passed through fully connected linear layer s of shape FC1 (65536), FC2 (1024), FC3 (512), and outputs tensor (3). Each of the linear layers are followed by 1D Batch Normalization and dropout with probability of 0.7. The activation function used is Relu for the entire network to enable faster training. The output layer has softmax activation, which gives us the probability for each type of class predicted by the model.

```
Net(
  (conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv4): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bn4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=65536, out_features=1024, bias=True)
  (fbn1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc2): Linear(in_features=1024, out_features=512, bias=True)
  (fbn2): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc3): Linear(in_features=512, out_features=3, bias=True)
  (dropout): Dropout(p=0.7, inplace=False)
)
```

Convolutional layer

A convolutional layer within a neural network should have the following attributes:

- Convolutional kernels defined by a width and height (hyper-parameters).
- The number of input channels and output channels (hyper-parameter).

- The depth of the Convolution filter (the input channels) must be equal to the number channels (depth) of the input feature map.

Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus. Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10,000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters. For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters. By using regularized weights over fewer parameters, the vanishing gradient and exploding gradient problems seen during backpropagation in traditional neural networks are avoided.

Pooling

Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer. In addition, pooling may compute a max or an average. Max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Average pooling uses the average value from each of a cluster of neurons at the prior layer.

Max Pooling

Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

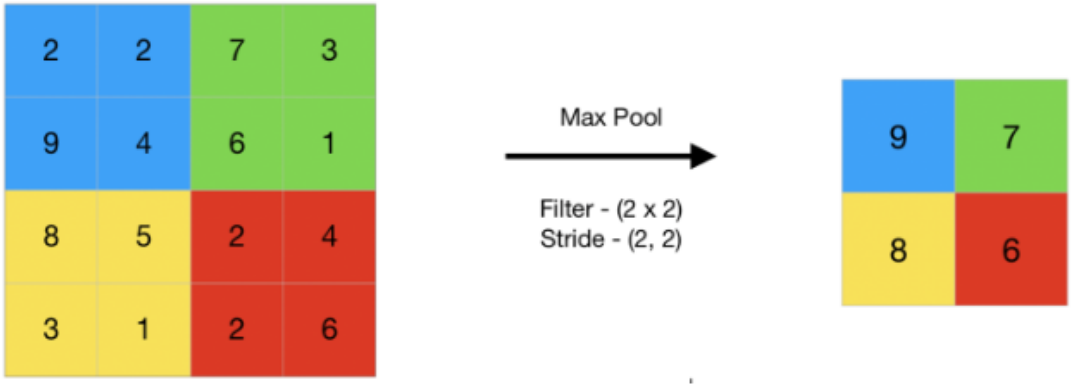


Figure 5.10: Example of Max Pooling

Average Pooling

Average pooling computes the average of the elements present in the region of feature map covered by the filter. Thus, while max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in a patch.

Fully connected

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

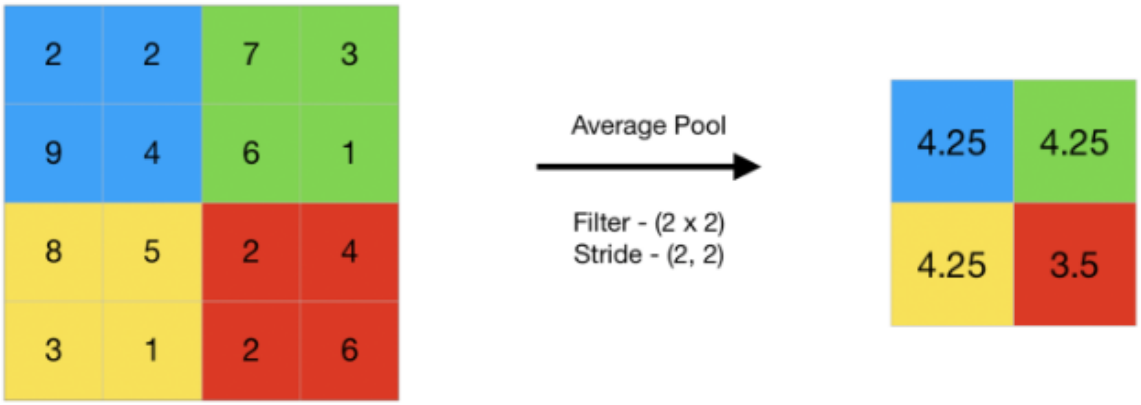


Figure 5.11: Example of Avg Pooling

Dropout

Dropout introduces regularization within the network, which ultimately improves generalization by randomly skipping some units or connections with a certain probability. In NNs, multiple connections that learn a non-linear relation are sometimes co-adapted, which causes overfitting. This random dropping of some connections or units produces several thinned network architectures, and finally, one representative network is selected with small Weights.

Softmax

The softmax activation is normally applied to the very last layer in a neural net, instead of using ReLU, sigmoid, tanh, or another activation function. The reason why softmax is useful is because it converts the output of the last layer in your neural network into what is essentially a probability distribution.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 256, 256]             896
         MaxPool2d-2         [-1, 32, 128, 128]               0
            Conv2d-3         [-1, 64, 128, 128]          18,496
       BatchNorm2d-4         [-1, 64, 128, 128]             128
         MaxPool2d-5           [-1, 64, 64, 64]               0
            Conv2d-6          [-1, 128, 64, 64]          73,856
       BatchNorm2d-7          [-1, 128, 64, 64]             256
         MaxPool2d-8          [-1, 128, 32, 32]               0
            Conv2d-9          [-1, 256, 32, 32]         295,168
      BatchNorm2d-10          [-1, 256, 32, 32]             512
        MaxPool2d-11          [-1, 256, 16, 16]               0
           Linear-12                [-1, 1024]      67,109,888
      BatchNorm1d-13                [-1, 1024]           2,048
          Dropout-14                [-1, 1024]               0
           Linear-15                 [-1, 512]         524,800
      BatchNorm1d-16                 [-1, 512]           1,024
          Dropout-17                 [-1, 512]               0
           Linear-18                   [-1, 3]           1,539
================================================================
Total params: 68,028,611
Trainable params: 68,028,611
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.75
Forward/backward pass size (MB): 51.54
Params size (MB): 259.51
Estimated Total Size (MB): 311.79
----------------------------------------------------------------
```

Figure 5.12: Model Summary: gives total number of trainable parameters

5.3 Explainable Model - Heatmap generation

Deep neural models based on Convolutional Neural Net- works (CNNs) have enabled unprecedented breakthroughs in a variety of computer vision tasks, from image classification, object detection, semantic segmenta- tion to image captioning visual question answering and more recently, visual dialog and embodied question answering. While these models enable superior performance, their lack of de- composability into individually intuitive components makes them hard to interpret. Consequently, when

today's intel- ligent systems fail, they often fail spectacularly disgracefully without warning or explanation, leaving a user staring at an incoherent output, wondering why the system did what it did. Interpretability matters. In order to build trust in intelligent systems and move towards their meaningful integration into our everyday lives, it is clear that we must build 'transparent' models that have the ability to explain why they predict what they predict.

A number of previous works have asserted that deeper repre- sentations in a CNN capture higher-level visual constructs . Furthermore, convolutional layers naturally retain spatial information which is lost in fully-connected layers, so we can expect the last convolutional layers to have the best com- promise between high-level semantics and detailed spatial information. The neurons in these layers look for semantic class-specific information in the image (say object parts). Grad-CAM uses the gradient information flowing into the last convolutional layer of the CNN to assign importance values to each neuron for a particular decision of interest. Although our technique is fairly general in that it can be used to explain activations in any layer of a deep network, in this work, we focus on explaining output layer decisions only.

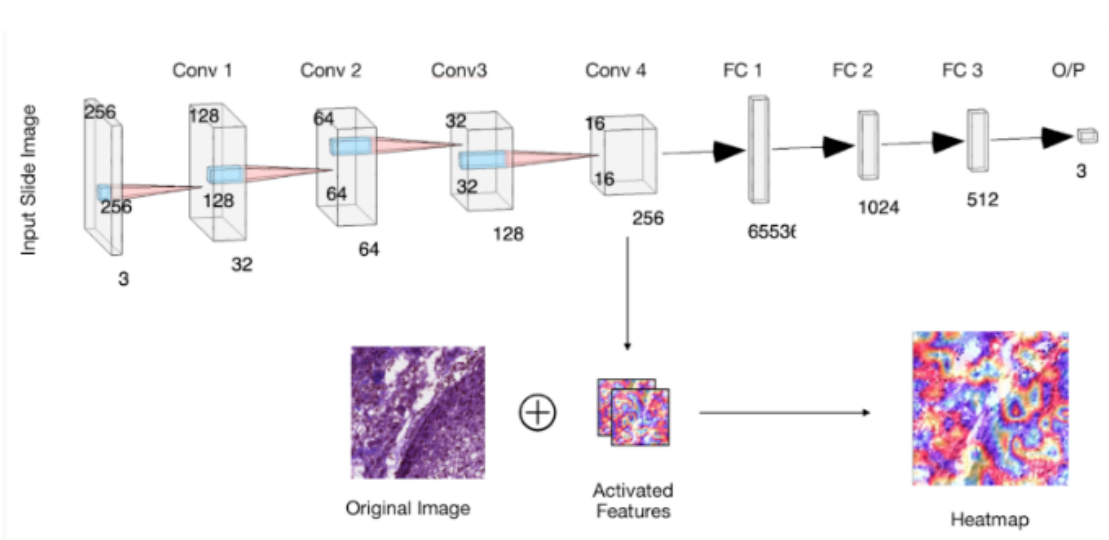


Figure 5.13: An overview of the heatmap generation using GradCAM technique. Gradient-weighted Class Activation Mapping when combined with the original image gives high class activation indicated with red on the heatmap scale.

As shown in the figure 5.12 with a trained model an whole slide image is passed in forward pass fashion and intercepted at convolution layer 4. Each of the features maps at layer 4 is stored in a variable.

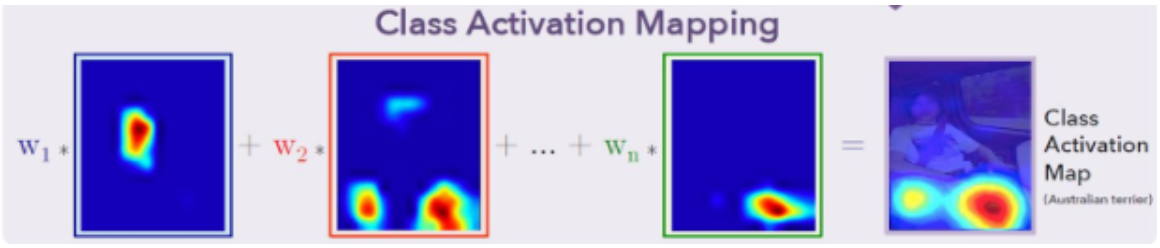


Figure 5.14: The above figure shows how GradCAM works. Each of the class activation obtained at the last convolution layer is rich in features. The product of features and weights and summation of all with superimposition gives class activation map.

Extract the value of the last layer of CNN and the weight of each channel corresponding to the target classification (w1, w2, w3...), multiply and superimpose to get the CAM of the image under this classification, the formula is:
We define *Mc* as class activation of class *c*, where each spatial element is given by,
Where wk is the weight of the GAP output from the *kth* channel of the last convo-

$$M_c(x,y) = \sum_k w_k^c f_k(x,y)$$

lution layer to the corresponding classification, and *fk* is the two-dimensional output of the *kth* channel of the last convolution layer.
An explainable model serves doctors to visually locate the regions of the whole slide images which have high malignancy.

Chapter 6

System Implementation

6.1 Dataset

The Cancer Genome Atlas (TCGA), a landmark cancer genomics program, molecularly characterized over 20,000 primary cancers and matched normal samples spanning 33 cancer types.

Sample images:

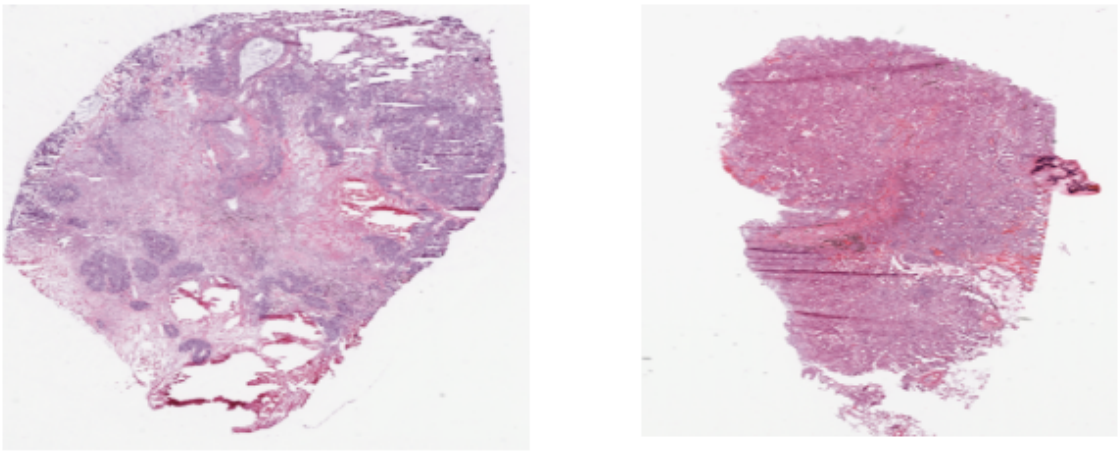


Figure 6.1: LUAD (left) , LUSC (right)

"Normal Tissue" and "Primary Tumor" slides using a set of eosin stained histopathology whole-slide images. Then, the "primary tumor" were classified between LUAD and LUSC types using a set of those whole-slide images.

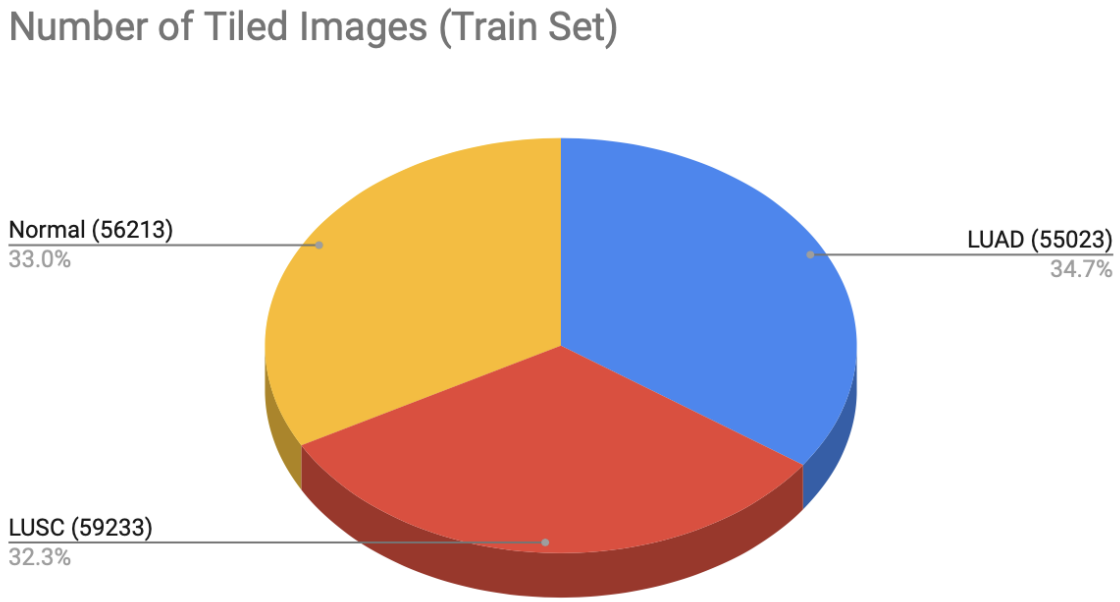


Figure 6.2: Chart showing number of tiled images(Train set)

Additional Datasets for evaluation of the trained network is procured from JSS Hospitals with the help of Dr. Ravi Krishnappan, Surgical Oncologist.

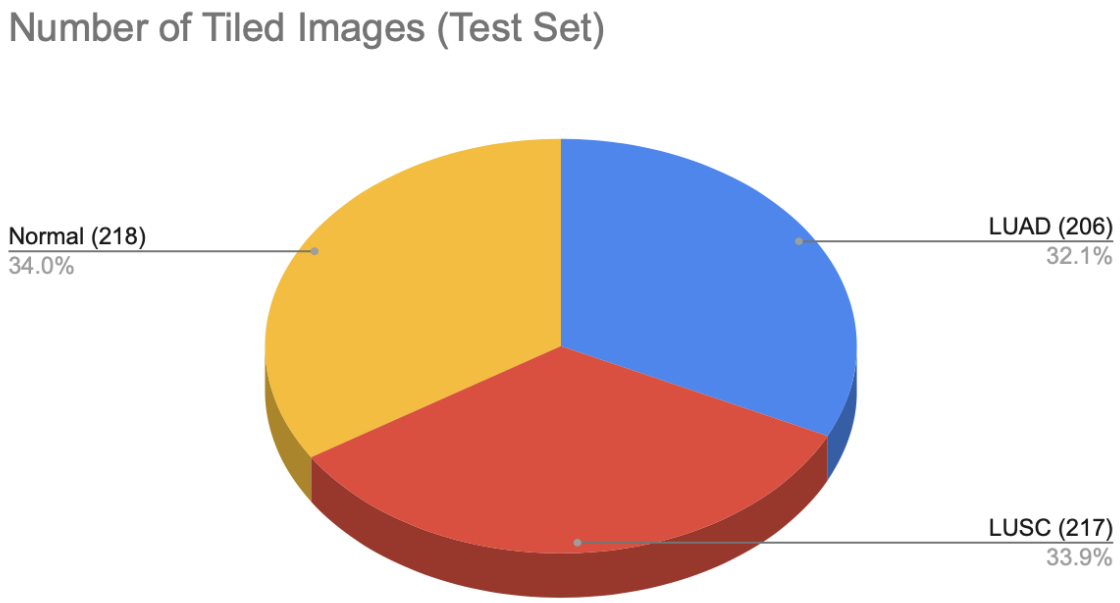


Figure 6.3: Chart showing Number of tiled images(Test set)

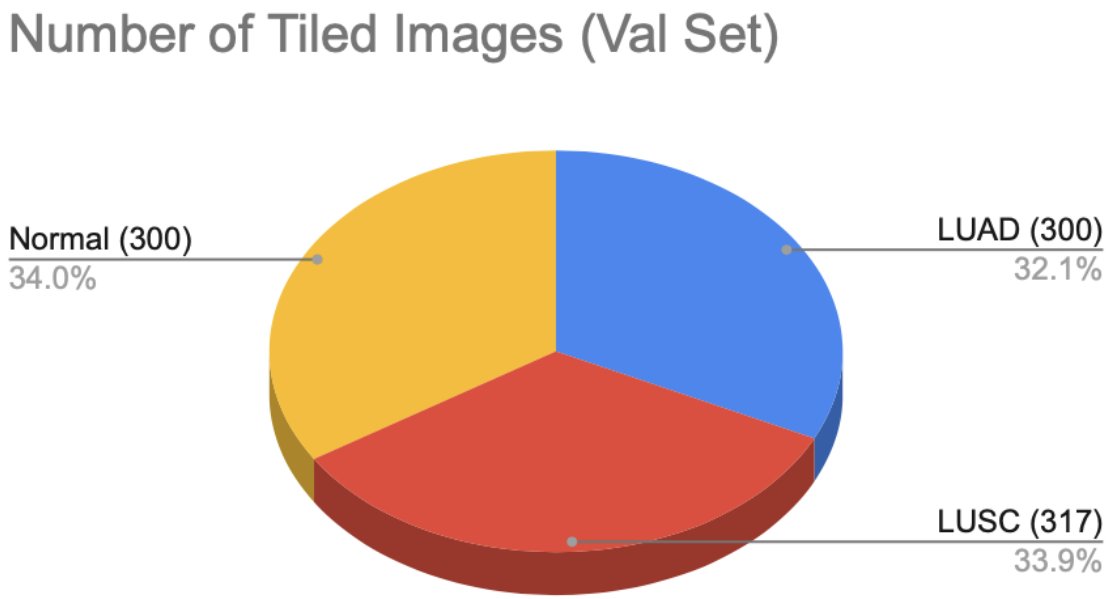


Figure 6.4: Chart showing Number of tiled images(Val set)

6.2 Data Preprocessing

6.2.1 Tiling

Tile-based processing consists of dividing an image into small, rectangular pieces called "tiles," processing each tile, and reassembling the image. The principle of locality applies, so an image-processing operation usually requires input tiles of approximately the same area as the output tile.

Since the InceptionV3 Model's input feature is 299x299, large svs images from the TCGA dataset cannot be fed to the network.So, we use OpenSlide library to divide the .svs image to multiple tiles of shape (256X256),the tiles are non - overlapping and background less than 20%. The number of tiles generated for 60 whole slide images

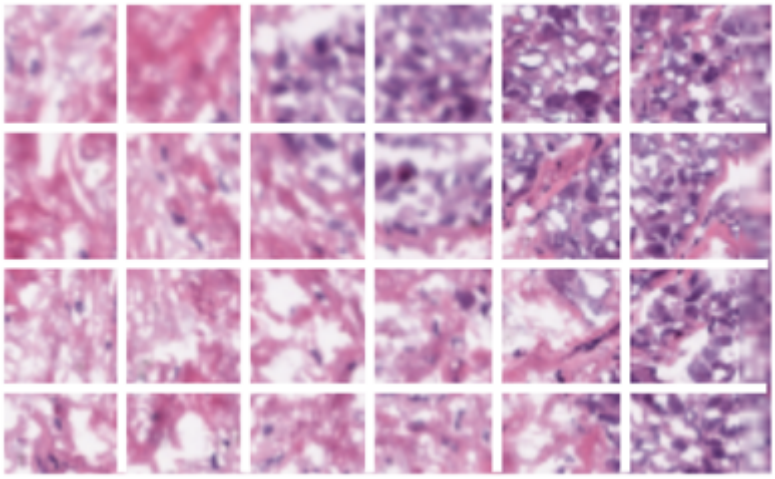


Figure 6.5: Tiled image into 256X256

are 1,69,000.

Command Line to count tiles: ls -lR /path/to/dir/*.jpg — wc –l

6.2.2 Convert SVS to JPG Format

SVS is an image format associated with Aperio medical equipment, such as microscopes or scanscopes. Though similar with the TIFF format in terms of features, SVS

has a lot less compatibility with regular image viewers and tends to have a very large size. SVS files contain multiple images in a predefined order, including a baseline tiled image, a thumbnail, intermediate "pyramid" images (made up of tiles), an optional slide label image etc.

JPG is one of the most popular image file formats currently in use, often referred to as a standard for uploading images online and for displaying photographs. One of its main advantages is providing a good image quality in a relatively small file size, which is easy to store and transfer. JPG uses a type of compression that prioritizes the quality of some image sections over others, thus assuring that most favorable quality/size ratio.

JPG image format is easy to compress just before feeding the tiles to the network. This format although is a lossy compression technique, the compressed images are sharper as only resolution is changed.

6.2.3 Sort each generated tiles into folders named after the class label which is used for training

The dataset downloaded from the TCGA are not annotated rather consists of meta-data. In order to train the network each type of data manual sorting of images into respective class labeled folders using terminal command are done.

Example:

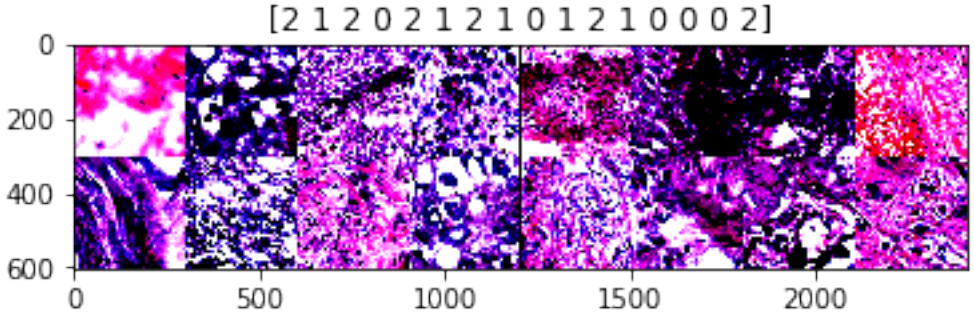
All tiled images of Squamous Cell Carcinoma are moved to the LUSC folder and the same is followed for LUAD. Each image is identified using an UUID. Each tile is mapped to one image with a label using protobuf.

Train	Valid	Test
LUSC LUAD NORMAL	LUSC LUAD NORMAL	LUSC LUAD NORMAL

6.2.4 Normalization

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges. Data normalization is an important step which ensures that each input parameter (pixel, in this case) has a similar data distribution. This makes convergence faster while training the network. Data normalization is done by subtracting the mean from each pixel and then dividing the result by the standard deviation. The distribution of such data would resemble a Gaussian curve centered at zero. For image inputs we need the pixel numbers to be positive, so we might choose to scale the normalized data in the range [0,1] or [0, 255]. For our data-set example, the following montage represents the normalized data.

There are several approaches in normalisation which can be used in deep learning models. They are mentioned below



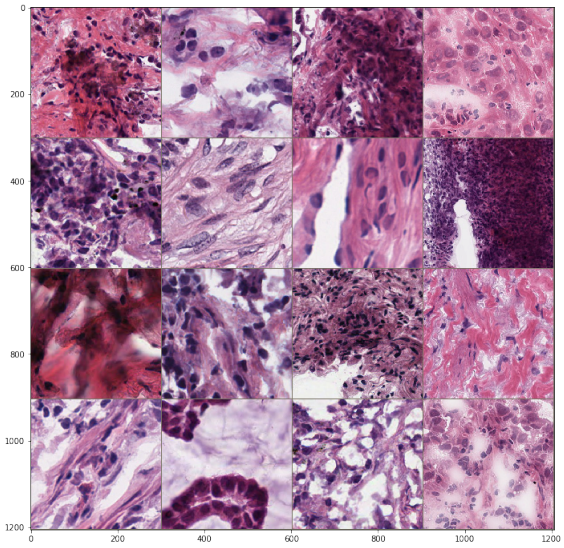


Figure 6.6: Original Image with batch size 16

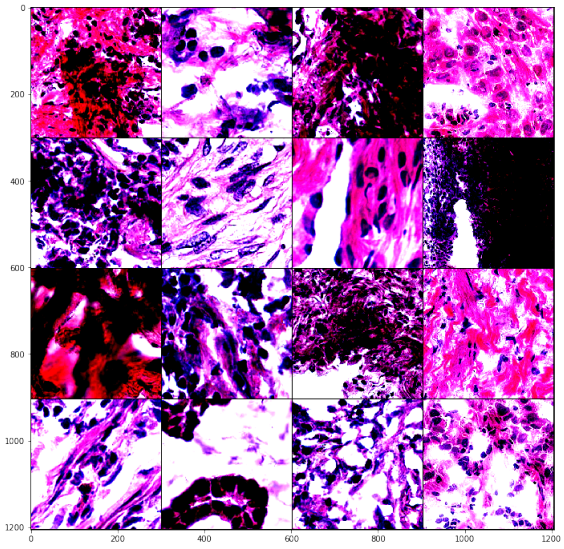


Figure 6.7: Normalized Image with batch size 16

Batch Normalization

Batch normalization is one of the popular normalization methods used for training deep learning models. It enables faster and stable training of deep neural networks by stabilising the distributions of layer inputs during the training phase. This approach is mainly related to internal covariate shift (ICS) where internal covariate shift means the change in the distribution of layer inputs caused when the preceding layers are

updated. In order to improve the training in a model, it is important to reduce the internal co-variant shift. The batch normalization works here to reduce the internal covariate shift by adding network layers which control the means and variances of the layer inputs.

Advantages

The advantages of batch normalization are mentioned below:

- Batch normalization reduces the internal covariate shift (ICS) and accelerates the training of a deep neural network.
- This approach reduces the dependence of gradients on the scale of the parameters or of their initial values which result in higher learning rates without the risk of divergence.
- Batch Normalisation makes it possible to use saturating nonlinearities by preventing the network from getting stuck in the saturated modes.

Weight Normalization

Weight normalization is a process of reparameterization of the weight vectors in a deep neural network which works by decoupling the length of those weight vectors from their direction. In simple terms, we can define weight normalization as a method for improving the optimisability of the weights of a neural network model.

Advantages

The advantages of weight normalization are mentioned below:

- Weight normalization improves the conditioning of the optimisation problem as well as speed up the convergence of stochastic gradient descent.
- It can be applied successfully to recurrent models such as LSTMs as well as in deep reinforcement learning or generative models.

Layer Normalization

Layer normalization is a method to improve the training speed for various neural network models. Unlike batch normalization, this method directly estimates the normalisation statistics from the summed inputs to the neurons within a hidden layer. Layer normalization is basically designed to overcome the drawbacks of batch normalization such as dependent on mini batches, etc.

Advantages

The advantages of layer normalization are mentioned below:

- Layer normalization can be easily applied to recurrent neural networks by computing the normalization statistics separately at each time step.
- This approach is effective at stabilising the hidden state dynamics in recurrent networks

Group Normalization

Group normalization can be said as an alternative to batch normalization. This approach works by dividing the channels into groups and computes within each group the mean and variance for normalization i.e. normalising the features within each group. Unlike batch normalization, group normalization is independent of batch sizes, and also its accuracy is stable in a wide range of batch sizes.

Advantages

The advantages of group normalization are mentioned below:

- It has the ability to replace batch normalization in a number of deep learning tasks.
- It can be easily implemented in modern libraries with just a few lines of codes.

Instance Normalization

Instance normalization, also known as contrast normalization is almost similar to layer normalization. Unlike batch normalization, instance normalization is applied to a whole batch of images instead for a single one.

Advantages

The advantages of instance normalization are mentioned below:

- This normalization simplifies the learning process of a model.
- The instance normalization can be applied at test time.

Why do we need Normalization?

Normalization has always been an active area of research in deep learning. Normalization techniques can decrease your model's training time by a huge factor. Let me state some of the benefits of using Normalization.

- It normalizes each feature so that they maintains the contribution of every feature, as some feature has higher numerical value than others. This way our network can be unbiased(to higher value features).
- It reduces Internal Covariate Shift. It is the change in the distribution of network activations due to the change in network parameters during training. To improve the training, we seek to reduce the internal covariate shift.
- Batch Norm makes loss surface smoother(i.e. it bounds the magnitude of the gradients much more tightly).
- It makes the Optimization faster because normalization doesn't allow weights to explode all over the place and restricts them to a certain range.
- An unintended benefit of Normalization is that it helps network in Regularization(only slightly, not significantly).

6.2.5 Augmentation

Another common pre-processing technique involves augmenting the existing data-set with perturbed versions of the existing images. Scaling, rotations and other affine transformations are typical. This is done to expose the neural network to a wide variety of variations. This makes it less likely that the neural network recognizes unwanted characteristics in the data-set.

Transforms on PIL Image

1. RandomChoice(transforms)

Apply single transformation randomly picked from a list.

2. RandomHorizontalFlip(p=0.5)

Horizontally flip the given image randomly with a given probability. The image can be a PIL Image or a torch Tensor, in which case it is expected to have [. . . , H, W] shape, where means an arbitrary number of leading dimensions.

Parameters

p (float) – probability of the image being flipped. Default value is 0.5.

3. RandomResizedCrop

Crop the given PIL Image to random size and aspect ratio.A crop of random size (default: of 0.08 to 1.0) of the original size and a random aspect ratio (default: of 3/4 to 4/3) of the original aspect ratio is made. This crop is finally resized to given size. This is popularly used to train the Inception networks.

For ex:

torchvision.transforms.RandomResizedCrop(size, scale=(0.08, 1.0), ratio=(0.75, 1.33), interpolation=2)

Parameters:

size – expected output size of each edge

scale – range of size of the origin size cropped

ratio – range of aspect ratio of the origin aspect ratio cropped

interpolation – Default: PIL.Image.BILINEAR

4. Resize(size, interpolation=2)

Resize the input PIL Image to the given size.

For ex:

CLASS torchvision.transforms.Resize(size, interpolation=2)

Parameters:

size (sequence or int) – Desired output size. If size is a sequence like (h, w), output size will be matched to this. If size is an int, smaller edge of the image will be matched to this number. i.e, if height ¿ width, then image will be rescaled to (size * height / width, size) interpolation (int, optional) – Desired interpolation. Default is PIL.Image.BILINEAR

Conversion Transforms

1. ToTensor()

Convert a PIL Image or numpy.ndarray to tensor.Converts a PIL Image or numpy.ndarray (H x W x C) in the range [0, 255] to a torch.FloatTensor of shape (C x H x W) in the range [0.0, 1.0] if the PIL Image belongs to one of the modes (L, LA, P, I, F, RGB, YCbCr, RGBA, CMYK, 1) or if the numpy.ndarray has dtype = np.uint8
In the other cases, tensors are returned without scaling.
__call__(pic)

Parameters

pic (PIL Image or numpy.ndarray) – Image to be converted to tensor. Returns - Converted image.

Return type - Tensor

https://pytorch.org/docs/stable/torchvision/transforms.html

6.2.6 Deploying the model using Flask and Docker

A Flask application is an instance of the Flask class. Everything about the application, such as configuration and URLs, will be registered with this class.

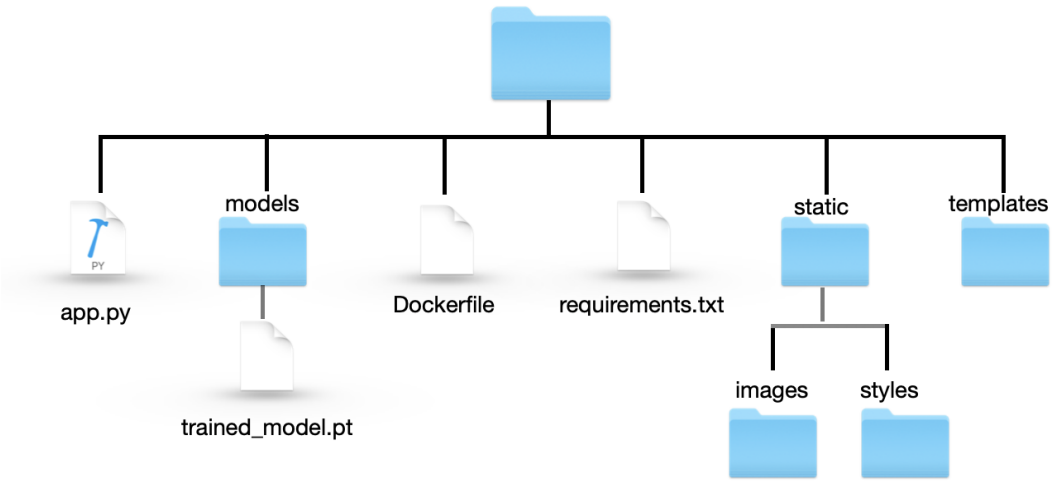


Figure 6.8: Folder structure of the app

Flask Request

When the Flask application handles a request, it creates a Request object based on the environment it received from the WSGI server. Because a worker (thread, process, or coroutine depending on the server) handles only one request at a time, the request data can be considered global to that worker during that request. Flask uses the term

context local for this. The Flask.wsgi_app() method is called to handle each request. It manages the contexts during the request. Internally, the request and application contexts work as stacks, _request_ctx_stack and _app_ctx_stack. When contexts are pushed onto the stack, the proxies that depend on them are available and point at information from the top context on the stack.
File uploading is the process of transmitting the binary or normal files to the server. Flask facilitates the uploading of files with ease. In the corresponding HTML form the encryption should be set to multipart/form-data.
The server-side flask script fetches the file from the request object using request.files[] Object. On successfully uploading the file, it is saved to the desired location on the server. The location can be configured using the app.config['UPLOAD_FOLDER'] setting.

Flask render_template

Flask configures the Jinja2 template engine automatically, which eases the process of rendering HTML pages.
To render a template, the render_template() method is used.The name of the template and the variables needed to be passed to the template engine are provided as keyword arguments to this method.
Inside templates there is access to the request, session and g objects as well as the get_flashed_messages() function.

Routes

Routing is a technique used in webapps to help the user remember the URLs. Routes in Flask are mapped to Python functions. The route() decorator, @app.route(), binds a URL to a function. By default, on opening the app, @app.route('/') is triggered, which renders the homepage using the render_template function.
The prediction from the model is triggered by a HTTP 'POST' method on the homepage, which triggers the @app.route('/predict') decorator, which is bound to the pre-

dict method.In predict method, the input image uploaded by the user is obtained using the request.files dictionary and it is pre-processed, fed as an input to the model wherein the predictions and heatmap are obtained. This is returned along with the render_template method which returns the results.html page.
The fully trained custom model with 98 accuracy is saved in the app/models path. By running the flask app, the decorator @app.route('/') is triggered, which is bound to a simple function which renders the homepage using the render_template function. On default, the connection is routed back to the system at 127.0.0.0 and the port mapping is configured to use port 5000. When the user uploads an image using the HTTP POST method, the decorator @app.route('/predict') is triggered. This decorator is bound to a method which performs the following functionality:

- Creates an instance of the custom defined CNN model class and loads the weights using load_state_dict and torch.load methods.
- The uploaded image is obtained from the request.files dictionary using request.files ['userinput'].
- The uploaded image is saved to the app.config['UPLOAD_FOLDER'] directory.
- The image is pre-processed using the same steps defined above and fed to the model.
- The output of the model is obtained and the explainable output image is generated using the GradCAM technique.
- The prediction of the model, the input image, the explainable output image is returned as arguments along with the results.html URL using the render_template function.

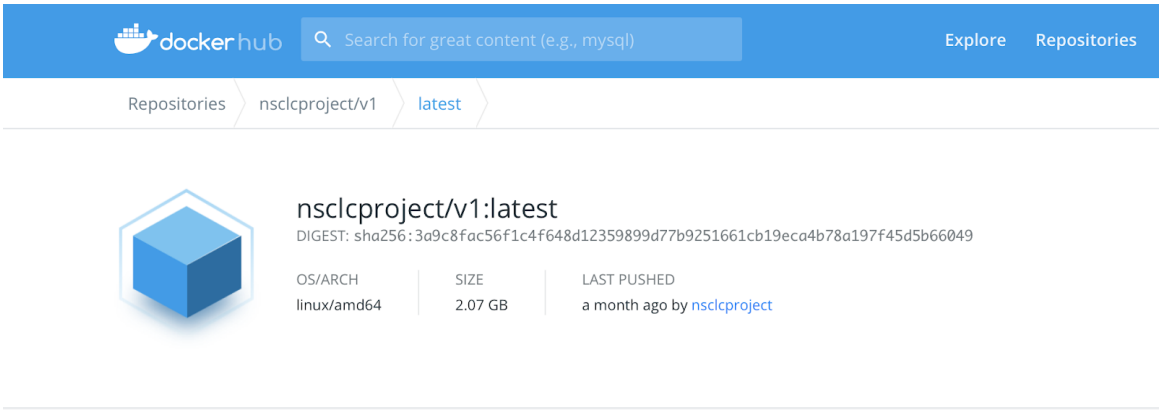


Figure 6.9: Repository hosted on docker hub

Chapter 7

System Testing and Result Analysis

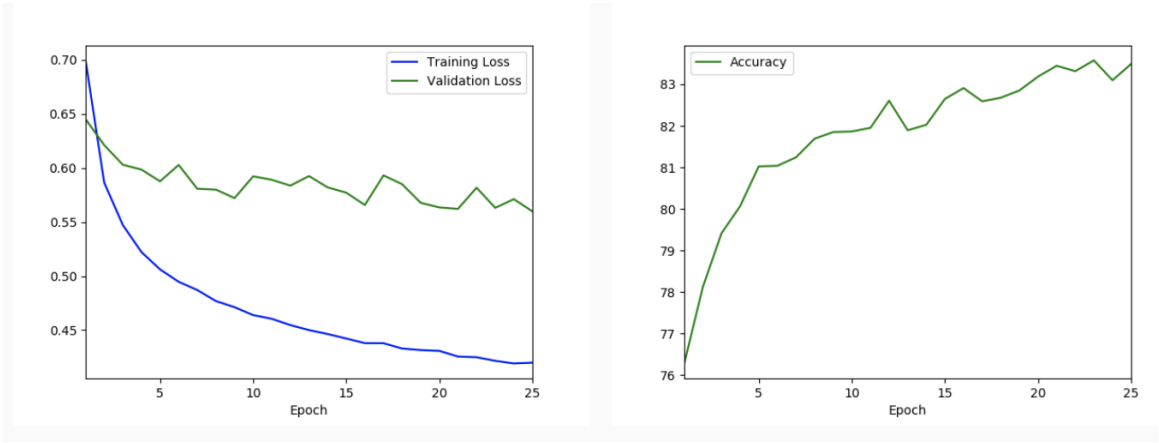


Figure 7.1: The training plot of Inception V3 model

The training plots for both the models are shown above. The Inception V3 model learns with dynamic learning rate(depends on scheduler) a gradual decrease in train loss(indicated in blue line) but the validation loss(orange line) struggles between 0.5 to 0.6 and shows no signs of decrease. The accuracy of the model keeps increasing up to 88.3% at epoch 19.

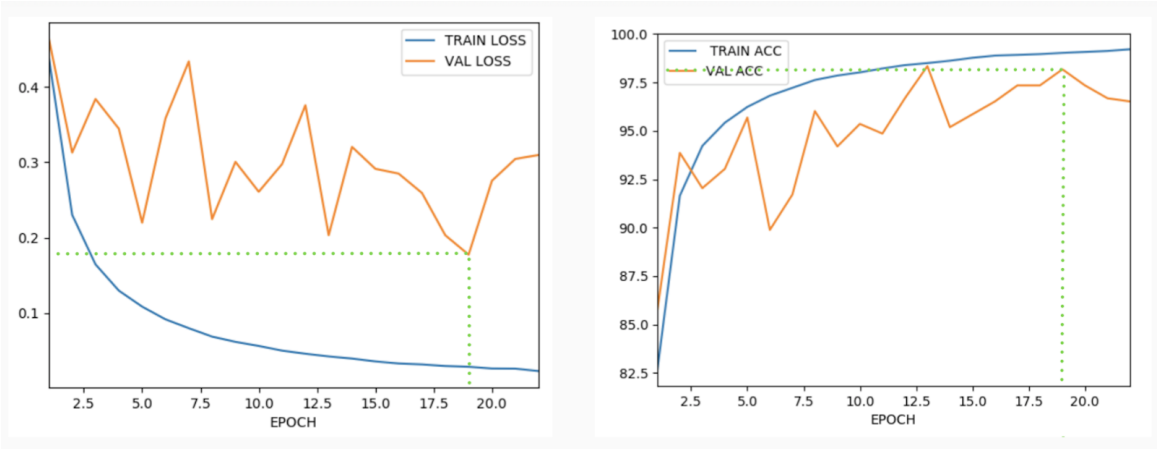


Figure 7.2: The training plot of Custom Net

The Custom Net model learns with dynamic learning rate(depends on scheduler) a gradual decrease in train loss(indicated in blue line) but the validation loss(orange line) although fluctuates has highest accuracy of 98.21% at epoch 19.

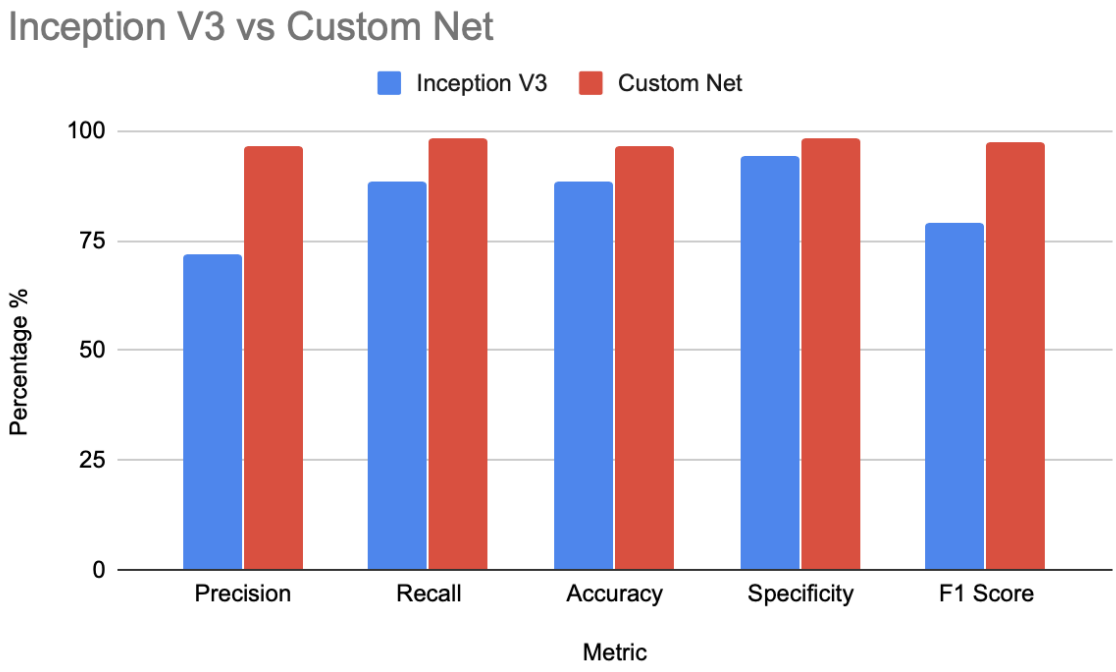


Figure 7.3: Chart showing Inception V3 vs Custom Net metric graph

Model	Precision	Recall	Accuracy	Specificity	F1 Score
Inception V3	71.88	88.3	88.3	94.1	79.24
Custom Net	96.71	98.40	96.52	98.21	97.54

Metric table

The model inferencing or testing is done on the local web server where validation set is from the same distribution as the training set and the test set is from a different distribution. They are passed to the model and checked for outputs. The metric table shows the evaluation matrix for different models on the validation set. Inception V3 has a precision 71.88%, recall 88.3%, accuracy of 88.3%, and specificity of 94.1% in contrast the designed custom net outperforms V3 architecture with precision 96.71%, recall 98.4%, accuracy of 96.52%, and specificity of 98.21%.

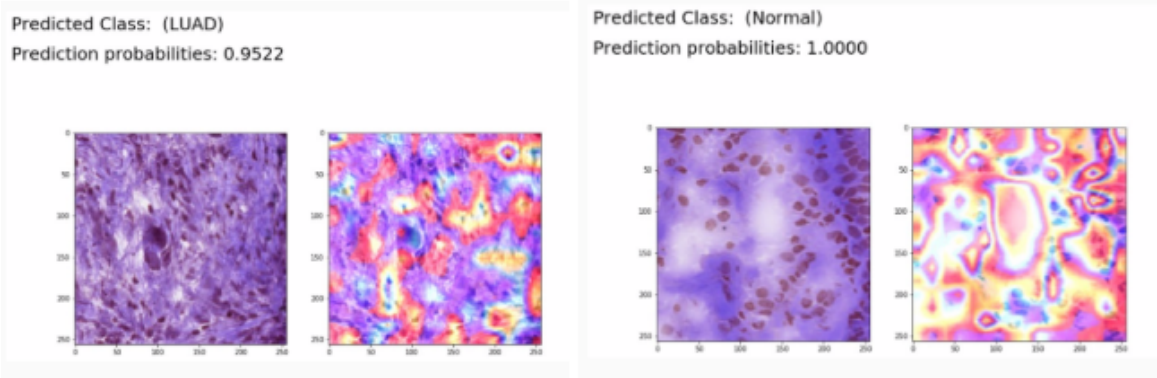


Figure 7.4: The image shown above is part of the whole slide image which consists of malignant cells. Custom net model accurately predicts the cancer type i.e LUAD with 95% probability. The red regions in the heatmap indicate the regions of high malignant and blue, purple regions show regions of least cancerous. The second image shows classification of a tiled image as Normal class(100% probability) with red regions showing why the model predicts it as a normal cell.

Having high recall and specificity from Custom Net ensures the model is able to identify the ground truth labels accurately. Referring to a medical test, specificity refers to the percentage of people who test negative for a specific disease among a group of people who do not have the disease. No test is 100% specific because some people who do not have the disease will test positive for it (false positive). An

histogram gives a performance comparison between models. From the metric table conclusive that Custom Net is our preference over V3.

For testing the model we used the data from link. The whole slides were tiled with no data augmentations. This is significant because it is necessary that our model is able to classify cancer tissues accurately based on all the augmentations implemented earlier during training. In order to localize the presence of malignancy using the GradCAM technique.

7.1 Webpage

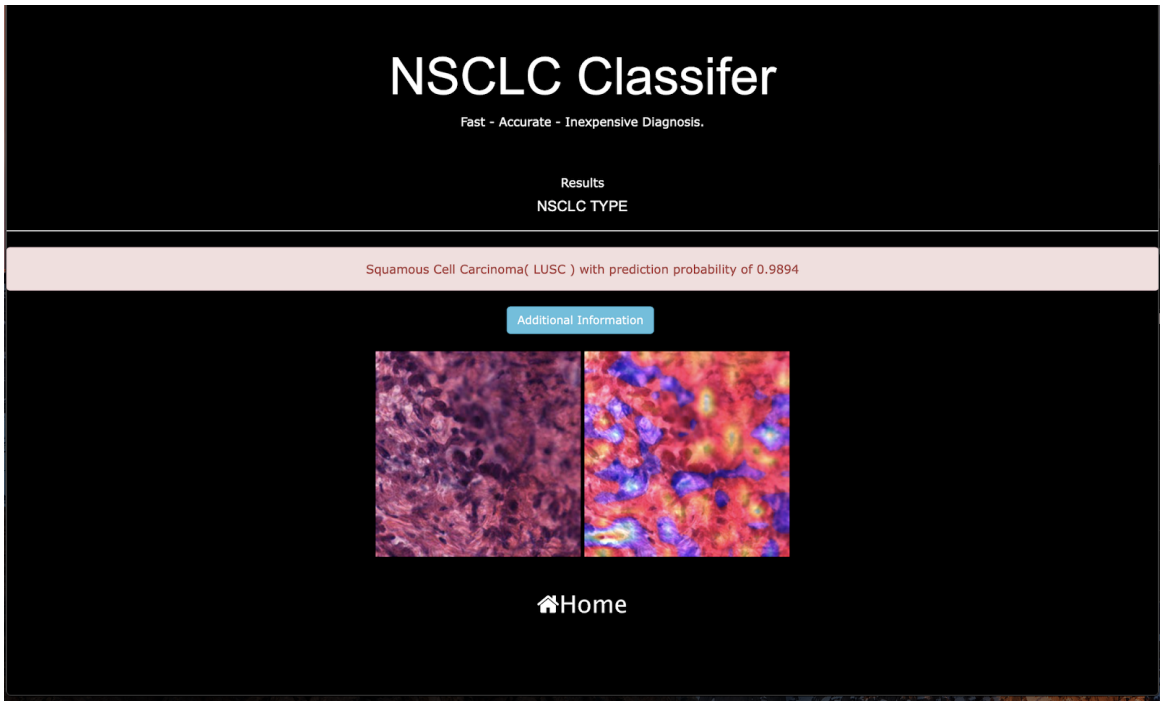


Figure 7.5: Sample result page showing the classification, prediction probability,link to additional information and explainable output

The resulting flask application is then containerized using the Platform as a Service(PaaS) platform Docker, so that the application can be run independent of the hardware and software configurations of various systems. The Docker image is hosted on docker hub, making it easy to pull from the repository and run as a container.

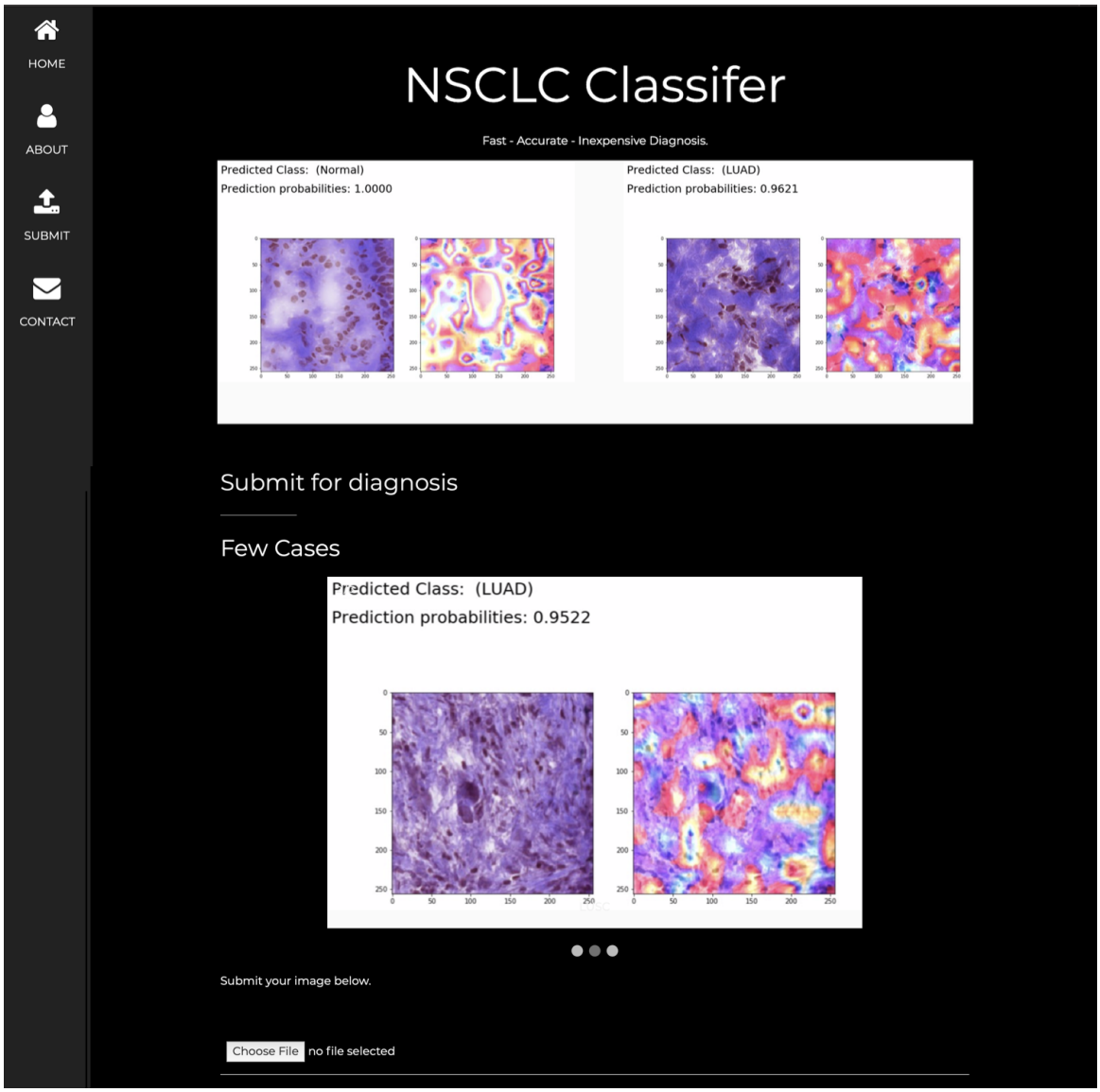


Figure 7.6: Landing page and upload button of the webpage

7.2 Patient Report Generation

The process of report generation follows the following steps:

- The uploaded svs file is tiled using the same process as above.
- Each tile is given as input to the model to obtain the type of cancer and the corresponding prediction score

- The count for each class and the corresponding sum of probabilities is stored as a vector.
- False Positive for a given tile is taken care by taking the class that has the maximum count
- The corresponding average probability is taken as the prediction probability of the model.
- Top three tiles of the resultant class with the highest probabilities are added to the report. This localizes the malignancies to a particular part of the whole slide.
- If the user uploads only a single tile, then it is processed as such and the output is presented.
- The report is available as a pdf to download.

7.2.1 Patient Report Sample

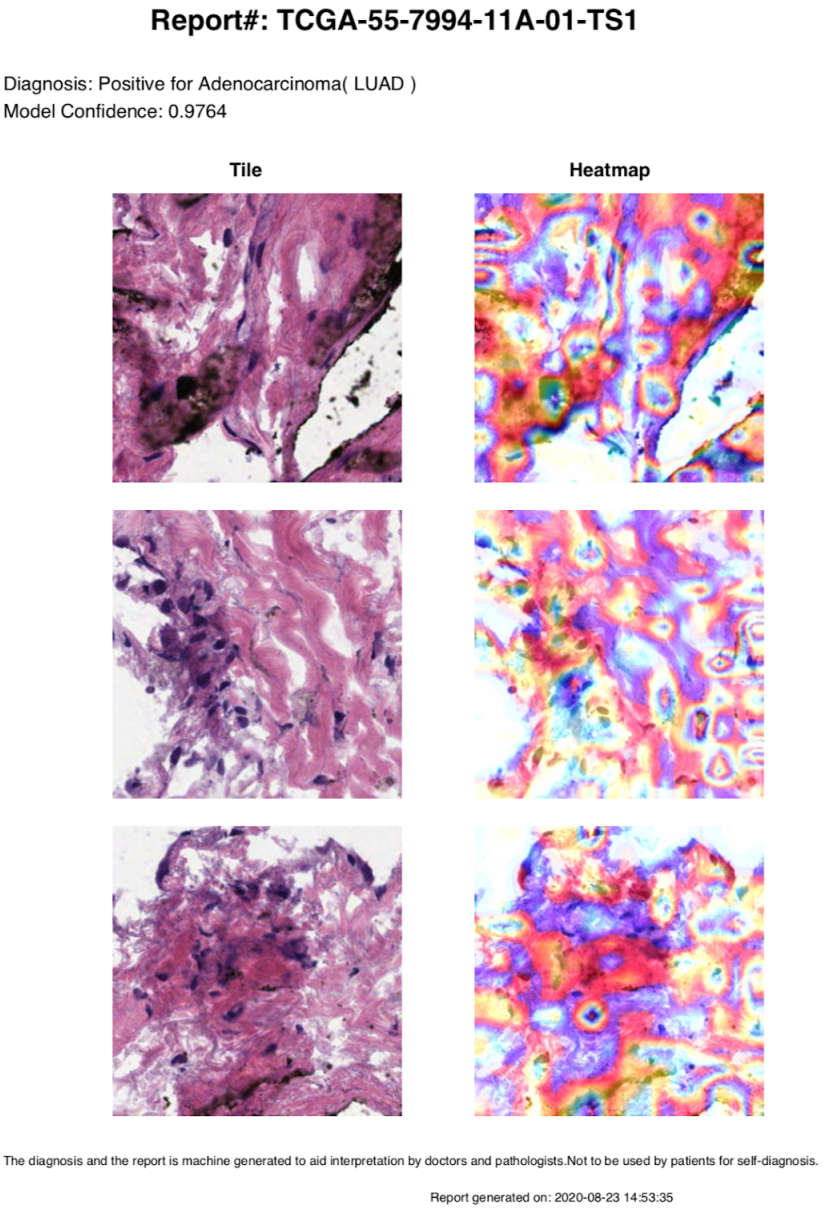


Figure 7.7: Sample report showing the patient id, diagnosis, three tiles with highest prediction probability and the corresponding heatmaps.

Chapter 8

Conclusion and Future Work

The domain of automated medical analysis has seen a lot of inventive solutions in the form of Deep Learning approaches. The problem of non small cell lung carcinoma classification is among the most challenging ones due to the poor differentiability even by experienced pathologists. We have explored the intricacies and issues involved in this problem statement and have successfully developed two models. The one based on Inception v3 architecture has achieved an accuracy of 84% and our custom model has an accuracy of 98%. This provides cost effective and accurate results for doctors and pathologists within seconds. The comparative heat map generated by the model servers as visual aid for pathologists and doctors to locate malignant cells. We have hosted the model as a product, so that immediate analysis can be performed even in areas where infrastructure and facilities are limited. In the future, we aim to remove histopathological slide format dependencies type of staining during the pre-processing stage by the use of Deep Generative Adversarial Networks. We will also explore the correlation of histopathological slide images with clinical biomarkers and survival chances.

Appendix A

Project Team details

Project Title: Deep Learning approach for subtype classification and localization from non small cell lung cancer histopathology images

USN	Team members name	Email-ID	Phone no
01JST16CS035	Hymavathi B U	hymabu@gmail.com	9164563737
01JST16CS132	Santosh Umesh Shet	santo.shet@gmail.com	7259419931
01JST17CS421	Tejas H R	tejashr244@gmail.com	8105317697

Appendix B

COs, POs and PSOs Mapping for the Project Work (CS84P)

Course Outcomes:

CO1: Formulate the problem definition, conduct literature review and apply requirements analysis.

CO2: Develop and implement algorithms for solving the problem formulated.

CO3: Comprehend, present and defend the results of exhaustive testing and explain the major findings.

Program Outcomes:

PO1: Apply knowledge of computing, mathematics, science, and foundation engineering concepts to solve the computer engineering problems.

PO2: Identify, formulate and analyze complex engineering problems.

PO3: Plan, implement and evaluate a computer-based system to meet desired societal needs such as economic, environmental, political, healthcare and safety within realistic constraints.

PO4: Incorporate research methods to design and conduct experiments to investigate real-time problems, to analyze, interpret and provide feasible conclusion.

PO5: Propose innovative ideas and solutions using modern tools.

PO6: Apply computing knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to professional engineering practice.

PO7: Analyze the local and global impact of computing on individuals and organizations for sustainable development.

PO8: Adopt ethical principles and uphold the responsibilities and norms of computer engineering practice.

PO9: Work effectively as an individual and as a member or leader in diverse teams and in multidisciplinary domains.

PO10: Effectively communicate and comprehend.

PO11: Demonstrate and apply engineering knowledge and management principles to manage projects in multidisciplinary environments.

PO12: Recognize contemporary issues and adapt to technological changes for lifelong learning.

Program Specific Outcomes:

PSO1: Problem Solving Skills: Ability to apply standard practices and mathematical methodologies to solve computational tasks, model real world problems in the areas of database systems, system software, web technologies and Networking solutions with an appropriate knowledge of Data structures and Algorithms.

PSO2: Knowledge of Computer Systems: An understanding of the structure and working of the computer systems with performance study of various computing architectures.

PSO3: Successful Career and Entrepreneurship: The ability to get acquaintance with the state of the art software technologies leading to entrepreneurship and higher studies.

PSO4: Computing and Research Ability: Ability to use knowledge in various domains

to identify research gaps and to provide solution to new ideas leading to innovations.

Write justification for the mapping

Since our's is machine learning project, to solve the computer engineering problems and to develop and implementing algorithms for solving the problem formulated, we have applied knowledge of computing, mathematics, and foundational engineering concepts. So CO2 has high relevance to the PO1 and PSO1 and this knowledge used less in formulating problems, and little more in Comprehend, present and defend the results of exhaustive testing and explain the major findings. Hence PO1 has medium relevance to CO3 and less relevance to CO3.

In this project we have Identified, formulated and analyzed complex engineering problems and Incorporate research methods to design and conduct experiments to investigate real-time problems, to analyze, interpret and provide feasible conclusion. Hence PO2 and PO4 has high relevance to CO1 and comparatively less relevance to the CO2 and CO3.

We have Planned, implemented and evaluated a computer-based system to meet desired societal needs and we have incorporated research methods to design and conduct experiments to investigate real-time problems, to analyze, interpret and provide feasible conclusion. along with that we have tried innovative ideas and solutions using modern tools. This has high relevance to C02. We have applied computing knowledge to assess societal, and cultural issues and the consequent responsibilities relevant to professional engineering practice. We analyzed the local and global impact of computing on individuals and organizations for sustainable development.

We adopted ethical principles and uphold the responsibilities and norms of computer engineering practice. And we tried to work effectively as an individual and as a member or leader in diverse teams and in multidisciplinary domains. These has some relevance in developing as well as formulating problem and even presenting that. Hence PO6, PO7, PO8, PO9 has some relevance to all the three COs.

We as a team, tried to communicate effectively and tried to demonstrate and apply engineering knowledge and management principles to manage projects in multidisciplinary environments. It has high relevance to CO3 and less relevance to other COs. We try to understand the structure and working of the computer systems with performance study of various computing architectures. And we tried to use knowledge in various domains to identify research gaps and to provide solution to new ideas leading to innovations. This has high relevance to CO1.We tried to get acquaintance with the state of the art software technologies leading to entrepreneurship and higher studies, This has medium relevance to all the COs.

Note:

1. Scale 1 – Low relevance
2. Scale 2 – Medium relevance
3. Scale 3 – High relevance

SEM	SUBJECT	CODE	CO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PS 01	PS 02	PS 03	PS 04
8	Project work	CS84P	Co 1	2	3	1	3	2	2	3	1	2	3	2	2	2	1	2	3
			CO2	3	3	1	2	2	1	2	1	2	3	3	3	2	2	1	2
			CO3	2	1	2	3	3	2	1	2	3	1	2	2	3	1	3	3

Appendix C

"Tracking and Preventing Diseases with Artificial Intelligence " to be published by Springer Book Chapters

Main Highlights

- Book series: Intelligent Systems Reference Library
- The books of this series are submitted to ISI Web of Science, SCOPUS, DBLP and Springerlink
- No publication fees

Important dates

- Chapter Submission Deadline: **15th October 2020**
- Acceptance Notification: **15th December 2020**
- Camera Ready Submission: **31st December 2020**

Link: https://philippe-fournier-viger.com/books/ai_disease_book/Call%20for%20Chapters%20V3.jpg

References

1. CancerStat0 https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga/studied-cancers/lung-adenocarcinoma

2. CancerStat1 https://www.cancer.org/cancer/lung-cancer/about/ key-statistics.html

3. Yu et al Yu, K., Zhang, C., Berry, G. et al. Predicting non-small cell lung cancer prognosis by fully automated microscopic pathology image features. Nat Commun 7, 12474 (2016). https://doi.org/10.1038/ncomms12474

4. AIHealthCare Esteva, A., Robicquet, A., Ramsundar, B. et al. A guide to deep learning in healthcare. Nat Med 25, 24–29 (2019). https://doi.org/10.1038/s41591-018-0316-z

5. DLRef H. Greenspan, B. van Ginneken and R. M. Summers, "Guest Editorial Deep Learning in Medical Imaging: Overview and Future Promise of an Exciting New Technique," in IEEE Transactions on Medical Imaging, vol. 35, no. 5, pp. 1153-1159, May 2016, doi: 10.1109/TMI.2016.2553401.

6. EstevaRef Esteva, A., Kuprel, B., Novoa, R. et al. Dermatologist-level classification of skin cancer with deep neural networks. Nature 542, 115–118 (2017). https://doi.org/10.1038/nature21056

7. MednikovRef Y. Mednikov, S. Nehemia, B. Zheng, O. Benzaquen and D. Lederman, "Transfer Representation Learning using Inception-V3 for the Detection

of Masses in Mammography," 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Honolulu, HI, 2018, pp. 2587-2590, doi: 10.1109/EMBC.2018.8512750.

8. WangRef C. Wang et al., "Pulmonary Image Classification Based on Inception-v3 Transfer Learning Model," in IEEE Access, vol. 7, pp. 146533-146541, 2019, doi: 10.1109/ACCESS.2019.2946000.

9. GuanRef Guan, Qing et al. "Deep convolutional neural network Inception-v3 model for differential diagnosing of lymph node in cytological images: a pilot study." Annals of translational medicine vol. 7,14 (2019): 307. doi:10.21037/atm.2019. 06.29

10. JinRef J. Li, P. Wang, Y. Li, Y. Zhou, X. Liu and K. Luan, "Transfer Learning of Pre- Trained Inception-V3 Model for Colorectal Cancer Lymph Node Metastasis Classification," 2018 IEEE International Conference on Mechatronics and Automation (ICMA), Changchun, 2018, pp. 1650-1654, doi: 10.1109/ICMA.2018.8484405.

11. YimRef Ding Y, Sohn JH, Kawczynski MG et al. A Deep Learning Model to Predict a Diagnosis of Alzheimer Disease by Using 18F-FDG PET of the Brain. Radiology 2019;290(2):456–464, doi: 10.1148/radiol.2020192224

12. GulshanRef Gulshan V, Peng L, Coram M, et al. Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. JAMA. 2016;316(22):2402–2410. doi:10.1001/jama.2016.17216

13. YurRef Yu, Kun-Hsing et al. "Predicting non-small cell lung cancer prognosis by fully automated microscopic pathology image features." Nature communications vol. 7 12474. 16 Aug. 2016, doi:10.1038/ncomms12474

14. Link: http://alexlenail.me/NN-SVG/AlexNet.html

15. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

16. A. Mahendran and A. Vedaldi. Visualizing deep convolutionalneural networks using natural pre-images. International Journal of Computer Vision, pages 1–23, 2016.

17. J.Long,E.Shelhamer,andT.Darrell.Fullyconvolutionalnetworks for semantic segmentation. In CVPR, 2015.

18. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning forimage recognition. In CVPR, 2016.

19. R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hier- archies for Accurate Object Detection and Semantic Segmentation. In CVPR, 2014.

20. Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. IEEE transactions on patternanalysis and machine intelligence, 35(8):1798–1828, 2013.

21. Z.C.Lipton.TheMythosofModelInterpretability.ArXive-prints, June 2016.