PREDICTING PROTEIN ORGANELLE LOCALIZATION

A Project Report
Submitted in partial fulfillment of the
requirements of the award of the degree of

BACHELOR OF SCIENCE (COMPUTER SCIENCE)
By
HAIDER ZAINUDDIN ALI

Seat Number:

Under the esteemed guidance of

Prof. Javed Pathan Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE

RIZVI COLLEGE OF ARTS SCIENCE AND COMMERCE

Affiliated to University of Mumbai
MUMBAI, PIN 400050
MAHARASHTRA
2020-2021

RIZVI COLLEGE OF ARTS SCIENCE AND COMMERCE

(Affiliated to University of Mumbai)

MUMBAI-MAHARASHTRA-50

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that the project entitled, **"Protein cell localization"**, is benefited work of **Haider Zainuddin Ali** bearing Seat Number:__01___submitted in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE in COMPUTER SCIENCE from University of Mumbai.

Internal Guide **Co-ordinator**

External Examiner

Date: **College Seal**

ABSTRACT

Predicting protein cells are hard to classify from a single blood image. Model will predict protein organelle localization labels for each sample. There are in total 10 different labels present in the dataset. The dataset is acquired in a highly standardized way using one imaging modality (confocal microscopy). However, the dataset comprises 10 different cell types of highly different morphology, which affect the protein patterns of the different organelles.

ACKNOWLEDGEMENT

I extremely grateful and remain indebted to our guide Prof. Javed Pathan for being a source of Inspiration and for his constant support in the Design, Implementation and Evolution of the Project. We are thankful to them for their constant constructive criticism and invaluable suggestions, which benefited us a lot while developing the project on “IDENTIFYING PROTEIN CELLS”.
I also thank my parents for supporting me and helping me in every way possible.
I also thank our Department teacher Prof. Arif Patel and who have helped in successful completion of the project.
I also offer our sincerest gratitude to our very own principal of college

Prof. Anjum Ara Ahmad for his constant support and consideration.

DECLARATION

I, hereby declare that the project entitled, "**Identifying protein cells"** done at **RIZVI COLLEGE OF ARTS SCIENCE AND COMMERCE,** has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfillment of the requirements for the award of degree of **BACHELOR OF SCIENCE (COMPUTER SCIENCE)** to be submitted as **Fifth Semester** project as part of our curriculum.

Haider Zainuddin Ali

____

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION....................1

1.1 Background of the project 2

1.2 Objective of the Project 3

1.3 Purpose and Scope of the Project 3

1.4 Project Schedule 3

CHAPTER 2:SYSTEM ANALYSIS....................4

2.1 Existing System 4

2.2 Proposed System.................... 4

2.3 Hardware Requirements 6

2.4 Software Requirements.................... 6

2.5 Justification of Technology Used 6

CHAPTER 3:SYSTEM DESIGN....................8

3.1 Methodology.................... 8

3.2 Module Division 10

3.3 Data Dictionary.................... 11

3.4 E-R Diagram.................... 12

3.5 DFD 13

CHAPTER 4: IMPLEMENTATION AND CODING 14

4.1 Coding.................... 15

4.2 Screenshot of the Project 22

4.3 Testing Approach.................... 27

4.4 Test Case.................... 31

CHAPTER 5: CONCLUSION AND FUTURE WORK 32

5.1 Limitation of the system.................... 34

CHAPTER 6: REFERENCES....................35

Chapter 1. Introduction

Doctors and researchers have lots of images of blood cells to find ways to tackle the impurity in the protein cells. Proteins are “the doers” in the human cell, executing many functions that together enable life. Historically, classification of proteins has been limited to single patterns in one or a few cell types, but in order to fully understand the complexity of the human cell, models must classify mixed patterns across a range of different human cells. Images visualizing proteins in cells are commonly used for biomedical research, and these cells could hold the key for the next breakthrough in medicine.
However, thanks to advances in high-throughput microscopy, these images are generated at a far greater pace than what can be manually evaluated. Therefore, the need is greater than ever for automating biomedical image analysis to accelerate the understanding of human cells and disease. The dataset of this project is taken from Human protein Atlas kaggle competition.

1.1 Background of the project

The purpose of this project is to develop an image classification model to accurately predict that which from which 9 categories of protein types falls into a protein organelle localization. The categories cannot be classified through naked eye. The Human Protein Atlas will use these models to build a tool integrated with their smart-microscopy system to identify a protein's location(s) from a high-throughput image.

Historically, classification of proteins has been limited to single patterns in one or a few cell types, but in order to fully understand the complexity of the human cell, models must classify mixed patterns across a range of different human cells. Images visualizing proteins in cells are commonly used for biomedical research, and these cells could hold the key for the next breakthrough in medicine. However, thanks to advances in high-throughput microscopy, these images are generated at a far greater pace than what can be manually evaluated. Therefore, the need is greater than ever for automating biomedical image analysis to accelerate the understanding of human cells and disease.

1.2 Objective of the Project

This project is intended to help Human Protein Atlas Corporation to build a tool integrated with their smart-microscopy system to identify a protein's location from a high-throughput image. This project helped me learn new skills like regularization techniques, Artificial neural network, Transfer learning.

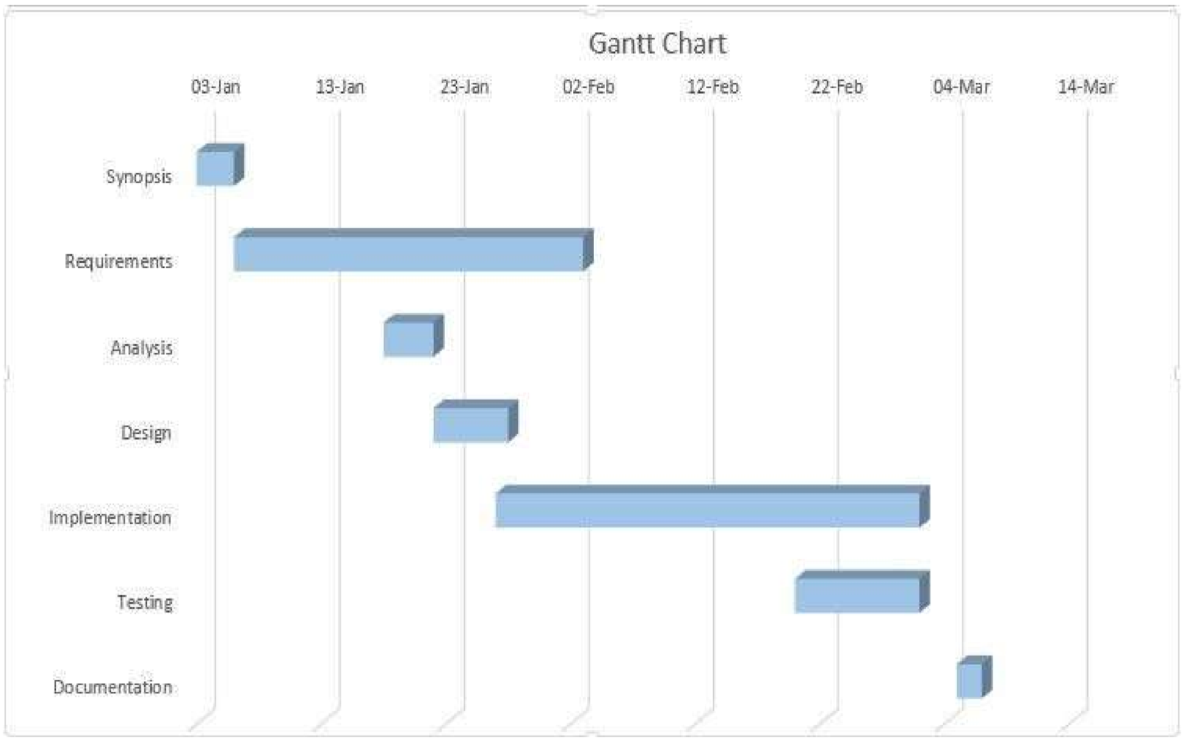
1.3 Purpose and Scope of the project

Purpose of this project is to help Human Protein Atlas to build a tool integrated with their smart-microscopy system to identify a protein's location(s) from a high-throughput image.

1.4 Project Schedule

Gantt chart is type of BAR CHART, which depicts the chronological development of various phases in project. It sometime also refers as the schedule chart because of which a team can keep track of the project in terms of time scheduling.

A typical Gantt chart gives the start and finish dates of the various phases of development and displays it in the graphical form of bars.



CHAPTER 2. SYSTEM ANALYSIS

System analysis is defined as "the process of studying a procedure or business in order to identify its goals and purposes and create systems and procedures that will achieve them in an efficient way". Another view sees system analysis as a problem-solving technique that breaks down a system into its component pieces for the purpose of the studying how well those component parts work and interact to accomplish their purpose.

The field of system analysis relates closely to requirements analysis or to operations research. It is also "an explicit formal inquiry carried out to help a decision maker identify a better course of action and make a better decision than she might otherwise have made."

System analysis is used in every field where something is developed. Analysis can also be a series of components that perform organic functions together, such as system engineering. System engineering is an interdisciplinary field of engineering that focuses on how complex engineering projects should be designed and managed.

2.1 Existing System

The Cell Atlas provides high-resolution insights into the expression and spatio-temporal distribution of proteins within human cells. Using a panel of 64 cell lines to represent various cell populations in different organs and tissues of the human body, the mRNA expression of all human genes are characterized by deep RNA-sequencing.The subcellular distribution of each protein can be determined in a subset of cell lines selected based on corresponding gene expression.

However, deep learning techniques can take these images and localize the classes present in a particular image with high accuracy.

2.2 Proposed System

The proposed system used for building image classification system used CNN a Deep learning technique to classify protein cells with their respective blood cells by the data collected by Human Protein Atlas. However it would be very difficult to classify images using a simple Neural Network using logistic regression (sigmoid). The CNN convolution operation makes it very simple for Neural Network as it detects low level features first, mid-level and high level features. I have used one cycle policy for better accuracy and with less number of iterations.

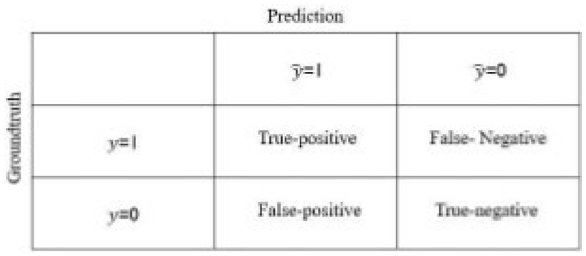
If the prediction and ground truth are equal, then it is either True-positive or True negative based on the classification labels. If the prediction is not equal to the ground truth, then it is either False positive or

False- Negative based on the classification labels. It gives the result by showing the recall, precision and accuracy of each different algorithms by comparing which has the best accuracy.

There is only one algorithm used i.e. Artificial Neural Network with convolution operation. I have tried different optimization algorithms.

- **Evaluation Metrics:**

To visualize the performance of an algorithm, typically a supervised learning confusion matrix is used. Also, known as error matrix, each column of the confusion matrix signifies an instance of a predicted class and each row signifies an instance of the actual class.



Prediction

Groundtruth

	$\bar{y}=1$	$\bar{y}=0$
$y=1$	True-positive	False- Negative
$y=0$	False-positive	True-negative

The above table is an example of a confusion matrix. If the prediction and ground truth are equal, then it is either True-positive or True negative based on the classification labels. If the prediction is not equal to the ground truth, then it is either False positive or False-Negative based on the classification labels. From the table we are calculating the Accuracy, Precision and Recall.

We are also determining the ROC curve to evaluate the performance of the algorithm.

- **Logistic Regression:**

Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable in which there are only two possible outcomes. The dependent variable is binary or dichotomous, i.e. it only contains data coded as 1 or 0. The binary logistic model is used to estimate the probability of a binary response based on one or more predictor variables.

The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest and a set of independent (predictor or explanatory) variables.

Logistic regression equation - Here p is the probability of presence of the characteristic of interest. The logit transformation is defined as the logged odds:

Odds=p/(1-p) and Logit(p)=ln(p/(1-p))

2.3. Hardware Requirements

Processor Type: Intel i3 or more, AMD E2-7110 APU

Processor Speed: 1.80 GHz or above

Hard disk Space: Minimum 1 GB free

RAM: 2 GB

2.4. Software Requirements

Operating System: Windows 7 or above, Linux.

Platform: Python IDLE v3.

Language: Python.

2.5. Justification of Technology Used

Technologies used to develop this software are as follows:

An Overview of Windows10:

It is the successor to Windows8.1, and was released to manufacturing on July 15, 2015, and broadly released for retail sale on July 29, 2015. Devices in enterprise environments can receive these updates at a slower pace, or use long-term support milestones that only receive critical updates, such as security patches, over their ten-year lifespan of extended support.

Windows 10 makes more consistent the user experience and functionality between different classes of device, and addresses shortcomings in the user interface that were introduced in Windows 8. Windows10 mobile, the successor to Windows Phone 8.1, shares some user interface elements and apps with its PC counterpart.The Windows Runtime app ecosystem was revised into the Universal Windows platform (UWP). These universal apps are made to run across multiple platforms and device classes, including smartphones, tablets, Xbox One consoles, and other compatible Windows 10 devices. Windows apps share code across platforms, have responsive designs that adapt to the needs of the device and available inputs, can synchronize data between Windows 10 devices (including notifications, credentials, and allowing cross-platform multiplayer for games), and are distributed through Microsoft store (rebranded from Windows Store since September 2017).

Python 3.7-32bit

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

IDLE has the following features:

- cross-platform: works mostly the same on Windows, Unix, and macOS.
- Python shell window (interactive interpreter) with colorizing of code input, output, and error messages.
- multi-window text editor with multiple undo, Python colorizing, smart indent, call tips, auto completion, and other features.
- search within any window, replace within editor windows, and search through multiple files (grep).
- debugger with persistent breakpoints, stepping, and viewing of global and local namespaces.
- configuration, browsers, and other dialogs.

CHAPTER 3. SYSTEM DESIGN

Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. To find out the requirements for the new system, it is always essential to study and recognize the problems of existing system, which will become the building blocks of this development.

To study different alternatives and to get a viable solution, Study of the system is necessary.

Here are the stages performed in preliminary study of the system:

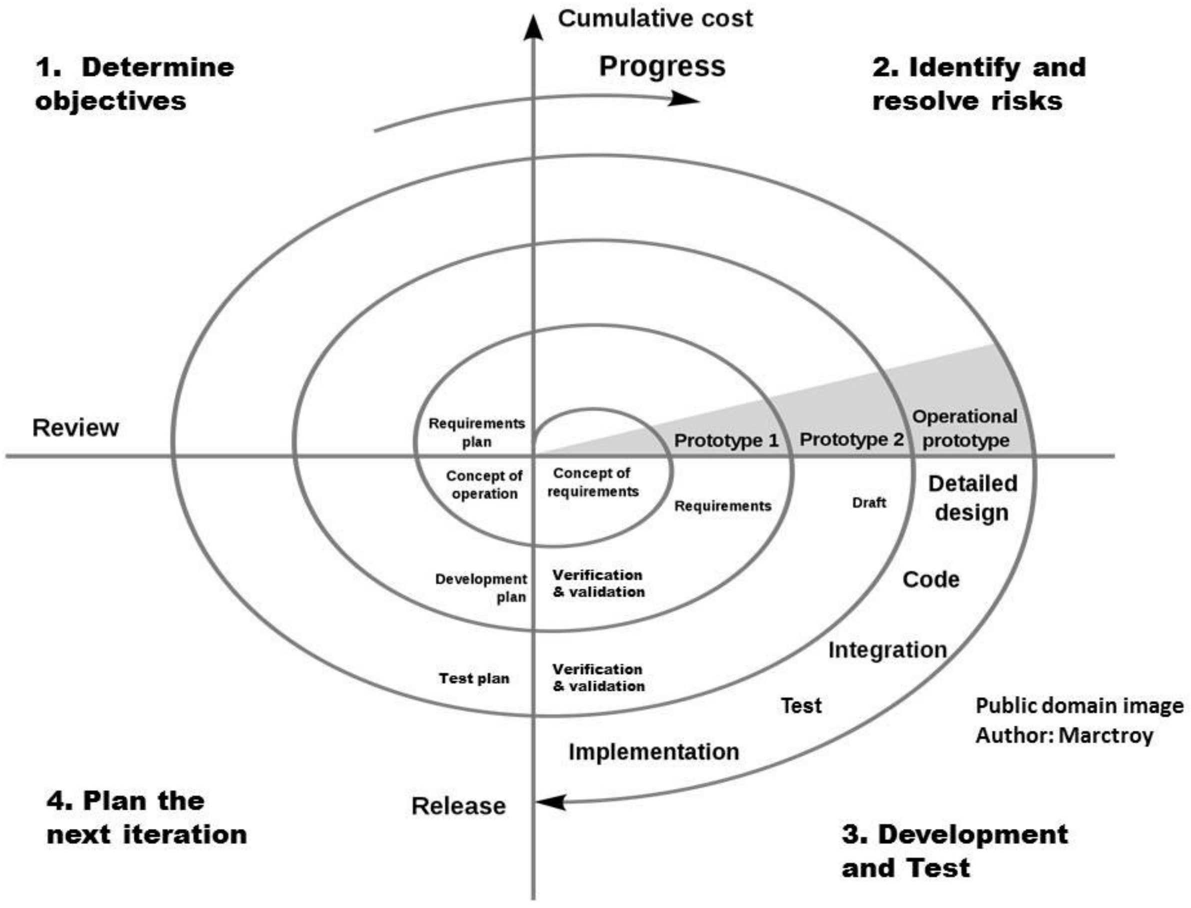
- Initially, when an enquiry comes, the data is saved in the database which can be later on follow up to convert it into a potential lead.
- When this potential student comes up, there is no need to fill in the data as it can be pulled up from the enquiry database.
- With the help of this software, everything (Data) in an organization can be digitally recorded into the system.
- Integrity can be maintained.

3.1 Methodology:

Software Development Model:

Spiral Model:

- A software development model serves the developer as a guiding tool during and throughout the development of the software, it facilitates to implement various steps / stages during the development of the overall project.
- Spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project.
- Each loop of the spiral is called a Phase of the software development process. The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using spiral model.
- The Radius of the spiral at any point represents the expenses (cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.



Spiral Model Approach:

Each phase of Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

1. **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
2. **Identify and resolve Risks:** During the second quadrant all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution is identified and the risks are resolved using the best possible strategy. At the end of this quadrant, Prototype is built for the best possible solution.
3. **Develop next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.
4. **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.

3.2 Module Division

Here are the modules used in this project

Modules Used:

- **Numpy:**

  It is a library for the Python programming language, adding support for large,multi-dimensional arrays and matrices. Along with the high collection of mathematical functions to operate on these arrays.

- **Pandas**:

  pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

- **Random**:

  Python offers random module that can generate random numbers. These are pseudo-random number as the sequence of number generated depends on the seed.

- **Matplotlib.pyplot:**

  Matplotlib is a plotting library for the python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like TKinter, WxPython, Ot, orGTK+.

- **from torchvision.utils import make_grid**:

  The make grip function is used to display images in a grid format. Also it can add many features which makes visualization more intuitive. Some of them are:

  Padding: Adding rows and columns around the image, the padding value depends upon pad_value parameter.

  Normalize: This parameter scales image in range between 0 to 1.

- **from torch.utils.data import Dataset, random_split, DataLoader:**

  Dataset: This method is used to make a torch dataset to bring images and csv files together.

  DataLoader: This method loads dataset in torch and also this is used to load dataset in cuda(NIVIDIA GPU).

  Random_split: This method used to decrease overfit the model.

- **import torchvision.models as models:**

  Inbuilt models can be used from torchvision models class for increasing accuracy

- **PIL:**
  It is used for image processing in python.

- **TQDM:**
  In addition to its low overhead, tqdm uses smart algorithms to predict the remaining time and to skip unnecessary iteration displays, which allows for a negligible overhead in most cases.

- **import torch.nn.functional as F:**
  This functional class from torch.nn is used to create a neural network. It contain optimizers, Loss f unction, sparse functions, etc.

- **import torch.nn as nn:**
  The nn mdule contains all the activation functions, pooling functions, normalization functions,etc.

- **from sklearn.metrics import f1_score:**
  The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:
  F1 = 2 * (precision * recall) / (precision + recall)

3.3 Data

Here is a description of the dataset. The dataset is made up of train and test dataset. Train dataset consists of 19,200 images and test dataset consists of 8243 images. I am predicting protein organelle localization labels for each sample. There are in total 10 different labels present in the dataset. The dataset is acquired in a highly standardized way using one imaging modality (confocal microscopy). However, the dataset comprises 10 different cell types of highly different morphology, which affect the protein patterns of the different organelles. Each image can have 1 or more labels associated to them.

The labels are represented as integers that map to the following:

- 0: ‘Mitochindria’
- 1: ‘Nuclear Bodies’
- 2: ‘Nuclioli’
- 3: ‘Golgi Apparatus’
- 4: ‘Nucleoplasm’
- 5: ‘Nucleoli fibrillar center’
- 6: ‘Cytosol’
- 7: ‘Plasma Membrane’
- 8: ‘Centrosome’
- 9: ‘Nuclear Speckles’

Every image in the train and the test dataset has its unique id in the csv file for which we will use pandas package to read.

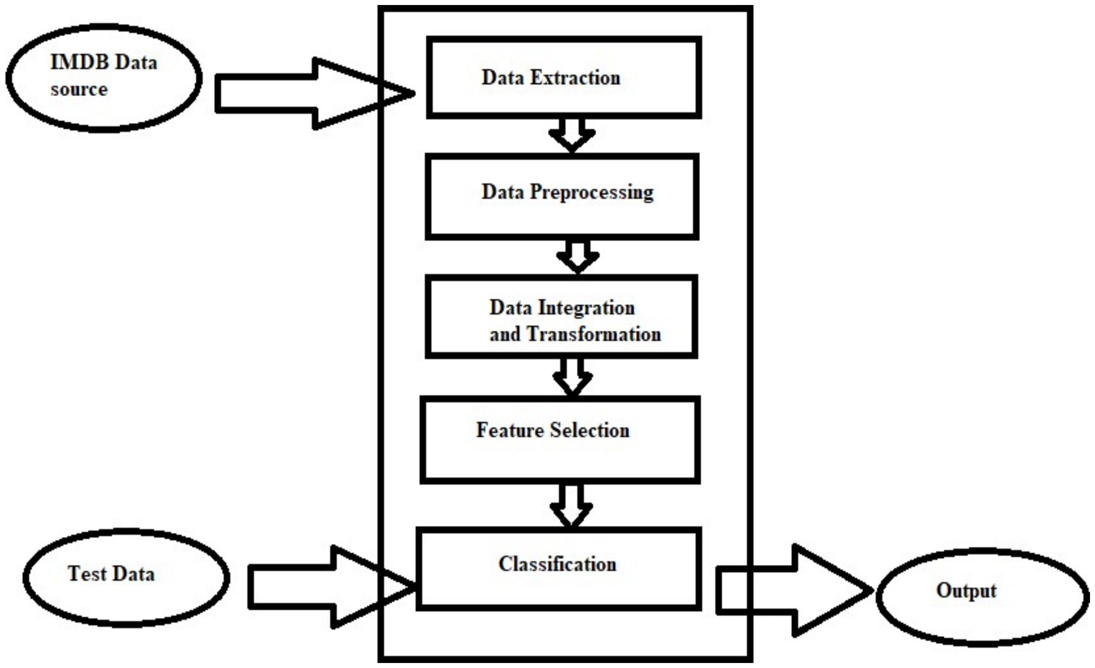
3.4. E-R Diagram

Entity Relationship Diagram:

E-R Model is a popular high level conceptual data model. This model and its variations are frequently used for the conceptual design of database application and many database design tools employ its concept. A database that confirms to an E-R diagram can be represented by a collection of tables in the relational system. The mapping of E-R diagram to the entities is:

Entity relationship diagram provides a visual starting point for database design that can also be used for determine information system requirements throughout an organization.

However, while an ERD can be helpful for organizing data that can be represented by a relational structure, it can’t sufficiently represent semi-structured and unstructured data.



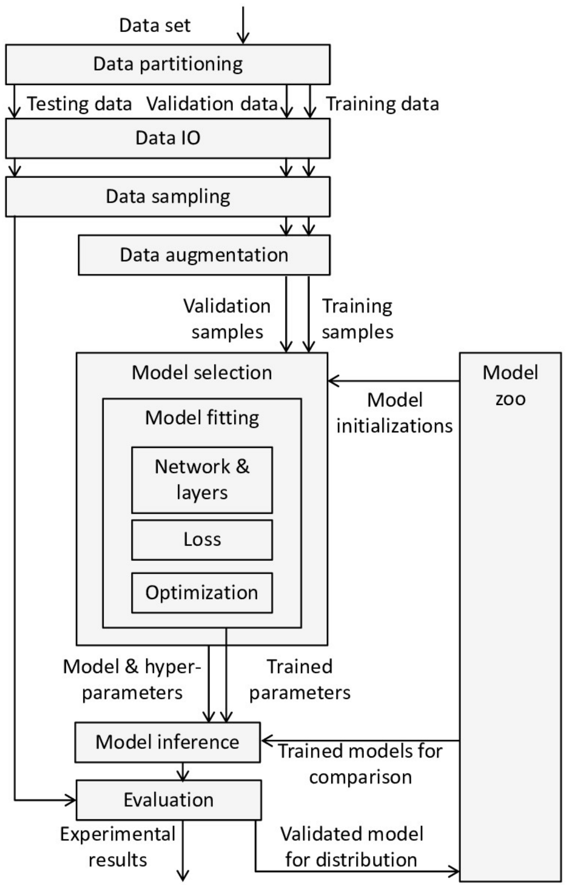
3.5 DFD

Dataflow Diagram:

Data flow diagram is the starting point of the design phase that functionally decomposes the requirements specification. A DFD consists of a series of bubbles joined by lines. The bubbles represent data transformation and the lines represent data flows in the system. A DFD describes what data flow rather than how they are processed, so it does not hardware, software and data structure. A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design).

A data flow diagram (DFD) is a significant modeling technique for analyzing and constructing information processes. DFD literally means an illustration that explains the course or movement of information in a process. DFD illustrates this flow of information in a process based on the inputs and outputs. A DFD can be referred to as a Process Model. The data flow diagram is a graphical description of a system's data and how to Process transform the data is known as Data Flow Diagram (DFD). Unlike details flow chart, DFDs don't supply detail descriptions of modules that graphically describe a system's data and how the data interact with the system.

DFD:



CHAPTER 4. IMPLEMENTATION AND CODING

Project implementation (or project execution) is the phase where visions and plans become reality. This is the logical conclusion, after evaluating, deciding, visioning, planning, applying for funds and finding the financial resources of a project.

Implementation simply means carrying out the activities described in your work plan. Executing a project in the water and sanitation sector is a very complex mission, as it requires the coordination of a wide range of activities, the overseeing of a team, the management of budget, the communication to the public, among other issues. Independent of whether it is a social project to raise the awareness and promote hygiene or it is a construction project for service delivery, there is a certain process that has to be followed.

The following description will give you an introduction into the implementation of projects in sustainable sanitation and water management, and highlights key aspects that have to be taken into account for a successful implementation.

It is important to take into account that independently of the nature of the project, implementation takes time, usually more than it is planned, and that many external constraints can appear, which should be considered when initiating the implementation step (i.e. seasonality in availability of community engagement/resources)

The experiment is as previously described aimed at evaluating the prediction performance of given algorithms for predicting movie ratings, and most importantly to investigate whether or not any significant advantages of using RF can be found.

The following sections will go through the process of collecting, preparing and mining the movie data in order to reliably evaluate the predictive performance of these algorithms. This includes selection of datasets as well as selection and preparation of data and attributes, but also the practical process of acquiring the results as in selection of metrics, configuration of parameters for the algorithms and design of the experiment itself.

1. Data Collection:

Data collection is very important and costly part of any project. In order to properly evaluate the relevant deep learning algorithms on protein labels predictions a number of relevant datasets has to be considered for the experiment.

The Human Cellular and Organelle Proteome chapters are a collection of interactive pages providing conceptual overviews, compilations and analyses of the data generated within the Cell Atlas from different perspectives.

2. Data Preparation

The data is in a zip compressed folder and is separated into different directories. For this we need to make a dataset which will contain images with its unique image id to make it easy for the model to train and make better predictions. Pytorch Dataset and Dataloader class helps us to make custom datasets. Dataloader helps to load the whole dataset in cuda (gpu from NVIDIA). GPU helps model to train much faster than cpu.

Dataset and Dataloader class helps to helps to randomly split the dataset into equal size of batches. Also they use more cpu blocks to distribute their work.

4.1 Code:

4.1.1 Importing Libraries:

```
import os
import torch
import pandas as pd
import numpy as np
from torch.utils.data import Dataset, random_split, DataLoader
from PIL import Image
import torchvision.models as models
import matplotlib.pyplot as plt
from tqdm.notebook import tqdm
import torchvision.transforms as T
from sklearn.metrics import f1_score

import torch.nn.functional as F
import torch.nn as nn
from torchvision.utils import make_grid
% matplotlib inline
```

```
def encode_label(label):
    target = torch.zeros(10)
    for l in str(label).split(' '):

        target[int(l)] = 1.
    return target

def decode_target(target, text_labels=False, threshold=0.5):
    result = []
    for i, x in enumerate(target):
        if (x >= threshold):
            if text_labels:
                result.append(labels[i] + "(" + str(i) + ")")
            else:
                result.append(str(i))
    return ' '.join(result)
```

4.1.2 Creating custom Pytorch dataset.

```
class HumanProteinDataset(Dataset):
    def __init__(self, df, root_dir, transform=None):
        self.df = df
        self.transform = transform
        self.root_dir = root_dir
    def __len__(self):
        return len(self.df)
    def __getitem__(self, idx):
        row = self.df.loc[idx]
        img_id, img_label = row['Image'], row['Label']
        img_fname = self.root_dir + "/" + str(img_id) + ".png"
        img = Image.open(img_fname)
        if self.transform:

            img = self.transform(img)
        return img, encode_label(img_label)
```

4.1.3 Data Augmentation:

```
imagenet_stats = ([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
train_tfms = T.Compose([
    T.RandomCrop(512, padding=8, padding_mode='reflect'),
#     T.RandomResizedCrop(256, scale=(0.5,0.9), ratio=(1, 1)),
#     T.ColorJitter(brightness=0.1, contrast=0.1, saturation=0.1, hue=0.1),
    T.RandomHorizontalFlip(),
    T.RandomRotation(10),
    T.ToTensor(),
#     T.Normalize(*imagenet_stats,inplace=True),
    T.RandomErasing(inplace=True)
])
valid_tfms = T.Compose([
#     T.Resize(256),
    T.ToTensor(),
#     T.Normalize(*imagenet_stats)
])
```

4.1.4 F Score Function:

```
def F_score(output, label, threshold=0.5, beta=1):
    prob = output > threshold
    label = label > threshold
    TP = (prob & label).sum(1).float()
    TN = ((~prob) & (~label)).sum(1).float()


    FP = (prob & (~label)).sum(1).float()
    FN = ((~prob) & label).sum(1).float()
    precision = torch.mean(TP / (TP + FP + 1e-12))
    recall = torch.mean(TP / (TP + FN + 1e-12))
    F2 = (1 + beta**2) * precision * recall / (beta**2 * precision + recall + 1e-12)
    return F2.mean(0)
```

4.1.5 Base class for classification:

```
class MultilabelImageClassificationBase(nn.Module):
    def training_step(self, batch):
        images, targets = batch
        out = self(images)
        loss = F.binary_cross_entropy(out, targets)
        return loss

    def validation_step(self, batch):
        images, targets = batch
        out = self(images)                           # Generate predictions
        loss = F.binary_cross_entropy(out, targets)  # Calculate loss
        score = F_score(out, targets)
        return {'val_loss': loss.detach(), 'val_score': score.detach() }

    def validation_epoch_end(self, outputs):
        batch_losses = [x['val_loss'] for x in outputs]
        epoch_loss = torch.stack(batch_losses).mean()   # Combine losses
        batch_scores = [x['val_score'] for x in outputs]
        epoch_score = torch.stack(batch_scores).mean()      # Combine accuracies
        return {'val_loss': epoch_loss.item(), 'val_score': epoch_score.item()}

    def epoch_end(self, epoch, result):
        print("Epoch [{}], last_lr: {:.4f}, train_loss: {:.4f}, val_loss: {:.4f}, val_score: {:.4f}".format(
            epoch, result['lrs'][-1], result['train_loss'], result['val_loss'], result['val_score']))
```

4.1.6 Convolution Model

```
class ProteinCnnModel(MultilabelImageClassificationBase):
    def __init__(self):
        super().__init__()

        self.network = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, padding=1),

            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),

            nn.MaxPool2d(3, 3),
            nn.Conv2d(64, 64, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Conv2d(64, 128, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),

            nn.Conv2d(128, 128, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Conv2d(128, 256, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.Conv2d(256,512,kernel_size =3, stride=1, padding=1),
            nn.ReLU(),
            nn.AdaptiveAvgPool2d(1),

            nn.Flatten(),
            nn.Linear(512, 256),
            nn.ReLU(),
            nn.Linear(256,128),
            nn.ReLU(),
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Linear(64, 10),
            nn.Sigmoid()
        )

    def forward(self, xb):
        return self.network(xb)
```

4.1.7 Resnet50 pretrained model:

```
class ProteinResnet(MultilabelImageClassificationBase):
    def __init__(self):
        super().__init__()
        # Use a pretrained model
        self.network = models.resnet50(pretrained=True)
        # Replace last layer
        num_ftrs = self.network.fc.in_features
        self.network.classifier = nn.Linear(num_ftrs, 10)
    def forward(self, xb):
        return torch.sigmoid(self.network(xb))

    def freeze(self):
        # To freeze the residual layers
        for param in self.network.parameters():

        param.require_grad = False
        for param in self.network.classifier.parameters():
            param.require_grad = True

    def unfreeze(self):

        # Unfreeze all layers

    for param in self.network.parameters():
            param.require_grad = True
```

4.1.8 Functions for changing device from CPU to GPU

```
def get_default_device():
    """Pick GPU if available, else CPU"""
    if torch.cuda.is_available():
        return torch.device('cuda')
    else:
        return torch.device('cpu')

def to_device(data, device):
    """Move tensor(s) to chosen device"""
    if isinstance(data, (list,tuple)):
        return [to_device(x, device) for x in data]
    return data.to(device, non_blocking=True)

class DeviceDataLoader():
    """Wrap a dataloader to move data to a device"""
    def __init__(self, dl, device):
        self.dl = dl
        self.device = device

    def __iter__(self):
        """Yield a batch of data after moving it to device"""
        for b in self.dl:
                yield to_device(b, self.device)

   def __len__(self):
        """Number of batches"""
       return len(self.dl)

model = to_device(ProteinResnet(), device)
history = [evaluate(model, val_dl)]
```

```
model.freeze()
epochs = 5
max_lr = 0.01

grad_clip = 0.1
weight_decay = 1e-4

opt_func = torch.optim.Adam


%%time
history += fit_one_cycle(epochs, max_lr, model, train_dl, val_dl,
                         grad_clip=grad_clip,
                         weight_decay=weight_decay,
                         opt_func=opt_func)
model.unfreeze()

%%time
history += fit_one_cycle(epochs, 0.001, model, train_dl, val_dl,
                         grad_clip=grad_clip,
                         weight_decay=weight_decay,
                         opt_func=opt_func)
def plot_scores(history):
    scores = [x['val_score'] for x in history]
    plt.plot(scores, '-x')
    plt.xlabel('epoch')
    plt.ylabel('score')
    plt.title('F1 score vs. No. of epochs');


plot_scores(history)

def plot_losses(history):
    train_losses = [x.get('train_loss') for x in history]
    val_losses = [x['val_loss'] for x in history]
    plt.plot(train_losses, '-bx')
    plt.plot(val_losses, '-rx')
    plt.xlabel('epoch')
    plt.ylabel('loss')
    plt.legend(['Training', 'Validation'])
    plt.title('Loss vs. No. of epochs');

plot_losses(history)


def plot_lrs(history):
    lrs = np.concatenate([x.get('lrs', []) for x in history])
    plt.plot(lrs)
    plt.xlabel('Batch no.')
    plt.ylabel('Learning rate')
    plt.title('Learning Rate vs. Batch no.');
```

```
plot_lrs(history)

def predict_single(image):
    xb = image.unsqueeze(0)
    xb = to_device(xb, device)
    preds = model(xb)
    prediction = preds[0]
    print("Prediction: ", prediction)
    show_sample(image, prediction)

test_df = pd.read_csv(TEST_CSV)
test_dataset = HumanProteinDataset(test_df, TEST_DIR, transform=valid_tfms)
predict_single(test_dataset[100][0])

test_dl =DeviceDataLoader(DataLoader(test_dataset, batch_size, num_workers=3, pin_memory=True, device)

@torch.no_grad()
def predict_dl(dl, model):
    torch.cuda.empty_cache()
    batch_probs = []
    for xb, _ in tqdm(dl):
        probs = model(xb)
        batch_probs.append(probs.cpu().detach())
    batch_probs = torch.cat(batch_probs)
    return [decode_target(x) for x in batch_probs]
test_preds = predict_dl(test_dl, model)

submission_df = pd.read_csv(TEST_CSV)
submission_df.Label = test_preds
submission_df.sample(20)
```

4.2. Screenshots of the project

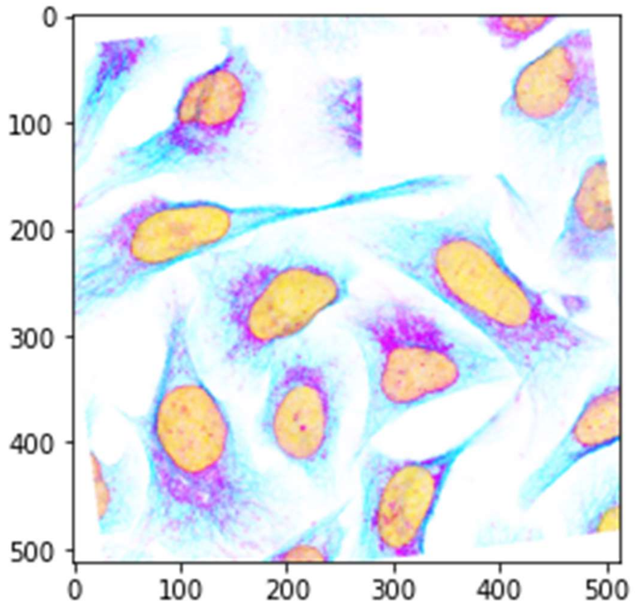


Fig – Mitochondria (0)

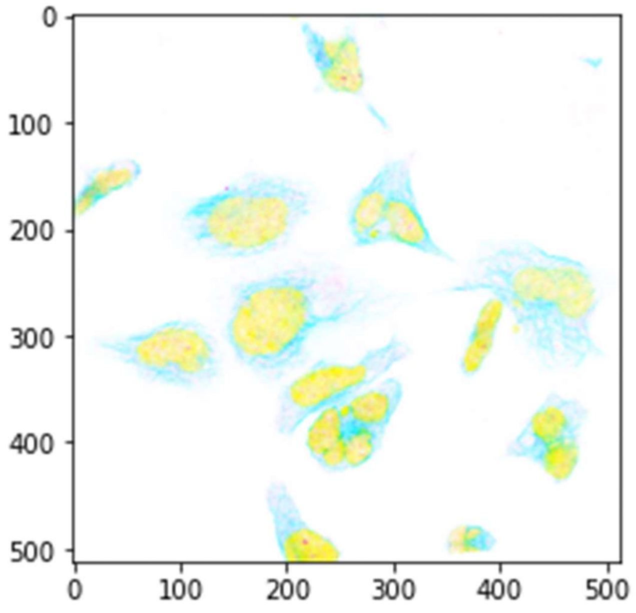


Fig - Nuclear Bodies (1)

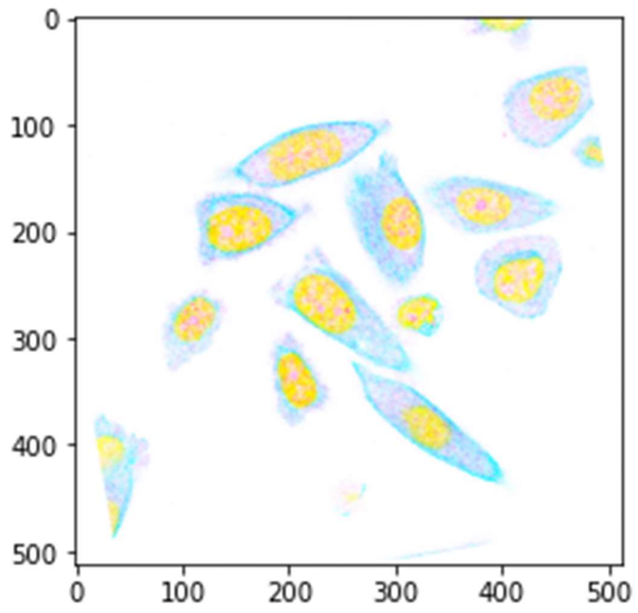


Fig – Nucleoli (2)

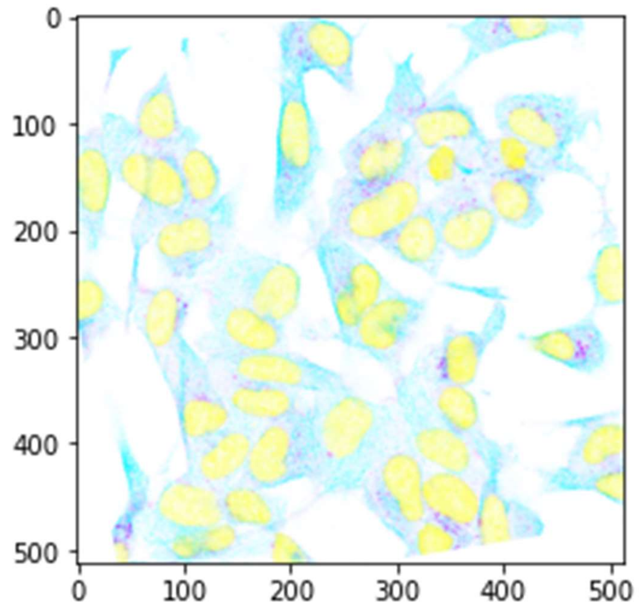


Fig - Golgi apparatus(3)

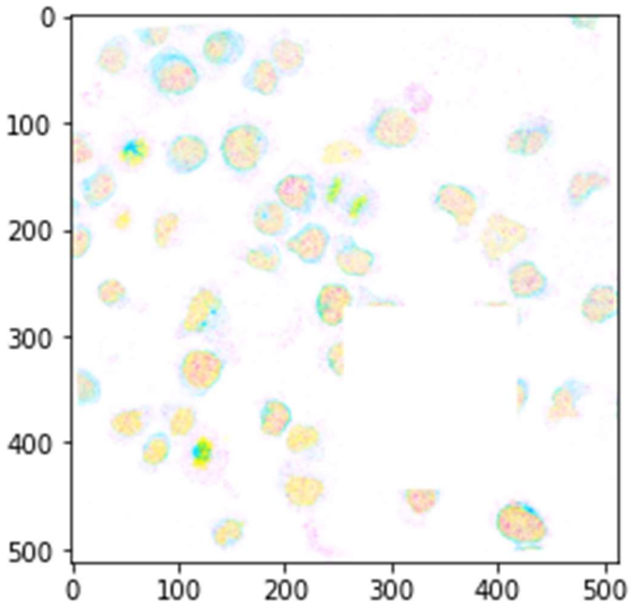


Fig - Nucleoplasm (4)

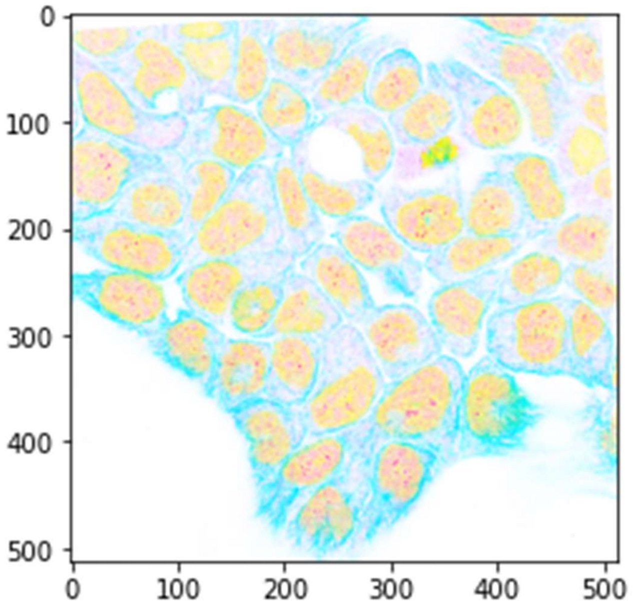


Fig - Nucleoli fibrillar center(5)

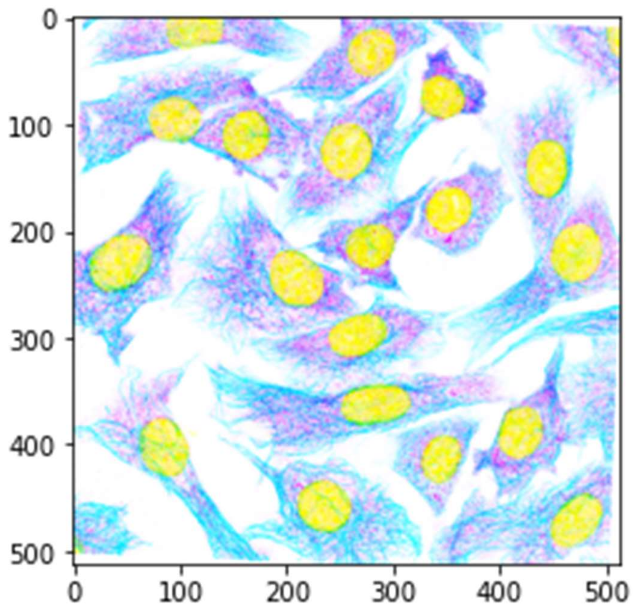


Fig - Cytosol (6)

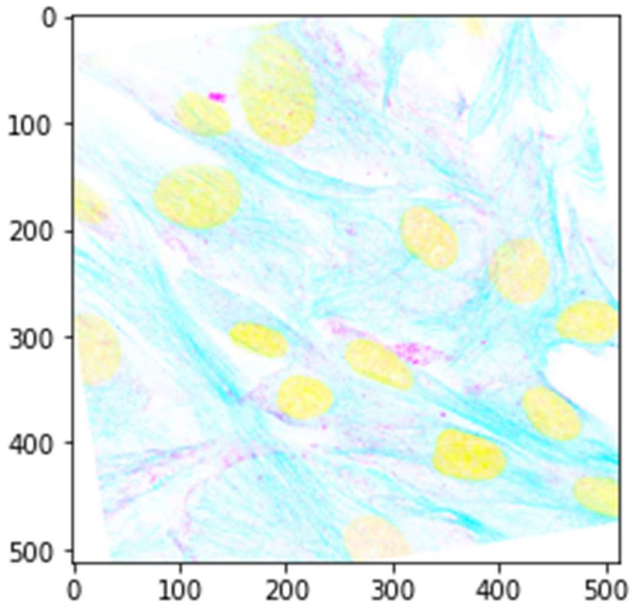


Fig - Plasma Membrane (7)

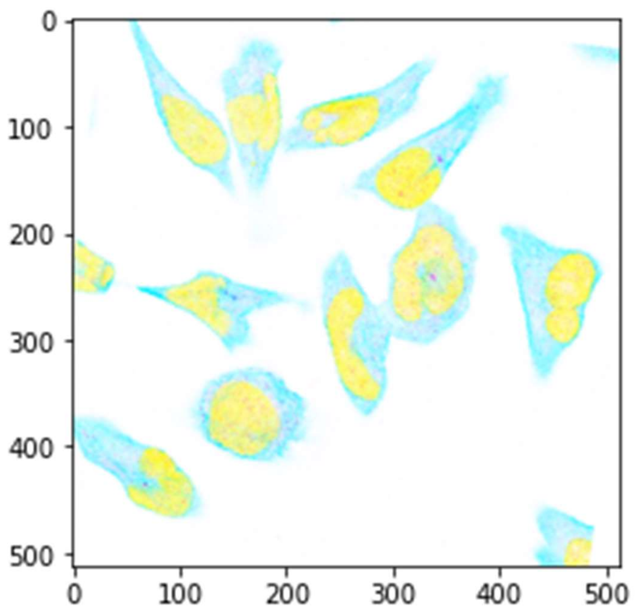


Fig - Centrosome (8)

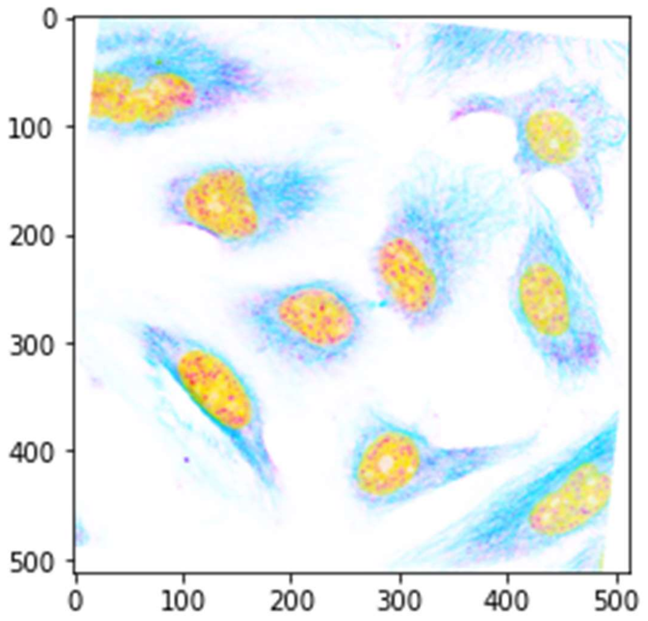


Fig - Nuclear Speckles (9)

4.3 Testing Approach

Testing is vital for the success of any software. No system design is ever perfect. Testing is also carried in two phases. First phase is during the software engineering that is during the module creation. Second phase is after the completion of software. This is system testing which verifies that the whole set of programs hanged together.

White Box Testing:

In this technique, the close examination of the logical parts through the software is tested by cases that exercise species sets of conditions or loops. All logical parts of the software checked once. Errors that can be corrected using this technique are typographical errors, logical expressions which should be executed once may be getting executed more than once and error resulting by using wrong controls and loops. When the box testing tests all the independent part within a module a logical decisions on their true and the false side are exercised, all loops and bounds within their operational bounds were exercised and internal data structure to ensure their validity were exercised once.

Black Box Testing:

This method enables the software engineer to device sets of input techniques that fully exercise all functional requirements for a program. Black box testing tests the input, the output and the external data. It checks whether the input data is correct and whether we are getting the desired output.

Unit Testing:

Each module is considered independently. It focuses on each unit of software as implemented in the source code. It is white box testing.

Implementation and Software Specification Testing:

Detailed Design of Implementation

This phase of the systems development life cycle refines hardware and software specifications, establishes programming plans, trains users and implements extensive testing procedures, to evaluate design and operating specifications and/or provide the basis for further modification.

Technical Design

This activity builds upon specifications produced during new system design, adding detailed technical specifications and documentation.

Test Specifications and Planning

This activity prepares detailed test specifications for individual modules and programs, job streams, subsystems, and for the system as a whole.

Programming and Testing

This activity encompasses actual development, writing, and testing of program units or modules.

User Training

This activity encompasses writing user procedure manuals, preparation of user training materials, conducting training programs, and testing procedures.

Acceptance Test

A final procedural review to demonstrate a system and secure user approval before a system becomes operational.

Installation Phase

In this phase the new computerized system is installed, the conversion to new procedures is fully implemented, and the potential of the new system is explored.

System Installation

The process of starting the actual use of a system and training user personnel in its operation.

Review Phase

This phase evaluates the successes and failures during a systems development project, and to measure the results of a new Computerized Tran system in terms of benefits and savings projected at the start of the project.

Development Recap

A review of a project immediately after completion to find successes and potential problems in future work.

Post-Implementation Review

A review, conducted after a new system has been in operation for some time, to evaluate actual system performance against original expectations and projections for cost-benefit improvements. Also identifies maintenance projects to enhance or improve the system.

THE STEPS IN THE SOFTWARE TESTING

The steps involved during Unit testing are as follows:

a. Preparation of the test cases.

b. Preparation of the possible test data with all the validation checks.

c. Complete code review of the module.

d. Actual testing done manually.

e. Modifications done for the errors found during testing.

f. Prepared the test result scripts.

The unit testing done included the testing of the following items:

1. Functionality of the entire module/forms.
2. Validations for user input.
3. Checking of the Coding standards to be maintained during coding.
4. Testing the module with all the possible test data.
5. Testing of the functionality involving all type of calculations etc.
6. Commenting standard in the source files.

After completing the Unit testing of all the modules, the whole system is integrated with all its dependencies in that module. While System Integration, We integrated the modules one by one and tested the system at each step. This helped in reduction of errors at the time of the system testing.

The steps involved during System testing are as follows:

1. Integration of all the modules/forms in the system.
2. Preparation of the test cases.
3. Preparation of the possible test data with all the validation checks.
4. Actual testing done manually.
5. Recording of all the reproduced errors.

6. Modifications done for the errors found during testing.
7. Prepared the test result scripts after rectification of the errors.

The System Testing done included the testing of the following items:

1. Functionality of the entire system as a whole.
2. User Interface of the system.
3. Testing the dependent modules together with all the possible test data scripts.
4. Verification and Validation testing.
5. Testing the reports with all its functionality.

After the completion of system testing, the next following phase was the Acceptance Testing. Clients at their end did this and accepted the system with appreciation. Thus, we reached the final phase of the project delivery.There are other six tests, which fall under special category. They are described below:

Peak Load Test: It determines whether the system will handle the volume of activities that occur when the system is at the peak of its processing demand. For example, test the system by activating all terminals at the same time.

Storage Testing: It determines the capacity of the system to store transaction data on a disk or in other files.

Performance Time Testing: it determines the length of time system used by the system to process transaction data. This test is conducted prior to implementation to determine how long it takes to get a response to an inquiry, make a backup copy of a file, or send a transmission and get a response.

Recovery Testing: This testing determines the ability of user to recover data or re-start system after failure. For example, load backup copy of data and resume processing without data or integrity loss.

Procedure Testing: It determines the clarity of documentation on operation and uses of system by having users do exactly what manuals request. For example, powering down system at the end of week or responding to paper-out light on printer. Human Factors Testing: It determines how users will use the system when processing data or preparing reports.

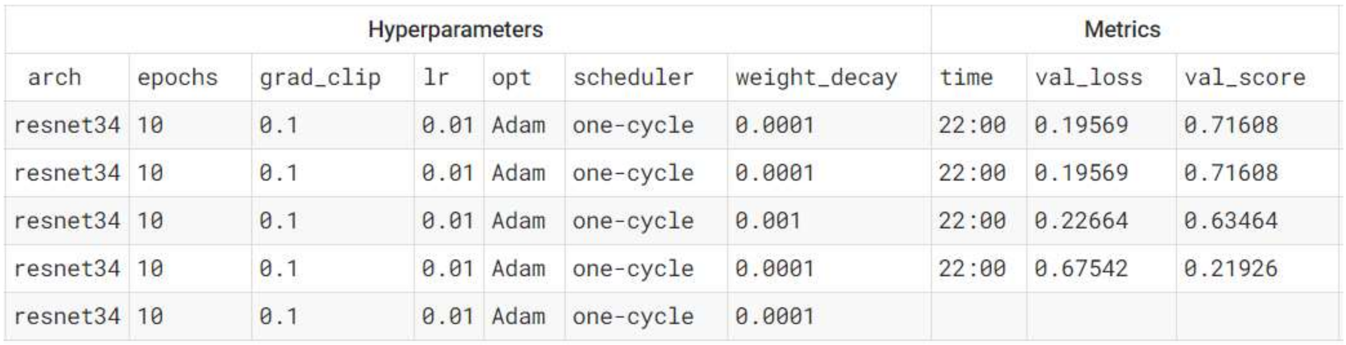
4.4 Test Case

Steps	Test Step	Test Data	Expected Result	Actual Result	Pass/Fail
1	Data collection	Different images collected through a research.	Data gathered Succesfull	Data gathered Succesful	Pass
2	Data preparation	Images and csv files are combined together to form a dataset. The images are normalized, resized, rotated,cropped,etc.	Reads successful	Reads Successful	Pass
3	Data Mining	Algorithms such as Logistic regression, ANN, CNN. Architectures such as Resnet 50 and Resnet 34.	Successfull y performed	Successfull y performed	Pass
4	Data Transfer	RawData converted into csv format	Successfull y converted	Successfull y converted	Pass

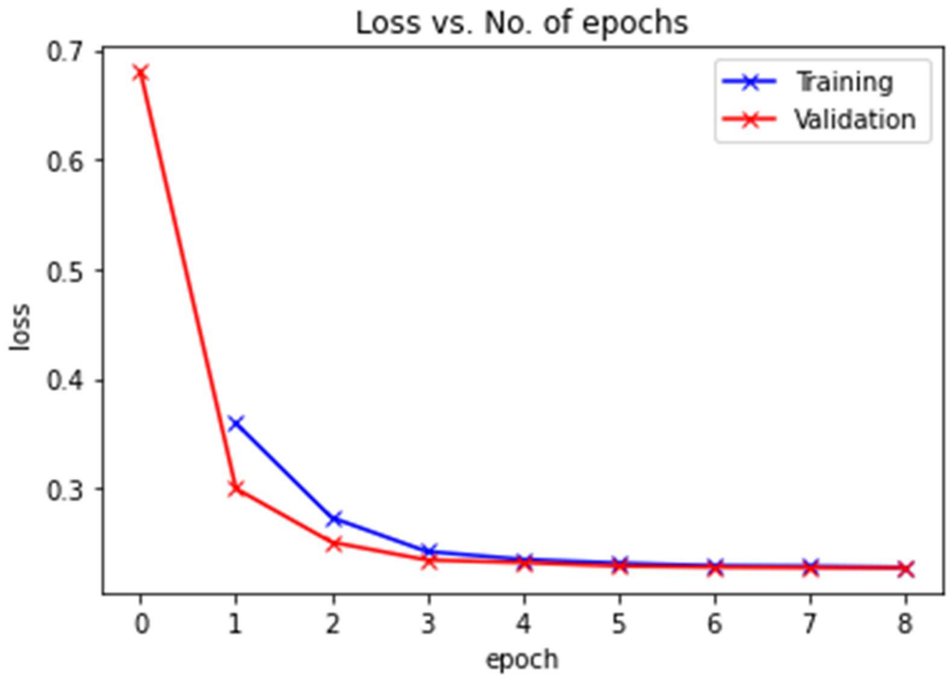
CHAPTER 5. CONCLUSION AND FUTURE WORK

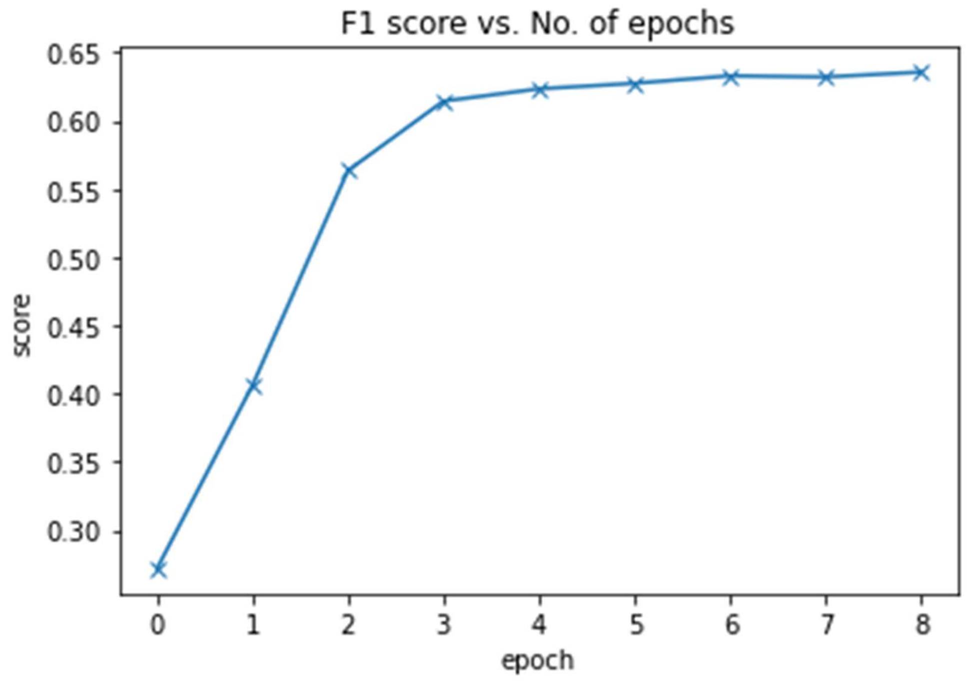
As mentioned above the dataset is collected from Human protein advanced. This company is a data analytics company use to solve problems related to biotechnology. It would be very much difficult for them to collect. I faced difficulties in combining the images and the csv files.

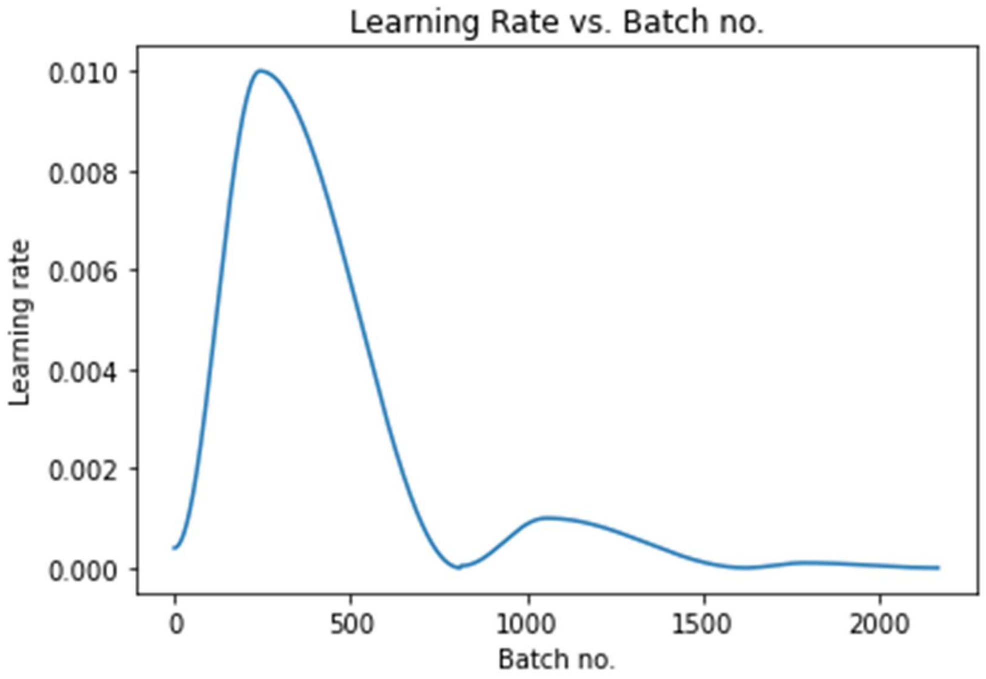
Still I have tried various combination of CNN models as well as pretrained models also tried transformation techniques but still not receiving good accuracy like 90 or above. I am still learning some new concepts to improve accuracy like ensemble learning techniques, Mask RCNN, Vgg net,etc.



Hyperparameters							Metrics		
arch	epochs	grad_clip	lr	opt	scheduler	weight_decay	time	val_loss	val_score
resnet34	10	0.1	0.01	Adam	one-cycle	0.0001	22:00	0.19569	0.71608
resnet34	10	0.1	0.01	Adam	one-cycle	0.0001	22:00	0.19569	0.71608
resnet34	10	0.1	0.01	Adam	one-cycle	0.001	22:00	0.22664	0.63464
resnet34	10	0.1	0.01	Adam	one-cycle	0.0001	22:00	0.67542	0.21926
resnet34	10	0.1	0.01	Adam	one-cycle	0.0001			







5.1. Limitation of the system

As mentioned above the proposed system has some limitations on which I'm studying that are MASK-RCNN, One Cycle Policy and many more.

CHAPTER 6. REFERENCES

[1] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[2] https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html#sphx-glr-beginner-blitz-neural-networks-tutorial-py

[3] https://jovian.ai/learn/deep-learning-with-pytorch-zero-to-gans