Honours Research Thesis

Multi-label Image classification on Imbalanced Data to predict mutational status in Rhabdomyosarcoma

Vivek Velivela – 12704286

Bachelor in computing Science (Honours)

Supervised by

Prof. Paul Kennedy and A/Prof. Daniel Catchpoole

December 6, 2021

Contents

List of Figures

List of Tables

Abstract

A significant amount of research has been done and many machine learning models have been developed to classify and identify major tumour types based on features extracted from histological images. Rhabdomyosarcoma (RMS) is the most common soft tissue cancer in children (Hiniker & Donaldson, 2015). It is important to understand the underlying molecular and mutational changes of RMS tumour to have a better understanding of the tumour. Various Genetically engineered mouse models (GEMM's) have been developed which resembles human Rhabdomyosarcoma tumour characteristics to analyse and examine the tumour behaviour. In this study, we trained a deep convolutional neural network (inception v3) with various weighted binary cross entropy loss functions on whole slide images of GEMM's obtained from children's cancer therapy development institute (cc-TDI) to predict various mutations present in each tumour image. We have developed a multi label classification model that can predict mutations among (Myf6, Pax7, Pax3, Cdkn2a, Trp53, PTC1), each tumour image can have more than one mutation involved. For training purposes, we only know the mutations involved in each WSI, we did not have any manual annotations on the image while most related research work on tumour classification requires manual region or nuclei segmentation. To predict the mutations involved in each WSI, we have divided each WSI into patches of size 256*256 at level 14 magnification.

We have used weighted binary cross loss function because of the high-class imbalance nature of our data. As one image can contain more than one mutation, on high level view over sampling or under sampling our dataset wouldn't change the distribution of sample per class. Conventional binary

cross entropy treats each class equally whereas weighted binary cross entropy assigns loss according to the overall positive and negative frequency of samples per class.

Keywords

Histology, Deep Learning, Rhabdomyosarcoma, Convolutional Neural Networks, Imbalanced data, Weighted Binary cross entropy.

1.Introduction

Rhabdomyosarcoma is one of the most common childhood soft tissue sarcoma and fourth most common paediatric solid tumour (Hiniker & Donaldson, 2015). There are various risk levels in treatment guidance of Rhabdomyosarcoma (RMS) tumour, these risk levels are based on various factors like tumour size, tumour invasion, patient age, tumour stage, disease site and histology of the tumour (Hiniker & Donaldson, 2015). Instigating biological underpinnings of Rhabdomyosarcoma has been an area of extensive research with promising results (Hiniker & Donaldson, 2015). There are various subtypes of RMS in which each subtype is driven by a certain mutation. Embryonal RMS (ERMS) is the most common subtype and Alveolar RMS (ARMS) is another subtype of RMS which is notoriously aggressive and genetically distinct disease (Kashi, et al., 2015). Both the subtypes have their own distinct features. Studies show that most ARMS express one of two oncogenic gene fusions: PAX3 or PAX7 with

FOXO1. PAX3 is the protein produced by chromosomal translocation (Hiniker & Donaldson, 2015).

Several research groups have utilised gene expression profiling, targeted sequencing, genetically engineered models to investigate RMS under the hypothesis that tumour biological signatures might improve risk stratification. Understanding the underlying molecular underpinnings would reveal the behaviour of the tumour which can be used for targeted treatment. Decades of targeted sequencing have led to the discovery of loss of heterozygosity, characteristic translocations involving PAX3-FOXO1, PAX7-FOXO1 have defined the genomic characteristics frequently associated with histological features of the disease (Shern, et al., 2015). So, Histological features have the necessary information that can reveal the mutations involved in a certain tumour image.

There has been a huge growth in applications of deep learning and convolutional neural networks (CNN) in histopathology assessment (Agarwal, et al., 2020). There are some state-of-the art deep learning models for breast cancer detection and many other challenges in histopathological imaging (Agarwal, et al., 2020). CNN's are responsible for extracting representative features from the image, these features can be used to train predictive models. Based on success driven by convolutional neural networks in medical image analysis we can use CNN's to extract features which can be used to predict the mutational status of a tumour image.

Each RMS tumour can be associated with more than one mutation which makes this a multi-label classification problem. CNN's models give good results with more data which made them data hungry but the data that is being sent into the model must be un-biased and balanced so that model

can treat each class equally and acquire good results. Based on most common and rare subtypes of RMS, different mutations are observed with various frequencies in each tumour which can lead to class imbalance where there are lot of samples with Trp53 and very less samples containing Cdkn2a.

In this paper, we propose a multi-label image classification model which can classify mutations present in each tissue image on a class imbalanced dataset with the help of modified weighted binary cross entropy loss function. We have used a weighted binary cross entropy loss function which can assign different loss to various under-represented and over-represented classes. This loss function helps the model to introduce bias towards under-represented classes by assigning higher loss whenever a misclassification happens.

The following chapters of this report are arranged as follows. A literature review on various computer aided techniques to predict mutations of various cancer types, techniques to handle class imbalanced data for multi label classification problems has been discussed in chapter 2. Chapter 3 and 4 contains hypothesis and research questions of this study. The proposed method has been featured in chapter 5. Chapter 6 contains details about experimental settings, information about dataset, evaluation metrics and results. Final chapter contains conclusion and future work.

2. Background and Related work

This chapter contains literature review of some of the related works to mutation prediction and background of the data that has been used in this

study. Chapter 2.1 discusses about the background work done by Children's Cancer Therapy Development Institute(cc-TDI) on developing Genetically engineered mouse models whose tissue images were crucial in this study. Chapter 2.2 contains an overview of some the existing methods, techniques for predicting various mutations in non-small lung cancer and other methods and techniques to predict genetic alterations from colorectal cancer. Chapter 2.3 discusses about techniques that deals with various data imbalance problems.

2.1 Dataset Background

Children's Cancer Therapy Development Institute (cc-TDI) has done extensive work on preparing this dataset. They have generated multiple Genetically engineered models of fusion positive alveolar RMS and fusion negative embryonal RMS and undifferentiated pleomorphic sarcoma subtype of NRSTS (C. Keller, viewed:2021). They have found out that Lymphatic and hematogenous metastasis is predominant feature and primary cause of mortality in these models. These models have been ensured to represent human diseases by histopathology, gene expression and other features (C. Keller, viewed:2021). They have also carefully characterized the GEM models with respect to tumour latency and growth rates. They have generated deep biological replicates of low-passage primary tumour cell cultures for chemical & genetic screens which can be used for drug discovery.

Alveolar Rhabdomyosarcoma GEM model is extensively characterized with respect to the natural history, histopathology, and molecular features. The ARMS GEM genotype is *Myf6Cre PAX3: FOXO1*$^{+}$ $p53^{NULL}$ which means from the fetal Myf6 cell lineage the pathognomonic PAX3:FOXO1

oncogene is activated and both copies of p53 tumour are conditionally deleted as shown in figure 1. This mouse model is bred into hairless background to improve bioluminescent imaging of the luciferase reporter gene that is conditionally activated in the tumour cells (C. Keller, viewed:2021).

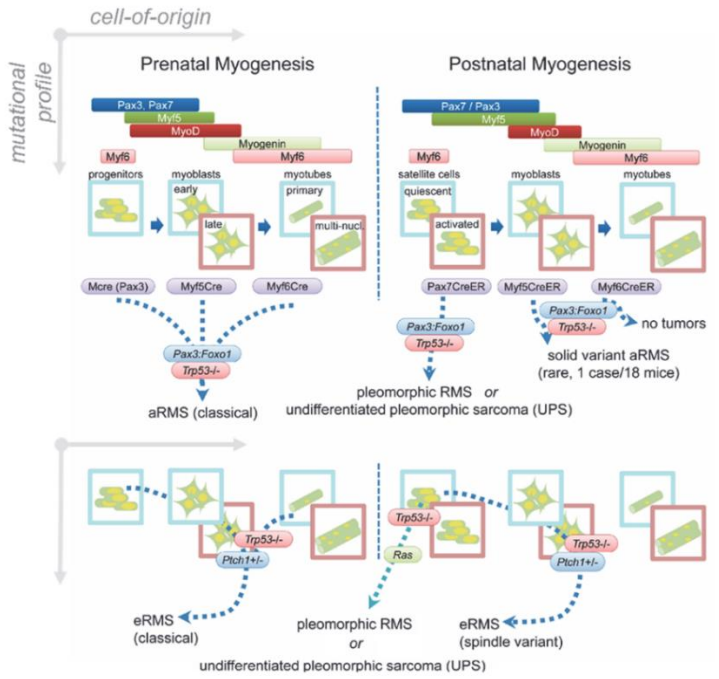


Figure 1: Conditional Mouse models of RMS and Non-RMS soft tissue sarcomas

cc-TDI has also performed preclinical therapeutic studies using small molecular inhibitors delivered by parenteral administration. They have assessed tumour growth by calipers, bioluminescent imaging.

2.2 Mutational Status Prediction and Data Pre-processing

Various researchers have proposed deep learning techniques to predict the mutational status of a tissue image. Authors from (Coudray, et al., 2018) have proposed a multi-label classification model using transfer learning from Inception v3 with RMSprop algorithm for optimization where each whole slide image is converted into 512*512 non-overlapping pixel

window at magnification of 20. To remove unnecessary features from images they also have removed all the patches where >80% of the surface is covered by background. Only Adenocarcinoma Whole slide images were used with training set of approx. 212,000 tiles from approx. 320 slides and testing set of approx. 44,000 tiles from approx. 62 slides. Out of all mutations TP53 and STK11 have highest allele frequency.

They have concluded that six frequently mutated genes seem predictable out of 10 mutations. It also has been concluded that low prevalent mutations like ALK haven't been predicted or detected by their model and high prevalent mutations like STK11, TP53 have high AUC's. The methods that have been used in this research are effective in terms of training time and computational power. A pattern can be observed in the results where model has produced high probability for frequent mutations and low probability for rare mutations. Although, there was no explicit technical reason mentioned in the paper for these kinds of results, class imbalance can be considered as one of the factors for this pattern in the results where frequent mutations like TP53, STK11 and EFGR are over-represented and mutations like ALK are under-represented.

Similarly, authors from (Jang, et al., 2020) have developed several binary classifiers to predict genetic alterations from colorectal cancer. The classifiers were built and tested on 629 The Cancer Genome Atlas (TCGA) CRC dataset and validated them with 142 Seoul ST Mary Hospital (SMH) CRC dataset. Only limited frequently mutated genes like APC, KRAS, PIK3CA, SMAD4 and TP53 were only selected for this study. The classifiers were trained with 360*360-pixel patches. The number of patients with respect to each mutation has been shown in the below table 1

Mutation	Number of patients
APC	436
KRAS	249
PIK3CA	133
SMAD4	74
TP53	340

Table 1: List of number of patients involved per mutation

They have also built simple CNN model with 12(5*5), 24(5*5) and 24(5*5) convolutional filters each followed by (2*2) max pooling layer a as a data pre-processing technique where they have built a tissue/non-tissue-based classifier for 360*360-pixel image patches at 20 times magnification to remove all the artefacts at once. They have used inception v3 pre trained CNN to train another classification model which can classify normal vs tumour tiles and have only used tiles more than 0.9 tumour probability. At last, they have used Inception v3 to train five different classifiers and validated with a patient level tenfold cross validation scheme.

While authors from (Coudray, et al., 2018) have performed image pre-processing with certain techniques like thresholding authors from (Jang, et al., 2020) have created two CNN models to classify tumour/normal and tissue/non-tissue respectively which proved that almost 99.9% contaminated patches were eliminated effectively. While authors from (Jang, et al., 2020) have developed an automated classifier to classify tissue/non-tissue images. The authors from (Jang, et al., 2020) haven't mentioned what kind of images are used to train that specific model and how much percent of tissue in a patch can be considered as a full-fledged

tissue patch by the model. Due to the opportunity provided by TCGA to reveal the genotype-phenotype relationship because of the extensive archives of digital pathology slides with multi-omics test results and because of difference in morphological features between the frozen and FFPE tissue WSI's, the authors had a chance to build a normal/tumour classifier which has helped to select specific patches with mutated tissue region in the whole slide level to be used for predicting mutations in further classifiers.

2.3 Techniques to deal with imbalanced data

There are two major ways to deal with data imbalance in multi label classification problem: data transformation and algorithm adaption (Charte, et al., 2015). Data transformation aims to produce a dataset or a group of datasets from a multi label dataset that can be processed with traditional classifiers. The objective of algorithmic adaption is to adapt the existing classification algorithms according to the data.

Authors from (Rezaei-Dastjerdehei, et al., 2020) have proposed a weighted cross entropy loss function in multi-label classification to address class imbalance. This is a kind of algorithmic adaption where the earning algorithm is reformed to detect minority class better, more penalty is considered for misclassifying minority class in loss function. A major problem in practice with conventional cross entropy loss functions is that they assign equal weights to all classes, thus equal loss is applied when a class is misclassified.

Weighted binary cross entropy partially solves the problem by introducing weight into the loss function as shown in below equation 1

$$WBCE = -(\beta * p * \log(\hat{p}) + (1 - p) * \log(1 - \hat{p}))\text{----------(1)}$$

$$p = Real\ label$$

$$\hat{p} = predicted\ label$$

$$\beta = Weight$$

First term is related to detection of positive label and second term is related to detection of negative label. Authors from (Rezaei-Dastjerdehei, et al., 2020) have also mentioned that values of β can affect the number of false positives and false negatives where β must be greater than 1 to decrease the number of false negatives and β must be less than 1 to decrease the number of false positives. They have conducted many experiments on COCO dataset with various β values. However, this loss function is addressing half of the problem where at a time where β must be either greater than 1 or less than 1 to reduce either false positives or false negatives respectively. This loss function does not address a criterion where the output must have least number of false positives and false negatives.

There are some constraints to the values of β where the given loss function gives a smaller number of false negatives only if the dataset contains majority of negative samples and minor positive samples and value of β is greater than 1.

Although authors from (Rezaei-Dastjerdehei, et al., 2020) have used a weighted binary cross entropy loss function to deal with data imbalance in an algorithmic level, authors from (Jang, et al., 2020) have mentioned that the deep learning model did not perform optimally when there was a huge data imbalance among colorectal mutational classes. Thus, the difference

in patient number between each mutation group is reduced by 1.4-fold through random sampling. To minimize model overfitting authors from (Jang, et al., 2020) have also used data augmentation including random rotations, random horizontal/vertical flipping and random perturbation of the contrast and brightness. Although most of the data augmentation techniques are useful, some of the techniques like image stretching and changing contrast, brightness might give a chance to elevate more unnecessary features which can be captured by the model affecting the model performance.

Both the techniques mentioned in (Rezaei-Dastjerdehei, et al., 2020) and (Jang, et al., 2020) can be potentially used to handle data imbalance but methods mentioned in (Rezaei-Dastjerdehei, et al., 2020) utilises all the available data whereas the methods in (Jang, et al., 2020) suggests performing under sampling at the beginning and oversampling to maintain balance among the classes to avoid overfitting. Methods from (Jang, et al., 2020) have provided a data level changes which might not take advantage of all the available representative data where some data might be lost in under sampling and some of the existing data is replicated in data augmentation.

3. HYPOTHESIS

Pre-trained neural networks with help of custom Binary cross entropy loss function can identify and predict the mutational status of Rhabdomyosarcoma tumours accurately within highly class imbalanced dataset.

4. Research Questions

- Can pre-trained convolutional neural networks predict the mutational status of RMS accurately?
- How does weighted loss function affects the prediction results of a model with highly class imbalanced dataset?

5. Methodologies and Proposed Method

In this chapter, I have provided information about the methods and techniques that have been used to create a machine learning model that can predict mutations involved in each RMS tumour image. I have adapted a similar approach from (Coudray, et al., 2018). A detailed description about image pre-processing techniques have been outlined in 5.1. A comprehensive specification about transfer learning and its uses have been

detailed in 5.2. Then I have outlined how weighted binary cross entropy function can be used to address data imbalance in 5.3.

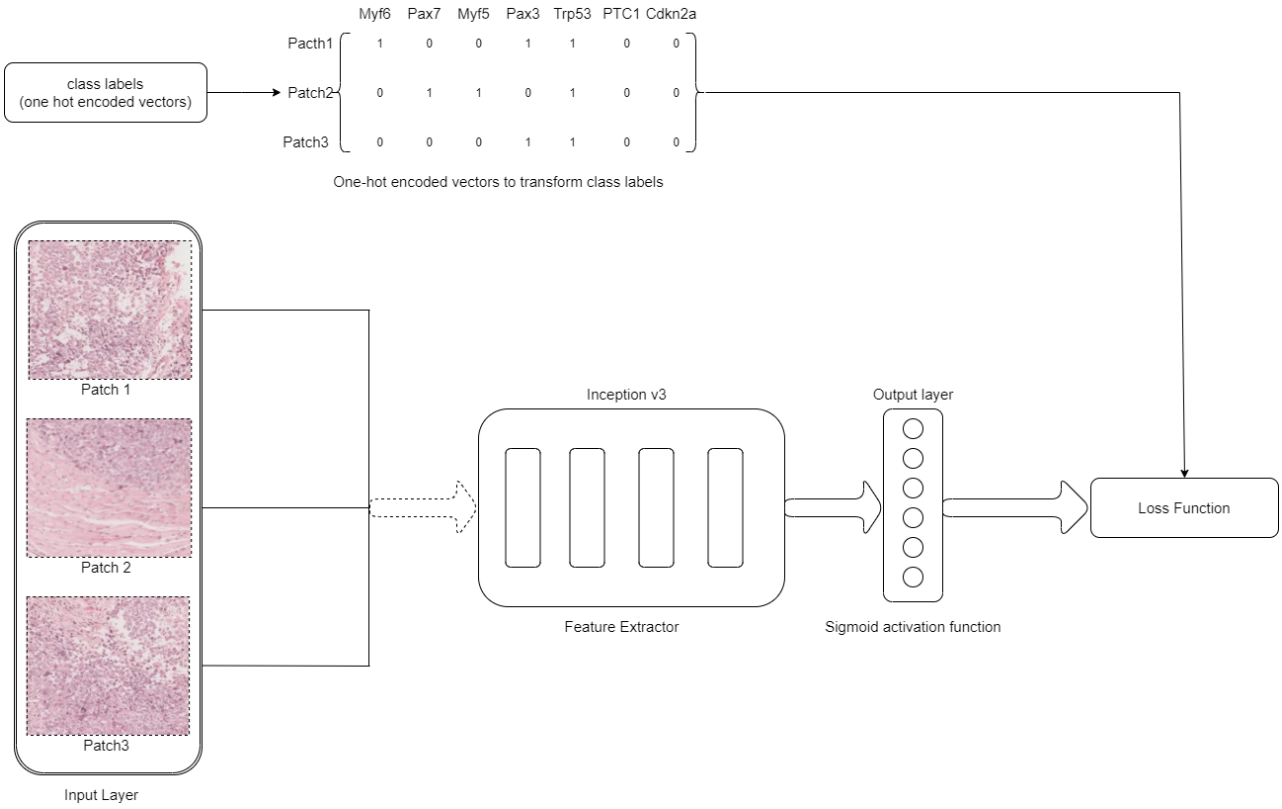


Figure 2: Training pipeline of the proposed method to predict mutational status of Rhabdomyosarcoma on multi-label imbalanced data.

The pipeline of the proposed method as shown in the figure 2 consist of several steps. Firstly, the input labels are converted from strings to one hot encoded vector. The input images will be associated with corresponding one hot encoders, these images will be sent into inception v3 model to extract important and necessary image features. The outputs predictions of these images are generated in the final layer which consist of Sigmoid activation function. Sigmoid activation function makes sure that the output prediction values are independent of each class (Each class probability

ranges between 0 to 1). Finally, the custom loss function is used to adjust the weights for back propagation.

5.1 Image Pre-processing Techniques

We had 124 whole slide tissue images in SVS format with an average of 30,000*30,000 pixels in width and height with different magnification levels. All the tissue images have white background in them. All the whole slide images are converted into 256*256 pixel of non-overlapping patches at magnification level 14 using open slide library so that important and necessary features of each patch can be learned by the model. To eliminate the whitespace, we have used binary thresholding technique with threshold of 225 where we have removed all the patches with more than 80% background. This process has produced approx. 120,000 patches where 20,000 patches have been removed due to image corruption, compatibility, readability issues. Patches with different stains and corrupted patches with ink stains and dust are also removed manually by hand.

The labels have been provided for each GEMM in a string format. All the labels are processed such that each patch has corresponding label associated with specific GEMM and all the labels are converted into one-hot encoded format which can be readable and interpretable by the model. Figure 3 shows the transformation of labels from string format to one-hot encoded format.

UID	Genotype
U21786	Mcre(Cre/WT) Pax3(P3Fm/P3Fm) Trp53(F2-10/WT)

Filename	Myf6	Pax7	Myf5	Pax3	Trp53	PTC1	Cdkn2a
U21786	0	0	0	1	1	0	0

**

Figure 3: Transformation of labels from string format to one-hot encoded.

5.2 Transfer Learning and Fine Tuning

We have performed transfer learning and fine tuning on Inception v3 as mentioned in (Coudray, et al., 2018). Inception v3 contains 11 stacks of inception modules where each module consists convolutional layers and pooling layers with rectified linear unit as activation function. Inception v3 contains different filter sizes to produce various feature maps. It avoids representational bottlenecks to define a clear information flow across the network (Szegedy, et al., 2016). All the experiments that we have conducted in this study have two results corresponding to transfer learning and fine tuning.

To maintain consistency, we have only used convolutional base of Inception v3 and added some dense layers with varied dense units per each layer with last dense layer containing dense units as per the number of mutations to be predicted. Convolutional base of inception v3 will act as feature extractor and all those features will be sent into newly added dense layers for training and final class prediction.

As last layers of pre-trained Inception v3 contains more generic features of various ImageNet dataset classes, Fine-Tuning unfreezes some of the last layers of incpetionv3 and training fine-tuned model will result in model

learning more generic features which are relevant to our dataset. We believe that fine-tuning inception v3 networks according to our data gives better results with respect to each class.

5.3 Custom Weighted Binary Loss Function

Conventional binary cross entropy loss function doesn't have any type of weights involved which means any kind of misclassification done by the model will be allocated same loss disregarding underrepresented and overrepresented class. The formula of conventional binary cross entropy is shown in equation 2

$$BCE = -(p * \log(\hat{p}) + (1-p) * \log(1-\hat{p})) \text{ ----------(2)}$$

$$p = Real\ label$$

$$\hat{p} = predicted\ label$$

Authors from (Rezaei-Dastjerdehei, et al., 2020) suggested a weighted cross entropy function which incorporates only one weight which is static with no mathematical formulation behind the selection of a specific weight whereas, the proposed custom Weighted Binary loss function consists of two separate weights. This loss function introduces weight on the cost of misclassifying a positive sample and another weight for misclassifying a negative sample. Weights in this loss function are allocated dynamically based on the inverse frequency of positive and negative samples in each class. The formula of the custom weighted binary cross entropy is shown in equation 3

$$CWBCE = -(\gamma^{+}(p * \log(\hat{p})) + (\gamma^{-} * (1-p) * \log(1-\hat{p}))) \text{ ---------(3)}$$

$$p = Real\ label$$

$$\hat{p} = predicted\ label$$

$$\gamma^{+} = positive\ weight,\ \gamma^{-} = positive\ weight$$

Positive weight is the inverse frequency of number of positive examples in a class ($\frac{Total\ number\ of\ samples}{Number\ of\ positive\ examples\ in\ a\ class}$). Negative weight is the inverse frequency if negative examples in a class ($\frac{Total\ number\ of\ samples}{Number\ of\ negative\ examples\ in\ a\ class}$). The below example gives a basic intuition on the working of custom binary loss function and compares the losses allocated using three loss functions.

EXAMPLE: Assume a multilabel classification problem where the prediction contains two classes with the statistics as shown in table 2

Class	Myf6	Pax7	Total
Number of positive examples	22229	9744	68929
Number of negative examples	46700	59185	68929

Table 2: List of positive and negative examples in Myf6 and Pax7

From table 2 we can see that Pax 7 is underrepresented in terms of number of positive samples.

Now let's assume the below two scenarios where model misclassifies on positive sample and negative sample of both classes.

The weights of each class for custom weighted binary cross entropy are shown in table 3

Class	Myf6	Pax7
Positive Weight (γ^+)	68929/22229=3.100	68929/9744=7.07
Negative Weight (γ^-)	68929/46700=1.47	68929/59185=1.16

Table 3: List of positive and negative weights of Myf6 and Pax7

SCENARIO 1: Model misclassifying a positive example of both classes.

Table 4 depicts the truth and predicted values of each class

Class	Myf6	Pax7
True Value	1	1
Predicted Value	~0 = 1e-7	~0 = 1e-7

Table 4: The ground truth and predicted values of Myf6, Pax7 in misclassifying positive examples

Calculating loss with Custom Weighted Binary cross entropy (CWBCE):

From equation 3 the loss of each class will be

Loss of Myf6 = -(3.100(1) *(log(1e-7)) + (1-1) *(1.47) *log (0.9999999)) = 49.960*

Loss of Pax7 =-(7.07(1) *log (1e-7) + (1-1) *1.16* log (0.9999999)) = 113.95*

Total Loss = average of Loss of Myf6 and Loss of Pax7 = 81.95

Calculating loss with Weighted Binary cross entropy (WBCE):

Let's assume the weight (β) be 2 then the loss of each class will be from equation 1 will be

Loss of Myf6 = -(2(1) *(log(1e-7)) + (1-1) *log (0.9999999)) = 32.23*

Loss of Pax7 =-(2(1) *log (1e-7) + (1-1) * log (0.9999999)) = ~32.23*

Total Loss = average of Loss of Myf6 and Loss of Pax7 = *32.23*

Calculating loss with Binary cross entropy (BCE):

Let's calculate the loss of each class using the equation 2

*Loss of Myf6 = -((1) *(log(1e-7)) + (1-1) *log (0.9999999)) = 16.11*

*Loss of Pax7 =-((1) *log (1e-7) + (1-1) * log (0.9999999)) = 16.11*

Total Loss = average of Loss of Myf6 and Loss of Pax7 = 16.11

SCENARIO 2: Model misclassifying a negative example of both classes

Table 5 depicts the truth and predicted values of each class

Class	Myf6	Pax7
True Value	0	0
Predicted Value	~1 = 0.9999999	~1 = 0.9999999

Table 5: The ground truth and predicted values of Myf6, Pax7 in misclassifying negative examples

Calculating loss with Custom Weighted Binary cross entropy (CWBCE):

From equation 3 the loss of each class will be

Loss of Myf6 = -(3.100(0) *log (0.9999999) + (1-0) *(1.47) *log (1e-7)) = 23.69*

Loss of Pax7 =-(7.07(0) *log (0.9999999) + (1-0) *1.16* log(1e-7)) = 18.69*

Total Loss = average of Loss of Myf6 and Loss of Pax7 =21.19

Calculating loss with Weighted Binary cross entropy (WBCE):

Let's assume the weight (β) be 2 then the loss of each class will be from the equation 1 will be

Loss of Myf6 = -(2(0) *log (0.9999999) + (1-0) *log (1e-7)) = 16.11*

Loss of Pax7 =-(2(0) *log (0.9999999) + (1-0) * log(1e-7)) = 16.11*

Total Loss = average of Loss of Myf6 and Loss of Pax7 = 16.11

Calculating loss with Binary cross entropy (BCE):

Let's calculate the loss of each class using the equation 2

*Loss of Myf6 = -((0) *log (0.9999999) + (1-0) *log (1e-7)) = 16.11*

*Loss of Pax7 =-((0) *log (0.9999999) + (1-0) * log(1e-7)) = 16.11*

Total Loss = average of Loss of Myf6 and Loss of Pax7 = 16.11

Class	Myf6	Pax7	Total
Misclassifying positive samples	CWBCE: *49.960* WBCE:32.23 BCE:16.11	CWBCE: *113.95* WBCE: 32.23 BCE: 16.11	CWBCE: *81.95* WBCE: 32.23 BCE: 16.11
Misclassifying negative samples	CWBCE: *23.69* WBCE: 16.11 BCE: 16.11	CWBCE: *18.69* WBCE: 16.11 BCE: 16.11	CWBCE: *21.19* WBCE: 16.11 BCE: 16.11

Table 6: The comparison of all the loss values is defined in below table

Based on the results in table 6 which indicates the equal loss allocation to any type of misclassification by BCE, it is evident that binary cross entropy is non-partial to all the classes. As the weighted binary cross entropy doesn't have another weight for misclassifying negative samples, it is semi-partial in misclassifying positive samples, and it shares similar behaviour with binary cross entropy in misclassifying the negative samples. custom weighted binary cross entropy is dynamically partial based on the frequency of each label type occurrence.

6. Experiments

In this chapter, I have provided detailed information about experiments that were conducted as part of this thesis. Firstly, detailed description about dataset has been provided in chapter 6.1. Experimental setup was explained in chapter 6.2. Chapter 6.3 outlines about several evaluation metrics like precision, recall and f1-score that were used to verify the performance of the model. Finally, discussion about all the experiments were conducted in chapter 6.4.

6.1 Dataset

All the experiments that were conducted as part of this study have utilised data that has been provided by Children's Cancer Therapy Development Institute(cc-TDI). The dataset contains 126 whole mount images involving 38 genetically engineered mouse models (GEMM's) where only 124 whole mount images involving 35 GEMM's were used due image, stain, label corruption issues. Totally seven mutations are involved in the entire dataset namely (Pax7, Pax3, Cdkn2a, PTC1, Trp53, Myf5, Myf6), each GEMM has almost 3 snaps on an average where tissue image belongs to more than one mutation.

Each whole mount tissue image is converted into small patches where each whole mount has produced 1000 patches on an average which combined assembled a dataset of approx. 120,000 patches where approx. 20,000 further patches were further removed due to image corruption issues. The whole dataset was split into training set, test set, and validation set with split percentage of 70%, 15%,15% respectively. Figure (4,5,6) contains

partition information with respect to each class among train, test and validation sets.

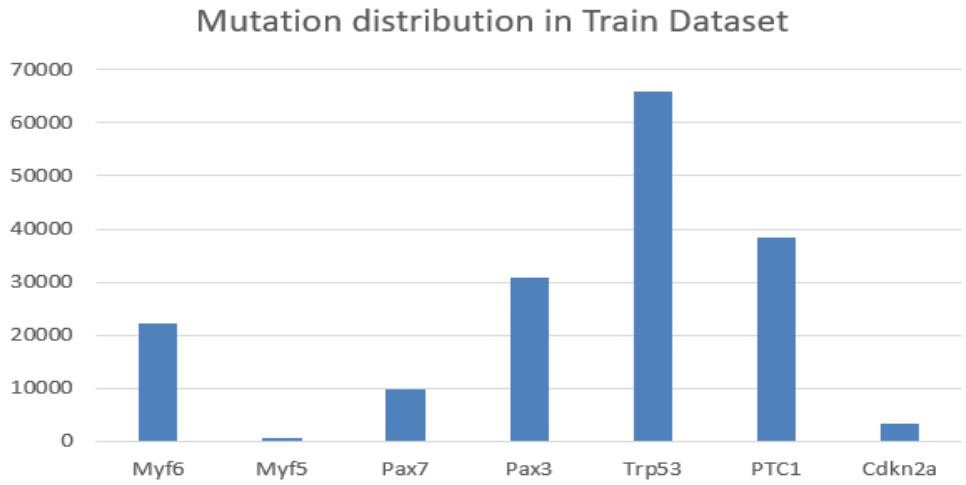


Figure 4: Distribution of each class in train dataset

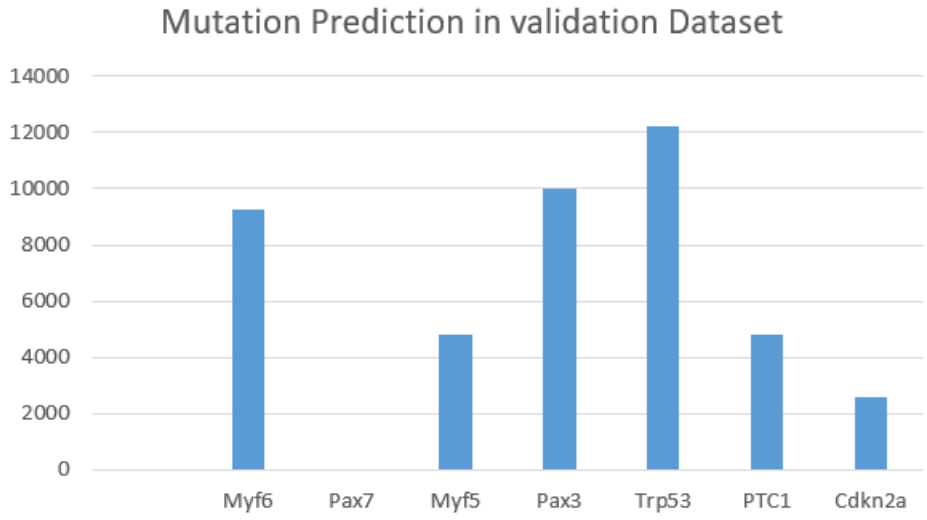


Figure 5: Distribution of each class in validation dataset

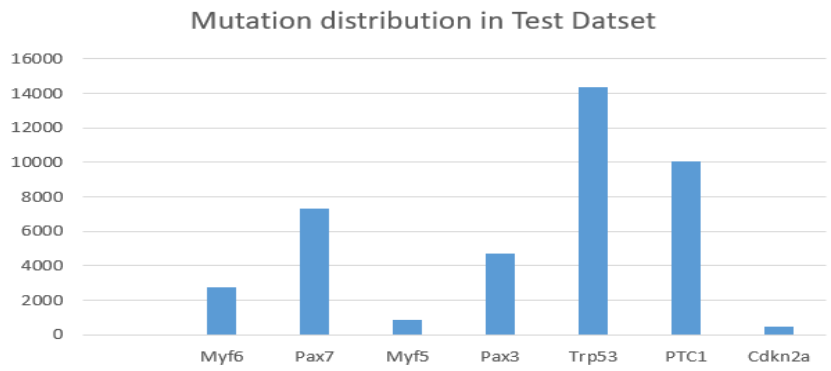


Figure 6: Distribution of each class in test dataset

Although Each mouse model has more than one snap there has been no data leakage among train, test, validation (We have confined all the images corresponding to a particular GEMM to one of the three sets). Figure 7 shows number of patches produced by each genetically engineered mouse model over train, test and validation sets and figure 8 is a pie chart depicting the distribution of patches per mutation in the whole dataset.

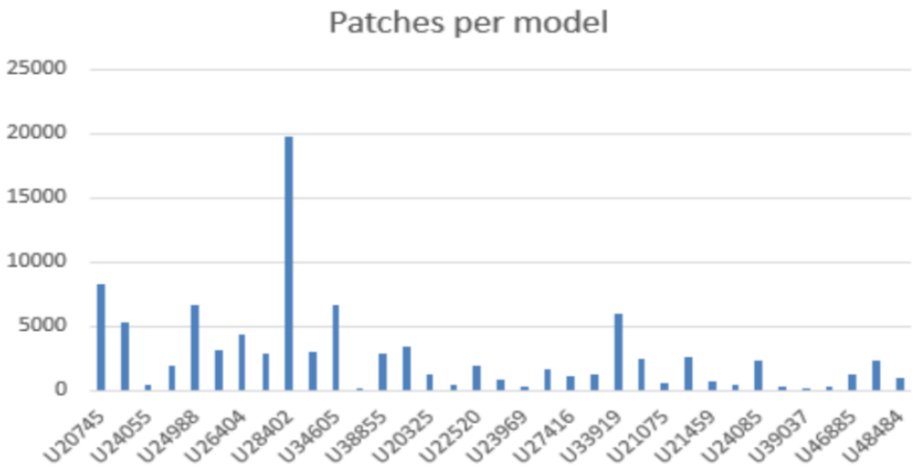
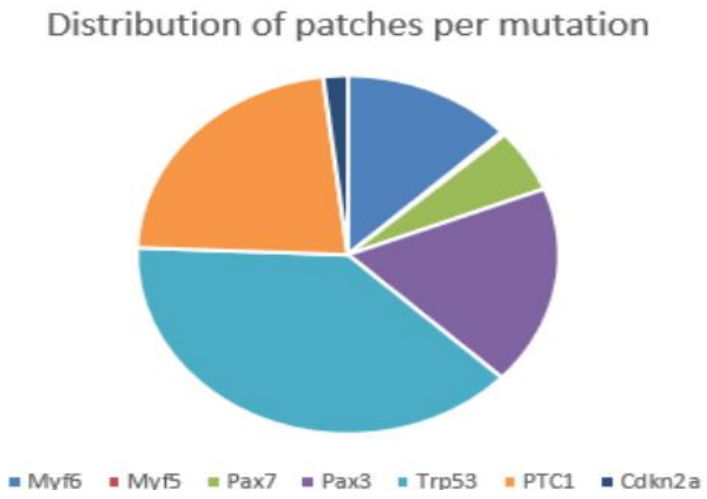


Figure 7:Histogram distribution showing number of patches per GEMM



~~FIG : Histogram distribution of patches~~ Figure 8: Pie chart distribution of showing the volume of patches per mutation

Figure 9 shows the distribution of patches among train, test, validation with Respect to each genetically engineered mouse model. These plots help to understand how each set has been divided.

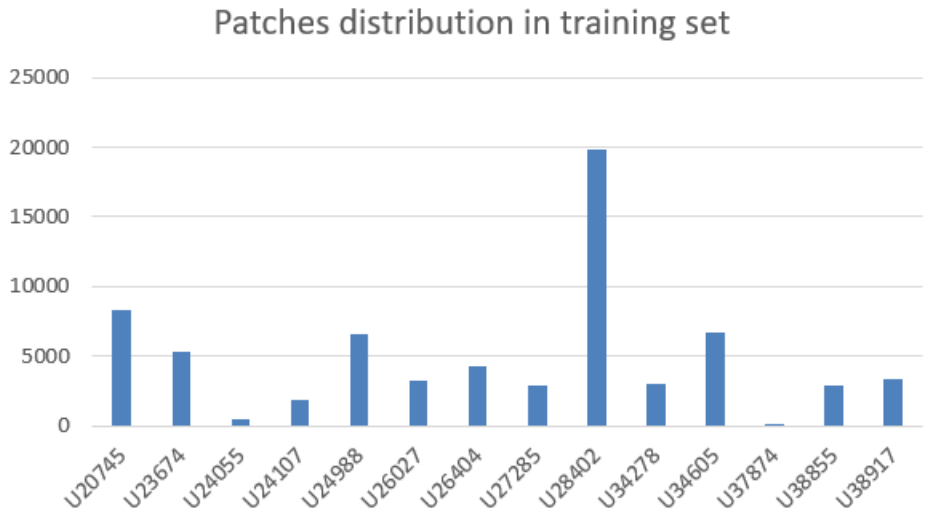


Figure 9: Patches Distribution in train set with respect to each GEMM.

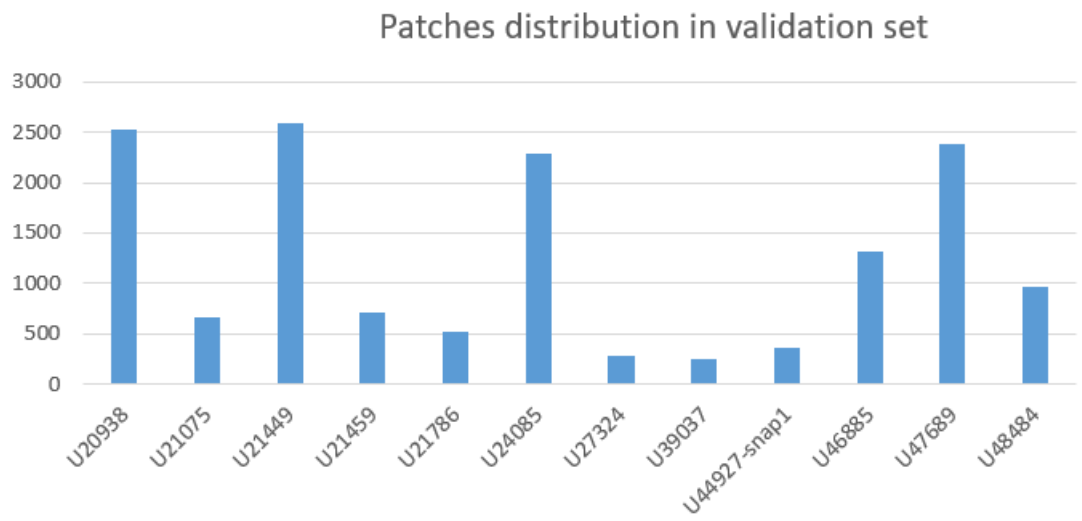


Figure 10：Patches distribution in validation set with respect to each GEMM.

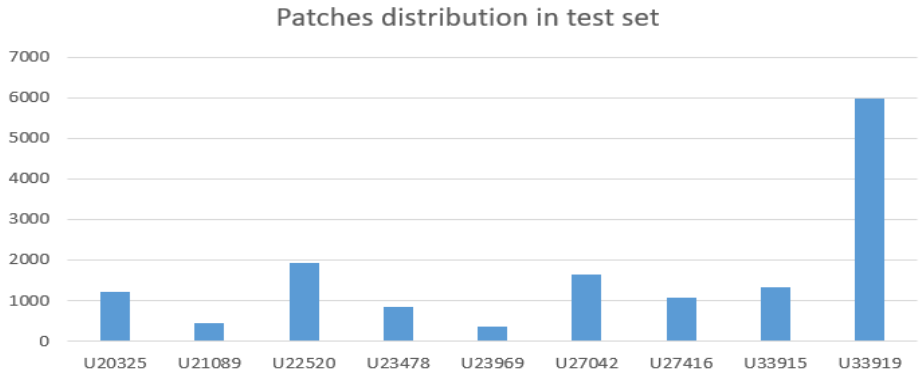


Figure 11： Patches Distribution in test set with respect to each GEMM.

6.2 Experimental Setup

We have performed various experiments with different parameters. Each of the experiments has highly class imbalanced data as input. To maintain consistency, All the experiments have utilised inception v3 convolutional base as feature extractor by using ImageNet weights (Coudray, et al., 2018). Compared to other CNN architectures Inception v3 possess lower computational cost which makes it suitable for big data scenario (Szegedy, et al., 2016). For fine-tuning, to maintain consistency over all experiments,

we have trained last 4 out of 11 blocks of Inception v3. We have attached custom fully connected layers as stated below

> CONV_BASE → FLATTEN→DENSE (1024, Relu) → DROPOUT (0.3)→DENSE (512, Relu) → DROPOUT (0.1)→DENSE (256, Relu)→ DENSE (128, Relu)→DENSE (Number of classes, sigmoid)

As the data is highly imbalanced, dropout layers have been added to avoid overfitting and some other dense layers are also added with gradually reducing number of neurons to make the weights adjust gradually from convolutional base to last dense layers.

As there are different levels of class imbalance among various classes in the dataset, almost all our experiments are run on only two of the imbalanced classes to observe, interpret and understand the results of each class.

We have experimented with three different loss functions where one of them is conventional binary cross entropy and two are weighted binary cross entropy, the difference between the other two loss function is allocation of weights. We have used Adam optimizer with learning rates of 0.0001 and 0.00001 for transfer learning and fine-tuning respectively.

6.3 Evaluation Metrics

Each experiment that has been conducted in this study has been evaluated by some of the standard and widely used evaluation metrics in machine learning domain. The metrics used are accuracy, precision, recall and F1 Score. Accuracy is most useful metric when the data is balanced among all

the classes. F1 score is harmonic mean of precision and recall, this metric can be used as an alternative to accuracy when the data among all the classes are imbalanced.

Recall or Sensitivity is the proportion of real positive cases that are correctly predicted positive (Powers, 2020). It helps to understand the proportion of positive samples which are identified correctly as positives (Carter, et al., 2016). It is also called as true-positive rate.

Precision or specificity is the proportion of predicted positive cases that are correctly real positives (Powers, 2020). It helps to understand the proportion of negative samples which are correctly identified as negatives (Carter, et al., 2016). It is also called true negative rate. The formulas of Recall, precision, F1-score, and accuracy are shown in equation (4,5,6,7) respectively based on table 7

	Actual Positive	Actual Negative
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

Table 7: Confusion Matrix

$$Recall = \frac{True\ Positives}{(True\ Positives + False\ Negatives)} \text{ ---------(4)}$$

$$Precision = \frac{True\ Positives}{(True\ Positives + False\ Positives)} \text{ -------(5)}$$

$$F1\text{- }Score = 2 * \frac{Precision * Recall}{Precision + Recall} \text{ ----------(6)}$$

Accuracy is the proportion of true samples from total number of samples. The formula of Accuracy is shown in below EQUATION

$$Accuracy = \frac{True\ Positives + True\ Negatives}{(True\ Positives + False\ Positives + True\ Negatives + False\ Negatives)} \quad (7)$$

Receiver Operator Characteristic (ROC) is a probability curve that plots true positive rate against false positive rate at various threshold values. Area Under the Curve (AUC) of an ROC curve is sed to quantify the overall ability of a test to discriminate between two outcomes (Carter, et al., 2016). This is a suitable metric for binary classification where the possible outcome is a probability between [0,1]. Higher AUC demonstrates good performance of the model at distinguishing between positives and negatives. A perfect model would have AUC as 1 which means no false positives and no false negatives. When AUC is between 0.5 and 1.0, there is a higher chance that the model can distinguish positive class values from the negative class values.

6.4 Results and Discussions

Each experiment was validated using several metrics. Frist, we have conducted experiment on all classes with custom weighted binary cross entropy (CWBCE) and based on the biased results as shown in figure 12 and figure 13 we have decided to test the CWBCE loss function by conducting experiments on only two mutational classes with class imbalance to observe and understand the behaviour of loss function with respect to each class. The classification report for each of the experiments and can be seen in Table 8

Experiment Type	Transfer Learning Results	Fine Tuning Results
Experiment with Binary Cross Entropy	precision recall f1-score support Myf6 0.19 0.56 0.29 2748 Pax7 0.92 0.32 0.48 7307	precision recall f1-score support Myf6 0.19 0.56 0.29 2748 Pax7 0.92 0.32 0.48 7307
Experiment with weighted Binary Cross Entropy(w=4)	precision recall f1-score support Myf6 0.18 0.60 0.28 2748 Pax7 0.92 0.37 0.53 7307	precision recall f1-score support Myf6 0.20 0.56 0.29 2748 Pax7 0.92 0.34 0.50 7307
Experiment with weighted Binary Cross Entropy(w=8)	precision recall f1-score support Myf6 0.17 0.61 0.27 2748 Pax7 0.89 0.45 0.60 7307	precision recall f1-score support Myf6 0.19 0.61 0.29 2748 Pax7 0.91 0.44 0.59 7307
Experiment with weighted Binary Cross Entropy(w=16)	precision recall f1-score support Myf6 0.20 0.72 0.31 2748 Pax7 0.86 0.48 0.62 7307	precision recall f1-score support Myf6 0.20 0.60 0.30 2748 Pax7 0.89 0.43 0.58 7307
Experiment with custom weighted Binary Cross Entropy	precision recall f1-score support Myf6 0.18 0.62 0.28 2748 Pax7 0.82 0.53 0.65 7307	precision recall f1-score support Myf6 0.19 0.62 0.30 2748 Pax7 0.90 0.47 0.61 7307

Table 8: Depicting results of all experiments that were conducted as part of this study

	precision	recall	f1-score	support
Myf6	0.19	0.58	0.28	2748
Pax7	0.95	0.24	0.39	7307
Myf5	0.25	0.40	0.31	850
Pax3	0.39	0.75	0.51	4744
Trp53	0.97	0.99	0.98	14380
PTC1	0.79	0.62	0.70	10085
Cdkn2a	0.13	0.10	0.11	449
micro avg	0.64	0.69	0.66	40563
macro avg	0.52	0.53	0.47	40563
weighted avg	0.78	0.69	0.68	40563
samples avg	0.65	0.70	0.66	40563

Figure 12: Results of all classes on pre-trained model on CWBCE

	precision	recall	f1-score	support
Myf6	0.22	0.60	0.32	2748
Pax7	0.94	0.35	0.51	7307
Myf5	0.17	0.34	0.23	850
Pax3	0.46	0.77	0.58	4744
Trp53	0.97	0.99	0.98	14380
PTC1	0.83	0.64	0.73	10085
Cdkn2a	0.07	0.04	0.05	449
micro avg	0.68	0.71	0.70	40563
macro avg	0.52	0.53	0.48	40563
weighted avg	0.79	0.71	0.71	40563
samples avg	0.69	0.73	0.70	40563

Figure 13: Results of all classes on fine-tuned model on CWBCE

	precision	recall	f1-score	support
Myf6	0.19	0.58	0.28	2748
Pax7	0.90	0.32	0.47	7307
Myf5	0.85	0.06	0.10	850
Pax3	0.38	0.72	0.50	4744
Trp53	0.97	1.00	0.98	14380
PTC1	0.77	0.47	0.58	10085
Cdkn2a	0.08	0.01	0.02	449
micro avg	0.64	0.65	0.65	40563
macro avg	0.59	0.45	0.42	40563
weighted avg	0.77	0.65	0.66	40563
samples avg	0.66	0.67	0.65	40563

Figure 14: Results of all classes on pre-trained model on BCE

	precision	recall	f1-score	support
Myf6	0.21	0.53	0.30	2748
Pax7	0.92	0.40	0.56	7307
Myf5	0.58	0.04	0.08	850
Pax3	0.44	0.71	0.55	4744
Trp53	0.97	0.99	0.98	14380
PTC1	0.81	0.59	0.68	10085
Cdkn2a	0.10	0.04	0.05	449
micro avg	0.70	0.69	0.70	40563
macro avg	0.58	0.47	0.46	40563
weighted avg	0.79	0.69	0.70	40563
samples avg	0.72	0.70	0.70	40563

Figure 15: Results of all classes on fine-tuned model on BCE

Firstly, figure 14 and 15 depicts the results of multi label classification model with binary cross entropy, a pattern can be observed where the results are skewed according to the number of samples in each group (Support in figure 12 depicts number of samples in each class) where Trp53 with the greatest number of classes have highest score and Cdkn2a which has lowest number of samples have lowest score. These results convey class imbalance problem in the dataset. So, we have experimented with custom weighted binary cross entropy loss function (CWBCE) to check the difference in performance.

Figures 12 and 13 depicts the results of multi label classification model with custom weighted binary cross entropy loss (CWBCE), a similar pattern of class imbalance results can be observed where under-represented classes have lowest score but the model with CWBCE has produced slightly better f1-score results with under-represented classes like Cdkn2a and Myf5 compared to model with BCE which shows that CWBCE has the potential to improve results of underrepresented classes by a slight chance.

The table 8 results show that pre-trained convolutional neural networks can accurately predict the over-represented mutations involved in certain patch.

The over-represented class has more positive examples and the under-represented have more negative examples. Among results in the above table a pattern can be observed where precision of under-represent class is lower than precision of over-represented class and recall of over-represented class is lower than recall of under-represented class, this phenomenon is because the model has biased itself into predicting positive examples of over-represented class and negative examples of under-represented class, thus justified according to the given formulas of precision and recall in equation (4,5) justifies the imbalance in precision and recall. The results of custom loss function are bit interesting where the recall of the over-represented class improved while the recall of under-represented classes stayed the same which is different to other experiments. Although the precision of each class almost remained the same or decreased across all the experiments, the custom weighted binary loss function has improved the results with respect to each class compared to remaining experiments. Improved results can be seen with both weighted binary cross entropy and custom weighted binary cross entropy compared to conventional binary cross entropy due to the added weights which tends the model to accurately predict more positive samples.

Experiment Type	Avg. AUC of Transfer Learning	Avg. AUC of Fine Tuning
Experiment with Binary Cross Entropy	75.85	74.5
Experiment with weighted Binary Cross Entropy(w=4)	74.3	74.5
Experiment with weighted Binary Cross Entropy(w=8)	75	77.1
Experiment with weighted Binary Cross Entropy(w=16)	75.42	78.2
Experiment with custom weighted Binary Cross Entropy	75	78.09

Table 9: Depicting Validation AUC results of experiments with two classes that were conducted as part of this study

Table 9 contains all the validation AUC results from the experiments that have been conducted on two classes. As shown in table 9 the results of all the experiments range from AUC 74 – AUC 78, the performance of model on overrepresented class might have increased the average AUC of the model. In our study, we have not considered accuracy of the model as a valid evaluation metric because of the nature of the data. Accuracy would be useful and necessary when there is less or no class imbalance.

7. Conclusion and Future Work

Based on the current results, it can be concluded that the experiment which was conducted using custom weighted binary loss function produced slightly better results compared to other experiments. The usual weighted binary cross entropy has also helped to produce some improved recall results due to the partial weighting. The data imbalance in this study played a crucial part in producing the skewed results as shown in table 8. To tackle this kind of class imbalance, various data level methods like random under and over sampling in conjunction to algorithmic level changes would present very good results. The loss functions that have been used in this study are solving intra class imbalance (positive and negative examples within each class) where the inter class imbalance still exist, some problem transformation techniques like label powerset where each set of unique labels can be considered as single label can help eliminate the inter class imbalance. Some automated data pre-processing techniques like image segmentation to eliminate the background more efficiently would have improved the results. Training one single multi label classification model for predicting all the mutations with custom weighted binary cross entropy loss function would be challenging as the losses with respect to the custom weighted BCE are very unstable for

each class because of the dynamic nature of the function. Thus, the model must adjust itself with respect to cost function which is the average of highly unstable losses of all the classes. Perhaps building binary classifiers for each class to check whether each mutation is present in a particular patch and combining result from each classifier to produce a final output might produce some good and decent results.

8. Acknowledgement

This research was supported by Children's Cancer Therapy Development Institute(cc-TDI). I specifically acknowledge the assistance of Charles Keller and Arthur Frankel for providing me with the necessary and required data and assisting me with various data related queries which were crucial part of this study.

This research was supported in part by the iHPC facility at UTS. I specifically acknowledge the assistance of Dr. Matthew Gaston for providing me the access to computing resources to run experiments.

This research was supported in part by Open slide, py-wsi opensource library and code repository which were used for image pre-processing techniques used in this study.

9. Ethics Declaration

This project was reviewed and approved by The Sydney Children's Hospital Network Human Research Ethics Committee (LNR/18/SCHN/195) and subsequently ratified by The University of Technology Sydney Human Research Ethics Committee (ETH21-6199).

10. Bibliography

Agarwal, S. et al., 2020. *Rhabdomyosarcoma Histology Classification using Ensemble of Deep Learning Networks.* s.l., s.n., p. 1–10.

Carter, J. V., Pan, J., Rai, S. N. & Galandiuk, S., 2016. ROC-ing along: Evaluation and interpretation of receiver operating characteristic curves. *Surgery*, Volume 159, p. 1638–1645.

Charte, F., Rivera, A. J., del Jesus, M. J. & Herrera, F., 2015. Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing*, Volume 163, p. 3–16.

Coudray, N. et al., 2018. Classification and mutation prediction from non–small cell lung cancer histopathology images using deep learning. *Nature medicine*, Volume 24, p. 1559–1567.

Hiniker, S. M. & Donaldson, S. S., 2015. Recent advances in understanding and managing rhabdomyosarcoma. *F1000prime reports*, Volume 7.

Ho, Y. & Wookey, S., 2019. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access*, Volume 8, p. 4806–4813.

Jang, H.-J.et al., 2020. Prediction of clinically actionable genetic alterations from colorectal cancer histopathology images using deep learning. *World Journal of Gastroenterology*, Volume 26, p. 6207.

Kashi, V. P., Hatley, M. E. & Galindo, R. L., 2015. Probing for a deeper understanding of rhabdomyosarcoma: insights from complementary model systems. *Nature Reviews Cancer*, Volume 15, p. 426–439.

Kikuchi, K., Rubin, B. P. & Keller, C., 2011. Developmental origins of fusion-negative rhabdomyosarcomas. *Current topics in developmental biology*, Volume 96, p. 33–56.

Powers, D. M. W., 2020. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*.

Rezaei-Dastjerdehei, M. R., Mijani, A. & Fatemizadeh, E., 2020. *Addressing Imbalance in Multi-Label Classification Using Weighted Cross Entropy Loss Function.* s.l., s.n., p. 333–338.

Shern, J. F., Yohe, M. E. & Khan, J., 2015. Pediatric rhabdomyosarcoma. *Critical Reviews™ in Oncogenesis,* Volume 20.

Szegedy, C. et al., 2016. *Rethinking the inception architecture for computer vision.* s.l., s.n., p. 2818–2826.

C. Keller, '*RMS-mouse-multidimensional-datasets*', viewed on: 2021, retrieved from: https://childrenscancertherapydeve.app.box.com/s/f82oib8p2yzmvdco3m9glxvl8qamu6eq

11. Appendix

- Below Code was used for whitespace eliminations

Eliminating Background Patches

November 30, 2021

[ ]:
```
import os
import pandas as pd
import glob
import shutil
from functools import partial
from multiprocessing.pool import ThreadPool,Pool
from tqdm import tqdm
```

[]:
```
def white_space_elimination(file):
    image = cv2.imread(file)
    h, w, _ = image.shape
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    thresh = cv2.threshold(gray, 223, 255, cv2.THRESH_BINARY)[1]
    total_pixels = h*w
    Non_zero_pixels = cv2.countNonZero(thresh)
    Zero_pixels = total_pixels - Non_zero_pixels
    ratio = (Zero_pixels/total_pixels) * 100
    if ratio >= 80:
    else:
        back = shutil.move(file,'Destination folder where the background patches␣
↪to be moved')
```

[]:
```
DIR = 'Source folder of all patches'
All_files = glob.glob(os.path.join(DIR, '*.png'))
with Pool(processes=10) as pool:
    results = pool.map(white_space_elimination,All_files)
```

- Below Code was used for Divide Whole Mount Images into patches

Patch Generation

November 30, 2021

1 Download libraries and Import necessary packages

```
[ ]: ! pip install py-wsi
     %cd py-wsi
     ! python setup.py install
```

```
[ ]: import openslide
     from matplotlib import pyplot as plt
     import os
     import py_wsi
     import py_wsi.imagepy_toolkit as tk
     import os
     import pandas as pd
     import numpy as np
     import glob
     import cv2
     import shutil
```

2 Patch generation

```
[ ]: file_dir ='Path to the WholeSlide Images'
     db_location ='Path Where the patches will be stored'
     level = 14
     db_name = "patch_db"
     patch_size = 256
     overlap = 0
```

```
[ ]: turtle = py_wsi.Turtle(file_dir, db_location, db_name,storage_type='disk'
```

```
[ ]: print("Total WSI images:    " + str(turtle.num_files))
     print("LMDB name:           " + str(turtle.db_name))
     print("File names:          " + str(turtle.files))
```

```
[ ]: print("Patch size:", patch_size)
     turtle.sample_and_store_patches(patch_size, level, overlap, load_xml=False,␣
      ↪limit_bounds=True)
```

- Below Code was used to convert string labels into binary format

labels

December 3, 2021

```
[ ]: from google.colab import drive
     drive.mount('/content/drive')

[ ]: import pandas as pd
     import numpy as np

[ ]: df = pd.read_excel('path to the excel file with labels')
     df

[ ]: column_1 = df.Genotype_1.unique()
     column_1 = column_1[~pd.isnull(column_1)]
     column_2 = df.Genotype_2.unique()
     column_2 = column_2[~pd.isnull(column_2)]
     column_3 = df.Genotype_3.unique()
     column_3 = column_3[~pd.isnull(column_3)]
     column_4 = df.UID.unique()

[ ]: d=np.concatenate((column_1, column_2,column_3), axis=None)

[ ]: new_df = pd.DataFrame(columns = d)

[ ]: new_df['UID'] = column_4
     new_df['UID_1'] = column_4

[ ]: new_df

[ ]: for row in df.iteritems():
       print(row[1][1])

[ ]: new_df = new_df.set_index('UID')

[ ]: for ind in df.index:
       for ind1 in range(len(new_df.index)):
          if(df['UID'][ind] == new_df['UID_1'][ind1]):
            if(df['Genotype_1'][ind] in new_df.columns):
              out = np.argwhere(new_df.columns.isin([df['Genotype_1'][ind]])).ravel()
              new_df.loc[new_df['UID_1'][ind1],new_df.columns[out]] = 1
            if(df['Genotype_2'][ind] in new_df.columns):
```



```
              out = np.argwhere(new_df.columns.isin([df['Genotype_2'][ind]])).ravel()
              new_df.loc[new_df['UID_1'][ind1],new_df.columns[out]] = 1
            if(df['Genotype_3'][ind] in new_df.columns):
              out = np.argwhere(new_df.columns.isin([df['Genotype_3'][ind]])).ravel()
              new_df.loc[new_df['UID_1'][ind1],new_df.columns[out]] = 1

[ ]: new_df

[ ]: new_df.to_excel('path to output the result')
```