African University of Science and Technology

Abuja

Knowledge is Freedom

A LIGHTWEIGHT CONVOLUTIONAL NEURAL NETWORK FOR BREAST CANCER DETECTION USING KNOWLEDGE DISTILLATION TECHNIQUES

A Dissertation presented to the Department of Computer Science,
African University of Science and Technology, Abuja-Nigeria
In partial fulfilment of the requirements for a Masters degree in Computer Science

By

Falmata Modu [40853]

Abuja, Nigeria

January, 2023

CERTIFICATION

This is to certify that the thesis titled A CONVOLUTIONAL NEURAL NETWORK KNOWLEDGE DISTILLATION TECHNIQUES FOR LIGHTWEIGHT DETECTION OF BREAST CANCER submitted to the school of postgraduate studies, African University of Science and Technology (AUST), Abuja, Nigeria for the award of the Master's degree is a record of original research carried out by Falmata Modu in the Department of Computer Science.

15.02.2023

SIGNATURE PAGE

A LIGHTWEIGHT CONVOLUTIONAL NEURAL NETWORK FOR BREAST CANCER DETECTION USING KNOWLEDGE DISTILLATION TECHNIQUES

By

Falmata Modu

A THESIS APPROVED BY THE COMPUTER SCIENCE DEPARTMENT

RECOMMENDED:

[illegible] 18 Feb 2023
——————————————-
Supervisor: Dr. Rajesh Prasad

[illegible] 18 Feb 2023
——————————————-
Head of Department: Dr. Rajesh Prasad

APPROVED:

——————————————-
Chief Academic Officer

January 26th, 2023
——————————————-
Date

ABSTRACT

The second most heterogeneous cancer ever discovered is Breast Cancer (BC). BC is a disease that develops from malignant tumors when the breast cells begin to grow abnormally. Although it grows in the breast, it can spread to other body parts or organs.through the lymph and blood vessels of the breast. Globally, more than two million new cases and about 600,000 women died from BC in 2020. Early detection increases the chance of survival by 99%. Deep Learning (DL) models have recorded remarkable achievements in disease diagnosis and treatments. However, it requires powerful computing resources. In this work, we propose a lightweight DL model that can detect BC using the knowledge distillation technique. The knowledge of a pre-trained deep neural network is distilled to a shallow neural work that is easily deployable in a low-power computing environment. We have achieved an accuracy of up to 99%. In addition, we recorded 99% reduction in trainable parameters compared to deploying with a deep neural network.

AKNOWLEDGEMENT

All thanks be to almighty Allah for giving me the ability and opportunity to this level of my academic carrier.

I would like to express my gratitude to *Dr.Rajesh Prasad* for his guidance, advice, time and efforts. Your helpful advice and suggestions were extremely beneficial to me as in making this work successful.

I like express my special thanks to my mentors *Dr. Farouq Muhammad Aliyu* and *Dr. Audu musa Mabu* for their undeniable support and push. Their advice led to the success of this work.

I am grateful to all of my lecturers who taught me over the course of my Master's degree; I value and treasure the invaluable knowledge and experience they have shared with me.

I also like to extend my special thanks to my loving parents, my husband and my siblings for their encouraging words, prayers, financial and emotional support in every aspect of my life.

Finally, I would like to thank Tertiary Education Trust Fund (TETFUND), Yobe State University (YSU), family and friends for their support.

DEDICATION

I dedicate this thesis to my lovely parents, my husband and my siblings for their immeasurable support in one way or the other.

List of Abbreviations and Terms

AI	Artificial Intelligence
ANN	Artificial Neural Network
BC	Breast Cancer
CNN	Convolutional Neural Network
DCIS	Ductal Carcinoma in Situ
DNNS	Deep Neural network with suport value
DL	Deep Learning
FN	False Negative
FP	False Positive
GT	Ground Truth
HA-BiRNN	Bidirectional Recurrent Neural Networks
IBC	Inflammatory Breast Cancer
IDC	Invasive ductal carcinoma
KD	Knowledge Distillation
KNN	K-Nearest Neighbor
LC	Lobular carcinoma
ML	Machine Learning
RCNN	Region-based Convolutional Neural Network
SM	Student Model
SVM	Support Vector Machine
TM	Teacher Model
VGG	Visual Geometry Group

Table of Contents

List of Figures

List of Tables

CHAPTER ONE

INTRODUCTION

Breast cancer (BC) detection is a vital area of research in the medical and healthcare sectors. BC develops from malignant tumors when breast cells' growth is abnormal [1, 2]. It grows in different parts of the breast and is common among women, but men can get it as well [3]. According to the World Health Organization, 2.3 million women are diagnosed with BC each year, which results in 29% deaths [4]. The higher the age, the more the chances of breast lump turning out to be malignant [5]. The survival chance increases to 99% if it is detected early. There are various common types of BC classified as invasive or non-invasive. Non-invasive is when the abnormal cells are bound to the milk passage (duct) or the milk-producing glands (lobules); invasive BC is when the abnormal cells spread beyond the duct or lobules, affecting the tissue connected to the breast or surrounding fatty tissues [6].

Infiltrating Ductal Carcinoma (IDC) is an invasive BC that starts in the milk duct, breaks through the duct wall, and spreads to the fatty tissue [5]. About 80% of BC diagnoses are IDC. Thus, making it the most common BC among women [6]. Infiltrating lobular carcinoma (ILC) is an invasive BC that usually starts in the milk glands (lobules) and extends to other breast parts. The most frequent non-invasive BC that accounts for almost 90% is ductal carcinoma in situ (DCIS) [6]. DCIS is when the cancer cells are bounded to the duct [5]. Inflammatory BC (IBC) and medullary carcinoma (MC) are some of the invasive BC types that occur less frequently; however, they are extremely fast-growing [7].

Some of the breast cancer diagnosis techniques include ultrasound-guided surgical biopsy, magnetic resonance imaging, mammography Xray, computed tomography, portion emission tomography, magnetic resonance imaging, and breast temperature [8, 9].

Researchers from computer disciplines proposed several models for detecting BC using a Convolutional neural network (CNN). CNN is widely used for the prediction of BC due to its excellent performance in feature extraction [10]. Deep CNN models have been developed in recent years to increase the effectiveness and improve the performance of BC detection [11]. Also, researchers have recorded remarkable achievements as some deep learning models detect BC with an accuracy of up to 98%. However, deep CNN architectures are cumbersome models with millions of trainable parameters and many hidden layers between the input and the output neurons. Thus, they require a lot of computing resources (such as GPU processors or an FPGA). Lightweight versions of deep CNN architectures require fewer hardware resources. Thus, it finds applications in real-time classification, recognition tasks, or resource-constrained environments.

In this work, we implore a lightweight ML model that can detect BC using knowledge distillation (KD) techniques. KD is a method of knowledge transfer where the knowledge of a pre-trained deep CNN model, referred to as the teacher model, is used during the training of a minor or shallow neural network, referred to as the student model. The typical use of the KD is capturing knowledge in a complex ML model and distilling it to a shallow model that can be deployed easily without sacrificing accuracy [12].

1.1 Problem Statement

The second most heterogeneous cancer ever discovered is breast cancer [13]. Worldwide, breast cancer is the fifth most common cause of mortality among women. Globally, more than two million new cases of breast cancer were diagnosed and about 600 thousand women died from breast cancer in 2020 worldwide [14].

Various factors contribute to the occurrence of breast cancer. However, its occurrence, survival, and mortality rates vary across different parts of the world. It could be due to many factors such as lifestyle (what you eat and how much you exercise), population structure, genetic factors, imbalances of hormones, environment, and older age [15]. Thus, there is no specific way of preventing breast cancer, but success is owed to early detection [16]. Hence, it is useful to have a system that will help us accurately detect it at an early stage which would reduce the mortality rate.

1.2 Aim and Objectives

This section of the proposal presents the aims and objectives of the research.

1.2.1 Aim

This thesis aims to develop a lightweight machine-learning model that can accurately detect breast cancer.

1.2.2 Objectives

1. Develop a DL model for breast cancer detection.
2. Develop a lightweight ML model for detecting breast cancer from the DL model through KD techniques.
3. Evaluate and compare the performance of the DL model in 1 and the lightweight model in 2
4. Compare the performance of the proposed system in 2 with some related work.

1.3 Scope of the Research

The scope of this research is limited to the following:

1. The thesis will focus on developing a model that can only classify or diagnose data related to breast features.
2. We will only consider training and testing of our model.

1.4 Significance of the Research

KD-based ML for breast cancer has many merits. Some of the benefits of this research are as follows:

1. KD will help develop accurate yet light ML models.
2. It can be easily deployed on resources with limited computing powers, such as laptops, mobile phones, and tablets.
3. This research will help pave the way for telemedicine in the field of oncology.

1.5 Expected Results and Deriverables

1.5.1 Expected Results

We expect to develop a lightweight machine-learning model that can detect breast cancer with an accuracy of at least 95 percent.

1.5.2 Deliverables

By the end of this research, we will publish at least a conference paper for a lightweight version of breast cancer detection.

1.6 Research Outline

The research outline is as follows:

1. **Chapter one** contains an introduction to breast cancer, a problem statement, aims & objectives, the scope of the research, the significance of the research, expected results & deliverables, and then the thesis outline.
2. **Chapter two** contains the background of the terms used in the thesis and a literature review on the machine learning techniques used in BC detection.
3. **Chapter three** contains a discussion of our proposed model and the method we used to meet the research objectives.
4. **Chapter four** contains the performance evaluation of our proposed model. First, we will compare the two models we build (deep and shallow NN) and then compare them with some results in the related work.
5. **Chapter five** contains the summary, conclusion of our research and possible open research directions.

CHAPTER TWO

BACKGROUND AND RELATED WORK

2.1 Introduction

This chapter prepares the reader for the coming chapters by explaining the concepts that form the foundation of this thesis. The chapter explains machine learning (ML), classification, neural networks, convolutional neural networks, and knowledge distillation (KD) techniques. In addition, we review work done in the past relating to breast cancer detection using various machine learning algorithms and some related work on knowledge distillation techniques. To our knowledge, knowledge distillation techniques have never been used for breast cancer diagnosis.

2.2 Machine Learning

Machine learning (ML) was first introduced by IBM computer scientist called Arthur et al. [17], where they defined ML as a type of artificial intelligence that allows software applications to learn and predict output based on experience without being explicitly programmed. It employs a variety of algorithms to construct mathematical models and make predictions using historical data or information as an input [18]. ML is capable of solving complex tasks at high speeds that are beyond human capabilities. With a large amount of data, a machine can be trained to make analysis, detect patterns, connections, or relationships which can use for improving decision-making, or optimizing efficiency [19]. A ML model is immune to cognitive bias and fatigue because one of its primary goals is to draw conclusions from a set of data with little human intervention. Figure 1 depicts various types of ML.

2.2.1 Supervised Learning

Supervised learning is a machine learning training method that uses labeled datasets for training. Cunningham et al. [20] described the aim of supervised learning as building an artificial system that can learn the relationship and connection between input and output, then anticipate the system's output given new inputs. The algorithms models the relationship, dependencies, and connections between the features and target class. After

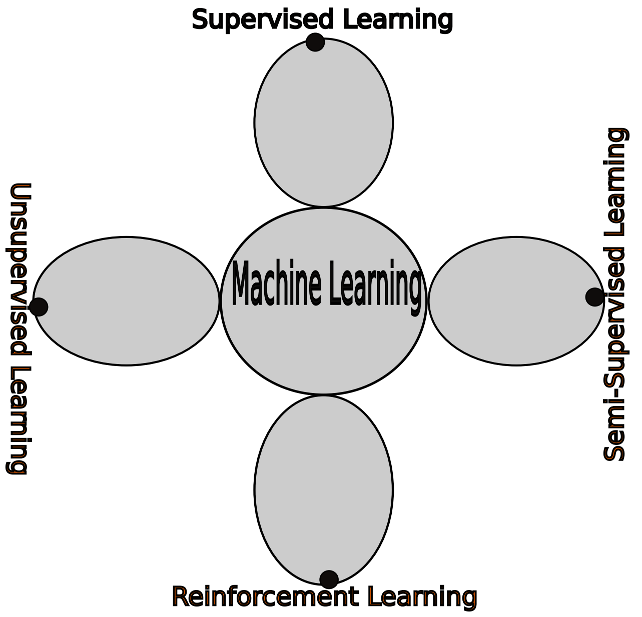


Figure 1. Types of Machine Learning

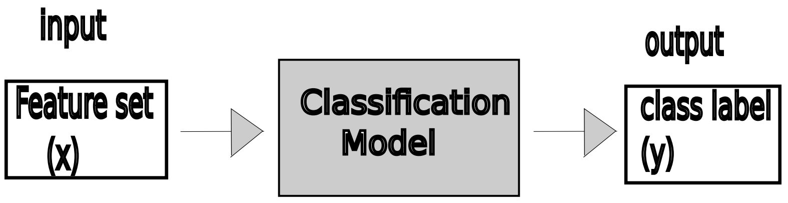
sufficient training, the algorithm should be able to detect the class label of a new data based on the key characteristics and the knowledge gained [21]. Supervised learning falls mostly into two categories which are *classification* and regression which we have used in this work.

2.2.2 Classification

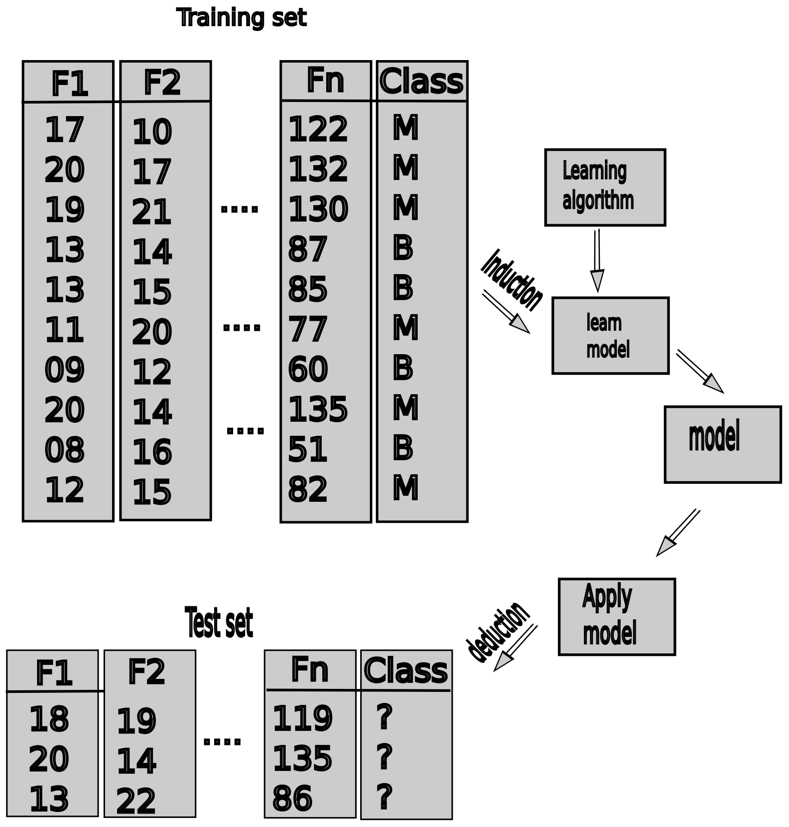
Classification is the task of determining the class label of a new instance (data) based on a training set of data with known class label [22, 23].The goal of classification is to build a classification model g that associates each feature set x with a particular class label y (refer to Equation 2.1 and Figure 2a). The feature set(x) consists of some properties or attributes of the task to classify, while the class label (y) is the target class [24]. When the class is two, then it is called binary classification and if the class is more than two then its referred to as multi-class classification.

$$\begin{gathered} y = g(x) \qquad (2.1) \\ where,\ y\ is\ the\ class\ label \end{gathered}$$

A systematic method for creating classification models from input data sets is known as a classification technique (or classifier). Examples include neural networks, naive Bayes, rule-based, decision trees and support vector machines. Each classifier determine a model



(a) Mapping an input x into class label y



(b) General Approach for building classification model

Figure 2. Classification

that better describes the connection between the input feature set and the target class using a learning algorithm. The identified model should be able to fit input features correctly and effectively predict the target class of a new instance (data).Hence, building models with a good generalization capabilities or one that correctly anticipate the class labels of previously unseen data is one of the primary objectives of a learning algorithm. [24, 25]. The general technique for building a classification model is depicted in figure 2b. A training set of data with known class labels must be provided first. The training set is used to create a classification model, and it is then applied to the validation set, which consists of data with unidentified class labels.

2.2.3 Neural Network

One of the enamous fields of the twenty-first century on which a significant amount of research has been done is the neural network. Haykin et.al. [26] described neural network as a type of ML where human biological neurons are simulated to build a computational network for solving problems and making decision. An essential component of a neural network's functionality is its basic information processing unit called the neuron [26]. The model of a neuron, which consists of weighted input signals, an activation function, and an output signal, is depicted in the block diagram of Figure3. The weight of a neuron is a parameter that set the standards for the neuron's signal strength.

ANN is gaining acceptance in different fields as a solution [27]. ANN is used to optimize the performance of both non-linear and linear control systems [28]. In medicine, ANN is used for drug discovery[29]. Moreover, ANN is also used in natural language understanding, speech recognition, video processing, and computer graphics [30].

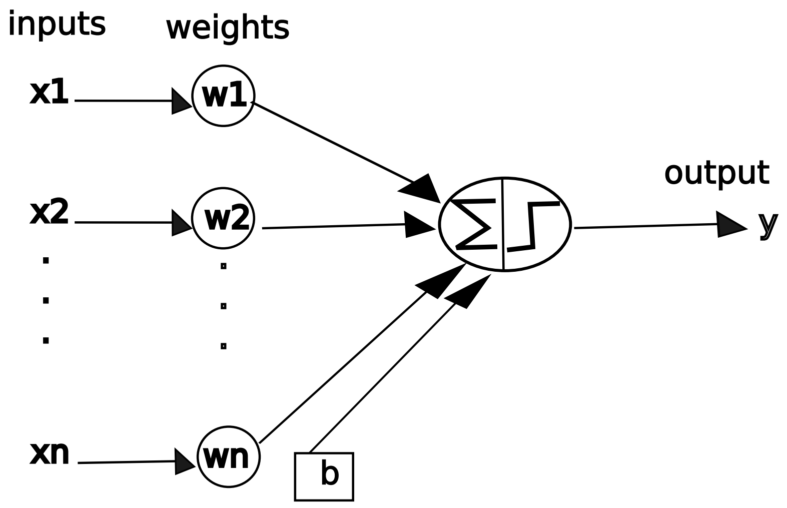


Figure 3. Architecture of a perceptron

Equation 2.2 and 2.3 represent the perceptron (a single neuron) shown in Figure 3. Equation 2.2 is the summation of the bias and the input signals (x_i) multiplied with their respective weights (w_i). Equation 2.3 uses an activation function to convert the effects of the input (z) to an output (y). The process of tuning the weights (w_1, w_2, . ., w_n) until desirable predictions for the output (y) are obtained compared to a sample dataset is known as training.

$$z = \left(b + \sum_{i=1}^{n} x_i w_i \right) \tag{2.2}$$

$$y = \varphi\left(z\right) \tag{2.3}$$

$$where,$$
$$\left(w_1, w_2, . . ., \ w_n\right) = weight\ of\ neurons$$
$$\left(x_1, x_2, \ . . ., \ x_n\right) = input\ signals$$
$$b = bias$$
$$z = intermediate\ output$$
$$\varphi = activationfunction$$
$$y = output\ signal\ of\ the\ neuron.$$

$$Y_1 = \varphi\left(W_1^T X + b_1\right) \tag{2.4}$$

$$Y_{i+1} = \varphi\left(W_{i+1}^T Y_i + b_{i+1}\right), \forall i \in]1, 2, ..j - 1[\tag{2.5}$$

$$o = \varphi\left(W_{j+1}^T Y_j + b_{j+1}\right) \tag{2.6}$$

$$where,$$
$$all\ subscript\ denotes\ the\ layer\ number$$
$$input\ vector(X) = x_1, x_2, ..., x_n$$
$$w_1^T = transpose\ of\ neurons\ weight\ signified\ by\ superscript\ T$$
$$b = bias$$
$$\varphi = activationfunction$$
$$Y_1 = output\ between\ input\ to\ hidden\ layer$$
$$Y_{i+1} = output\ between\ hidden\ to\ hidden\ layer$$
$$o = final\ output\ signal\ of\ the\ network.$$

Furthermore, there are large neural networks with one or more layers between the input and the output layer of the network, those layers are called the hidden layers. A neural network architecture containing only one or not more than two hidden layers is referred to as a shallow neural network [31, 32], and a network with more than two hidden layers as shown in Figure 4 or even larger is referred to a deep neural network (DNN). It could be modeled with Equation 2.4-2.6. Equation 2.4 computes the output between an input layer and the first hidden layer using matrix multiplication, Equation 2.5 computes the outputs between hidden layer to hidden layer and Equation 2.6 computes the final prediction (last hidden layer to output layer). The deeper the DNN, the more hidden layers it has which exponentially increases the complexity of the prediction. Hence, it can solve more complex problems. Uzair et al.[33] discover that the ideal number of hidden layers is three. Additionally, they discovered that decreasing the number of hidden layers below this ideal value decreased accuracy while increasing the number of hidden neurons above the ideal value increased accuracy and computational complexity.

2.2.4 Convolutional Neural Network

Convolutional Neural Network (CNN) is a group of a deep neural network specially designed to work with grid-structure input with strong spatial dependencies and some special cases of it, such as temporal or spatiotemporal data [31]. CNN's are analogous to ANN in that they contain a set of neurons that self-optimize through learning. In addition, CNN requires at least one convolutional layer as it is the crucial step for feature extraction, although it can have other types of layers like fully connected and pooling layers [34]. The primary foundation of a CNN is the convolutional layer. It has a parameter that serves as a weight which is known as a filter (or kernel) that needs to be learned during the training. In the convolution layer, the kernel slides through the input vector from top to bottom and computes a dot product to identify features at all spatial positions. The weighted summation of the dot product represents the input of the next layer. Each convolutional operations is define by a sliding step (stride), filter size, and padding. A stride is a parameter of a filter (kernel) that specify the amount of movement in each step-in convolution; using the default stride length which is implies that the kernel will move one unit at a time. Padding is also a parameter in the convolutional operation that add zero's to the input vector symmetrically; it is used when we want to maintain same dimension for the input and the output after convolution (feature map).

Figure 4 shows an architecture of a CNN. It has one input layer, six (6) hidden layers (two of which are convolutional layers (conv1D) and four (4) fully connected dense layers), and an output layer. Between the convolutional layers is batch normalization, which is used to sets the pixels in all feature maps in a convolution layer to a new mean and a new standard deviation. A fully connected layer then performs the same task as in standard ANN. It multiplies the input by a weight matrix, then adds the bias vector used for classification. The last layer is the output layer which gives the predictions of the target class.

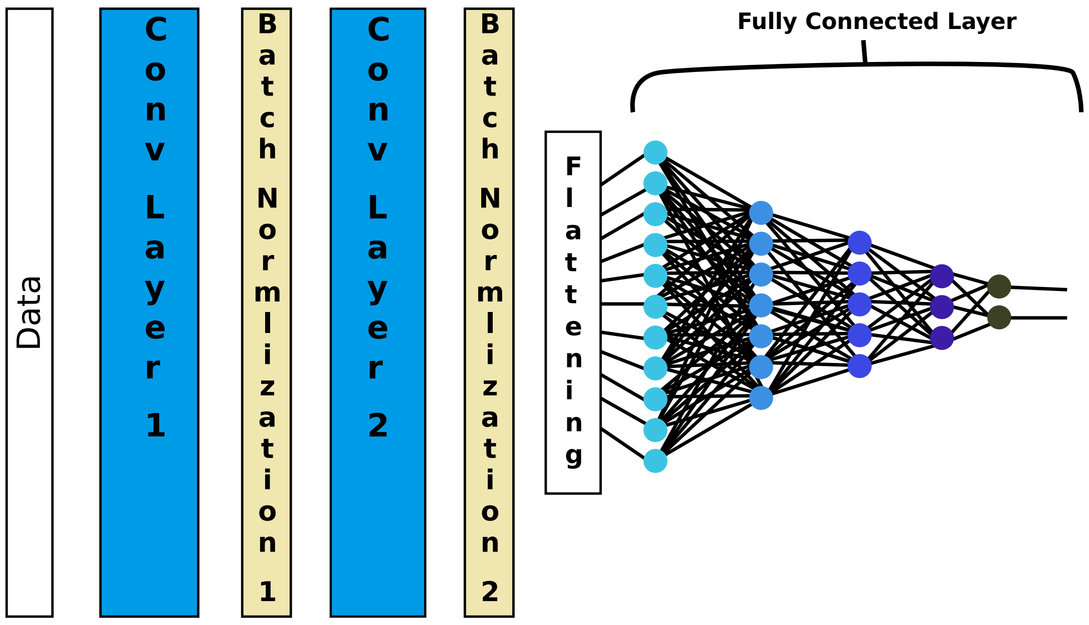


Figure 4. Architecture of a CNN with 6 hidden layers

2.2.5 Knowledge Distillation

Knowledge distillation is a model compression method in which the knowledge of a deep neural network is distilled to a small or shallow neural network, as shown in Figure 5. Hinton et al.[35] described that distillation is training where knowledge of a pre-trained large network referred to as the teacher model will be transferred to a shallow network called student model to enable suitable deployment on a low computing power or embedded devices. The soft probabilities of the pre-trained teacher network have more information than the target class label. Thus, training the student with soft probabilities will enable it to absorb more information discovered by the teacher beyond training with the class label alone [36].

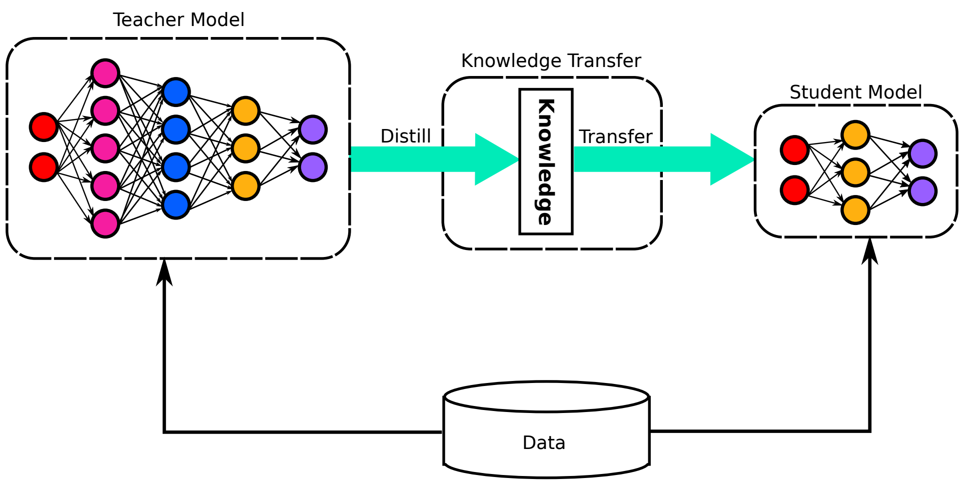


Figure 5. Simple Model of Knowledge Distillation

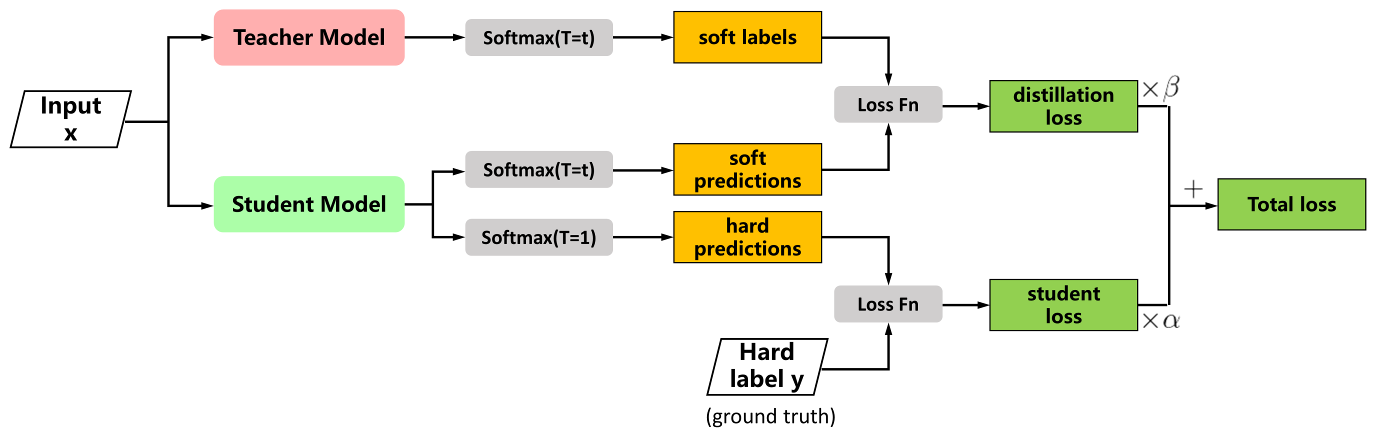
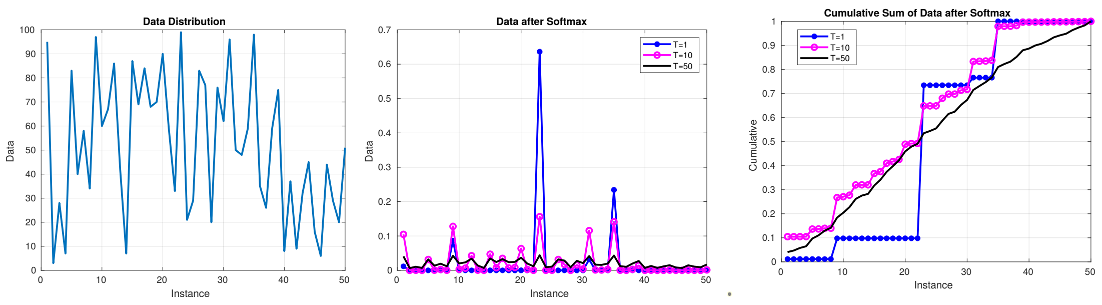


Figure 6. Teacher-Student Model [37]

Figure 6 shows the flowchart of how knowledge distillation works. The figure has two models; a **Teacher model** and a **Student model**. On the one hand, the teacher model is a CNN with numerous parameters. They allow it to model the dataset accurately. On the other hand is the student model, which has very few parameters. The parameters are so few that CNN cannot accurately model the dataset. Scientists desire fewer parameters because it means fewer computations, which translates to lesser resource (e.g., energy, memory, and computation time) requirements. Thus, the student model learns from the teacher.



(a) Data before Softmax (b) Data after Softmax (c) Cumulative distribution

Figure 7. Effect of the Softmax Function on Data

$$Softmax(x) = \frac{e^{x_i}/T}{\sum_{j=1}^{N}(e^{x_j}/T)} \tag{2.7}$$

$$Where,$$

$$x_i = Prediction\ for\ instance\ i$$

$$N = Number\ of\ instances$$

$$T = Temperature$$

The flowchart shows that both models learn from the same attributes, **Input x**. The predictions of the teacher model are then passed through a **Softmax** function shown in Equation 2.7. Softmax converts the predictions into a probability distribution, which bounds the data between zero and one [38]. To demonstrate how the Softmax function works, we used the MATLAB code in Listing 2.1. The code generated a uniformly distributed sample of numbers between 0–100. Then it plots the generated data, as shown in Figure 7a. The figure shows that the sample size is 50. Next we calculate the Softmax for the data at temperature 10, 50, and 1 and store them in the variables sf10, sf50 and sf1, respectively. They are ploted in Figure 7b and their respective cumulative distribution is shown in Figure 7c. The figures show how the Softmax function works and how the temperature variable affects the results. The Softmax operation converts the data into a probability distribution of the predictors — it fills the probability axioms; (1) All probabilities are between zero and one as shown in Figure 7b, and (2) The sum of all the samples is unity as shown in Figure 7c. The softmax function reduces the variance in the data, thus making it easier for the smaller student model to learn.

Listing 2.1. Matlab Code Simulating the Softmax Function

```
dt=round(rand(50,1)*100);

plot(dt,'linewidth',2)
title('Data␣Distribution');
ylabel('Data')
xlabel('Instance')
grid on;

sf10=softmax(dt/10);
sf50=softmax(dt/50);
sf1=softmax(dt);

figure;
plot(sf1,'-*b','linewidth',2);
hold on;
plot(sf10,'-om','linewidth',2);
plot(sf50,'k','linewidth',2);
title('Data␣after␣Softmax');
ylabel('Data');
xlabel('Instance');
legend('T=1','T=10','T=50');
grid on;

figure;
plot(cumsum(sf1),'-*b','linewidth',2);
hold on;
plot(cumsum(sf10),'-om','linewidth',2);
plot(cumsum(sf50),'k','linewidth',2);
title('Cumulative␣Sum␣of␣Data␣after␣Softmax');
ylabel('Cumulative');
xlabel('Instance');
legend('T=1','T=10','T=50');
grid on;
```

The Softmax of the Teacher's predictions is the **Soft Labels** because the student model aims to model them. The Softmax of the student model (at the same temperature (t) as the teacher model) is called the **Soft Predictions**. The Soft Predictions and the Soft Labels are inputted to a **Loss Function** for performance evaluation of the student model. The result of the loss function is known as the **Distillation Loss**.

$$Total\ loss = (\alpha \times distillation\ loss) + (\beta \times student\ loss) \tag{2.8}$$

$$Where,$$

$$\alpha + \beta = 1 \tag{2.9}$$

Our primary aim in knowledge distillation is for the student model to predict the dataset. Thus, we must compare the student model's performance with the actual dataset. Hence, the predictions of the student model are operated on a Softmax with a temperature of $T = 1$. The temperature acts as a smoothing factor for the Softmax function where T is proportional to the smoothness of the result, as Figure 7b and 7c show. The result of the Softmax operation is known as the **Hard Prediction**. A loss function compares the hard prediction with the normalized labels from the dataset. The result is known as the **Student Loss**. A weighted sum of the distillation loss and the student loss is the **Total loss** as shown in Equation 2.8 and 2.9. The values of α and β are proportional to the importance the student model gives to the Hard labels (i.e., labels from the dataset) and soft labels (i.e., labels from the teacher model), respectively. The total loss is the fed back to CNN for further optimization [39].

2.3 Related works

Authors in [40], developed BC detection using a deep learning algorithm. The authors used mammogram images of 400 women and apply deep learning neural network algorithm to identify the BC and type of tumor.

Gupta et al. [41] developed a BC prediction model using six supervised learning algorithms. They trained and evaluated each algorithm twice or three times by varying some hyperparameters. In their results, the deep learning algorithm outperformed the other five algorithms they used, achieving 98.24% accuracy and 98.0% for precision, recall, and F1-score.

Anji et al. [42] developed a novel pseudocode for BC detection using a deep neural network

(DNN) with support value. The authors employed a histosigmoid-based fuzzy clustering technique for data segmentation on the histopathology image dataset of over 683 patients. The performance of their model (DNNS) is compared to the five most widely used ML algorithms, which are support vector machine (SVM), Naive Bayes, Bi-clustering & Ada boost, HA-BiRNN, and RCNN) for BC detection. Results proved that DNNS is better in terms of efficiency, performance, and image quality. Its classification accuracy was 97.21%, a precision of 97.90%, and 97.01% recall.

Gong et.al.[43] developed a self-distilled supervised learning (SDSCL) model to enhance the predictive performance of a CNN-based computer-aided design for BC. The authors apply hematoxylin and Eosin (H and E) stain view techniques to some histopathological images. The decomposed H and E stain views are then fed into the SDSCL algorithm for learning more intrinsic feature representation. Hematoxylin & Eosin (H and E) stain views help to identify cells and breast tissues. It also provides more information about the structures and shape of the cells in a tissue, which help them in improving the classification performance of SDSCL.

In [44], BC classification using discrete wavelet transformation (DWT) and DL was developed from the numerical dataset. The authors used DWT techniques for data pre-processing and then applied a feed-forward neural network (FFNN) for BC classification. The model obtained a great BC classification accuracy of up to 98.84%. However, one major drawback of FFNN is time consumption in both development and deployment.

Authors in [45, 46], use Knowledge Distillation to obtain a lightweight model of their work in computer vision. They show that knowledge distillation can be used to enhance a CNN model, and can be applied in a situation where there is a limited training dataset or resource constraints without losing accuracy.

[47] developed breast cancer prediction using ML. The authors compared the performance of four ML algorithm: SVM, KNN, Random Forest, and logistics regression on different data set to predict. The results were evaluated using 10-fold cross-validation. They found out that SVM was more effective than the other classifier and has an accuracy of 97%. However, the authors mentioned that the variables used were few. Adding more variables will yield better performance.

In [48], a lightweight deep learning (DL) pipeline for detecting anomalies in mammogram images was developed. To reduce training time and extensive data processing, the authors used transfer learning techniques. Transfer learning is a technique that allows an ML to transfer knowledge obtained while solving one task to a different task. The result is obtained using INbreast public database. To produce lesion area, the authors modified a pre-trained CNN and for the mammogram classification, they modified a pre-trained VGG16 model.

Authors in [49] reviewed different AI techniques for the detection of BC. 80 most recent research papers on BC were reviewed. The authors stated that the most common method for BC diagnosis was histopathology imaging and only a few used genes. They stated that for both binary and multi-class classification, CNN and ANN models are the most often employed models and CNN exhibits outstanding performance for both imaging and gene expression. In conclusion, the authors mentioned that most of the papers only consider accuracy while assessing their performance. However, the accuracy matrix does not distinguish between FP and FN classification.

Zhigiang et al.[50] described various techniques for CNN model compression. They experimented with knowledge distillation, model pruning, and model quantization. The performance of the models was compared, where knowledge distillation outperforms the other two methods as it provides higher accuracy and faster deployment time.

Similarly, authors in [51] employed knowledge distillation to reduce a complex DL model to a lightweight version while maintaining performance. According to their findings, KD can effectively improve or compress a CNN model to fit a low computing power environment or embedded devices without sacrificing accuracy, as other compression methods do.

DL was designed to examine the key features influencing the diagnosis and treatment of serious diseases. All the related works in this section used ML techniques in one way or another to detect BC. However, none consider developing a lightweight model to reduce deployment time. Moreover, DL is computationally intensive in both training and deployment. Hence, in this research, we will use KD to distill the knowledge obtained by training a DL model to a shallow model. The SM model will be used for deployment. Hence, it achieves DL accuracy while maintaining the lightweight features of the shallow model.

CHAPTER THREE

METHODOLOGY

3.1 Introduction

In this chapter, we discuss the methodology and the algorithms we have used in our research. The first part of the chapter describes the data set used, and then the last part discusses how the proposed model is built to achieve the research objectives.

3.2 Dataset Discription

The dataset used in this research was retrieved from Kaggle's Wisconsin Diagnostic Breast Cancer (WDBC) [52]. The dataset contains some features of the breast cell nuclei that were determined from a digital image of a fine needle aspirate (FNA) [52]. In addition, FNA is a type of biopsy where a thin needle is inserted into the breast tissue of the suspicious area with the help of an ultrasound monitor. The biopsy tissue will then be checked by a pathologist under a microscope to find out if there are cancer cells in it. The dataset contains 569 cases, of which 212 are malignant (cancer) and 357 are benign (non-cancer).

Each of the features has three pieces of information: mean, standard error, and mean of the three largest values, thus making a total of 30 features per image.

Table 1. Characteristics of the dataset.

label	Features
i	radius
ii	perimeter
iii	texture
iv	area
v	compactness
vi	smoothness
vii	concavity
viii	symmetry
ix	concave points
x	fractal dimension
Total dimension: 30	

3.2.1 Data Exploration and Visualization

One of the vital topics in machine learning is data exploration and visualization. It represents the data in a graphical format, which helps in understanding the data, how the data looks, and what kind of correlation is held by the features of the data. Overall, it is easier to grasp the information expressed by the dataset. Figure 8 depicts the count plot of the class to be predicted. Correlation Coefficient (r) is a way of describing how two

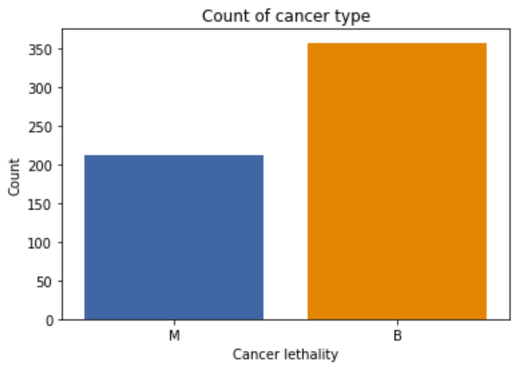


Figure 8. Class Distribution

features are closely related. The value of r ranges between ±1 where positive or negative values indicate that the least-squares line has a positive or negative relationship and if it is exactly zero, then it's said to be uncorrelated. Figure 9 shows a correlation heat map among the features of benign and malignant classes.

Highly correlated features reduce the performance of some machine learning models. Our data is valuable to acknowledge as the number of highly correlated features in the breast cancer dataset is low.

3.2.2 Data-preprocessing

Data preprocessing is a technique in data mining that is used to prepare the raw data making it useful, efficient, and appropriate for a machine-learning algorithm. WDBC dataset used has a single column with missing values, a column 'id' which is a unique identifier for each patient, thus has no impact on the cancer classification, and a column 'diagnosis', which is the class we expect to predict in our model. The 3 columns were removed using dataframe.drop() of pandas which left us with 30 features.

Feature Scaling helps algorithms quickly reach the minima of the cost function. Standardization is a technique of feature scaling that rescale the features to have the same characteristics of a conventional normal distribution with a mean of zero and a standard de-

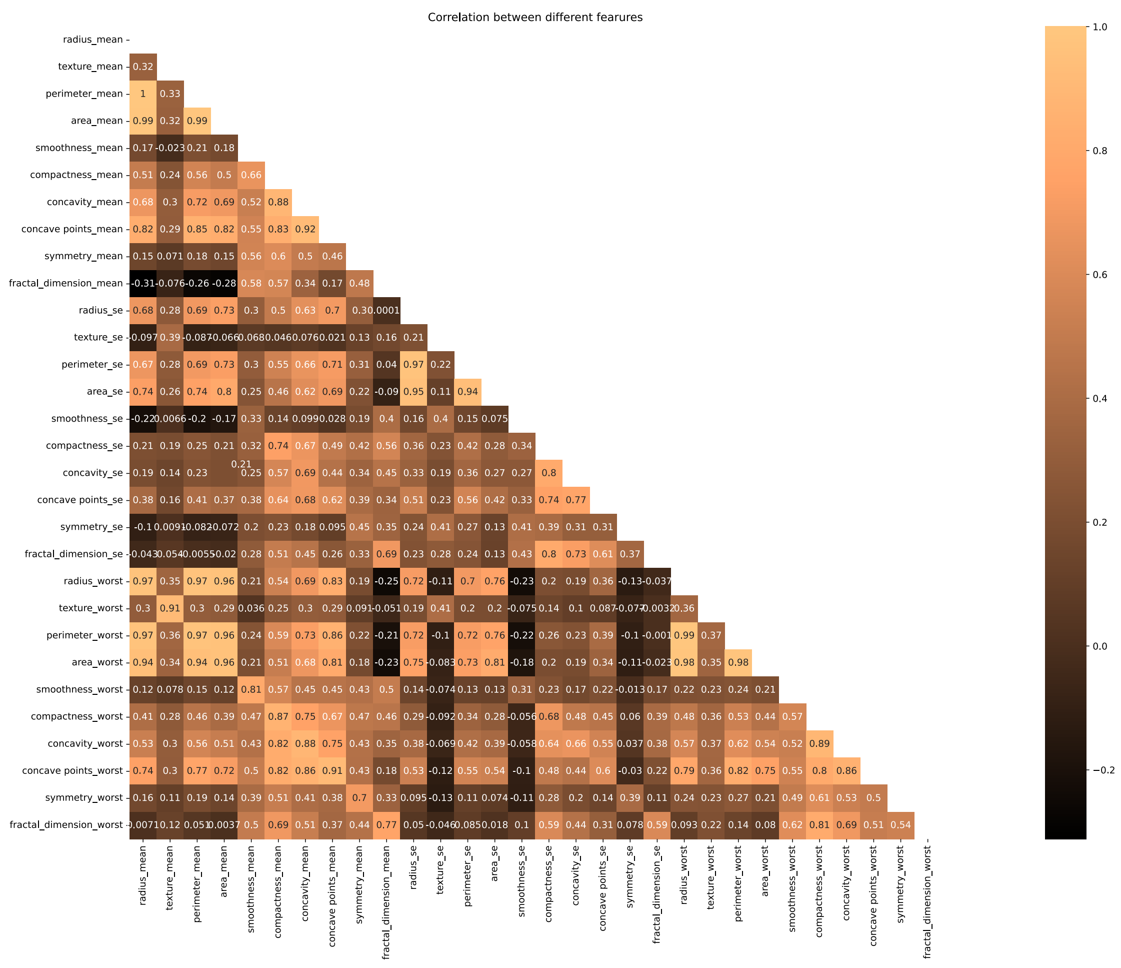


Figure 9. Correlation matrix

viation of one. For this work, we standardized the features using the Z-score normalization equation below :

$$Xscaled = \frac{xi - \mu}{\sigma} \tag{3.1}$$

Where, $x_i$is the feature to be standardized, μ is the mean, and σ is the standard deviation of the distribution. This was implemented using StandardScaler().fit_transform() of sklearn.

3.2.3 Model training and validation

Model training is a process of providing a clean and sufficient dataset to the machine learning algorithm to extract features, discover and learn patterns from the features involved. Validation testing is carried out immediately after training the model to evaluate the performance of the prediction model against the ground truth data. K-Fold Cross Validation is a model evaluation technique that divides the input dataset into k subset of data referred to as folds where a single subset is used for testing or validation and the remaining (k-1) is allocated to training the model. This process will be repeated k times, in each iteration new subset of data that was not assigned in the past will be used for the testing phase (see figure 10.

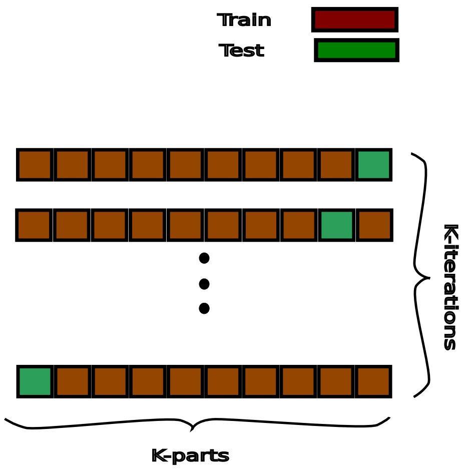


Figure 10. k-Fold cross-validation

Cross Validation techniques are used to flag problems like overfitting or input-output selection bias in the dataset. In our work, the dataset is randomly divided into ten equal parts using 10-fold cross-validation, we take out 1 unique part for validation testing and the remaining 9 parts are added together to represent the training model dataset. We fit our model with the training dataset and evaluate the performance with the validation dataset. This process is repeated ten times where a different subset of the dataset is used for validation in each iteration. The final evaluation score is the mean of the evaluation score from the ten iterations.

3.3 Proposed Model

In this research, we used a Convolutional Neural Network (CNN) model to detect BC. This research aims to develop a lightweight model for detecting breast cancer using the knowledge distillation technique. In this work, we have used Keras sequential API (Tensorflow backend). Figure 11 shows the experiment carried out.

A deep CNN model — the teacher model (TM) is defined with over 863 thousand total parameters of which 862 thousand are trainable parameters, the summary is presented in Table 2. Before building the model, we used the hyper-band grid search to identify the best parameters, such as the filter sizes, number of hidden layers, learning rate, and dropout units. The TM has one input layer, three (3) hidden layers (two of which are convolutional layers (Conv1D) and a dense layer), and an output layer.

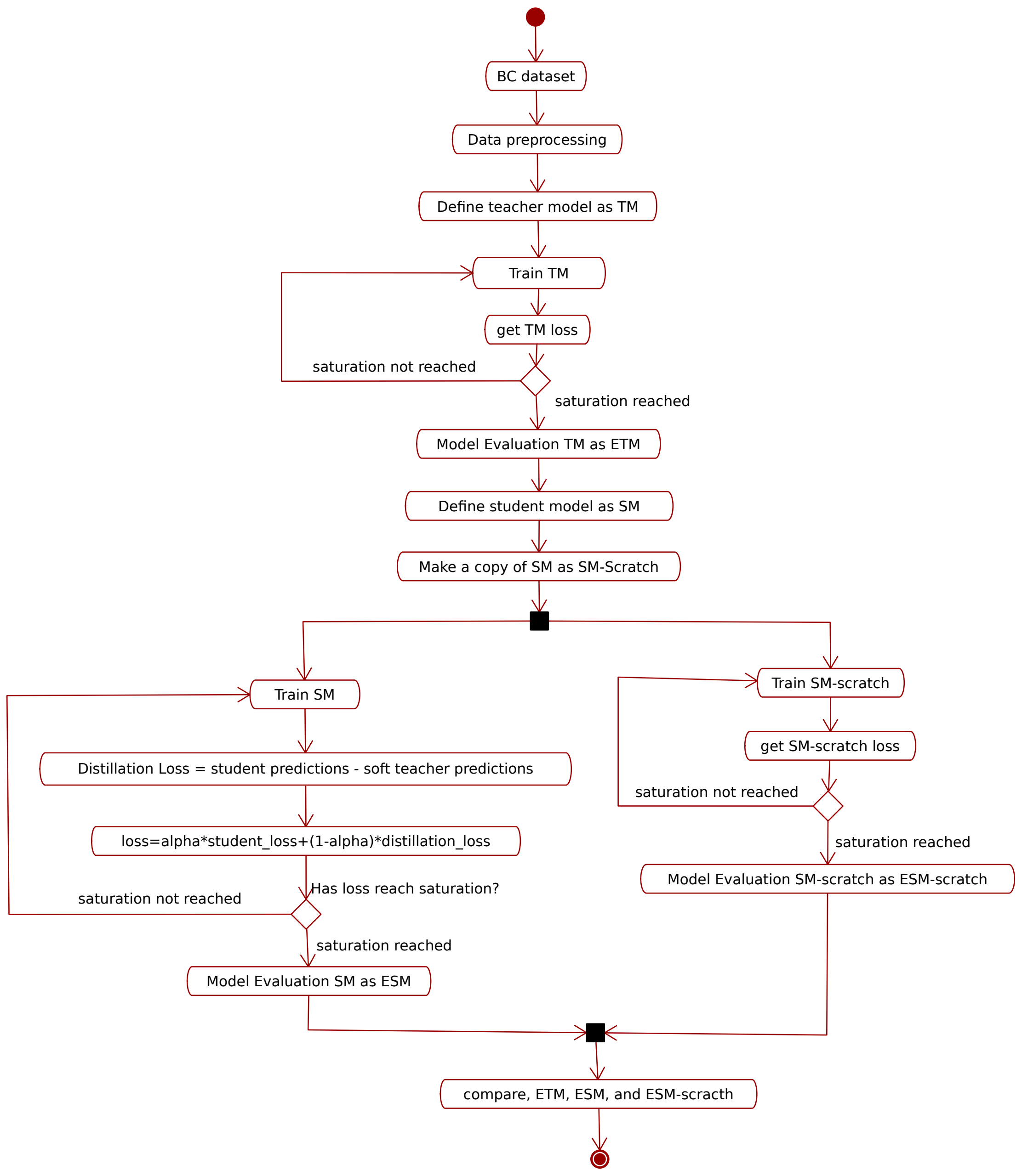

BC dataset
Data preprocessing
Define teacher model as TM
Train TM
get TM loss
saturation not reached
saturation reached
Model Evaluation TM as ETM
Define student model as SM
Make a copy of SM as SM-Scratch
Train SM
Distillation Loss = student predictions - soft teacher predictions
loss=alpha*student_loss+(1-alpha)*distillation_loss
Has loss reach saturation?
saturation not reached
saturation reached
Model Evaluation SM as ESM
Train SM-scratch
get SM-scratch loss
saturation not reached
saturation reached
Model Evaluation SM-scratch as ESM-scratch
compare, ETM, ESM, and ESM-scracth

Figure 11. Experiment Carried Out

Both convolutional layers use a kernel size of 2 and a ReLU activation function. The first convolutional layer has 64 filters. It takes the feature vector with a dimension of (30×1) as input and transforms the sample into a (29×64) shape vector. The second convolutional layer has 448 filters. It takes the output of the first convolutional layer, processes it, and produces a 28×448 output vector.

For the convolutional operation, the kernel slides through the input vector and weigh each feature to extract meaningful information (feature maps). This is done by computing the dot product of the filter matrix sliding through some portion of the feature vector; an illustration is shown in Figure 12. We used batch normalization between each hidden

Table 2. Summary of teacher model

Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 29, 64)	192
batch_normalization_1 (Batch	(None, 29, 64)	256
dropout_1 (Dropout)	(None, 29, 64)	0
conv1d_2 (Conv1D)	(None, 28, 448)	57,792
batch_normalization_2 (Batch	(None, 28, 448)	1,792
dropout_2 (Dropout)	(None, 28, 448)	0
flatten_1 (Flatten)	(None, 12544)	0
dense_1 (Dense)	(None, 64)	802,880
dropout_3 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 2)	130
Total params: 863,042		
Trainable params: 862,018		
Non-trainable params: 1,024		

layer to improve accuracy. The third in the table is a dropout which minimizes over-fitting or enhances generalization. Dropout is a process where some portions of neurons in the hidden layer are randomly ignored by setting their weights to zero, which drives the network to learn features differently. Furthermore, it forces the CNN to learn robust features that do not rely on the presence of other neurons. We used it to randomly turn off 10 percent of the neurons during the training phase.

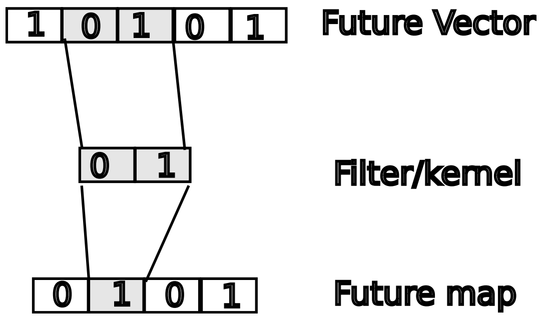


Figure 12. Conv1D Operation

Flatten layer involves transforming the last feature map (28x448) into one dimension with 12,544 data. Afterward, two dense layers were employed using ReLU and sigmoid activation functions. Hence, the TM has over 863 thousand total parameters, of which 862 thousand are trainable. The preprocessed BC dataset discussed in 3.2 was used to train the teacher model until full convergence, and the final predictions will be used to train the SM later.

Figure 13 presented the lightweight architecture of a shallow CNN model called the Student model (SM). The SM has fewer parameters than the TM. It consists of 2 conv1D, a filter of 4 & 8 with a kernel size of 2, summary is presented in Table 3. SM will be

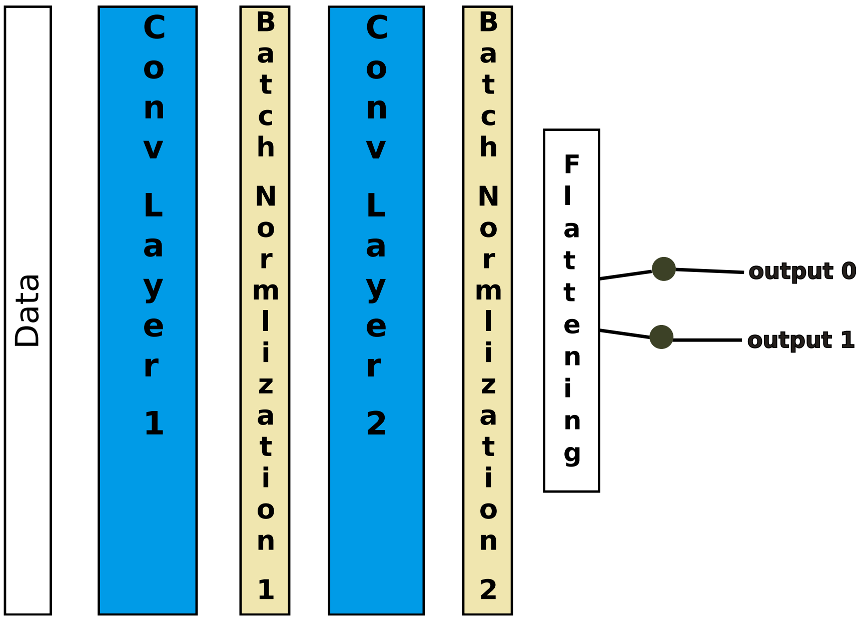


Figure 13. Lightweight Architecture of the Student Model

Table 3. Summary of student model

Layer(type)	OutputShape	Param#
Conv1d_1(Conv1D)	(None,29,4)	12
batch_normalization_1(Batc	(None,29,4)	16
dropout_1(Dropout)	(None,29,4)	0
conv1d_2(Conv1D)	(None,28,8)	72
batch_normalization_2(Batc	(None,28,8)	32
dropout_2(Dropout)	(None,28,8)	0
flatten_1(Flatten)	(None,224)	0
dense_29(Dense)	(None,2)	450
Total params:582		
Trainable params:558		
Non-trainable params:24		

trained in coordination with the fully trained TM. SM was trained in coordination with the fully trained TM. The Knowledge is distilled from large, complex TM to SM using the Distillation loss (D_{loss}) Equation below 3.2 :

$$D_{loss} = \mathfrak{H}_{KLD}(q_t, q_s) \tag{3.2}$$

$$Where,$$

$$\mathfrak{H}_{KLD} = Kull - Libler\ divergence\ loss$$

$$q_t = softer\ logit\ of\ TM\ predictions$$

$$q_s = softer\ logit\ of\ SM\ predictions$$

D_{loss} is the minimize squared difference between the TM and SM softer logits. We have used keras's Kullback-Liebler (KL) divergence loss function to obtain the squared differences. The KD uses a softmax function to soften the logits of both the TM and the SM. A temperature parameter (T) is added to the softmax function to control the softness of the probability distribution of the logits. The soft logits provides us with more information in each training case than the hard target because it has strong entropy. Also, the variance in the gradient between training cases is less.

The SM's performance was evaluated using Equation 3.4 by combining two loss functions. One loss function is the D_{loss}, and the second loss function is the student loss ($Student_{loss}$). $Student_{loss}$ is the comparison between the ground truth and the SM prediction as presented in Equation 3.3. Figure 14 summarized the TM and SM learning process.

$$Student_{loss} = \mathfrak{H}_{BCE}(p, Z_s) \tag{3.3}$$

$$\begin{aligned} &Where, \\ &\quad \mathfrak{H}_{BCE} = Binary\ cross\ enropy\ loss \\ &\quad p = ground\ truth \\ &\quad Z_s = SM\ predictions \end{aligned}$$

$$\mathfrak{H}_{KD} = \alpha * \mathfrak{H}_{BCE}(p, Z_s) + (1 - \alpha) * \mathfrak{H}_{KLD}(q_t, q_S) \tag{3.4}$$

$$\begin{aligned} &Where, \\ &\quad \mathfrak{H}_{BCE}(p, z_s) = S_{loss} \\ &\quad \mathfrak{H}_{KLD}(q_t, q_s) = D_{loss} \\ &\quad \alpha = hyperparameter\ to\ weigh\ the\ importance\ of\ S_{loss}\ and\ D_{loss} \end{aligned}$$

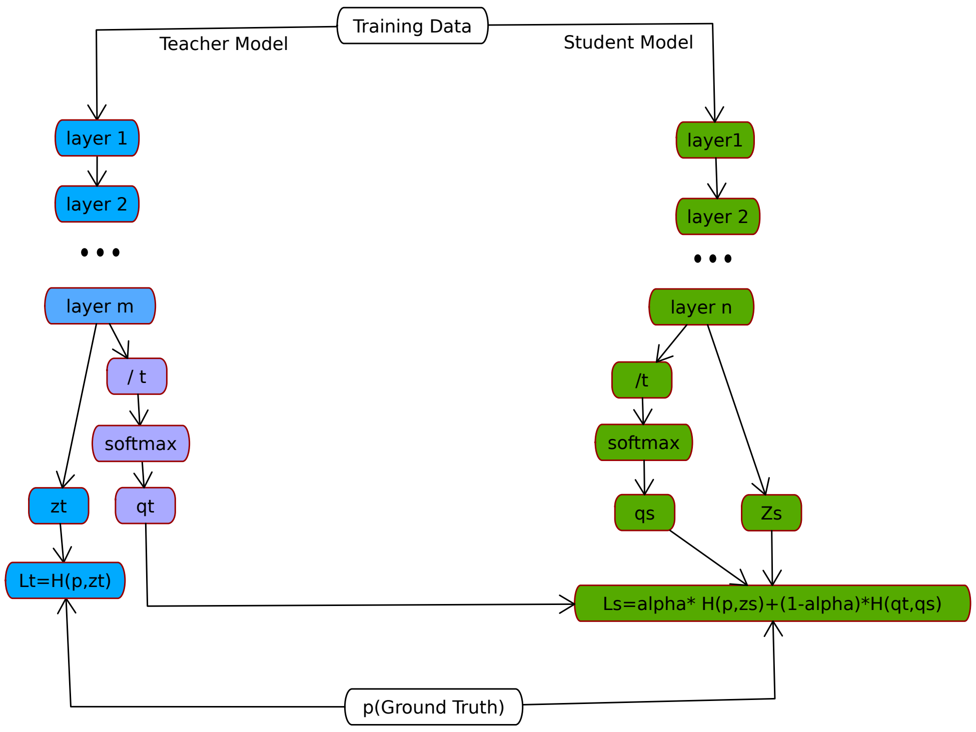


Figure 14. Teacher-Student learning process

CHAPTER FOUR

RESULTS and DISCUSSION

4.1 Introduction

In this chapter, we present the result and discussion of our proposed system. First, we present the performance comparison between the TM and the SM, and then we present the performance comparison of our proposed model and some prior related approaches at last. The implemented results were carried out using a python programming language in Kaggle's notebook, imploring Keras, Tensorflow, and sci-kit-learn frameworks. Kaggle is a cloud service provider that offers free memory and computational power to run and process deep learning programs. It has an intel Xeon 2.2 processor with 16.4 GB RAM and 220GB disk space.

4.2 Experimental Result

4.2.1 Comparison of TM and SM Results

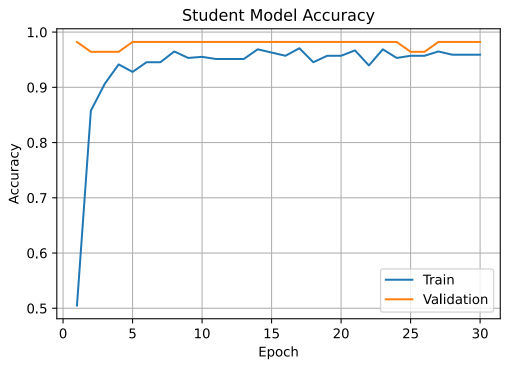
In this section, we used Kaggle's Wisconsin Diagnostic Breast Cancer (WDBC) dataset [52]. The dataset contains some features of the breast cell nuclei from digital images of a Fine Needle Aspirate (FNA) [52]. It has 569 cases; 212 are malignant (cancer), and 357 are benign (non-cancer).

We evaluated the performance of the TM and SM CNN classifier using 10-fold cross-validation discussed in subsection 3.2.3. The average classification accuracy of SM was found to be 98.07 $\pm 1.99\%$ and 97.54 $\pm 1.95\%$ for the TM. Table 4 shows the model evaluation metrics of our model, it follows that a larger model can be made lighter with less deployment time and without compromising accuracy.

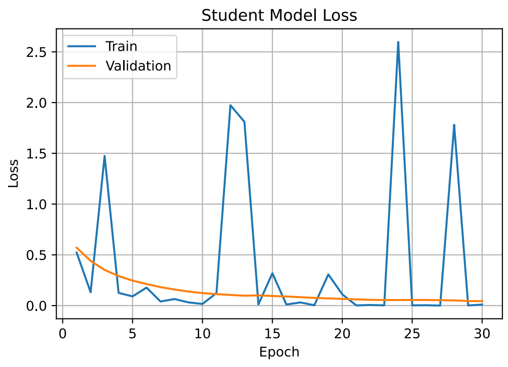
Table 4. Model Evaluation Metric

	Average Accuracy	Parameters	Hidden Layers	Running Time
TM	97.54%	863042	3	130 sec
SM	98.07%	582	2	70 sec

In addition, we plotted the accuracy and loss curves for training and validation of the TM and the SM using the model evaluation score of the last iteration. The accuracy curves show that more training implies better accuracy. Given that we repeat our training and validation 30 times (30 epochs), both training and validation accuracy become stable at greater accuracy than 97%. Figure 15 - 17 shows the accuracy and loss plot of SM and TM.

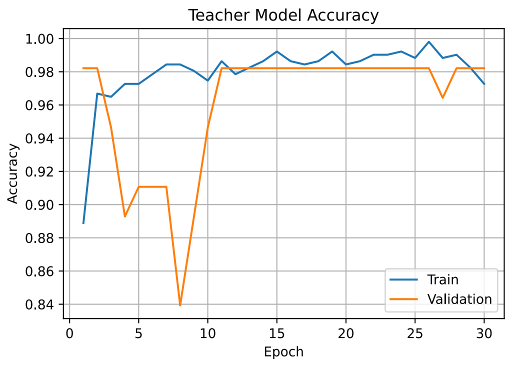


(a) Student Model Accuracy

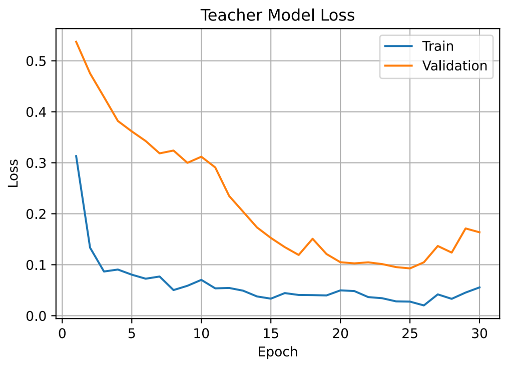


(b) Student Model Loss

Figure 15. SM Accuracy and Loss

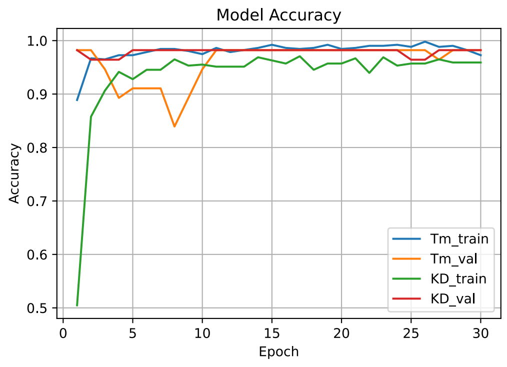


(a) Teacher Model Accuracy

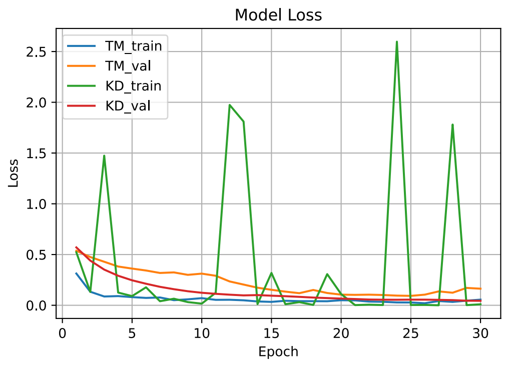


(b) Teacher Model Loss

Figure 16. TM Accuracy and Loss



(a) Teacher-Student Model Accuracy



(b) Teacher-Student Model Loss

Figure 17. TM-SM Accuracy and Loss

4.2.2 Comparison of SM Results with Prior Approaches

This section compares our work with the works in the literature. In addition to the WDBC dataset used in Section 4.2.1, we tested the proposed system with two more datasets: Breast Cancer Diagnosis (BCD) [53] and "Primary Breast Cancer vs Normal Breast Tissue (PBCT)" [54]. Adding these datasets allows us to determine the reliability of the system's performance. Most of the existing works have not used cross-validation for model evaluation. Thus, we present student model-2 (SM2) to have a fair comparison — it is SM without the cross-validation technique. We use 85% of our preprocessed dataset for training and 15% for validation. In a separate set of experiments, we used KMeansSmote oversampling on all our datasets to have balanced datasets. Table 5 shows the nomenclature used for all the experiments in this section.

Table 5. Nomenclature for Expriments Carried out

Experiments	Description
SM2-WDBC	WDBC Dataset
SM2-WDBC-O	WDBC Dataset with oversampling
SM2-BCD	BCD Dataset
M2-BCD-O	BCD Dataset with oversampling
SM2-PBCT	PBCT Dataset
SM2-PBCT-O	PBCT Dataset with oversampling

We evaluated the performance using different performance metrics explained in Equations 4.1 – 4.3 in addition to the Area Under the ROC Curve (AUC). A Receiver Operating Characteristics (ROC) curve is a graph that shows a model's classification performance at various classification thresholds. Therefore, an AUC reflects a classifier's reliability because it provides an aggregate performance measure across all possible classification thresholds.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{4.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

$Where,$

True positive (TP)= number of cancer that are correctly predicted as cancer

False Positive (FP) = number of not-cancer that are predicted as cancer

True Negative (TN) = number of not-cancer that are classified as not-cancer.

False Negative (FN) = number of cancer that are predicted as not-cancer.

Table 6. Performance comparison

Sn	Source	Algorithm	Accuracy(%)	Precision(%)	Recall(%)	AUC
1	SM2-WDBC	KD-CNN	98.8	100	97.0	0.98
2	SM2-BCD	KD-CNN	98.4	100	97.0	0.98
3	SM2-PBCT	KD-CNN	100.0	100	100	1.0
4	SM2-WDBC-O	KD-CNN	99.1	100	98.2	0.99
5	**SM2-BCD-O**	**KD-CNN**	**99.3**	**100**	**99.0**	**0.99**
6	SM2-PBCT-O	KD-CNN	97.3	100	95.0	0.97
7	[44]	DWT-ANN	98.8	98.3	98.3	-
8	[55]	SVM	97.2	0.99	0.95	0.97
		RF	96.5	0.97	0.92	0.96
		LR	95.8	0.98	0.95	0.95
		DT	95.1	0.90	0.92	0.95
		KNN	93.7	0.97	0.95	0.95
9	[56]	ANN	96.0	-	-	-

Table 6 compares the performance of some BC detection techniques in the literature with our proposed model. The results SM2-WDBC, SM2-BCD, and SM2-PBCT are from the SM2 model trained with WDBC, BDC, and PBCT datasets, respectively. The results show that SM2-PBCT is the best. However the dataset is highly imbalanced; only 20 of the 133 records are benign. Thus, we used oversampling to balance the datasets; SM2-WDBC-O, SM2-BCD-O, and SM2-PBCT-O. The results show that PBCT dataset is affected by accuracy paradox while WDBC and BCD shows more better performance after oversampling. Overall, SM2 is a good model as its tested with three different datasets and the accuracy ranges between 97-100%.

Masa et. al [44] used Discrete Wavelength Transformation - Artificial Neural Network (DWT-ANN) on BCD dataset. The authors used Discrete Wavelength Transformation techniques in their data preprocessing to ensure accurate results. It has similar performance with SM2-BCD. However, it is outperformed by SM2-BCD-O. This shows that oversampling gives better performance.

Naji et. al [55], and Hajiabadi et. al [56] trained and validated their model with WDBC dataset. In [55], SVM has the best performance, but it is marginally outperformed by both SM2-WDBC and SM2-WDBC-O. Similarly, [56] did not perform well. This shows that the light weight CNN had indeed learn from the complex ANN model (i.e., the TM).

CHAPTER FIVE

SUMMARY, CONCLUSION AND FUTURE WORK

5.1 Introduction

In this thesis, we proposed a CNN KD method for the lightweight detection of BC. This chapter concludes the thesis by summarizing our work, the method we have used to combat the problems, and the findings. Finally, we end the chapter by highlighting possible open research directions in this field.

5.2 Summary

A convolutional neural network is among the most effective algorithms used for finding patterns in data to recognize classes, objects, and categories. However, CNN architecture is large as it requires computing power (such as a GPU processor or an FPGA), among others. In a circumstance where real-time classification or recognition tasks need to be deployed in a resource constraint environment, lightweight versions of the CNN architectures that can be used on limited hardware are required. This work presented a lightweight CNN- Knowledge Distillation technique for the detection of BC, which can be used for embedded systems implementation. Knowledge Distillation is a model-agnostic technique that compresses and transfers knowledge from a computationally expensive deep neural network (TM) to a single shallow neural work (SM) with better inference efficiency.

We have sourced available datasets from an online source ([52], [53] and [54]), preprocessed them by cleaning, dropping, and applying Z-score normalization to bring the values of different features to a common scale. After the preprocessing, the TM architecture was built and trained with one input layer, three (3) hidden layers (two of which are convolutional layers(Conv1D) and a dense layer), and an output layer. To improve the performance of our TM, we used a grid search technique called hyper-band to identify the best parameters, such as the number of filters, number of hidden layers, learning rate, and dropout units. We applied the values that give the best performance to train and evaluate the teacher model.

The shallow SM architecture is built with only two convolutional (conv1D) hidden layers and a few filters. The total trainable parameters used by the SM [52] were only 0.06% of the training parameters used by the TM model. We applied a softmax function to the predictions of TM and obtained smooth and softer labels, which we used along with soft predictions of the SM to obtain the distillation loss using a Kullback–Leibler divergence loss function. In addition, we also obtained the hard prediction of the SM using the ground truth data. A weighted sum of the distillation loss and the student loss is then computed to evaluate the student model predictions. We found that our proposed system provides an average accuracy of 98.07±1.99% and reduced the training time of TM by up to 46.2%.

Moreover, we tested the proposed system with two more datasets: Breast Cancer Diagnosis (BCD) [53] and "Primary Breast Cancer vs Normal Breast Tissue (PBCT)" [54]. Adding these datasets allows us to determine the reliability of the system's performance. In [**bcd**] and [54], the SM used 0.12% and 0.11% trainable parameters of the TM. SM outperforms TM in both cases (see the performance of the SM in Table6).

5.3 Conclusion

The main aim of this work is to develop a lightweight CNN model that can mimic the performance of a deep neural network and accurately classify breast cancer data. The advantages of the proposed model include; obtaining a highly accurate model with a reduced running time, requiring few trainable parameters, and easy deployment on low-power computing devices (such as laptops, mobile phones, and tablets). Hence, this research can help pave the way for telemedicine in oncology.

5.4 Future Works

For future work, we propose deploying this model on the low computing power device for use in practice.

References

[1] Ahmed Mohammed and N Arunachalam. “Imbalanced Machine Learning Based Techniques for Breast Cancer Detection”. In: *2021 International Conference on System, Computation, Automation and Networking (ICSCAN)*. IEEE. 2021, pp. 1–4.

[2] Momna Hejmadi. *Introduction to cancer biology*. Bookboon, 2014.

[3] *About Breast Cancer*. 2022. URL: `https://www.cancer.org/content/dam/CRC/PDF/Public/8577.00.pdf`.

[4] World Health Ogranization. *Breast Cancer Statistics*. 2021. URL: `https://www.who.int/news-room/fact-sheets/detail/breast-cancer`.

[5] AKTH Kano Imam Mohammed Ibrahim; Consultant Pathologist Dept. Of Histopathology. *Pathology of breast.*

[6] Ganesh N Sharma et al. “Various types and management of breast cancer: an overview”. In: *Journal of advanced pharmaceutical technology & research* 1.2 (2010), p. 109.

[7] Fredika M Robertson et al. “Inflammatory breast cancer: the disease, the biology, the treatment”. In: *CA: a cancer journal for clinicians* 60.6 (2010), pp. 351–375.

[8] Xiaomin Zhou et al. “A comprehensive review for breast histopathology image analysis using classical and deep neural networks”. In: *IEEE Access* 8 (2020), pp. 90931–90956.

[9] Yash Amethiya et al. “Comparative analysis of breast cancer detection using machine learning and biosensors”. In: *Intelligent Medicine* 2.02 (2022), pp. 69–81.

[10] Seung Seog Han et al. “Augmented intelligence dermatology: deep neural networks empower medical professionals in diagnosing skin cancer and predicting treatment options for 134 skin disorders”. In: *Journal of Investigative Dermatology* 140.9 (2020), pp. 1753–1761.

[11] Yongwei Wang et al. “Ssd-kd: A self-supervised diverse knowledge distillation method for lightweight skin lesion classification using dermoscopic images”. In: *Medical Image Analysis* 84 (2023), p. 102693.

[12] 2022. URL: `https://neptune.ai/blog/knowledge-distillation`.

[13] Md Islam et al. “Breast cancer prediction: a comparative study using machine learning techniques”. In: *SN Computer Science* 1.5 (2020), pp. 1–14.

[14] 2022. URL: `https://www.cancer.net/cancer-types/breast-cancer/statistics`.

[15] Zohre Momenimovahed and Hamid Salehiniya. "Epidemiological characteristics of and risk factors for breast cancer in the world". In: *Breast Cancer: Targets and Therapy* 11 (2019), p. 151.

[16] Carol E DeSantis et al. "Breast cancer statistics, 2015: Convergence of incidence rates between black and white women". In: *CA: a cancer journal for clinicians* 66.1 (2016), pp. 31–42.

[17] Arthur L Samuel. "Machine learning". In: *The Technology Review* 62.1 (1959), pp. 42–45.

[18] 2021. URL: `https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML`.

[19] expert.ai. 2022. URL: `https://www.expert.ai/blog/machine-learning-definition/`.

[20] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. "Supervised learning". In: *Machine learning techniques for multimedia*. Springer, 2008, pp. 21–49.

[21] Rebala Gopinath, Ravi Ajay, and Churiwala Sanjay. *An Introduction to machine learning*. Springer Nature, Switzerland, 2019.

[22] Sinno Jialin Pan. "Transfer Learning." In: *Data Classification: Algorithms and Applications* 21 (2014).

[23] Tan pang-ning, Michael Steinbach, and kumar vipin kumar. *Introduction to data mining*. Pearson Education, Inc, 2006.

[24] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India, 2016.

[25] Allan David Gordon. *Classification*. CRC Press, 1999.

[26] Simon Haykin and N Network. "A comprehensive foundation". In: *Neural networks* 2.2004 (2004), p. 41.

[27] Imad A Basheer and Maha Hajmeer. "Artificial neural networks: fundamentals, computing, design, and application". In: *Journal of microbiological methods* 43.1 (2000), pp. 3–31.

[28] Kenneth J Hunt et al. "Neural networks for control systems—a survey". In: *Automatica* 28.6 (1992), pp. 1083–1112.

[29] Natalie Stephenson et al. "Survey of machine learning techniques in drug discovery". In: *Current drug metabolism* 20.3 (2019), pp. 185–193.

[30] Zonghan Wu et al. "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[31] Aggarwal Charu-C. *Neural Network and deep learning*. Springer International Publishing AG, part of Springer Nature 2018, 2018.

[32] 2019. URL: `Available%20at:%20https://towardsdatascience.com/shallow-neural-networks`.

[33] Muhammad Uzair and Noreen Jamil. "Effects of hidden layers on the efficiency of neural networks". In: *2020 IEEE 23rd international multitopic conference (INMIC)*. IEEE. 2020, pp. 1–6.

[34] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network". In: *2017 international conference on engineering and technology (ICET)*. Ieee. 2017, pp. 1–6.

[35] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* 2.7 (2015).

[36] Jang Hyun Cho and Bharath Hariharan. "On the efficacy of knowledge distillation". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 4794–4802.

[37] *Knowledge Distillation on NNI - Neural Network Intelligence*. 2022. URL: `https://nni.readthedocs.io/en/stable/sharings/kd_example.html`.

[38] Khandelwal Renu. *Knowledge Distillation in a Deep Neural Network*. 2021. URL: `https://medium.com/analytics-vidhya/knowledge-distillation-in-a-deep-neural-network-c9dd59aff89b`.

[39] Rikiya Yamashita et al. "Convolutional neural networks: an overview and application in radiology". In: *Insights into Imaging* 9.4 (Aug. 2018), pp. 611–629. ISSN: 1869-4101. DOI: `10.1007/s13244-018-0639-9`. URL: `https://doi.org/10.1007/s13244-018-0639-9`.

[40] S Prasath Alias Surendhar and RJMTP Vasuki. "Breast cancers detection using deep learning algorithm". In: *Materials Today: Proceedings* (2021).

[41] Puja Gupta and Shruti Garg. "Breast cancer prediction using varying parameters of machine learning models". In: *Procedia Computer Science* 171 (2020), pp. 593–601.

[42] Anji Reddy Vaka, Badal Soni, and Sudheer Reddy. "Breast cancer detection by leveraging Machine Learning". In: *ICT Express* 6.4 (2020), pp. 320–324.

[43] Ronglin Gong et al. "Self-Distilled Supervised Contrastive Learning for diagnosis of breast cancers with histopathological images". In: *Computers in Biology and Medicine* (2022), p. 105641.

[44] Emmanuel Masa-Ibi and Rajesh Prasad. “Breast Cancer Classification Using Discrete Wavelet Transformation and Deep Learning”. In: *Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science)* 14.7 (2021), pp. 2103–2112.

[45] Junho Yim et al. “A gift from knowledge distillation: Fast optimization, network minimization and transfer learning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4133–4141.

[46] Sajjad Abbasi et al. “Modeling teacher-student techniques in deep neural networks for knowledge distillation”. In: *2020 International Conference on Machine Vision and Image Processing (MVIP)*. IEEE. 2020, pp. 1–6.

[47] Ramik Rawal. “Breast cancer prediction using machine learning”. In: *Journal of Emerging Technologies and Innovative Research (JETIR)* 13.24 (2020), p. 7.

[48] Hugo S Oliveira, João F Teixeira, and Hélder P Oliveira. “Lightweight deep learning pipeline for detection, segmentation and classification of breast cancer anomalies”. In: *International Conference on Image Analysis and Processing*. Springer. 2019, pp. 707–715.

[49] Ali Bou Nassif et al. “Breast cancer detection using artificial intelligence techniques: A systematic literature review”. In: *Artificial Intelligence in Medicine* (2022), p. 102276.

[50] Chen Xi, Xing Zhiqiang, and Cheng Yuyang. “Introduction to Model Compression Knowledge Distillation”. In: *2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP)*. IEEE. 2021, pp. 1464–1467.

[51] Hao Ni, Jie Shen, and Chong Yuan. “Enhanced knowledge distillation for face recognition”. In: *2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*. IEEE. 2019, pp. 1441–1444.

[52] 2022. URL: `https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data`.

[53] UCI. *Breast Cancer Diagnosis*. 2022. URL: `https://www.kaggle.com/datasets/thedevastator/uncovering-breast-cancer-diagnosis-with-wisconsi`.

[54] N Matamala et al. *Primary breast cancer vs Normal breast tissue*. 2015. URL: `https://www.kaggle.com/datasets/rhostam/primary-breast-cancer-vs-normal-breast-tissue`.

[55] Mohammed Amine Naji et al. “Machine learning algorithms for breast cancer prediction and diagnosis”. In: *Procedia Computer Science* 191 (2021), pp. 487–492.

[56] Hamideh Hajiabadi et al. “Combination of loss functions for robust breast cancer prediction”. In: *Computers & Electrical Engineering* 84 (2020), p. 106624.

Appendix 1

Implementation Code

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
from matplotlib import cm
import seaborn as sns
import os
print(os.listdir("../input"))
%matplotlib inline
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv1D, MaxPool1D,Flatten,Dense,Dro
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import utils

#Import models from scikit learn module:
from sklearn import datasets,metrics
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import VarianceThreshold


#load dataset
data = pd.read_csv("../input/breast-cancer-wisconsin-data/data.csv",hea
data.head()

Y=data.diagnosis
```

```
print(Y.value_counts())
plt.title('Count␣of␣cancer␣type')
sns.countplot(data['diagnosis'])
plt.ylabel('Count')
plt.show()

Y=Y.map({'B':0,'M':1})
Y = utils.to_categorical(Y, num_classes=2)

data.isnull().any().describe()
data.info()

#drop id and un named 32 colums from the features
data.drop(['id','Unnamed:␣32'],axis=1,inplace=True)
data.describe()

correlation=data.corr()
# Getting the Upper Triangle of the co-relation matrix
matrix = np.triu(correlation)
plt.figure(figsize=(40,16))
sns.heatmap(correlation, vmax=1, square=True,annot=True,cmap='copper',m
plt.title('Correlation␣between␣different␣fearures')
plt.savefig("cor.svg")

for i in (data.columns[1:6]):
    plt.subplot(1,2,1)
    data[i][data['diagnosis']=='B'].plot.hist(alpha=0.5,title=i,color='
    data[i][data['diagnosis']=='M'].plot.hist(alpha=0.5,color='red')
    plt.legend(['B','M'],loc='upper␣right')
    #plt.grid(visible=True)


    plt.subplot(1,2,2)
    sns.boxplot(x="diagnosis", y=i, data=data)
    plt.show()


#data.drop('diagnosis',axis=1,inplace=True)
X=data.iloc[:,1:]
```

```
del data

scaler=StandardScaler()
data = scaler.fit_transform(data)
data=data.reshape(569,30,1)


#construct distiller class
class Distiller(keras.Model):
    def __init__(self, student, teacher):
        super(Distiller, self).__init__()
        self.teacher = teacher
        self.student = student

    def compile(
        self,
        optimizer,
        metrics,
        student_loss_fn,
        distillation_loss_fn,
        alpha=0.1,
        temperature=3,
    ):
        """ Configure the distiller.

        Args:
            optimizer: Keras optimizer for the student weights
            metrics: Keras metrics for evaluation
            student_loss_fn: Loss function of difference between studen
                predictions and ground-truth
            distillation_loss_fn: Loss function of difference between s
                student predictions and soft teacher predictions
            alpha: weight to student_loss_fn and 1-alpha to distillation
            temperature: Temperature for softening probability distribu
                Larger temperature gives softer distributions.
        """
        super(Distiller, self).compile(optimizer=optimizer, metrics=met
        self.student_loss_fn = student_loss_fn
        self.distillation_loss_fn = distillation_loss_fn
        self.alpha = alpha
        self.temperature = temperature

    def train_step(self, data):
        # Unpack data
        x, y = data

        # Forward pass of teacher
        teacher_predictions = self.teacher(x, training=False)

        with tf.GradientTape() as tape:
            # Forward pass of student
            student_predictions = self.student(x, training=True)

            # Compute losses
            student_loss = self.student_loss_fn(y, student_predictions)
            distillation_loss = self.distillation_loss_fn(
                tf.nn.softmax(teacher_predictions / self.temperature, a
                tf.nn.softmax(student_predictions / self.temperature, a
            )
            loss = self.alpha * student_loss + (1 - self.alpha) * disti

        # Compute gradients
        trainable_vars = self.student.trainable_variables
        gradients = tape.gradient(loss, trainable_vars)

        # Update weights
        self.optimizer.apply_gradients(zip(gradients, trainable_vars))

        # Update the metrics configured in `compile()`.
        self.compiled_metrics.update_state(y, student_predictions)

        # Return a dict of performance
        results = {m.name: m.result() for m in self.metrics}
        results.update(
            {"student_loss": student_loss, "distillation_loss": distilla
        )
        return results
    def test_step(self, data):
        # Unpack the data
        x, y = data

        # Compute predictions
        y_prediction = self.student(x, training=False)

        # Calculate the loss
        student_loss = self.student_loss_fn(y, y_prediction)

        # Update the metrics.
        self.compiled_metrics.update_state(y, y_prediction)

        # Return a dict of performance
        results = {m.name: m.result() for m in self.metrics}
        results.update({"student_loss": student_loss})
        return results

    def call(self, x):
        return self.student(x)


# Define the K-fold Cross Validator
# Define per-fold score containers <-- these are new
import time
T_acc_per_fold = []
T_loss_per_fold = []
TM_time=[]

S_acc_per_fold = []
S_loss_per_fold = []
SM_time=[]

TS_acc_per_fold = []
TS_loss_per_fold = []
KD_time=[]
num_folds=10
epoch=30
kfold = KFold(n_splits=num_folds, shuffle=True)
```

```
# K-fold Cross Validation model evaluation
fold_no = 1

for train, test in kfold.split(X, Y):
    # Create the teacher
    teacher = keras.Sequential(
    [
        keras.Input(shape=(30, 1)),
        layers.Conv1D(filters=64,kernel_size= 2,activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.2),
        #layers.LeakyReLU(alpha=0.2),

        layers.Conv1D(448, 2,activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.2),

        layers.Flatten(),
        layers.Dense(64,activation='relu'),
        layers.Dropout(0.2),

        layers.Dense(2,activation='sigmoid'),
    ],
    name="teacher",
    )

    # Create the student
    student = keras.Sequential(
    [
        keras.Input(shape=(30, 1)),
        layers.Conv1D(4, 2),
        layers.BatchNormalization(),
        layers.Dropout(0.2),
       # layers.LeakyReLU(alpha=0.2),

        layers.Conv1D(8, 2),
        layers.BatchNormalization(),
        layers.Dropout(0.2),
        layers.Flatten(),
        layers.Dense(2, activation='sigmoid'),
    ],
    name="student",)

    # Clone student for later comparison
    student_scratch = keras.models.clone_model(student)

    #compile teacher
    teacher.compile(optimizer=Adam(learning_rate=0.0001),loss='binary_c
              metrics=['accuracy'])

    # Generate a print
    print( sep='\n')
    print( sep='\n')
    print('--------------------+++++++++++++++++++++++++++++----------
    print( sep='\n')
    print(f'Training␣for␣fold␣{fold_no}␣...')
    print( sep='\n')
    print('TEACHER')
    start_time_TM = time.time()
    # Fit data to model
    teacherHistory = teacher.fit(X[train], Y[train],epochs=epoch, valida
              #batch_size=batch_size,

              verbose=1)
    end_time_TM = time.time()
    # Generate generalization
    scores = teacher.evaluate(X[test], Y[test])
    print(f'Score␣for␣fold␣{fold_no}:␣{teacher.metrics_names[0]}␣of␣{sc
    T_acc_per_fold.append(scores[1] * 100)
    T_loss_per_fold.append(scores[0])
    TM_time.append(end_time_TM - start_time_TM)

    print("Execution␣time:␣", end_time_TM - start_time_TM,"secs")
```

```
#compile STUDENT
#student_scratch.compile(optimizer=Adam(learning_rate=0.0001),loss=
 #            metrics=['accuracy'])
#print( sep='\n')
#print('STUDENT')
#start_time_SM = time.time()
#studentHistory=student_scratch.fit(X[train],Y[train],epochs=epoch,
#end_time_SM = time.time()
# Generate generalization
#Scores= student_scratch.evaluate(X[test],Y[test])
#print(f'Score for fold {fold_no}: {student_scratch.metrics_names[0
#S_acc_per_fold.append(Scores[1] * 100)
#S_loss_per_fold.append(Scores[0])
#SM_time.append(end_time_SM - start_time_SM)

#print("Execution time: ", end_time_SM - start_time_SM,"secs")



# Initialize and compile distiller
distiller = Distiller(student=student, teacher=teacher)
distiller.compile(optimizer=keras.optimizers.Adam(),
metrics=['accuracy'],
student_loss_fn= keras.losses.BinaryCrossentropy() ,
distillation_loss_fn=keras.losses.KLDivergence(),
alpha=0.1,
temperature=10,
)


# Distill teacher to student
print( sep='\n')
print('DISTILL␣TEACHER␣TO␣STUDENT')
start_time_KD = time.time()
history=distiller.fit(X[train], Y[train], epochs=epoch, verbose=1,
end_time_KD = time.time()
# Generate generalization
Scores= distiller.evaluate(X[test],Y[test])
#print(Scores)
```

```
    print(f'Score␣for␣fold␣{fold_no}:␣loss␣of␣{Scores[1]};␣{distiller.m
    #print(distiller.metrics_names)
    TS_acc_per_fold.append(Scores[0] * 100)
    TS_loss_per_fold.append(Scores[1])
    KD_time.append(end_time_KD - start_time_KD)

    print("Execution␣time:␣", end_time_KD - start_time_KD,"secs")



    # Increase fold number
    fold_no = fold_no + 1



print( sep='\n')
# == Provide average scores ==
print('------------------------------------------------------------
print('Teachers␣Score␣per␣fold')
for i in range(0, len(T_acc_per_fold)):
  print('----------------------------------------------------------
  print(f'>␣Fold␣{i+1}␣-␣Loss:␣{T_loss_per_fold[i]}␣-␣Accuracy:␣{T_acc_
  print('Running␣time:',TM_time[i],"secs")
print('------------------------------------------------------------
print('Teachers␣Average␣scores␣for␣all␣folds:')
print(f'>␣Accuracy:␣{np.mean(T_acc_per_fold)}␣(+-␣{np.std(T_acc_per_fol
print(f'>␣Loss:␣{np.mean(T_loss_per_fold)}')
print(sep='\n')
print('------------------------------------------------------------
print('Average␣running␣time:',np.mean(TM_time))

print( sep='\n')
print( sep='\n')
# == Provide average scores ==
#print('-----------------------------------------------------------
#print('Students Score per fold')
#for i in range(0, len(S_acc_per_fold)):
 # print('----------------------------------------------------------
  #print(f'> Fold {i+1} - Loss: {S_loss_per_fold[i]} - Accuracy: {S_acc
  #print('Running time:',SM_time[i],"secs")
#print('-----------------------------------------------------------
#print('Students Average scores for all folds:')
#print(f'> Accuracy: {np.mean(S_acc_per_fold)} (+- {np.std(S_acc_per_fo
#print(f'> Loss: {np.mean(S_loss_per_fold)}')
#print('-------------------------------------------------------------------
#print('Average running time:',np.mean(SM_time))


#print( sep='\n')
print( sep='\n')
# == Provide average scores ==
print('-------------------------------------------------------------------
print('knowledge␣distill␣Score␣per␣fold')
for i in range(0, len(TS_acc_per_fold)):
  print('-----------------------------------------------------------------
  print(f'>␣Fold␣{i+1}␣-␣Loss:␣{TS_loss_per_fold[i]}␣-␣Accuracy:␣{TS_ac
  print('Running␣time:',KD_time[i],"secs")
print('-------------------------------------------------------------------
print('knowledge␣distill␣Average␣scores␣for␣all␣folds:')
print(f'>␣Accuracy:␣{np.mean(TS_acc_per_fold)}␣(+-␣{np.std(TS_acc_per_f
print(f'>␣Loss:␣{np.mean(TS_loss_per_fold)}')
print('-------------------------------------------------------------------
print('Average␣running␣time:',np.mean(KD_time))




teacher.summary()
student.summary()

# teacher cm
#Make predictions
y_probs=teacher.predict(X[test])

#Convert prediction probabilities into integers
y_preds = y_probs.argmax(axis=1)

#Confusion matrix
cm=metrics.confusion_matrix(Y[test].argmax(axis=1),y_preds)
```

```
#Plot
disp=ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['B', 'M'
fig, ax = plt.subplots(figsize=(5,5))
disp.plot(ax=ax);

# student cm
#Make predictions
y_probs=distiller.predict(X[test])

#Convert prediction probabilities into integers
y_preds = y_probs.argmax(axis=1)

#Confusion matrix
cm=metrics.confusion_matrix(Y[test].argmax(axis=1),y_preds)
#Plot
disp=ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['B', 'M'
fig, ax = plt.subplots(figsize=(5,5))
disp.plot(ax=ax);

# plots of accuracy and loss


# accuracy and loss of teacher model
def plotLearningCurve(history,epochs):
  epochRange = range(1,epochs+1)
  plt.plot(epochRange,history.history['accuracy'])
  plt.plot(epochRange,history.history['val_accuracy'])
  plt.title('Teacher␣Model␣Accuracy')
  plt.xlabel('Epoch')
  plt.ylabel('Accuracy')
  plt.legend(['Train','Validation'],loc='lower␣right')
  plt.grid(visible=True)
  plt.savefig("TmAcc.svg")
  plt.show()



  plt.plot(epochRange,history.history['loss'])
  plt.plot(epochRange,history.history['val_loss'])
  plt.title('Teacher␣Model␣Loss')
```

```
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend(['Train','Validation'],loc='upper right')
    plt.grid(visible=True)
    plt.savefig("TmLoss.svg")
    plt.show()

plotLearningCurve(teacherHistory,epoch)

# Student knowledge distilled accuracy and loss
def plotKDCurveD(history,epochs):
    epochRange = range(1,epochs+1)
    plt.plot(epochRange,history.history['accuracy'])
    plt.plot(epochRange,history.history['val_accuracy'])
    plt.title('Student Model Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend(['Train','Validation'],loc='lower right')
    plt.grid(visible=True)
    plt.savefig("KdAcc.svg")
    plt.show()

    plt.plot(epochRange,history.history['student_loss'])
    plt.plot(epochRange,history.history['val_student_loss'])
    plt.title('Student Model Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend(['Train','Validation'],loc='upper left')
    plt.grid(visible=True)
    plt.savefig("KdLoss.svg")
    plt.show()

plotKDCurveD(history,epoch)

# Accuracy and loss of Teacher vs Student KD model
def plotKDCurveProf(his,tHis,epochs):
    epochRange = range(1,epochs+1)
    plt.plot(epochRange,tHis.history['accuracy'])
    plt.plot(epochRange,tHis.history['val_accuracy'])
    plt.plot(epochRange,his.history['accuracy'])
    plt.plot(epochRange,his.history['val_accuracy'])

    plt.title('Model␣Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend(['Tm_train','Tm_val','KD_train','KD_val'],loc='lower␣right
    plt.grid(visible=True)
    plt.savefig("Tm-SmAcc.svg")
    plt.show()


    plt.plot(epochRange,tHis.history['loss'])
    plt.plot(epochRange,tHis.history['val_loss'])
    plt.plot(epochRange,his.history['student_loss'])
    plt.plot(epochRange,his.history['val_student_loss'])
    plt.title('Model␣Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend(['TM_train','TM_val','KD_train','KD_val'],loc='upper␣left'
    plt.grid(visible=True)
    plt.savefig("Tm-SmLoss.svg")
    plt.show()

plotKDCurveProf(history,teacherHistory,epoch)
```