ADVANCING INDUSTRIAL ENGINEERING IN NIGERIA THROUGH TEACHING, RESEARCH AND INNOVATION

A BOOK OF READING

Edited By
Ayodeji E. Oluleye
Victor O. Oladokun
Olusegun G. Akanbi

ADVANCING
INDUSTRIAL ENGINEERING IN NIGERIA
THROUGH
TEACHING, RESEARCH AND INNOVATION

Edited By
Ayodeji E. Oluleye
Victor O. Oladokun
Olusegun G. Akanbi

ADVANCING
INDUSTRIAL ENGINEERING IN NIGERIA
THROUGH
TEACHING, RESEARCH AND INNOVATION

(A Festchrift in honour of Professor O. E Charles-Owaba)

Professor O. E. Charles-Owaba

Advancing Industrial Engineering in Nigeria through Teaching, Research and Innovation.

ISBN : 978-078-515-9

Published by

Department of Industrial and Production Engineering
University of Ibadan.

Printed by:
Leading Edge Printers & Publisher
Ibadan

FOREWORD

It gives me great pleasure writing the foreword to this book. The book was written in recognition of the immense contributions of one of Nigeria's foremost industrial engineers, respected teacher, mentor, and lover of youth – Professor Oliver Charles-Owaba.

His commitment to the teaching and learning process, passionate pursuit of research and demonstration of excellence has prompted his colleagues and mentees to write this book titled – Advancing Industrial Engineering in Nigeria through Teaching, Research and Innovation (A Festschrift in honour of Professor O. E Charles-Owaba) as a mark of honour, respect and recognition for his personality and achievements.

Professor Charles-Owaba has written scores of articles and books while also consulting for a medley of organisations. He has served as external examiner to various programmes in the tertiary educational system. The topics presented in the book cover the areas of Production/Manufacturing Engineering, Ergonomics/Human Factors Engineering, Systems Engineering, Engineering Management, Operations Research and Policy. They present the review of the literature, extension of theories and real-life applications. These should find good use in the drive for national development.

Based on the above, and the collection of expertise in the various fields, the book is a fitting contribution to the corpus of knowledge in industrial engineering. It is indeed a befitting gift in honour of erudite Professor Charles-Owaba.

I strongly recommend this book to everyone who is interested in how work systems can be made more productive and profitable. It represents a resourceful compilation to honour a man who has spent the last forty years building up several generations of industrial engineers who are part of the process to put Nigeria in the rightful seat in the comity of nations. Congratulations to Professor Charles-Owaba, his colleagues and mentees for this festschrift.

Professor Godwin Ovuworie
Department of Production Engineering
University of Benin

TABLE OF CONTENTS

CHAPTER 8

Application of Deep Learning in Disease Prediction

S. C. Nwaneri[1] * and C. O. Anyaeche[2]
[1,] Department of Biomedical Engineering, Faculty of Engineering, University of Lagos, Akoka, Lagos State, Nigeria
[2,] Department of Industrial & Production Engineering, University of Ibadan, Oyo State, Nigeria
*Corresponding Author
E-mail addresses: [1]snwaneri@unilag.edu.ng, [2]osita.anyaeche@ui.edu.ng

1. Overview of Deep Learning

1.1 Definition of Deep Learning

Traditional computers are limited by their inability to reason like humans. With advancements in the field of Artificial Intelligences (AI), modern computers can be programmed to reason like humans with the ability to understand patterns and take decisions based on available information. An essential component of AI is machine learning (ML), generally defined as a field of AI that finds patterns from empirical data using algorithms (Wittek, 2014). Deep Learning (DL) is a very important branch of ML composed of artificial neural networks (ANNs).

In ANNs, learning could be deep or shallow. DL is a term that refers to a class of ML algorithms mostly ANNs that makes use of multiple layers in the extraction and transformation of higher level features from the raw input (Deng and Yu, 2012). DL involves learning with deep architectures. Deep learning originated based on the need to overcome the limitations of shallow architectures. Shallow architectures are not fully able to handle complex real-life problems due to their limited modelling and representation power. The substantial increase in the amount of data available, as well as improved computational power have necessitated the use of DL in recent times as traditional ML methods are incapable of handling big data efficiently (Aggarwal, 2018). The performance of DL

networks is further aided by graphics processing units (GPUs) in modern computers. DL proves very useful when simpler networks are unable to give the desired level of accuracy.

Interestingly, DL can be applied in several areas of human life such as in: speech and language processing, programming of driverless cars, customer relationship management, detection of financial fraud, healthcare and bioinformatics. For instance, in driverless cars, a huge amount of data can be fed into DL system which could be trained to function autonomously having learnt from experience.

1.2 Classical Deep Learning Neural Networks

Classical DL networks are made up of several features similar to conventional multilayer ANNs. The distinguishing features include the number of layers, GPUs, and the type of activation functions that enhance their performance. Common examples of DL networks are briefly described.

i. **AlexNet** – A Deep Convolutional Neural Network (CNN) model that was developed to classify about 1.2 million high resolution images in the ImageNet Large Scale Visual Recognition Challenge (LSVRC) 2010 competition into the 1000 different classes. It was developed with the ReLU activation functions and consists 60 million parameters, 650,000 neurons in 8 layers (Krizhevsky *et al.*, 2012).

ii. **GoogleNet –**This is a Deep CNN developed through a collaborative research partnership between Google and some universities developed for image classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). The network is 22 layers deep and was developed using ReLU activation function. In addition, it performs dimension reduction 1×1 convolution with 128 filters (Szegedy *et al.*, 2015).

1.3 Activation Functions for Deep Learning

Activation functions enhance the ability of ANNs to handle complex problems by adding non-linearity to the network. The power of multilayer ANNs is increased by activation functions. Conventional

activation functions are generally used in the initial development of ANNs. A well-known example is the Sigmoid activation function. However, their performance is limited by the vanishing gradient problem (VGP). The VGP is responsible for the difficulty in training ANNs with many layers, as it can cause the gradient of the loss function to be too small for training.

The Sigmoid activation function is calculated thus: (Wang *et al.*, 2020):

$$f(x) = \frac{1}{1+e^{x}} \quad (1.1)$$

The derivative of the activation can be written as:

$$\frac{dy}{dx} = \frac{exp(-x)}{(1+ exp(-x))^{2}} \quad (1.2)$$

Tanh activation function is an improved version of Sigmoid activation function and is computed as:

$$f(x) = \frac{1-e^{-2x}}{1+e^{-2x}} \quad (1.3)$$

The gradient of Tanh activation function is computed as:

$$\frac{dy}{dx} = \frac{4.exp(2x)}{(\exp(2x)+1)^{2}} \quad (1.4)$$

However, in DL and most multi-layered neural networks **Rectified Linear Activation function (ReLU)** and Hard tanh activation functions are mostly preferred.

1.3.1 Rectified Linear Activation Function

This refers to a piecewise linear function commonly used in DL to generate an output that is exactly the value of the input if it is positive, otherwise, it will output zero. ReLU is mostly preferred in networks with many layers due to its ability to overcome the VGP and given as (Wang *et al.*, 2020):

$$\text{RELU(x)} = \begin{cases} 0 \ if \ x < 0 \\ x \ if \ x \geq 0 \end{cases} \quad (1.5)$$

1.3.2 Leaky RELU

The Leaky RELU activation function is designed to return the same value if the input is a *positive* number, but it gives a *negative* value scaled by

0.01, if the input value is negative. It was initially developed to address the dying neuron problem of RELU (Iuhaniwal, 2019).

1.3.3 Hard Sigmoid

The Hard-Sigmoid function can be approximated to generate improved outputs than the Sigmoid. It is defined mathematically as (Nwankpa *et al.*, 2018):

$$f(x) = \max\left(0, \min\left(1, \frac{(x+1)}{2}\right)\right) \quad (1.6)$$

1.3.4 Hard Tanh Activation Function

Hard Tanh is a variant of Tanh Function used in DL applications. It is computationally cheaper and more efficient than Tanh Function with a range of between -1 and +1 (Ingole, 2020).

1.4 Backpropagation Algorithm

Backpropagation algorithm is often used in training multilayer neural networks to minimise the errors between desired and actual outputs by adjusting its weights and biases. For this to be achieved, it computes the partial derivative of the cost function using the chain rule by a combination of the forward and backward pass approaches (Aggarwal, 2018). The forward pass through the network computes the overall output across the various layers of the network which is then compared with the desired output. The derivative of the cost function is then computed. The Backward pass follows immediately thereafter with the application of the chain rule to learn the gradient of the cost function in the backward direction beginning from the output node (Goodfellow and Courville, 2016). Consequently, the weights are adjusted. Consider a multilayer feedforward neural network with an input vector x, an activation function, N layers, h^{N-1} hidden layers, $W_{jk}^{(i)}$ weights from node j in layer N-1 to node k in layer N,

The output of the first hidden layer is calculated as:

$$h^{(1)} = wx + b \quad (1.7)$$

The cost function, C is computed as:

$$C = \frac{1}{N}\sum_{i=1}^{n}(y - \hat{y})^2 \quad (1.8)$$

The cost function is minimised using the Gradient descent algorithm.

$$\frac{\partial C}{\partial W^N} = \frac{\partial C}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial W^N} \quad (1.9)$$

The gradient in the penultimate layer is calculated as (Rawat and Wang, 2017):

$$\frac{\partial C}{\partial W^{N-1}} = \frac{\partial C}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h^{N-1}} \cdot \frac{\partial h^{N-1}}{\partial W^{N-1}} \quad (1.10)$$

The gradient in the layer before the penultimate layer is calculated as:

$$\frac{\partial C}{\partial W^{N-2}} = \frac{\partial C}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h^{N-2}} \cdot \frac{\partial h^{N-2}}{\partial W^{N-2}} \quad (1.11)$$

The gradient in the first layer is computed as:

$$\frac{\partial C}{\partial W^2} = \frac{\partial C}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h^2} \cdot \frac{\partial h^2}{\partial W^2} \quad (1.12)$$

1.5 Deep Learning Optimisation Algorithms

DL networks are trained to minimise the loss function using various optimisation algorithms. The most common are Gradient descent, Stochastic gradient descent (SGD), Newton's method, Batch stochastic gradient descent with momentum, Adagrad, Adaptive moment estimation (Adam), Adadelta and Root Mean Squared Propagation (RMSprop) (Sun and Scanlon, 2019).

1.5.1 Stochastic Gradient Descent

Stochastic Gradient Descent is an optimisation technique, that calculates the degree of change of a variable in response to the changes of another variable using a few randomly selected samples. In the conventional Gradient descent algorithm, the parameter θ, of the loss function, $J(\theta)$ is updated as (Sun *et al.*, 2018):

$$\theta = \theta - \alpha \nabla_\theta E[J(\theta)] \quad (1.13)$$

In equation (1.13), the expectation, E is obtained by estimating the cost and gradient on the entire training data. SGD eliminates the expectation in the update and calculating the gradient of the parameters using only a single or a few training examples as indicated in equation (1.14)

$$\theta = \theta - \alpha \nabla_\theta J\left(\theta; x^{(i)}, y^{(i)}\right) \quad (1.14)$$

1.5.2 Adaptive Moment Estimation (Adam)

The Adam optimisation algorithm is an adaptive moment estimation algorithm that combines the benefits of Adagrad and adaptive learning rates. Adaptive learning rates are considered to be modifications to the learning rate during the training phase by reducing it to a pre-defined value. Root Mean Squared Propagation, was designed to reduce the learning rate at a slower rate in relation to AdaGrad. The Adam optimisation algorithm developed by Kingma and Ba (2015) is shown in Figure 1.

Consider a stochastic scalar differentiable objective function f(θ) with each function $f_1(\theta),\ldots,,f_T(\theta)$ achieved at specific timesteps 1, ……T. The main aim is to minimise the expected value of the objective function, E[f(θ)].

The gradient at timestep, g_t is given by

$$g_t = \nabla\theta f_t(\theta) \qquad (1.15)$$

Let m_t represent exponential moving averages of the gradient, and v_t the squared gradient, where the hyper-parameters $\beta_1, \beta_2 \in [0, 1)$ control the exponential decay rates of these moving averages (Kingma and Ba, 2015).

The algorithm is summarised as:

Set initial parameter vector v_0
Initialise 1st moment vector, $m_0 \leftarrow 0$
Initialise 2nd moment vector, $v_0 \leftarrow 0$
Initialise timestep, $t \leftarrow 0$
While θ_t not converged do
$t \leftarrow t + 1$
Compute gradient, $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
Update biased first moment estimate, $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
Update biased second moment estimate, $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
Calculate biased corrected first moment estimate, $\hat{m}_t \leftarrow {m_t}/{(1 - \beta_1^t)}$
Calculate biased-corrected second moment estimate, $\hat{v}_t \leftarrow {v_t}/{(1 - \beta_2^t)}$
Update parameters, $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot {\hat{m}_t}/{(\sqrt{\hat{v}_t} + \epsilon)}$
end while
return θ_t (Resulting parameters)

Fig. 1. Adam Optimisation Algorithm (Kingma and Ba, 2015)

2. Deep Learning Types and Architectures

2.1 Classes of Deep Learning Networks

The three major classes of DL networks are:

i. Deep networks for supervised learning
ii. Deep networks for unsupervised learning
iii. Hybrid deep networks

2.1.1 Deep Networks for Supervised Learning

Supervised learning is a learning technique that provides a target or set of examples for the network to learn from. It is also known as learning from examples or learning from the teacher. This is achieved by labelling each input with a desired output value, thereby providing a set of target values of output (Masolo, 2017). Deep networks for supervised learning are also

known as discriminative deep networks as demonstrated by CNNs (Deng and Yu, 2012).

2.1.2 Deep Networks for Unsupervised Learning

In unsupervised deep learning, the deep network is only provided with a set of inputs with no target values. Learning is without the support of a teacher or examples. There is usually no feedback from the environment with regard to what should be the desired output and whether it is correct or incorrect. They are also referred to as generative networks because they have the ability to generate samples by sampling from the network. Common examples are autoencoders, deep belief networks and generative adversarial networks (Hinton *et al.*, 2012; Patterson and Gibson, 2020).

.

2.1.3 Hybrid Deep Networks

Hybrid deep networks have a combination of discriminant and generative abilities associated with both supervised and unsupervised learning. For example, a speech recognition is generative in nature, nevertheless it may have parameters that are discriminative in translating speech to text (Deng and Yu, 2012).

2.2 Deep Feed-Forward Networks

In Feedforward Neural Networks (FFNs), successive layers feed into one another in the forward direction from input to output (Aggarwal, 2018). This is based on the assumption that all nodes in one layer are connected to those of the next layer. FFNs are acyclic graphs. Fig. 2 shows an FFN with 4 inputs, 4 hidden layers and an output layer.

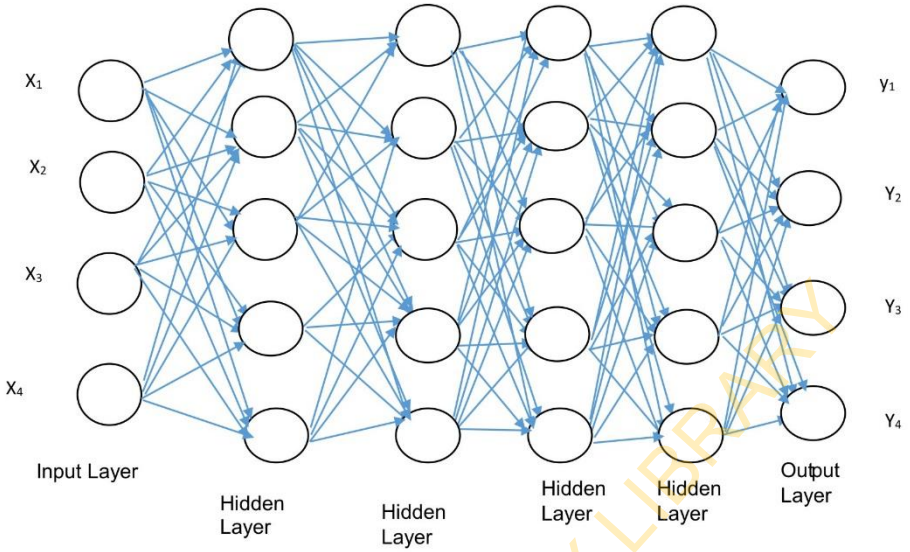


Fig. 2. Feedforward Neural Network

2.3 Deep Convolutional Neural Networks

Convolutional Neural Networks (CNNs) transform input data by p lution operations in a minimum of one layer of the network. Convolution involves the application of a matrix to the input image to generate modified filtered data. Deep CNNs have several layers, with each layer able to compute convolutional transforms, after that non-linearities and pooling operators (Wiatowski and Bolcske, 2017). A typical CNN consists basically of convolution, pooling and full-connection (FC) layers (Wang *et al.*, 2020). The convolution layer consists of neurons organised as feature maps that are trained to extract features from the input data. Each neuron in a feature map is connected to other surrounding neurons in the previous layer known as a filter bank. All neurons within a feature map have approximately equal weights. The pooling layer decreases the spatial resolution of the feature maps (Rawat and Wang, 2017). High-level reasoning and the interpretation of feature representations are the main functions of the full-connection layer (Hinton *et al.*, 2012). CNNs are mostly applied in image classification and considered to work with grid-structured inputs (Aggarwal, 2018). A

good activation function can efficiently map data in dimensions (Dubey and Jain, 2019). Figure 3 depicts a typical CNN architecture of an image classification problem. As the input image is fed into the network, the features are extracted at the convolution layer. The neurons at this layer form a feature map. The jth output Y_j of the feature map is given by:

$$Y_j = f(W_j * x) \quad (2.1)$$

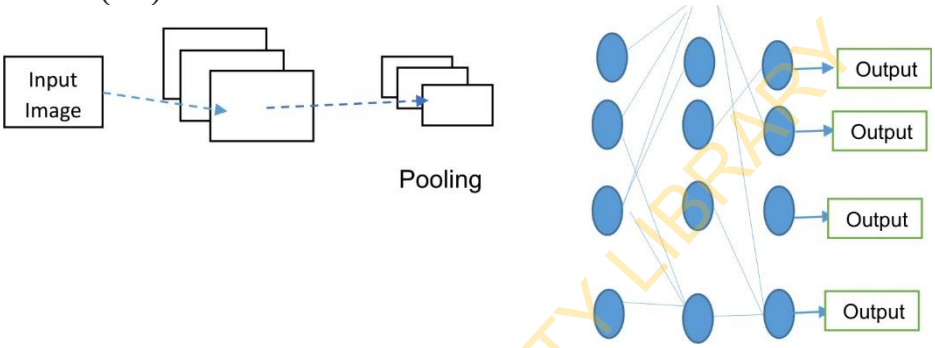


Fig. 3. Convolutional Neural Network Architecture (Rawat and Wang, 2017)

The next stage after convolution is pooling which aims to decrease the spatial resolution of the feature map (LeCun *et al.*, 2015). The output of the k^{th} feature of the map is given by:

$$Y_{kij} = \max x_{kpq} \quad (2.2)$$

Where x_{kpq} represents the element at location (p,q)

After the pooling operations, the fully connected layer performs the function of interpreting the features as well as high level reasoning (Hinton *et al.*, 2012). The Softmax operator is generally used for classification problems (Krizhevsky *et al.*, 2012).

2.4 Recurrent Neural Networks

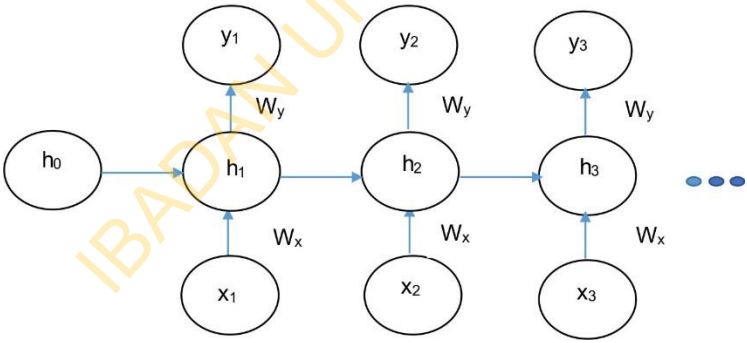
Recurrent Neural Networks (RNNs) are considered to accept a series of input with no fixed limit on size (Venkatachalam, 2019). RNNs are

described as “the deepest of all ANNs due to their computing power and ability to process sequential information” (Siegelmann and Sontag, 1991; Schmidhuber, 1990a). RNNs use the inputs of the previous layer and previous information observed by the individual neurons because they also have the ability of memorising previous inputs (Wang and Huang, 2019). The performance of RNNs in time series data is known to be better than other neural network architectures like CNNs. RNNs are therefore used in the classification of biological sequences Fig. 4 is a typical architecture of an RNN that shows the input, hidden and output layers of the network. Inputs are fed into the network at specific timestamps. The output $y^{(t)}$ of an RNN given input x^t at time is computed as (Torti *et al.*, 2019):

$$y^{(t)} = wg\left(Wx^{(t)} + Uh^{(t-1)} + b\right) + c \quad (2.2)$$

Where g is a nonlinear function while W, U, w, b and c represent network parameters. $h^{(t)}$ is the hidden state which is computed as:

$$h^{(t)} = g\left(Wx^{(t)} + Uh^{(t-1)} + b\right) + c \quad (2.3)$$



Fig, 4. Recurrent Neural Network

2.5 Deep Learning Implementation

Deep Learning data computations can be performed with a variety of open-source and propriety software available for deep learning application. We shall focus on the use of Python software, a simple open-source programming software. Implementation of DL in Python is achieved with the following important libraries Numpy, Pandas, Keras, Scikit-lean, matplotlib. The file containing the datasets should be in csv format. The file is loaded using pandas followed by data normalisation (Zhang, 2019). The Keras library runs on Tensorflow or Theano and is used to develop the models including defining the layers. Keras is a DL application programming interface (API) designed to enhance the speed of computation. The main data structures of Keras are **layers** and **models**.

i. Data Collection

DL networks are usually implemented with large datasets and loaded into the software.

ii. Data Pre-processing

a. Missing Values

Various techniques are used to handle missing data.

b. Covariance Matrix

Covariance matrix is computed to determine variables that are highly correlated. The covariance of two random variables X and Y having values (x_i, y_i) with probability p_{ij} is defined as:

$$COV(X,Y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_X)\left(y_j - \mu_Y\right) \quad (2.4)$$

c. Data Normalisation

Data normalisation is performed to ensure that all the features are measured on the same scale.:

iii. Feature extraction

Important features of the data are extracted leading to considerable reduction in the size of data to a controllable size and improved accuracy. Common feature selection techniques include correlation matrix, F-test, feature importance and variance thresholding.

iv. Splitting the dataset

The dataset is split into training and testing datasets. This is achieved with the train_test_split package imported from scikitilearn.

iv. **Building the model:**

This process involves creating the type of deep neural network, its network architecture and the activation functions to be used at the output of each layer of the network. Optimization technique to reduce cost function (error between the target output and the local output from each layer). Network structure is specified by importing dense from keras.layer (from keras.layers import Dense).

v. **Evaluating the model**

The performance of DNNs can be determined by the following evaluation metrics including, accuracy, sensitivity, specificity, positive predictive value, negative predictive values.

a. Accuracy is defined as the proportion of the total number of predictions that were correct.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.5)$$

b. Sensitivity is defined as proportion of positive class correctly classified.

$$\text{Sensitivity} = \frac{TP}{TP+FN} \quad (2.6)$$

c. Specificity is defined as the ratio of true negatives, rightly predicted as a negative class or true negative.

$$\text{Specificity} = \frac{TN}{TN+FP} \quad (2.7)$$

d. Positive predictive value or precision is defined as the ratio of true positives to the sum of true positive and false positives.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (2.8)$$

Where

$TP = \text{True positive value}$

TN = True negative value
FP = False positive value
FN = False negative value

3. Predictive Analytics in healthcare

In recent times, concerted efforts have been made to improve medical diagnosis through use of predictive analytics. This is borne out of the need to improve diagnostic accuracy and speed.

3.1 Overview of Predictive analytics

Predictive analytics are generally used to develop effective methods that can accurately predict future observations from historical data while also trying to understand the relationship between the features and response for scientific purposes (Bickel and Roy, 2008). The various applications of predictive analytics include forecasting of disease outbreaks and epidemics. disease diagnosis and therapy. The application of predictive analytics in big data has the potential not only to improve profits but also to minimise wastes in healthcare operations management. Deep learning as well as other machine learning algorithms are very useful tools in predictive analytics for disease classification. For instance, Nwaneri *et al.,* (2014) developed an ANN based model for the prediction of breast cancer.

3.2 Limitations of Predictive Analytics in Healthcare

Notwithstanding the several benefits associated with the use of predictive analytics in healthcare, it has the following limitations:

i. Data collection is often characterised by some challenges such as incomplete, inconsistent and heterogenous data.
ii. Data access is limited by privacy issues
iii. Predictive analytics of high dimensional data (data with too many attributes in relation to the number of observations) are usually susceptible to the "curse of dimensionality" problem (Dinov, 2016).
iv. Since large datasets is a major requirement for deep learning, the use of deep learning in healthcare is usually

limited by the availability of large amount of data from patients.

v. Deep neural networks are difficult to train as they are susceptible to overfitting.

3.3 Selected Deep Learning Applications in Disease Prediction

3.3.1 Application of Deep Learning in Diabetes Prediction

Diabetes mellitus (DM) is defined as "a chronic, metabolic disease characterized by elevated levels of blood glucose" (WHO, 2020). Therefore, early diagnosis and prediction of DM in persons with certain risk factors is necessary. Deep Learning algorithms can be used to develop predictive models for DM prediction based on certain risk factors of patients. In a typical DL model, the input nodes consist of risk factors such as age, weight, hypertension, family history of DM, waist circumference, consumption of sweet foods etc. The network is trained to help the network classify data as diabetic or non-diabetic. Various computations are done in the hidden layers with the results forwarded to the output layer which indicates if the person is diabetic or not.

3.3.2 Application of Deep Learning in Breast Cancer Classification

Breast cancer is a malignant tumour that affects mostly women. It accounts for 10.4% of all cancer incidences among women and is the fifth most common cause of cancer death globally (Sharma *et al.*, 2010). Early diagnosis of the disease is known to improve the chances of survival from the disease (Murtaza, 2019). Mammography is an imaging technique used in diagnosing breast cancer. However, diagnosis of mammographic breast images is subjective and difficult to interpret and requires a lot of skill and experience. Therefore, DL can be used in the prediction of the status of breast image to determine if it is benign or malignant. This is implemented by first digitizing breast images of fine needle aspirates of the breast mass of patients. Features of interest in the images are considered and their values computed. The features commonly used include uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, brand chromatin, and

normal nucleoli (Dhanya 2020). The data are usually pre-processed. The features values are usually fed as inputs to the DL model. Activation functions are used in the computation of the output of the various layers of the model. The model trains the input data and generates the output which may be classified as a benign or malignant tumour. The models are evaluated in terms of the accuracy, specificity, sensitivity and positive predictive value (PPV).

3.3.3 Application of Deep Learning in Chronic Kidney Disease Prediction

Chronic Kidney Disease (CKD) refers to kidney damage or glomerular filtration rate (GFR) $< 60\ \mathrm{mL/min/1.73\ m^2}$ for three (3) months or more (Inker et al., 2014). It is associated with high rates of mortality and morbidity. Some of the risk factors of CKD include: high blood pressure, old age, heart and blood vessel disease, smoking, diabetes, obesity, abnormal urinary structure and history of family of kidney disease. Early diagnosis and detection of CKD helps patients to know their health status and commence early treatment in order to improve chances of survival and quality of life of CKD patients. Deep Learning algorithms can be used for early prediction of CKD. Chen *et al.*, (2020) developed an Adaptive hybridised deep CNN for early detection of CKD.

3.4 Data Computations and Results

Example 1: What is the output of a rectified linear unit of Deep Convolutional Neural Network if the input is:

(a) -550 (b) 500

Solution:

$$\text{RELU(x)} = \begin{cases} 0 \ if \ x < 0 \\ x \ if \ x \geq 0 \end{cases}$$

Therefore, based on the above equation, the output is determined by the input. If the input is less than 0, the output is equal to zero. However, if the input is greater than or equal to zero, the output is equal to the input.

Therefore, (a) 0 (b) 500

Example 2: Consider a 4 by 4 matrix representing the patients and attributes in a dataset for the prediction of diabetes. Calculate the covariance between the 2nd (Age) and the 4th Column (Glucose).

Solution:

The covariance of two random variables X and Y having values (xi, yi) with probability pij is defined as:

$$COV(X,Y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_X)(y_j - \mu_Y)$$

$$A = \begin{matrix} 3 & 28 & 10 & 90 \\ 5 & 39 & 12 & 88 \\ 8 & 45 & 8 & 120 \\ 6 & 65 & 6 & 130 \end{matrix}$$

Let 2nd column = X, 4th column =Y

$$X = \begin{matrix} 28 \\ 39 \\ 45 \\ 65 \end{matrix} \qquad Y = \begin{matrix} 90 \\ 88 \\ 120 \\ 130 \end{matrix}$$

$E[X] = 44.25$

$E[Y] = 107$

$COV(X,Y)$

$$= \frac{(28-44.25)(90-107)+(39-44.25)(88-107)+(45-44.25)(120-}{4}$$

$$= 212.86$$

Question 3: Compute the image X by convolving the image I of size 6×6 with a 2 × 2 filter.

I =

1	8	4	7	8	2
9	1	21	10	3	7
2	1	2	8	17	9
18	5	10	5	5	3
5	12	5	9	22	7
1	6	7	11	6	5

*

F =

1	0
1	1

Solution

$(1 \times 1 + 8 \times 0 + 9 \times 1 + 1 \times 1) = 11$
$(8 \times 1 + 4 \times 0 + 1 \times 1 + 21 \times 1) = 30$
$(4 \times 1 + 7 \times 0 + 21 \times 1 + 10 \times 1) = 35$
$(7 \times 1 + 8 \times 0 + 10 \times 1 + 3 \times 1) = 20$
$(8 \times 1 + 2 \times 0 + 3 \times 1 + 7 \times 1) = 18$
$(9 \times 1 + 1 \times 0 + 2 \times 1 + 1 \times 1) = 12$
$.(1 \times 1 + 21 \times 0 + 1 \times 1 + 2 \times 1) = 4$
$(21 \times 1 + 10 \times 0 + 2 \times 1 + 8 \times 1) = 31$
$(10 \times 1 + 3 \times 0 + 8 \times 1 + 17 \times 1) = 35$
$(3 \times 1 + 7 \times 0 + 17 \times 1 + 9 \times 1) = 29$
$(2 \times 1 + 1 \times 0 + 18 \times 1 + 5 \times 1) = 25$
$(1 \times 1 + 2 \times 0 + 5 \times 1 + 10 \times 1) = 16$
$(2 \times 1 + 8 \times 0 + 10 \times 1 + 5 \times 1) = 17$
$(8 \times 1 + 17 \times 0 + 5 \times 1 + 5 \times 1) = 18$
$(17 \times 1 + 9 \times 0 + 5 \times 1 + 3 \times 1) = 25$
$(18 \times 1 + 5 \times 0 + 5 \times 1 + 12 \times 1) = 35$
$(5 \times 1 + 10 \times 0 + 12 \times 1 + 5 \times 1) = 22$
$(10 \times 1 + 5 \times 0 + 5 \times 1 + 9 \times 1) = 24$
$(5 \times 1 + 5 \times 0 + 9 \times 1 + 22 \times 1) = 36$
$(5 \times 1 + 3 \times 0 + 22 \times 1 + 7 \times 1) = 34$
$(5 \times 1 + 12 \times 0 + 1 \times 1 + 6 \times 1) = 12$
$(12 \times 1 + 5 \times 0 + 6 \times 1 + 7 \times 1) = 25$
$(5 \times 1 + 9 \times 0 + 7 \times 1 + 11 \times 1) = 23$
$(9 \times 1 + 22 \times 0 + 11 \times 1 + 6 \times 1) = 26$
$(22 \times 1 + 7 \times 0 + 6 \times 1 + 5 \times 1) = 33$

After convolution the image is transformed to:

11	30	35	20	18
12	4	31	35	29
25	16	17	18	25
35	22	24	36	34
12	25	23	26	33

Example 4:

A deep CNN was recently developed for breast cancer screening. The model was tested on a large dataset comprising 10,000 digitised breast images of women who participated in a recent screening conducted in 3 tertiary hospitals in Southwest, Nigeria. Of the 10,000 breast images, 9100 were benign while 900 were confirmed to be malignant from confirmed biopsy procedures. The model correctly classified 8,950 breast images as benign, and 780 as malignant. It wrongly classified 120 images as benign and 150 as malignant. Calculate:

i. The accuracy ii. Sensitivity iii. Specificity iv. Precision

Solution

Total number of breast images =10,000
Total number of positive cases = 9100
Total number of negative cases = 900
True positive cases = 780
False positive cases = 150
True negative cases = 8,950
False negative cases = 120

i. $\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} = \frac{\mathbf{780+8950}}{\mathbf{10000}} = 0.973$

ii. $\text{Sensitivity} = \frac{TP}{TP+FN} = \frac{780}{780+120} = 0.867$

iii. $\text{Specificity} = \frac{TN}{TN+FP} = \frac{8950}{8950+150} = 0.984$

iv. $Precision = \frac{\text{TP}}{\text{TP+FP}} = \frac{780}{780+150} = 0.839$

Example 5:
In a recent study, a deep learning model was used to predict CKD. The model was tested on a large dataset comprising 20,000 adults who participated in a nationwide screening conducted across the 6 geographical zones of Nigeria. Of the 200,000 subjects, 170,000 tested negative while 30,000 subjects were diagnosed with CKD. The model correctly classified 165,000 negative cases and 28000 positive cases. It misclassified 5000 samples as positive and 2000 as negative.

Solution
Total number of datasets =200,000
Total number of positive cases = 30,000
Total number of negative cases = 170,000
True positive cases = 28,000
False positive cases = 5,000
True negative cases = 165,000
False negative cases = 2,000

i. $\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} = \frac{28,000+165,000}{\mathbf{200000}} = 0.965$

ii. $\text{Sensitivity} = \frac{\text{TP}}{\text{TP+FN}} = \frac{28,000}{28,000+2000} = 0.933$

iii. $\text{Specificity} = \frac{\text{TN}}{\text{TN+FP}} = \frac{165,000}{165,000+5000} = 0.971$

iv. $\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} = \frac{28,000}{28,000+5,000} = 0.848$

4. Conclusion

Deep learning and its application in disease prediction have been discussed in this chapter. Deep learning was defined as a class of machine learning (ML) algorithms mostly artificial neural networks (ANNs) that make use of multiple layers in the extraction and transformation of higher level features from the raw input. Important deep learning concepts were extensively discussed and clearly illustrated with practical examples.

References

Aggarwal, C.C. (2018) Neural networks and deep learning. Springer International Publishing, NewYork.

Bickel, P.F. and Roy, L. J. editors. (2008) Discussion on the paper "Sure independence screening for ultrahigh dimensional feature space". *Journal of the Royal Statistical Society*: Series B 70(5):883–884.

Chapelle, O. Scholkopf, B. and Zien, A. (2009) Semi-supervised learning. *IEEE Transactions on Neural Networks*, 20(3):542–542.

Chen, G. Ding, C. Li, Y. Hu, X. *et al.*, (2020) Prediction of chronic kidney disease using adaptive hybridized deep convolutional neural network on the internet of medical things platform. *IEEE Access*, 8:100497-100508.

Deng, L. and Yu, D. (2012) Deep learning methods and applications. *Foundations and Trends in Signal Processing*, 7(3-4):198 -383.

Dhanya, R. Irene, R.P. Sai, S.A. Madhumathi, S. Jyothisha, J.N. (2020) F-Test feature selection in stacking ensemble model for breast cancer prediction. *Procedia Computer Science* 171:1561–1570.

Dinov, I.D. (2016) Methodological challenges and analytic opportunities for modelling and interpreting Big Healthcare Data. *Gigascience*, 5:12. Retrieved October 13, 2020, from

https://gigascience.biomedcentral.com/articles/10.1186/s13742-016-0117-6

Dubey, A.K. and Jain, V. (2019) Comparative Study of Convolution Neural Network's ReLu and Leaky-ReLu Activation Functions. *In Applications of Computing, Automation and Wireless Systems in Electrical Engineering;* Springer: Singapore, 873–880. 23.

Deng, L. Yu, D. (2014). Deep Learning: Methods and Applications. *Foundations and Trends in Signal Processing*. 7 (3–4):1–199.

Goodfellow, I. and Bengio, Y. and Courville, A. (2016) *Deep Learning*. MIT Press: Massachusetts.

Hinton, G. E. Srivastava, N. Krizhevsky, A. Sutskever, I. and Salakhutdinov, R. R.
(2012). Improving neural networks by preventing co-adaptation of feature detectors.
Retrieved October 12, 2020, from https://arxiv.org/pdf/1207.0580.pdf

Hinton, G. Krizhevsky, A. and Wang, S. (2011) Transforming autoencoders. *In Proceedings of International Conference on Artificial Neural Networks*, 44-51.

Ingole, K. (2020) Performance Analysis of Various Activation Functions. *Journal of Emerging Technologies and Innovative Research*, 7(6):269 -276.

Iuhaniwal, V. (2019) Analyzing different types of activation functions in neural networks — which one to prefer? Retrieved October 5, 2020 from https://towardsdatascience.com/analyzing-different-types-of-activation-functions-in-neural-networks-which-one-to-prefer-e11649256209

Kingma, D. and Ba, J. (2015) *Adam: A method for Stochastic Optimization*. Retrieved October 5, 2020 from https://arxiv.org/pdf/1412.6980.pdf.

Krizhevsky, A. Sutskever, I. and Hinton, G. E. (2012) Imagenet classification with deep convolutional neural networks*. Advances in Neural Information Processing Systems. **1**: 1097–1105.*

LeCun, Y. Bengio, Y. and Hinton, G. (2015) Deep learning. *Nature*, *521*(7553):436–
444.

Masolo, C. (2017) Supervised, unsupervised and deep learning. Retrieved October 3, 2020 from https://towardsdatascience.com/supervised-unsupervised-and-deep-learning-aa61a0e5471c.

Murtaza, G. Shuib, L. Wahab, A.W.A. Mujtaba, G. Mujtaba. G. Nweke, H.F. Al-garadi, M.A. Zulfiqar, F. Raza, G. and Azmi, M.A. (2019) Deep learning-based breast cancer classification through medical imaging modalities: state of the art and research challenges. *Artificial Intelligence Review* 53:1655–1720.

Nwaneri, S.C. Nwoye, E.O. Iruhe, N.K. Babatunde, A.M. (2014) Application of Artificial Neural Networks in Breast Cancer Classification: A Comparative Study. *University of Lagos Journal of Basic Medical Sciences*, 2(1):32 – 38.

Nwankpa, C.E. Ijomah, W. Gachagan, A. and Marshall, S. (2018) Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. Retrieved October 8, 2020 from https://arxiv.org/pdf/1811.03378.pdf.

Patterson, J. and Gibson, A. (2020) *Deep Learning*. O'Reilly Media, Inc

Rawat, W. and Wang, Z. (2017) Deep Convolutional Neural Network for Image Classification: A Comprehensive Review. *Neural Computation*, 29:2352 – 2449.

Schmidhuber, J. (2015) Deep Learning in Neural Networks: An Overview. *Neural Networks*. 61:85–117.

Sharma, G.N. Dave, R. Sanadya, J. Sharma, P. and Sharma, K.K. (2010) Various types and management of breast cancer: an overview. *Journal of Advanced Pharmaceutical Technology and Research*, 1(2):109 – 126.

Siegelmann, H. T. and Sontag, E. D. (1991). Turing computability with neural nets. *Applied Mathematics Letters*, 4(6):77–80.

Sun, A.Y. and Scanlon, B.R. (2019) How can Big Data and machine learning benefit environment and water management: a survey of methods, applications, and future directions. *Environmental Research Letters*, Retrieved October 11, 2020, from https://iopscience.iop.org/article/10.1088/1748-9326/ab1b7d/pdf

Sun, S. Cao, Z. Zhu, H. and Zhao, J. (2019) A Survey of Optimization Methods from a Machine Learning Perspective. Retrieved October 6, 2020, from https://arxiv.org/pdf/1906.06821.pdf.

Szegedy, C. Liu, W. Jia, Y. Sermanet, P. *et al*., (2015) Going deeper with convolutions, *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, 1-9.

Torti, E. Musci, M. Guareschi, F. Leporati, F. Piastra, M. (2019) Deep Recurrent Neural Networks for Edge Monitoring of Personal Risk and Warning Situations. *Scientific Programming*, Retrieved October 7,2020, from https://www.hindawi.com/journals/sp/2019/9135196/

Wang, Y. Li, Y. Song, Y. and Rong, X. (2020) The Influence of the Activation Function in a Convolution Neural Network Model of Facial Expression Recognition. *Applied Sciences*, 10(5): Retrieved October 2, 2020, from https://doi.org/10.3390/app10051897**.**

Wang, S. and Wuang, T. (2019) Application of Deep Learning in Biomedicine.

"About diabetes". *World Health Organization.*

World Health Organisation (2020) Diabetes. Retrieved October 5, 2020 from https://www.who.int/health-topics/diabetes.

Wittek, P. (2014) Quantum Machine Learning. Elsevier Inc.

Wiatowski and Bolcskei, (2017) A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction. Retrieved October 8, 2020, from https://arxiv.org/pdf/1512.06293.pdf

Zhang, Y. (2019) Deep Learning in Wisconsin Breast Cancer Diagnosis. Retrieved October 5, 2020, from https://towardsdatascience.com/deep-learning-in-winonsin-breast-cancer-diagnosis-6bab13838abd.