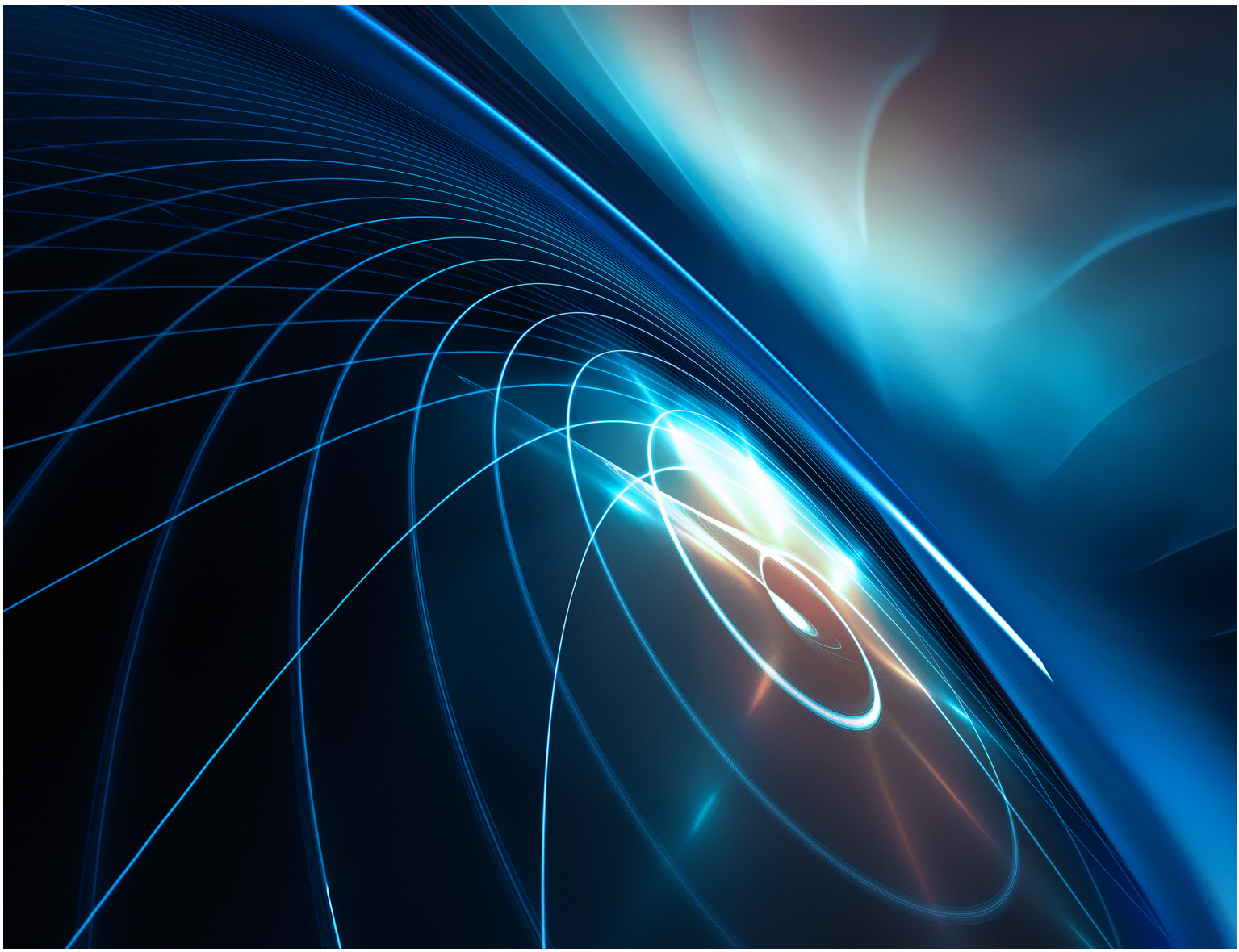
Lattice

THE MACHINE LEARNING JOURNAL

VOLUME-2, **ISSUE**-1 (January - March 2021)



AN INTERNATIONAL PEER REVIEWED OPEN ACCESS JOURNAL
IN DATA SCIENCE & MACHINE LEARNING

ADaSci | THE ASSOCIATION OF DATA SCIENTISTS

www.adasci.org/lattice

Index

PUBLISHED BY:

ADaSci | THE ASSOCIATION OF DATA SCIENTISTS

BANGALORE, INDIA

ONLINE CONTENTS AVAILABLE: EVERY QUARTER ON **www.adasci.org/lattice**

About THE JOURNAL

Lattice is an international peer-reviewed and refereed journal on machine learning. The journal is hosted and managed by the Association of Data Scientists (ADaSci). Lattice intends to publish high-quality research articles of the researchers and professionals working in the field of data science and machine learning. All the articles published by Lattice have to pass an in-depth doubly blinded review process before publishing. The journal maintains a list of reviewers and editors all belonging to the prestigious institutions/organizations that take part in the functioning of the journal

Scope OF LATTICE

The Association of Data Scientists was formed with the intent to develop, disseminate and implement knowledge, basic and applied research and technologies in analytics, decision-making, and management. Lattice, hosted under the flagship of ADaSci follows the same vision and aims to provide a platform for sharing and exchanging the knowledge and research outcomes in the field of data science and machine learning. Lattice publishes scholarly articles that come under the aim and scope of the journal. The article submitted for consideration must consist of new concepts, theories, methodologies, and applications that are unpublished. It considers articles from the following key areas of data science and machine learning.

Ethics Concerns (PLAGIARISM, MISCONDUCT, ETC.)

The publication of an article in Lattice is considered as a building block in the development of a coherent and respected network of knowledge. The publication is a direct reflection of the quality of the contribution of an author and the organization that supports them. It is, therefore, necessary to adhere to certain standards of expected ethical behaviour. The important points that must be considered before submission are given below.

AUTHORSHIP: Authorship should be limited to those persons only who have made a significant contribution to the conception, design, execution, or interpretation of the reported study. Transparency about the contributions of authors is encouraged.

ORIGINALITY AND PLAGIARISM: The authors should ensure that they have written entirely original works, and if the authors have used the work and/or words of others, that this has been cited or quoted appropriately..

DATA ACCESS AND RETENTION: Authors may be required to provide the raw data in connection with a paper for editorial review, and should be prepared to provide public access to such data.

ACKNOWLEDGEMENT OF SOURCES: Proper acknowledgement of the work of others must always be given. Any funding received for the research must also be acknowledged.

DISCLOSURE AND CONFLICTS OF INTEREST: All submissions must include disclosure of all relationships with any member of the Lattice's editorial team that could be viewed as presenting a potential conflict of interest.

Published by:

ADaSci | THE ASSOCIATION OF DATA SCIENTIST

www.adasci.org

Copyright POLICY

The Lattice requires the transfer of copyrights from the author to the journal. On successful acceptance of every paper to the Lattice, the authors are required to submit a copyright transfer form. The copyright is transferred from the author to the publisher that is meant for the contents in the article. The algorithms and research work is always the intellectual property of the researcher or writer. Consider the case that in future, the author claims that ADaSci has published my work without my knowledge and consent or the author publishes the work with any other publisher and the other publisher claims that ADaSci has published its contents by violating the copyright rules.

The copyright aims to ensure that the researcher has published his work with the publisher to whom the researcher has transferred the copyrights. Now the publisher is the owner of the contents and the publisher of the intellectual property that actually belongs to the researcher. After getting the copyrights transferred from the author, the publisher becomes authorised to publish the contents on his publication mediums such as website, journal, video series etc. The copyright also ensures that the same content is not being published by the author with any other publication without the consent of the current publisher. It also ensures that no other person can publish the contents which are already published where the publisher has the copyrights for the same.

Disclaimer

The Association of Data Scientists (ADaSci) believes that the manuscripts submitted to Lattice by the corresponding authors are their original work as the authors have acknowledged the same while transferring the copyright to the journal. In future, if it is found that the content has been published with any other publication without the knowledge of ADaSci, the Lattice will discontinue the publication of that manuscript from the website.

The Editorial Board

Pneumothorax Detection and Classification on Chest Radiographs using Artificial Intelligence

Tejas Haritsa V K*
AI Engineer
Telerad Tech Pvt. Ltd.
Bengaluru, India
tejastejatej@gmail.com

Naveen Raju S G*
AI Engineer
Telerad Tech Pvt. Ltd.
Bengaluru, India
naveenraju100@gmail.com

Kishore Rajendra
AI Engineer
Telerad Tech Pvt. Ltd.
Bengaluru, India
kishorerajendra000@gmail.com

Dr. Arjun Kalyanpur
CEO and Chief Radiologist
Teleradiology Solutions and
Image Core Lab
Bengaluru, India
arjun.kalyanpur@telradsol.com

Dr. Pallavi Rao
Senior Scientific Officer and
Consultant Radiologist
Image Core Lab
Bengaluru, India
pallavi.rao@imagecorelab.com

Abstract:

In recent years, Computer Aided Diagnosis (CAD) systems have been designed for the detection of lung space anomalies. Pneumothorax is an abnormal collection of air in the pleural space between the lung and the chest wall that can result in partial or complete lung collapse [1]. This is a medical emergency in which quick detection and timely intervention can be life-saving. Pneumothorax detection on chest radiographs is important & may be facilitated with the help of image processing and deep learning algorithms. In this study we aim to evaluate the performances of two artificial intelligence systems in detection of pneumothorax on chest radiographs. The AI system was trained on open source datasets obtained from from the US National Institutes of Health (NIH) [2], Society for Imaging Informatics in Medicine (SIIM) [3][4] & private datasets. Two unique approaches were used, one involved processing high-resolution complete images of size 1024x1024px and other involved feeding medium resolution images in portions (segments), each of size 448x448px. The trained AI systems was trained with binary mask as ground truth evaluated by a team of radiologists where, the segmental approach yielded a dice coefficient of 0.72, sensitivity of 0.986, specificity of 0.95, accuracy of 0.9683 and with Area Under Receiver Operating Characteristic Curve (AUROC) of 0.95, while the full image approach yielded an accuracy of 0.9417, dice coefficient of 0.865, sensitivity of 0.9084, specificity of 0.9510 with AUROC of 0.93.

Keywords:

Computer Aided Diagnosis (CAD) systems, Image Processing, Artificial Intelligence (AI) System, Deep Learning, Chest Radiographs, US National Institutes of Health (NIH), The Society for Imaging Informatics in Medicine (SIIM), Two Unique approaches, Segmental approach, Full Image approach, RetinaNet, ResNet, U-Net.

I. INTRODUCTION

A pneumothorax is an abnormal collection of air in the pleural space between the lung and the chest wall [1]. This air pushes on the outside of the lung, causing it to collapse. A pneumothorax can be caused by a blunt or penetrating chest injury, certain medical procedures, or from underlying lung disease, typically emphysema. Depending on its size, pneumothorax can result in complete lung collapse or collapse of only a portion of the lung. Occasionally it may occur for no obvious reason (idiopathic). Pneumothorax can potentially be life-threatening and is considered to represent a critical finding in Emergency Radiology (ER), requiring immediate reporting to the treating physician to ensure immediate medical attention. Hence, Pneumothorax detection is of critical importance in clinical care. Pneumothorax may be detected with the help of image processing and deep learning algorithms. If utilized effectively, deep learning techniques can assist radiologists with quick detection, segmentation, classification and quantification of pneumothorax. In this paper, we evaluate two deep learning architectures for the detection and segmentation of pneumothorax regions on chest radiograph images. The AI system detects regions of pneumothorax in a chest radiograph and may assist the radiologist to review on priority the cases that contain a pneumothorax and thus facilitate early management of patients.

* Equal Contributors

II. DATASET AND METHODS

A. DATASET

The data set was curated from an open source repository from the US National Institutes of Health (NIH) [2], The Society for Imaging Informatics in Medicine (SIIM) [3][4] & private data sets and repositories, totaling 18,252 chest radiographs of varying quality. Which was approved by our Institutional Ethics Committee (IEC) at Image Core Lab (ICL). A total of 12,954 chest radiographs comprising 3,576 positive, 9,378 negative were obtained from Society for Imaging Informatics in Medicine (SIIM) [3][4] and a total of 5,298 chest radiographs comprising 2,146 positive, 3,152 negative were obtained from private data repositories each of dimensions 1024x1024. This was used to train the AI system excluding 3,720 radiographs which was used only for validation during the training phase. The AI system was later tested on a portion of the data set which was initially excluded from both training and validation sets and consisted of a batch of chest radiographs numbering 1,578. An initial analysis of the data set revealed that the mean age of patients was 48 years with a male-female ratio of 52:48.

B. METHODS

1) Full Image Approach

In this approach high resolution images were trained along with their respective binary masks as ground truths. The U-Net architecture [6] has been used for segmentation of several

abnormalities such as "Intracranial Hemorrhage Segmentation Using Deep Convolutional Model" [7], "The 2ST-UNet for Pneumothorax segmentation in chest radiographs using Residual Network with 34 layers (ResNet34) as a backbone for U-Net" [8], "Multi-Path Recurrent U-Net Segmentation of Retinal Fundus Image" [9], "MRI Breast Tumor Segmentation Using Different Encoder and Decoder CNN Architectures" [10] and has been proved to give better accuracy. Observing its versatility in the above scenarios, U-Net architecture [6] was chosen for this problem statement.

2) Segmental Approach

As pneumothorax is an abnormality which appears as a lucency (variation in the exposure), Image augmentation techniques were used to enhance the visibility of the regions of pneumothorax. Adjustments to gamma values of the image in particular played an important role in enhancing the visibility of the pneumothorax boundaries. For the Segmental approach each chest radiograph was sliced into nine image segments as shown in Fig. 2.2.1 and Fig.2.2.2.

The segmented images were classified into positive segments and negative segments based on the ground truth binary masks [5]. The AI system was then trained on the segmented chest radiographs in 3 folds (Here, a fold refers to a cycle of training). The 1st fold of training was carried out on only positive segments of pneumothorax for 10 epochs, The 2nd fold of training was carried out with a mixture of positive and negative pneumothorax segments with a ratio of 0.8:0.2 for 10 epochs, The 3rd fold of training was carried out on a mixture of positive and negative pneumothorax segments with a ratio of 0.4:0.6 for 10 epochs.

This particular approach was used in training to ensure that the model picked up the key regions of pneumothorax, so that it first gets skewed towards high sensitivity as a false negative prediction could have severe implications in the Emergency Radiology (ER) environment. This was achieved by first exposing the model to learn the appearance and patterns of pneumothorax in the 1st fold and later on slowly exposing the model to learn on pneumothorax negative segments while annealing the positive regions of pneumothorax in the 2nd and 3rd folds.

Fig.2.2.1 Chest Radiograph

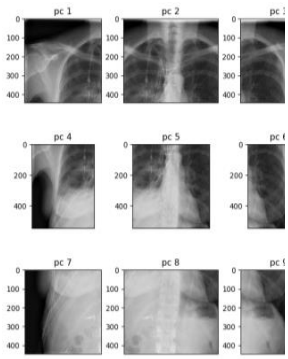


Fig.2.2.2 Segmental image instances

III. MODEL ARCHITECTURE AND TRAINING

A. MODEL ARCHITECTURE

1) Full Image Approach

In this approach U-Net architecture [6] with an efficient backbone such as ResNet-34 [11] [19] was used as encoder and decoder because the features of pneumothorax are very subtle. A U-Net [6] is a Convolutional Neural Network (CNN) architecture that was developed for bio medical image segmentation. A ResNet [11] is used for the encoder/down sampling section of the U-Net [6]. In this model, We have used a ResNet-34, a 34 layer ResNet architecture [11], as this was found to be very effective, was faster to train and used lesser computational memory than a ResNet-50 architecture [11]. ResNet is a CNN architecture, made up of a series of residual blocks (ResBlocks) with skip connections. When using only U-Net architecture [6] the predictions tend to lack fine detail. To help address this, cross or skip connections are made from the same four residual blocks in down-sampling path to its respective corresponding residual blocks in the up-sampling path. The model utilizes an input image and a ground truth mask both of dimensions 1024x1024x3 (Height x Width x Channels) and outputs a binary mask of the same dimensions. A graphical representation of the model architecture can be viewed in Fig.3.1.1

2) Segmental Approach

A 10 block custom U-Net [6] with 39 layers was used with each block consisting of 2 Convolution 2D and 1 MaxPooling 2D layers. At the end of the model architecture the last block consisted of a Convolution 2D layer with 3x3 kernel and a Convolution 2D layer with 1x1 kernel (with the former layer using 2 filters) which used sigmoid activation for pixel level classification. Adam optimizer was used to optimize the weights selection for the convolutional kernel. The model architecture followed a strict layout where each block contained 2x exponentially more filters than its predecessor. The input to the model is a 448x448x1 grayscale segmental instance of a chest radiograph and its output is a 448x448x1 feature map (binary mask) confining to the boundaries of pneumothorax positive regions. Predictions from this architecture were further subjected to a simple thresholding to reduce False Positive (FP) predictions. A graphical representation of the model architecture can be viewed in Fig.3.1.2

The quantification of the pneumothorax is done by dividing the area of pneumothorax by the area of the lung space. Where, the area of pneumothorax is a product of model output and the pixel spacing of individual radiographs.

B. TRAINING

1) Full Image Approach

Images and their corresponding binary ground truth masks of dimension 1024x1024x3 were used as inputs for training using NVIDIA 1080Ti of 11gb GPU [12]. The training was performed in 5 folds cross-validation split strategy. Each fold was trained in 2 stages, 1st stage and 2nd stage had positive and negative images in the ratio 0.8:0.2 and 0.6:0.4 respectively. All folds in both stages were trained with mini batch size of 2 with the following parameters: Adam optimizer, reduce learning rate on plateau, early stopping with patience of 3, combined loss which includes focal loss [14], binary cross entropy, and dice loss for 50 epochs per fold.

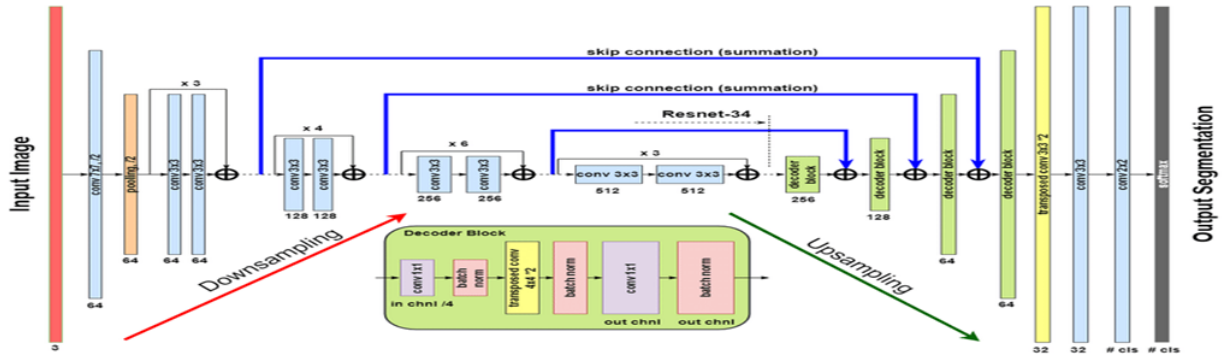


Fig.3.1.1 Full Image Approach Architecture [15]

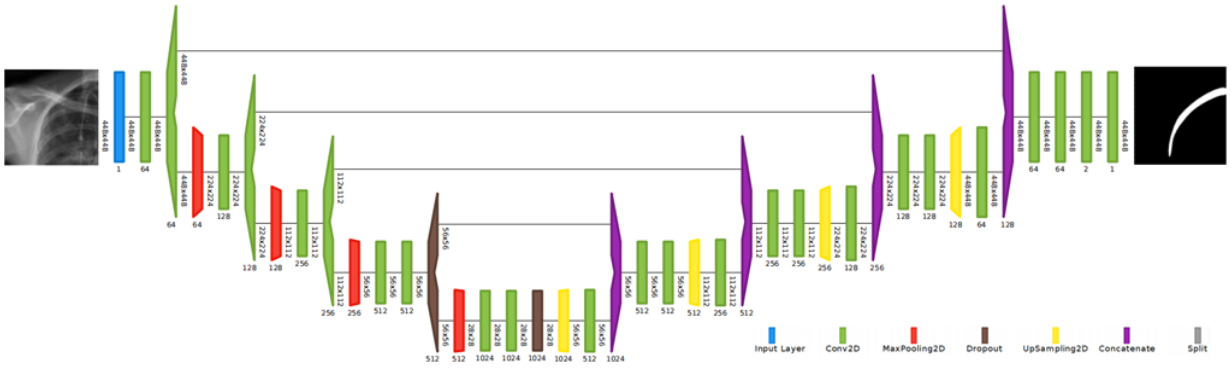


Fig.3.1.2 Segmental Image Approach Architecture [16]

To avoid over-fitting and to make it more generalized on different kinds of image quality, various augmentation techniques were used during training. Dice coefficient [13] was used as the main validation metrics in callbacks used to track the improvement of model in each epoch. The dice coefficient [13] was calculated on multiple predefined binary thresholds on final output feature map and the models with the best dice coefficient with its corresponding threshold were recorded. After training of all folds in first stage, best weights were selected from each fold and used for transfer learning for 2nd stage with cosine annealing learning rate. The same process of selection on the basis of dice coefficient is used to choose the best model from each fold for better accuracy in the second stage. Finally, the best model from each fold is chosen for ensembling for better accuracy.

2) Segmental Approach

The model was trained on the segmental instances of chest radiographs of dimensions 448x448x1 (Height x Width x Channels) with binary masks of dimensions 448x448x1 as the ground truth, in 3 folds as described in the "Methods" section above. The model was initialized with random weights using He-Normal kernel initializer. The network was trained with Adam as the optimizer with a learning rate of 10^{-3}, epsilon of 0.01, and a decay rate of 10^{-12}. A custom loss function bin_dice_loss was used to gain a measure of both classification and segmentation losses, This was achieved by using a combination of binary cross-entropy and dice_loss [13] (bin_dice_loss = binary_cross-entropy + dice_loss). Two metrics: Accuracy and Dice Co-efficient [13] were used to keep track and evaluate the models performance throughout the training phase. The training was carried out methodically to replicate the "cosine annealing learning rate" effect over all the 3 folds. Each fold of the training phase was carried out by choosing the best performing model from the preceding folds, a total of 2 iterations (3 + 3 folds) were performed to obtain the final model. The complete training (all 3 folds) was carried out with a batch size of 4 and input image dimensions of 448x448x1 on a 11gb NVIDIA 2080Ti GPU [12]. Augmentation techniques were used to avoid over-fitting of the model and to make it more generalized to various quality of input images as seen on a live work list of ER environment. The model was validated by feeding carefully curated and augmented images.

IV. RESULTS

Both the approaches were tested on a portion of the data set from the sources previously mentioned in section II, which was initially excluded from both training and validation sets and consisted of a batch of chest radiographs numbering 1,578. For calculating the results, performance metrics such as binary Accuracy, Dice Co-Efficient [13], Focal Loss [14], Bin-Dice Loss, Sensitivity and Specificity are used as shown in equations (1), (2), (3), (4), (5) and (6) respectively.

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{TN} + \text{FN} + \text{FP})} \quad -- \text{Equation (1)}$$

$$\text{Dice loss} = 1 - \frac{(\text{GT} * \text{P})}{(\text{GT} + \text{P})} \quad -- \text{Equation (2)}$$

$$\text{Focal loss} = [-GT * (\alpha * ((1-P)^{\gamma}) * \log(P))] + [-(1-GT) * ((1-\alpha) * ((P)^{\gamma}) * \log(1-P))] \quad --\text{Equation (3)}$$

$$\text{Binary cross entropy (BCE)} = GT * (-\log(P)) + (1-GT) * (-\log(1-P))$$

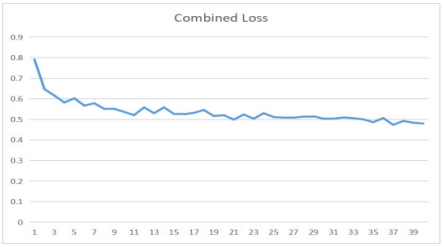
$$\text{Bin} - \text{Dice Loss} = (\text{BCE} + \text{Dice Loss}) \quad -- \text{Equation (4)}$$

$$\text{Sensitivity} = \frac{TP}{(TP+FN)} \quad -- \text{Equation (5)}$$

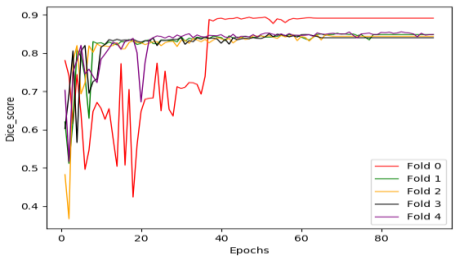
$$\text{Specificity} = \frac{TN}{(TN+FP)} \quad -- \text{Equation (6)}$$

Where, TP (True Positive) - Chest Radiographs containing Pneumothorax flagged as positive.
FP (False Positive) - Chest Radiographs without Pneumothorax flagged as positive.
FN (False Negative) - Chest Radiographs with Pneumothorax flagged as negative.
TN (True Negative) - Chest Radiographs without Pneumothorax flagged as negative.
GT (Ground Truth) - Actual regions of pneumothorax
P (Prediction) - Regions predicted as pneumothorax by model
α - Scaling Factor
γ - Focusing Parameter
Dice Coefficient - Represents the percentage of overlap between the ground truth and model prediction.

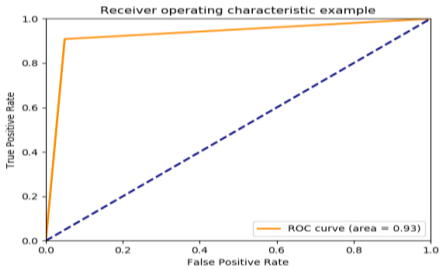
The full image approach after training gives the accuracy of 0.9417, sensitivity of 0.9084, specificity of 0.9510, combined loss of 0.51, dice coefficient of 0.865 and AUROC as shown in Fig.4.1(a.) through Fig.4.1(c.) respectively.



(a.) Combined Loss



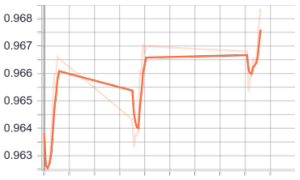
(b.) Dice Coefficient



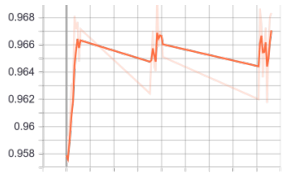
(c.) AUROC

Fig.4.1

The segmental approach after training gives the accuracy of 0.9683, bin-dice loss of 0.35, dice coefficient of 0.72, learning rate, sensitivity of 0.986, specificity of 0.95 and AUROC as shown in Fig.4.2(a.) through Fig.4.2(h.) respectively.

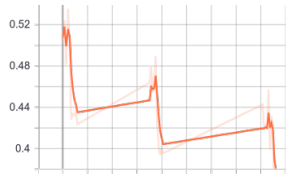


(a.) Accuracy

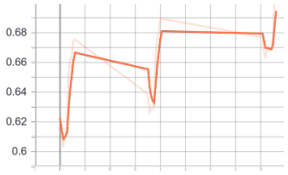


(b.) Validation Accuracy

(c.) Bin-Dice Loss

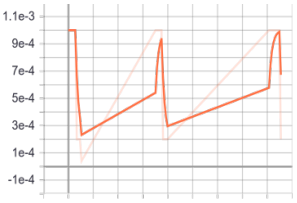


(d.) Validation Bin-Dice Loss

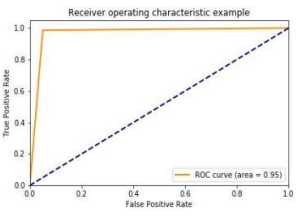


(e.) Dice Coefficient

(f.) Validation Dice Coefficient



(g.) Learning Rate



(h.) AUROC

Fig.4.2

A. *Full Image Approach*

While both the models yield good results overall, the full image approach yields better detection and quantification on instances which stretch over multiple segmental instances as it has a better overview of the surroundings of the abnormality

and hence performs better in such scenarios. A few sample outputs for the full image approach can be viewed below in Fig. 5.1.1 and 5.1.2.

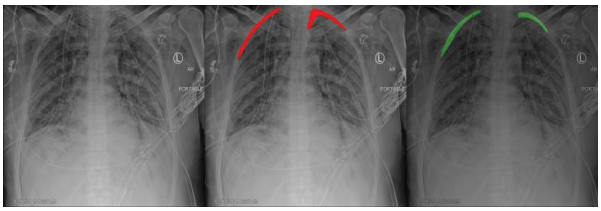


Fig.5.1.1 Full Image Approach Model Prediction

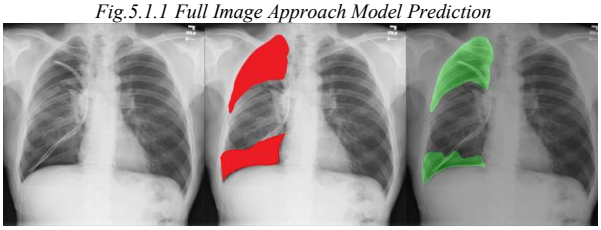


Fig.5.1.2 *Full Image Approach Model Prediction*

In the above figures the regions highlighted in red represent the ground truth i.e. actual regions of pneumothorax and the regions highlighted in green represent the model prediction of pneumothorax regions.

B. Segmental Approach

While both the models yield good results overall, the segmental approach performs better in detecting subtle and smaller instances on pneumothoraces which are often missed by the radiologists during heavy workloads. While the segmental approach has better performance in detecting smaller and subtle regions of pneumothoraces, it is challenged when it comes to the detection of pneumothoraces which spread over multiple regions or fall in between the boundaries of the segmental instances. To counteract this, issue the segmental model included regions of overlaps in between the segmental image instances which helped it to mitigate the previously mentioned issues. A few sample outputs for the full image approach can be viewed below in Fig. 5.1.1 and 5.1.2.

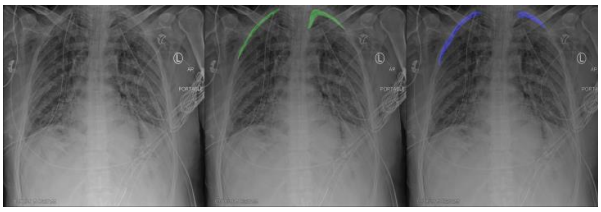


Fig.5.2.1 *Segmental Approach Model Prediction*

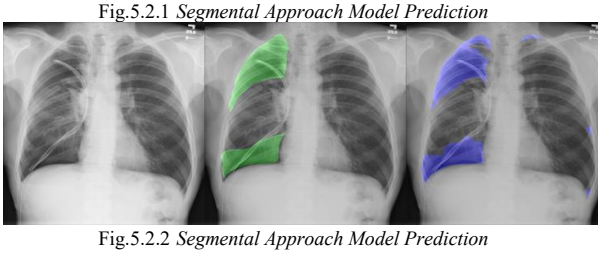


Fig.5.2.2 *Segmental Approach Model Prediction*

In the above figures the regions highlighted in green represent the ground truth i.e. actual regions of pneumothorax and the regions highlighted in blue represent the model prediction of pneumothorax regions.

V. CONCLUSION

This study was conducted with the aim of improving the efficiency of pneumothorax detection and quantification. Integrating these models into the workflow of radiologists can aid them by highlighting pneumothorax and accurately quantifying it, thereby enhancing the reporting accuracy and reducing the turn around time. The two approaches presented in this paper hold promising results in detection and quantification of pneumothorax regions. Of the two approaches discussed in this study the Full Image approach provided higher confidence in terms of localization while, the Segmental approach performed higher in terms of AUROC as shown in the "Results" section above. We hypothesize that an ensemble of both the approaches could outperform their individual results that inherits the best of both. Future work could include the various augmentation techniques that can be used to improve both the approaches or training a single model using a combined architecture.

VI. ACKNOWLEDGMENT

We would like to thank Keras [17] and Tensorflow [18] for providing us a reliable framework, which eases the work involved in prototyping and experimentation of deep learning algorithms. We would like to thank Kaggle [4] for hosting the "SIIM-ACR Pneumothorax Segmentation" competition which reduced the work involved in data sourcing, annotation and provided us a platform to learn from many bright minds. We would like to thank Image Core Lab for providing and annotating the data set.

VII. REFERENCES

[1] Zarogoulidis P, Kioumis I, Pitsiou G, Porpodis K, Lampaki S, Papaiwannou A, Katsikogiannis N, Zaric B, Branislav P, Secen N, Dryllis G, Machairiotis N, Rapti A, Zarogoulidis K., "Pneumothorax: from definition to diagnosis and treatment", J Thorac Dis. 2014 Oct;6(Suppl 4):S372-6. doi: 10.3978/j.issn.2072-1439.2014.09.24. PMID: 25337391; PMCID: PMC4203989.

[2] Dataset, Wang X, Peng Y, Lu L, Lu Z, Bagheri M, Summers RM. ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. IEEE CVPR 2017, http://openaccess.thecvf.com/content_cvpr_2017/papers/Wang_ChestX-ray8_Hospital-Scale_Chest_CVPR_2017_paper.pdf

[3] Dataset, https://siim.org/page/pneumothorax_challenge

[4] Dataset, https://www.kaggle.com/c/siim-acr-pneumothorax-segmentation/overview

[5] André Gooßen, Hrishikesh Deshpande, Tim Harder, Evan Schwab, Ivo Baltruschat, Thusitha Mabotuwana, Nathan Cross, Axel Saalbach, "Deep Learning for Pneumothorax Detection and Localization in Chest Radiographs", arXiv:1907.07324 [eess.IV]

[6] Olaf Ronneberger, Philipp Fischer, Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", arXiv:1505.04597v1 [cs.CV]

[7] Murtadha D. Hssayeni, M.S., Muayad S. Croock, Ph.D., Aymen Al-Ani, Ph.D., Hassan Falah Al-khafaji, M.D., Zakaria A. Yahya, M.D., Behnaz Ghoraani, Ph.D, "Intracranial Hemorrhage Segmentation Using Deep Convolutional Model", arXiv:1910.08643v2 [eess.IV]

[8] Ayat Abedalla, Malak Abdullah, Mahmoud Al-Ayyoub, Elhadj Benkhelifa, "The 2ST-UNet for Pneumothorax Segmentation in Chest X-Rays using ResNet34 as a Backbone for U-Net", arXiv:2009.02805v1 [eess.IV]

[9] Jiang, Yun; Wang, Falin; Gao, Jing; Cao, Simin. 2020. "Multi-Path Recurrent U-Net Segmentation of Retinal Fundus Image" Appl. Sci. 10, no. 11: 3777.

[10] El Adoui, Mohammed; Mahmoudi, Sidi A.; Larhmam, Mohamed A.; Benjelloun, Mohammed. 2019. "MRI Breast Tumor Segmentation Using Different Encoder and Decoder CNN Architectures" Computers 8, no. 3: 52.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition", arXiv:1512.03385v1 [cs.CV]

[12] GPU, https://www.nvidia.com/en-in/

[13] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sébastien Ourselin, M. Jorge Cardoso, "Generalised Dice overlap as a deep learning loss function for highly unbalanced segmentations", arXiv:1707.03237v3 [cs.CV]

[14] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár, "Focal Loss for Dense Object Detection", arXiv:1708.02002v2 [cs.CV]

[15] Shvets, Alexey & Iglovikov, Vladimir & Rakhlin, Alexander & Kalinin, Alexandr. "Angiodysplasia Detection and Localization Using Deep Convolutional Neural Networks.", arXiv:1804.08024v1 [cs.CV]

[16] Alex Bäuerle, Christian van Onzenoodt, Timo Ropinski, "Net2Vis -- A Visual Grammar for Automatically Generating Publication-Ready CNN Architecture Visualizations", arXiv:1902.04394 [cs.LG]

[17] Framework, https://keras.io/

[18] Framework, https://www.tensorflow.org/.

[19] Sneddy, https://github.com/sneddy/pneumothorax-segmentation/

Machine learning approach to predict patient position for preventing bedsores

Aditya Aggarwal
Advanced Analytics Practice
Abzooba Inc.
Pune, India
aditya.agarwal@abzooba.com

Sujoy De
Advanced Analytics Practice
Abzooba Inc.
Pune, India
sujoy.de@abzooba.com

Abstract - The Agency for Healthcare Research & Quality estimates more than 2.5 million individuals in the United States develop bedsores annually that costs $9.1-$11.6 billion to the healthcare system. In this paper, we develop a low-cost solution to reduce the risk of bedsores for bed-ridden patients using machine learning. Elderly patients with mobility impairments are the highest risk population segment for bedsores (also known as pressure ulcers). Currently, smart beds that send alarms when patients have not changed their position on bed for long time are used to manage the risk of developing bedsores. However, such smart devices are cost prohibitive. The proposed affordable solution uses low-cost load-cells' readings to accurately estimate the patient position with an accuracy of 98.8%. Specifically, the solution manages bedsore risk by deriving meaningful intuitive features that are used by machine learning model to generate alerts when a patient has been in the same position for a prolonged period of time.

Keywords – bedsores, pressure ulcers, smart beds, patient position, predictive modeling

I. INTRODUCTION

Hospital smart-beds are integrated solutions for patient care, assistance and monitoring. It seamlessly integrates into the healthcare system and enables caregivers by providing many functionalities to care for their patients effectively. One of such functionality provided is alerting caregiver if a patient is at risk of developing pressure ulcers. Pressure ulcers are developed if a patient lies in same position for a long period of time. Its treatment is costly, with an average charge per stay of $37,800 [1]. Preventing pressure ulcers has been a nursing concern for many years. Many clinicians believe that pressure ulcer development is not simply the fault of the nursing care, but rather a failure of the entire heath care system. However, Hospital smart-beds are too big and expensive, and mats that enable beds detecting pressure sores alone cost ~$6000.

Our objective is to reduce the cost of smart-beds through the use of predictive analytics. In this paper, we will explain the use of predictive analytics to predict the position of patient lying on the bed. The estimation of the patient position on bed will help in generating real time alerts that can be sent to healthcare professionals to change the position of a patient if he/she is in a particular position for a long period of time.

Our design involves a bed consisting of 4 planks on top of bed with 4 load cell on each plank measuring the weight readings of the patient. The 4 planks cover the entire length of the bed as shown in fig 1. Distribution of patient weight on these 16 load cells are captured and processed through a machine learning algorithm to estimate the patient position. There are total 5 positions that our algorithm will be predicting. Our algorithm predicts patient position with overall accuracy of more than 98%.

II. LITERATURE REVIEW

Sensing the patient position on bed is not a new problem that the researchers are working on. We can find literatures where researchers used pressure sensor readings to detect the patient position as also discussed in the below 2 papers:

1. Smart care beds for elderly patients with impaired mobility [2]: This paper discusses the design of a smart care bed using the pressure reading from sensors for caring elderly patients with impaired mobility. A total of 45 sensors capturing the accumulated pressure were used for this purpose. The smart bed prototype can detect the posture of a patient into three positions namely supine, left lateral, and right lateral by detecting which side of bed are accumulating pressure via sensors. However the bed can only

determine the posture if the patient has been lying in the same position for more than one minute. Also the performance of the bed in detecting the posture of various patients is affected by the size of the patient and its parameters need to be adjusted for the same.

2. A Smart Bed Platform for Monitoring & Ulcer Prevention [3]: Researchers leveraged two categories of features, posture independent features such as blood pressure and posture dependent features such as center of pressure. With these set of features, a binary classifier for classifying whether a patient is at high risk of developing pressure ulcer or not is developed. Machine learning algorithms namely SVM classifier is used to train the model that classify the risk of developing the pressure ulcer.

In this paper, we will talk about our contribution that is focused towards

1. Minimize the number of load cells on bed without compromising the classifier accuracy to bring down overall cost.
2. Simplifying the feature engineering by crafting intuitive features and develop a robust position detection classifier.

III. OUR DESIGN

Hospital smart bed platform consists of 4 sections of different size to enable the movement of patient on bed (as shown in fig 1). Our smart-bed design consists of planks equipped with load cells placed on top of platform of the bed. These planks are equipped with total 16 load cells (LC1 – LC16), i.e. 4 load cells on each plank.

Our system takes the reading from load cells at an interval of 10 seconds each. Our trained patient position prediction model, deployed onto the bed embedded device, processes the load cell readings as input and provides the estimated patient position. If the patient is not found to change his/her position for more than specified configured time then an alert is triggered to caregiver through this system. This data is also sent to a centralized database using wireless services.

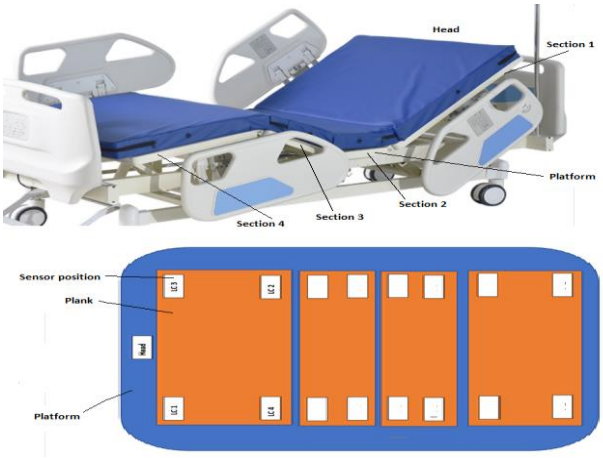


Fig 1: Top image - smart bed; bottom image - plank and sensor placement between bed flat surface and mattress

IV. METHODOLOGY

A machine learning solution is prepared to estimate the patient position on bed using 16 pressure load cell readings on 4 planks placed between bed platform and the mattress. Machine learning model is trained to predict a total of 5 positions (as shown in fig 2) namely, Supine, Left Lateral, Partial Left Lateral, Right Lateral and Partial Right Lateral.

Two pipelines are created with components as shown in fig 3. Pipelines are as follow:

1. Learning pipeline to train the model
2. Classification pipeline to predict the position

Fig 2: Different positions on bed

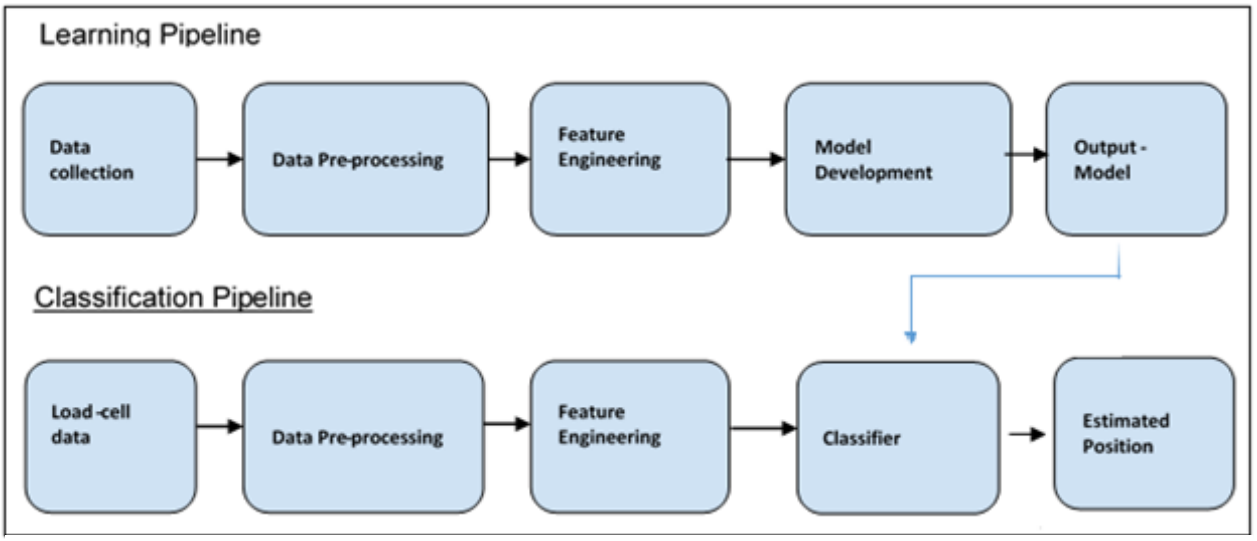


Fig 3: Learning and Classification pipeline

We will discuss details of each of these components in the further sections.

A. Data collection

Data was generated in a controlled environment with predetermined subjects whose weight ranged from 60 kg to less than 100 kg. Each subject was put in 5 different positions (as shown in fig 2) and then readings of 16 load cells along with their positions were logged. This data was used to build a predictive model that estimates patient position on the bed.
Sensor readings distribution by position is given in table 1.

TABLE 1
SENSOR READINGS DISTRIBUTION BY POSITION

#	Subject position on bed	Total readings
1	Supine	1563
2	Left Lateral	1032
3	Right Lateral	965
4	Partial Left	488
5	Partial Right	512

Summary of the data collected

- Total subjects considered for data collection = 28
- Total sensor readings = 4560

B. Data preprocessing

Before we start feature engineering or modeling, data is cleaned and normalized. This preprocessing steps is applied to the whole dataset. Following steps were done in order of given sequence-

1. Missing value treatment: Removed samples with missing values for any of the load cell
2. Outlier removal:
    a. Removed reading where cumulative readings from load cells is less than 50.0 kg and more than 110.0 kg. Please note "cumulative readings from load cells" should be same as weight of the subject.
    b. Remove samples with outliers in them. Outliers have been defined as load cell reading greater than 3.5 standard deviation away from the mean

TABLE 2
DATA DISTRIBUTION BY POSITION

Load Cell name	Sample Raw reading	Normalized reading
LC1	5.9	0.081
LC2	6.2	0.085
LC3	13.2	0.181
LC4	14.2	0.194
LC5	6.7	0.092
LC6	5.9	0.081
LC7	4.1	0.056
LC8	4.3	0.059
LC9	1.0	0.014
LC10	1.2	0.016
LC11	1.3	0.018
LC12	1.0	0.014
LC13	2.6	0.036
LC14	3.8	0.052
LC15	0.5	0.007
LC16	1.2	0.016
Total	73.1	1.000

3. Data Normalization: Normalized the load cell readings by converting them into percentages (i.e. "load cell reading" / "cumulative readings from load cells") as shown in table 2.

C. Feature engineering

Features to explain position using 16 load cell normalized readings are created after data pre-processing. Load cell normalized readings shows the distribution of mass on the bed. Using mass distribution on bed and calculated center of mass (COM) on each plank, a lot of features are created to explain patient position. The calculation for center of mass on plank 1 is as shown below.

COM x co-ordinate on plank 1 = (m1 * x1 + m2 * x2 + m3 * x3 + m4 * x4) / (m1 + m2 + m3 + m4)
COM y co-ordinate on plank 1 = (m1 * y1 + m2 * y2 + m3 * y3 + m4 * y4) / (m1 + m2 + m3 + m4)

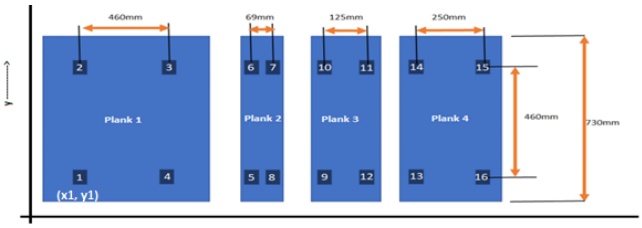


Fig 4: Different positions of load cells on the bed

Where,
On Plank-1 if we plot the position of load cells using coordinate system such that LC1 is at (x1, y1) as shown in fig 4, then
Mass "m1" is present on (x1, y1)
Mass "m2" is present on (x2, y2)
Mass "m3" is present on (x3, y3)
Mass "m4" is present on (x4, y4)

Similarly, COM calculation for other planks are also done.

1st set of features is created based on hypothesis that "for Supine position" all planks COM will fall in a straight line. However, for left lateral position, plank 3 COM will fall above fitted straight line and for right lateral below straight line" as shown in fig 5. Using this hypothesis below features are created.

- A straight line is fitted using 4 planks COM. Error is calculated in y direction between each plank COM and the fitted line. Summation of this error is named as "y_error". Distribution of mean of y_error with various positions is shown in fig 6.
- A straight line is fitted using plank 1 and plank 2 COM (as shown in fig 5). Distance in y-direction between plank 3 COM and the fitted line is calculated. This feature is named as plank_3_dev_bucket.

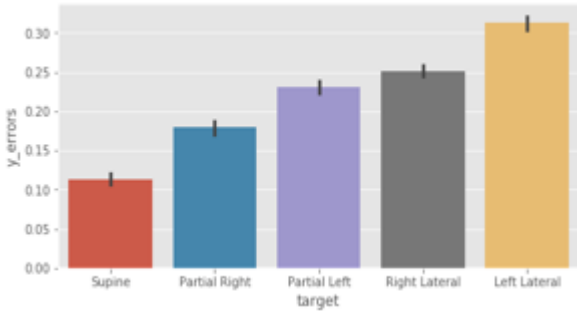


Fig 6: Distribution of mean of y_errors with various positions

- Similarly, a straight line is fitted using plank 1 and plank 2 COM. Distance in y-direction between plank 4 COM and the fitted line is calculated. This feature is named as plank_4_dev_bucket.

2nd set of features is created with hypothesis that "patient in left lateral position will have more weight towards left side of the bed and similarly, patient in right lateral position will have more weight towards right side of the bed" as shown in fig 7.

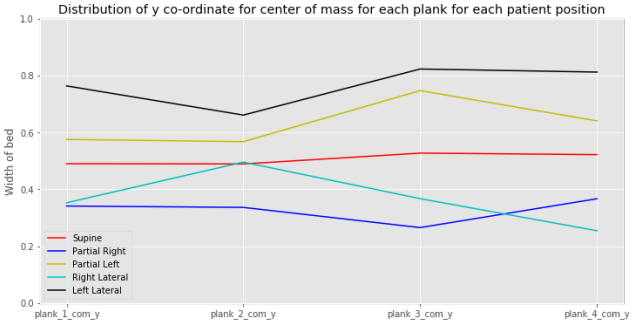


Fig 7: Plot of y coordinate of COM for each plank and position

- Percentage of total weight of the patient covered by sensors situated on left side of the bed. This feature is named as "left_sensors_pct"
- Y co-ordinates of center of mass for each plank (namely, plank_1_com_y, plank_2_com_y, plank_3_com_y, plank_4_com_y).

As can be seen in fig 7,

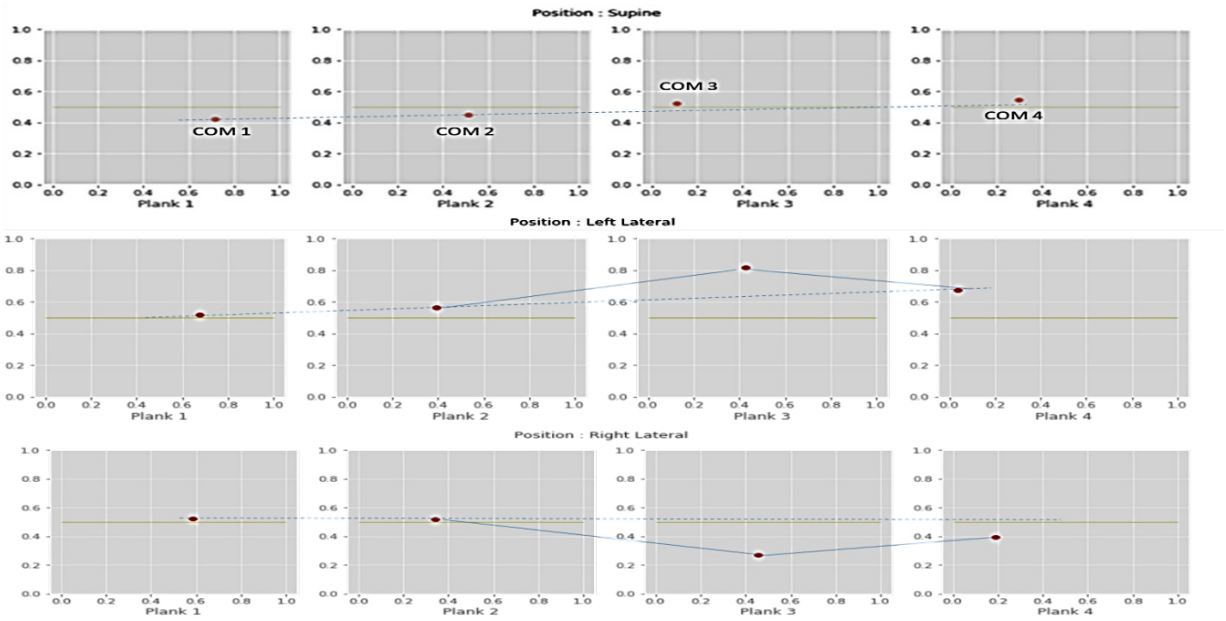


Fig 5: Center of mass on plank 1 (head plank), plank 2, plank 3 and plank (leg plank) for supine, left and right positions. The plank 3 COM is above the fitted line through plank 1 and plank 2 COM for left lateral position and below for right lateral position

- For supine position, the y co-ordinates of the COM is near the middle of the bed (0.5) for each of the plank
- Similarly, for left lateral and partial left positions, we are seeing the y co-ordinates being more towards the left side of the bed and
- Similarly, for right lateral and partial right positions, we are seeing the y co-ordinates being more towards the right side of the bed

3rd set of features is aimed to differentiate right lateral position from partial right lateral position, and, left lateral position from partial left lateral position. It is found that standard deviation of normalized sensor readings from each plank is able to differentiate these positions as shown in fig 8.

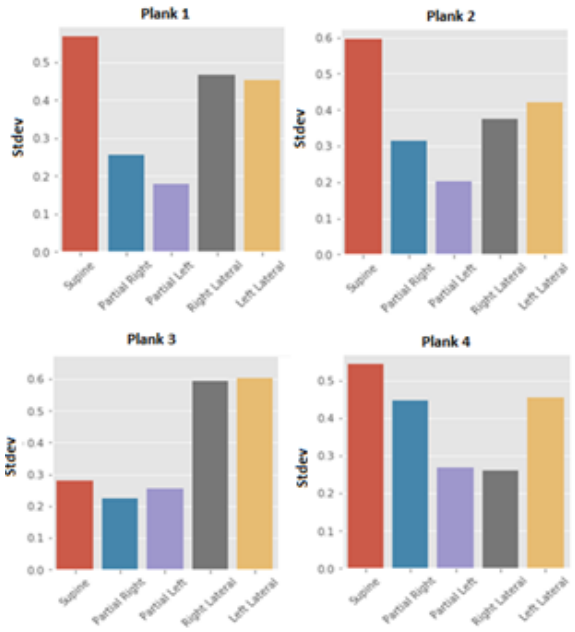


Fig 8: Distribution of standard deviation of planks reading with various positions

D. Model iterations

After data pre-processing, we were left with 4560 data points spanning across 28 number of subjects. Out of 28 subjects, 26 were taken to train the model and the remaining subject data to test the model. It accounts for 3908 data points for training and 652 data points for training. Distribution of train and test datasets by position is given in table 3.

Model building and feature preparation was done in iterative manner. We came to the best model and best set of features in 3 different iterations.

TABLE 3
DISTRIBUTION OF VARIOUS POSITION IN TRAIN AND TEST DATA

	Overall data	Training data	Test data
Supine	1563 (34%)	1396 (36%)	167 (26%)
Left Lateral	1032 (23%)	865 (22%)	167 (26%)
Right Lateral	965 (21%)	801 (20%)	164 (25%)
Partial Left	488 (11%)	417 (11%)	71 (11%)
Partial Right	512 (11%)	429 (11%)	83 (13%)
Overall	4560 (100%)	3908 (100%)	652 (100%)

Iteration 1: Best accuracy on test dataset was obtained with Random Forest model using 6 set of features as shown in table 4.

Observation from Iteration 1: It was found that model was struggling to differentiate left lateral position from right lateral position and vice versa as shown in table 5. In next iteration, focus was to introduce such features that will differentiate between these 2 positions.

Iteration 2: Two more features that were aimed to differentiate left lateral position from right lateral position was added that boosted the accuracy further as shown in table 6.

Observation from iteration 2: it was found that model was struggling to differentiate left lateral position from partial left lateral position as shown in table 7. In next iteration, focus was to introduce such features that will differentiate between these 2 positions.

Iteration 3: In this iteration, we focused on features that should be able to differentiate partial left lateral position from partial right lateral position. Post these feature additions, model accuracy reached to 98.8% on test data as shown in table 8.

Observation from iteration 3: Model is able to classify all the positions accurately as shown in table 9.

********** Iteration 1 **********

TABLE 4

#	Classifier	Hyper-parameters	Features used	Accuracy	Log Loss
1	Random Forest	Number of trees = 20	1. left_sensors_pct 2. y_errors 3. plank_1_std 4. plank_2_std 5. plank_3_std 6. plank_4_std	81.9%	0.69
2	Decision Tree	Default		79.0%	7.26
3	Logistic Regression	Default		56.6%	1.15
4	KNN*	Number of neighbors = 5		56.3%	8.94
5	SVM*	Kernel = Radial Basis; C (penalty) = 0.025		21.0%	1.56

**Data was not standardized*

TABLE 5
CONFUSION MATRIX AFTER ITERATION 1

Confusion Matrix		Predictions					
		Left Lateral	Supine	Right Lateral	Partial Left	Partial Right	Overall
Actual	Left Lateral	121	0	46	0	0	167
	Supine	0	164	3	0	0	167
	Right Lateral	22	2	140	0	0	164
	Partial Left	44	0	0	27	0	71
	Partial Right	0	1	0	0	82	83
	Overall	187	167	189	27	82	652

********** Iteration 2 **********

TABLE 6

#	Classifier	Hyper-parameters	Features	Accuracy	Log Loss
1	Random Forest	Number of trees = 20	1. - 6. All features in iteration 1 7. plank_3_dev_bucket 8. plank_4_dev_bucket	87.1%	0.46
2	KNN**	Number of neighbors = 5		75.6%	5.2
3	SVM**	Kernel = Radial Basis; C (penalty) = 0.025		75.8%	0.7
4	Random Forest with early stopping	Number of trees = 20; Min samples to split a node = 25; Minimum samples required at a leaf node = 5		87.7%	0.42

**Data standardized using min max scaler

TABLE 7
CONFUSION MATRIX AFTER ITERATION 2

Confusion Matrix		Predictions					
		Left Lateral	Supine	Right Lateral	Partial Left	Partial Right	Overall
Actual	Left Lateral	119	0	0	48	0	167
	Supine	0	164	3	0	0	167
	Right Lateral	0	1	163	0	0	164
	Partial Left	12	0	0	59	0	71
	Partial Right	0	11	5	0	67	83
	Overall	131	176	171	107	67	652

********** Iteration 3 **********

TABLE 8

#	Classifier	Hyper-parameters	Features	Accuracy	Log Loss
1	Random Forest with early stopping	1. Number of trees = 20 2. Minimum samples to split a node = 25 3. Minimum samples required at a leaf node = 5	1. - 8. All features of iteration 2 9. Y co-ordinate of COM of plank 1, plank 2, plank 3, plank 4	98.8%	0.35

TABLE 9
CONFUSION MATRIX POST ITERATION 3

Confusion Matrix		Predictions					
		Left Lateral	Supine	Right Lateral	Partial Left	Partial Right	Overall
Actual	Left Lateral	165	1	0	1	0	167
	Supine	0	164	2	0	0	167
	Right Lateral	0	1	163	0	0	164
	Partial Left	1	0	0	70	0	71
	Partial Right	0	0	1	0	82	83
	Overall	166	166	166	71	83	652

V. RESULTS

Our approach is able to predict patient position on bed with an accuracy of 98.8% on test dataset. Best model selected post model iterations is Random forest using feature as shown in table 8. Most important features comes out to be "Summation of error from a fitted line through COM", "Y coordinate of COM on plank 3" and "Distance in y-direction between plank 3 COM and the fitted line passing through Plank 1 COM and Plank 2 COM" as shown in fig 9. Learning curve showing the plot of training and test error by number of training instances suggests that test error is remarkably close to training error and within desired tolerance limit as shown in fig 10.

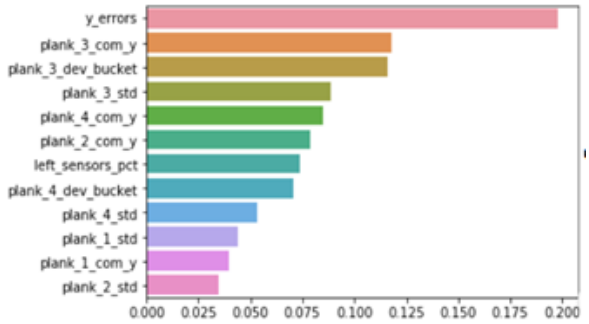


Fig 9: Feature importance

Our solution is able to predict each position with more than 0.98 precision and recall as shown in table 10.

TABLE 10
PERFORMANCE ON TEST DATASET

Description	Precision	Recall	f1-score
Left Lateral	0.99	0.99	0.99
Supine	0.99	0.98	0.98
Right Lateral	0.98	0.99	0.99
Partial Left	0.99	0.99	0.99
Partial Right	0.99	0.99	0.99

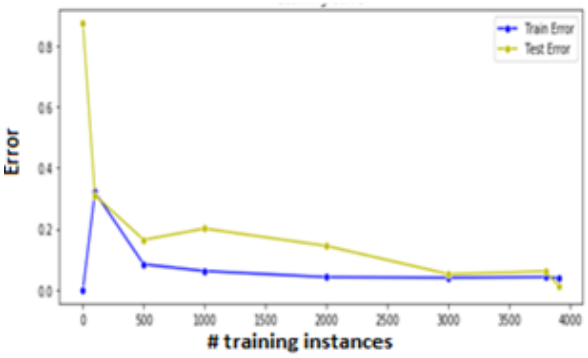


Fig 10: Learning curve

VI. CONCLUSION

Our solution is able to bring down the cost of bed that is able to assist caregiver by providing alarm if patient is lying in same position for a long time. It uses machine learning solution that estimates the patient position on bed using low cost load-cells. Our machine learning algorithm's high accuracy helped in reducing false alarms and hence boosted the confidence of caregiver for the machine.

Our solution is able to bring down the cost of bed that is able to assist caregiver by providing alarm if patient is lying in same position for a long time. It uses machine learning solution that estimates the patient position on bed using low cost load-cells. Our machine learning algorithm's high accuracy helped in reducing false alarms and hence boosted the confidence of caregiver for the machine.

Currently, our model gives the solution to reduce bed-ulcers only, we can fine tune the same model to predict if the patient is at risk of falling from the bed using the same design but with additional collected data.

VII. REFERENCE

1. C Allison Russo, Anne Elixhauser, "Hospitalizations Related to Pressure Sores, 2003", Healthcare cost and utilization project, April 2018 https://www.hcup-us.ahrq.gov/reports/statbriefs/sb3.pdf

2. Youn-Sik Hong, "Smart Care Beds for Elderly Patients with Impaired Mobility", Wireless Communications and Mobile Computing, vol. 2018, Article ID 1780904, 12 pages, 2018. https://doi.org/10.1155/2018/1780904

3. R. Yousefi et al., "A smart bed platform for monitoring & Ulcer prevention," 2011 4th International Conference on Biomedical Engineering and Informatics (BMEI), Shanghai, 2011, pp. 1362-1366, doi: 10.1109/BMEI.2011.6098589.

Solution Approach to Resolve Vehicle Routing Problem using Deep Reinforcement Learning

Monika Singh
Affine Analytics Pvt. Ltd
Bengaluru, India
monika.singh@affineanalytics.com

Sourav Mazumdar
Affine Analytics Pvt. Ltd
Bengaluru, India
sourav.mazumdar@affineanalytics.com

***Abstract*—In this work, we present a Deep Reinforcement Learning based approach as a soluion to one of the popular optimization problems, namely "Capacitated Vehicle Routing Problem". We have benchmarked the results againt genetic algorithm and have evaluated the performance using two KPIs- Travelling cost (distance covered) and Computational time. The comparison shows a 5X-20X reduction in cost and a 100X–1000X reduction in computational time. The Deep Reinforcement Learning based solution adheres to an adaptive learning framework where the system automatically thrives for optimality rather than being explicitly programmed.**

Keywords- Vehicle Routing Problem (VRP), Capacitated Vehicle Routing Problem (CVRP), Reinforcement Learning, Optimization, Genetic Algorithm

I. INTRODUCTION

Vehicle Routing Problem (VRP) -VRP can be defined as a problem for finding the optimum routes for a given set of vehicles to fulfill delivery and collection for a specified set of customers based on some pre-defined demand. Finding the optimal routes may involve business constraints like serving each customer only once. Our solution focuses on mapping the optimal routes for a single vehicle; hence the problem reduces to a simple Travelling Salesman Problem [8]. Optimal routes serve the purpose of minimizing the overall transportation cost, minimizing the number of vehicles, minimum distance travelled, minimum travel time, or other objectives. Some of the most important applications of VRP are Supply Chain Management, Mail delivery, Bus and railway route optimization, vehicle optimization, etc.
One of the variants of VRP is Capacitated VRP (CVRP). CVRP states that m capacitated vehicles initially at depot locations are required to fulfill discrete demands n customer nodes. The objective is to design the set of least distance paths with known customer demands.
CVRP is one of the most popular problems in Network Optimization and is classified as NP-hard problem. It means the size of the problem that can be solved optimally using mathematical programming or combinatorial optimization may be limited. Hence, most of the commercial solutions tend to use a meta-heuristic approach to solve real-world VRPs.
In this whitepaper, we have discussed one of the effective ways to solve the CVRP problem using Deep Reinforcement Learning. Below are the highlights of the whitepaper:

- Benchmarked the results of DeepRL against a popular algorithm called Genetic Algorithm.
- Compared the performance using 2 KPIs – Travelling cost (distance covered) and Computational time.
- During the comparison of KPIs performance, we have observed a 5X-20X reduction in cost and a 100X–1000X reduction in computational time.
- The overall solution adheres to an adaptive learning framework where the system itself identifies the optimal solution without explicitly programming.

II. RELATED WORK

Algorithms for solving VRP

VRP is an NP-hard optimization problem and various exact and approximation algorithms have been used to solve it. The algorithms are based on linear programming techniques, Branch and Bound, Branch and Cut; the approximate algorithms are heuristic-based viz. Genetic algorithms, Evolutionary algorithms, Tabu search, Dynamic Programming, and Neural networks, etc.
For n customer nodes and one depot nodes, there are n! numbers of paths that exist. For large VRP instances, using exact methods for solving VRP is computationally expensive and time-consuming. Due to this reason, approximate algorithms that employ heuristic search are often employed to solve VRP problems. The choice of using exact vs. approximate algorithms is determined by the trade-off between optimality and computational cost.

Following are the recent important algorithmic developments for solving VRP problem in chronological order:

i) Pointer Networks [2]: Pointer Networks (PN) is a kind of Recurrent Neural Networks that can be trained to produce outputs of variable lengths, which is vital in the case of VRP as the route lengths shall very. The PN is used in a supervised way to find near-optimal tours to Travelling Salesman Problem from the given ground truth optimal solutions.

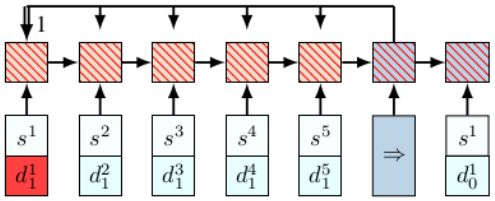


Fig.1: Pointer Networks (Reference: Nazari et al. 2018)

The drawback of this architecture is its sensitivity to small changes in the sequences due to which it gave inconsistent results. For instance, in Fig 1, if the demand d_1 of node S^1 is partially/completely fulfilled, the change should be incorporated. For this, the entire Pointer network is required to be updated to compute the probabilities in the next decision point (with a new d_1 value). Another drawback is the dependence on supervision which prohibits PN from finding better solutions than the ones provided during training.

ii) Actor-Critic Networks [1]: The actor-critic model proposed by Bello et al. [1] is the first work that lays the foundation for using reinforcement learning and neural networks to model VRP.
- The pointer network is trained without supervised solutions.
- The drawback of this architecture is that it assumes that the systems to be static over time. However, VRP should include the dynamic nature of the problem, i.e. suppose the demand of the customer node is fulfilled, its demand should be updated. This is the generic problem in Pointer Networks and is discussed in the previous section.

iii) Reinforcement Learning for VRP [3]:

i. This method is an extension of the Actor-Critic method [1], and it solves dynamic VRP, which considers customer node's demand changes over time.
ii. Long Short-Term Memory (LSTM) encoder in the pointer networks is replaced by element-wise projections. These are invariant to the input sequence ordering and will discard off unnecessary sequential information. Changing the order of any two inputs will not affect the network.
iii. This framework in Fig 2 has an RNN decoder with an attention mechanism that is capable of handling both static and dynamic elements of the system and does the combinatorial optimization using both components.
iv. The reward is calculated based on the generated outputs, which means that if we can verify the feasibility of the generated output sequences, we can learn the desired meta-algorithm.
v. The unique value addition provided by this architecture is that if new VRP instances are generated with the same number of nodes and vehicle capacity, and the same location and demand distributions as the ones used in the training, then the trained policy will work well. As this framework doesn't require an explicit distance matrix calculation and learns meta-algorithm based on the output sequences, this
vi. is suited for constructing routes based on the new data.

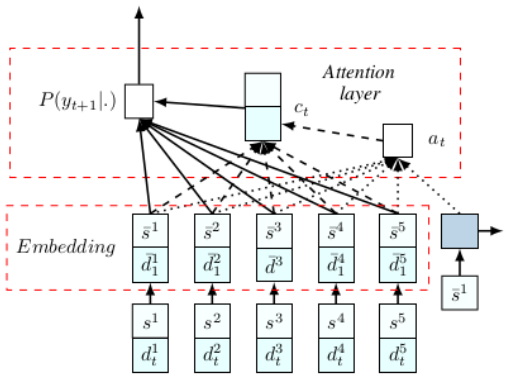


Fig.2: Model invariant of input sequence ordering (Nazari et al. 2018)

iv) Deep Reinforcement Learning (Pang et al., 2020) for VRP

i. This architecture proposed in this work has a dynamic attention model (AM-D) with dynamic encoder and decoder architecture. Each node is characterized dynamically in the context of the graph. This means that the node information can be adapted to the changes in demand, and the state change of instance is also done.
ii. It is based on encoder-decoder architecture, where the encoder extracts structural features of the input instance, and the decoder incrementally constructs the solution. At each construction step of the decoder, the decoder predicts a distribution over nodes, and one node at a time is selected and appended to the partial solution.
iii. The advancement introduced in this architecture is the dynamic encoder-decoder, the encoder and decoder are used alternately to re-construct the embedding of each node and construct a partial solution.
iv. Feature of each node is learned using the embedding learned progressively based on the selected output paths during traversal.

3. Setting up the CVRP problem using Deep Reinforcement Learning

CVRP- problem definition

There are depot nodes and customer nodes in a graph. A depot node is responsible to fulfil the demands of each customer node and return to itself in case of supply over. This is shown in the following figure.

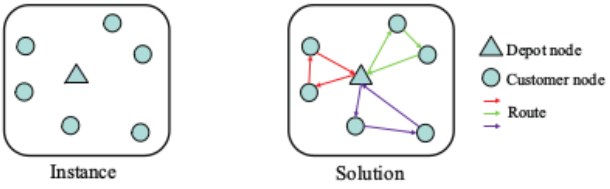


Fig.3: Vehicle Routing Problem (Pang et al., 2019)

Reinforcement learning for solving CVRP problem: In this work, we used deep reinforcement learning for solving CVRP. For CVRP, it is required to pick up the best routes during training and augmenting the learning in a current timestamp from the goodness of results produced in the previous timestamps. This makes using Neural Networks and Reinforcement learning to approach this problem as the combination can lead to better prediction and decision-making processes. Neural networks can do automatic feature selection, in VRP case selecting the routes which give the shortest distances.

In Reinforcement learning-based solutions to VRP, learning an optimal route is cast as a Markov Decision Process (MDP). An MDP is a discrete-time dynamic system model, with three components:

i) S_t-State of the system at time t
ii) a_t-the Action was taken at time t
iii) r_t-the reward at time t

Let S be the set of possible states, A the set of possible actions, and R the set of possible rewards, such that $s_t \epsilon S$, $a_t \epsilon A$, and $r_t \epsilon R$. We assume that A is finite, such that we can choose one of a set of discrete actions at each time point, and that $R = \mathbb{R}$. The reward is meant to represent the value to us (the system controller) of a particular state. The system is Markovian, the current state of the system depends on the state of the system at the previous timestamp and the action taken at the previous timestamp. The transition to the next state is given by:

$$T: P(s_t+1|s_t, a_t)$$

The solution to MDP is to find out the optimal path, where a policy is a function π: S->A, which tells about the action to take in a particular state. The heuristics for making decisions are represented in the form of rules, which can be interpreted as policies to make decisions. These policies are parameterized using neural networks. The idea of learning policy is discussed next.

Learning policies:

i) Policy iteration [5]: Policy iteration is a dynamic programming-based algorithm that combines Iterative Policy evaluation and Policy improvement. Iterative policy evaluation is an algorithm for evaluating the goodness of a policy by iteratively going through each state and updating expected returns for state-values. Policy improvement is an algorithm that compares outputs retrieved from old policy π and a new one π' and deterministically pick the one with higher expected returns over action values.

ii) Monte Carlo method: Instead of computing every single action-value pair in a state space, giving the true expectation, the Monte Carlo sampling method is used to pick up samples of action-value pairs which can yield the best policy for the agent. Sampling the action-value pairs from the state space is requires consciously picking those pairs which are optimal. If the samples are finite, the probability of finding optimal pairs also decreases. Similarly, if the number of pairs is increased, the computational cost will rise. The trade-off between computational cost and optimal results is also known as the exploitation versus exploitation trade-off. Either exploit a small space, have lesser samples, and thus cost, but it may not be near to the optimal solution. Or explore the space more, get more samples, and thus cost and enhance the likelihood of getting a near-optimal solution.

State, Action, Rewards, and Policy:

From a Reinforcement learning standpoint, the following are the definitions of States, Actions, Rewards, and Policy.

i. **State:** Partial solution of the instance (set of nodes represented as a graph) and the feature of each node of the graph.

ii. **Action:** Choice of choosing the next node to visit.

iii. **Rewards:** Negative of total tour length.

iv. **Policy:** Heuristic strategy parameterized by a neural network.

3.1 Data for CVR

We have arbitrarily taken 1 depot node and 20 customer nodes. The depot nodes and the customer nodes can be defined by the coordinates in X-Y plane. The distance between each node can be calculated using Euclidean distance. Table 1 shows the location of 1 depot nodes: Each customer node in a graph has a demand associated.

Similarly, we have arbitrarily created multiple graphs for training (called graph instance). Each customer node is defined by the position in X, Y coordinates.

These are also randomly assigned demand values between 0-1. The capacity of the vehicle starting from the Depot node is taken as 1, and as a constraint to the CVRP the demand at any customer node cannot exceed the capacity of the vehicle. So, the demand values range in 0-1.

3.2 The cost function

The loss is the difference between the baseline REINFORCE [4] path length value and the value calculated on the paths obtained by the RL algorithm. The optimal route is the sequence of nodes traversed by the vehicle, where the traversal cost is found minimum.

3.3 The Environment of the CVRP agents

nodes will be marked visited. The functions in the environment file are written to ensure that any customer node once visited would be masked and a vehicle can return to either the depot node or any other customer node except the visited nodes.

Node No	X Coordinate	Y Coordinate	Demand
Depot	0.848307	0.32357132	
Node1	0.29036105	0.3107828	0.16666667
Node2	0.84289145	0.9391912	0.13333334
Node3	0.7035593	0.43545485	0.06666667
Node4	0.20168412	0.5328256	0.13333334
Node5	0.06545448	0.50693643	0.23333333
Node6	0.53601944	0.5814208	0.1
Node7	0.76553667	0.54667234	0.2
Node8	0.33509946	0.4530629	0.1
Node9	0.29120958	0.8327706	0.1
Node10	0.5238005	0.6762059	0.2
Node11	0.94022834	0.8333516	0.16666667
Node12	0.68327105	0.21968603	0.03333334
Node13	0.392851	0.72069836	0.2
Node14	0.84294224	0.36046684	0.03333334
Node15	0.8818841	0.04076123	0.06666667
Node16	0.6190139	0.31246388	0.23333333
Node17	0.71776164	0.03977752	0.3
Node18	0.9660357	0.4579624	0.23333333
Node19	0.39407074	0.259192	0.03333334
Node20	0.46775055	0.49120617	0.2

Table 1. Coordinates of depot nodes and customer nodes and demands of each customer node

4. Results

Using Deep Reinforcement Learning (Pang et al., 2020) for CVRP. Refer Fig 4.

Optimal path: For a CVRP problem, the following is the optimal path obtained for a graph instance and a depot. The cost of the path is 8.83766, which can be further improved by hyperparameter tuning.

Current path [0.0, 19.0, 1.0, 8.0, 4.0, 5.0, 9.0, 13.0, 14.0, 0.0, 18.0, 11.0, 2.0, 7.0, 0.0, 3.0, 6.0, 10.0, 20.0, 16.0, 0.0, 12.0, 17.0, 15.0, 0.0]

Where 0.0 indicates the depot node. The links between the paths are directed.

Cost and loss values vs. Epochs: Refer Fig 5.

The following plot shows the cost and loss vs. the number of epochs. As the training continues, we can observe that the training loss decreases. This can be further improved by hyperparameter tuning. Similarly, it can be observed that the training cost and validation cost (which is negative of the path length) are decreasing with training.

For comparison purpose we have used Genetic Algorithm (GA-VRP):

Genetic Algorithm-VRP
Genetic Algorithm (GA) is a search-based optimization technique based on the principles of Genetics and Natural Selection. It is used in finding optimal or near-optimal solutions to difficult problems that otherwise would take a long time to solve. Refer [6, 7] for details.

Solution Process:

i. Initialization: In the beginning, an initial generation must be defined. This can be done using a random initialization.

ii. Selection: First we select a proportion of the existing population to breed a new generation. The selection is done on a fitness-based approach where fitter individuals are more likely to breed than others.

iii. **Reproduction:** During the reproduction phase the next generation is created using the two basic methods, crossover, and mutation. For every new child, a pair of parents is selected from which the child inherits its properties. In the crossover, process genotype is taken from both parents and combined to create a new child. With a certain probability, the child is further exposed to some mutation, which consists of modifying certain genes. This helps to further explore the solution space and ensure, or preserve, genetic diversity. The occurrence of mutation is generally associated with low probability. A proper balance between genetic quality and diversity is therefore required within the population to support efficient search.

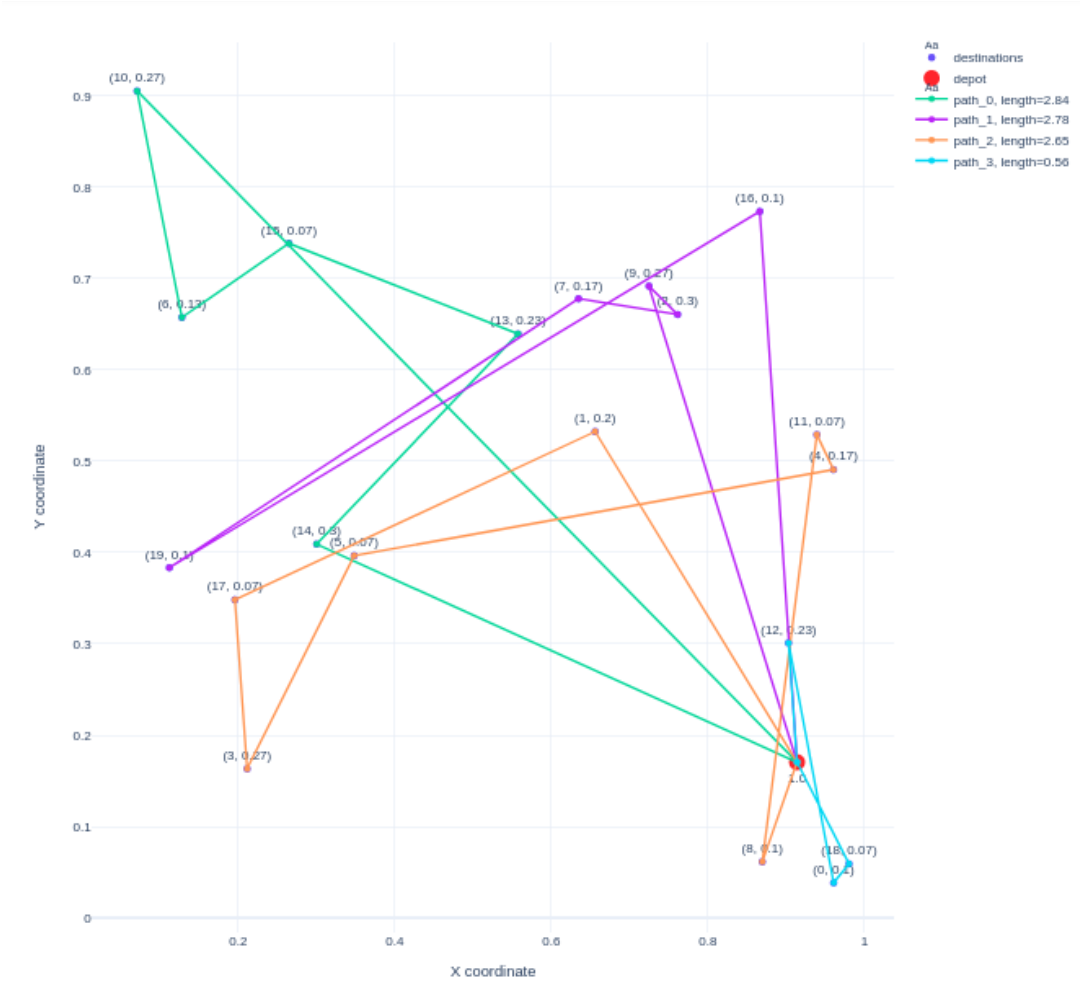


Fig.4: Graph solution for 20 nodes generated by DeepRL

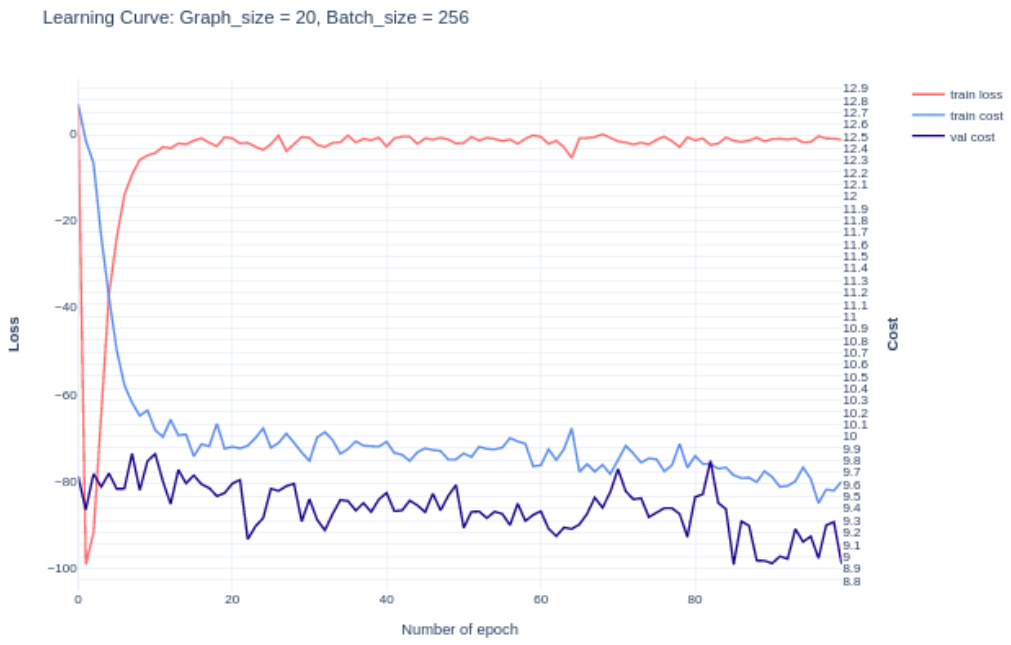


Fig.5: Cost and loss values vs. Epochs

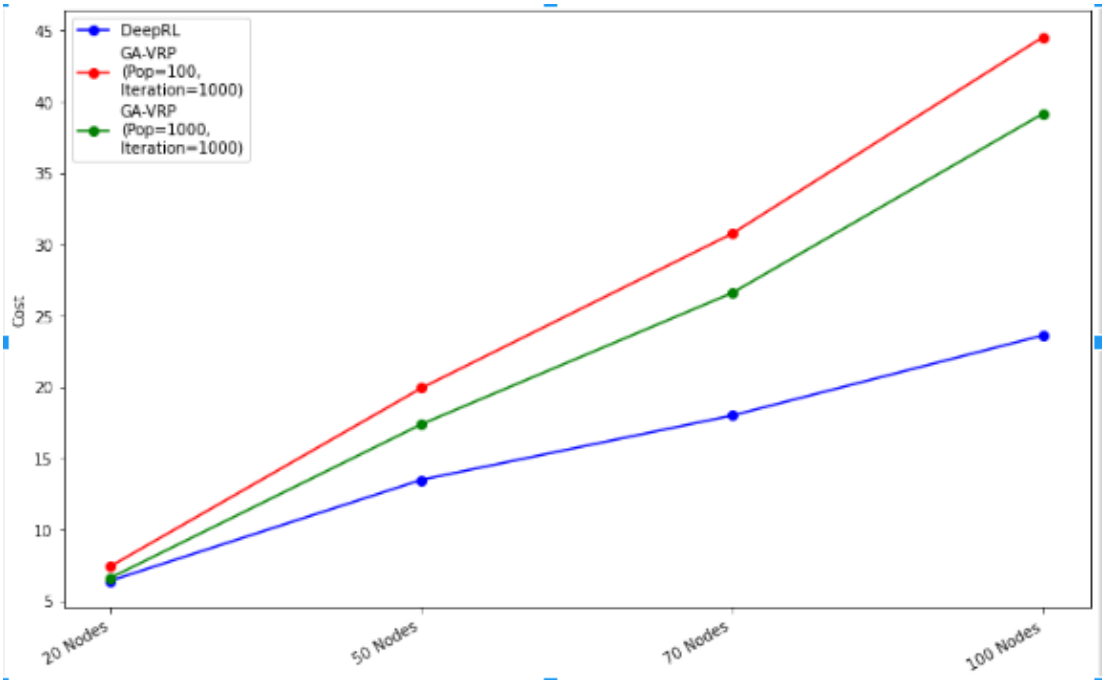


Fig.6: Cost vs. Nodes plot with different Algorithms to solve CVRP

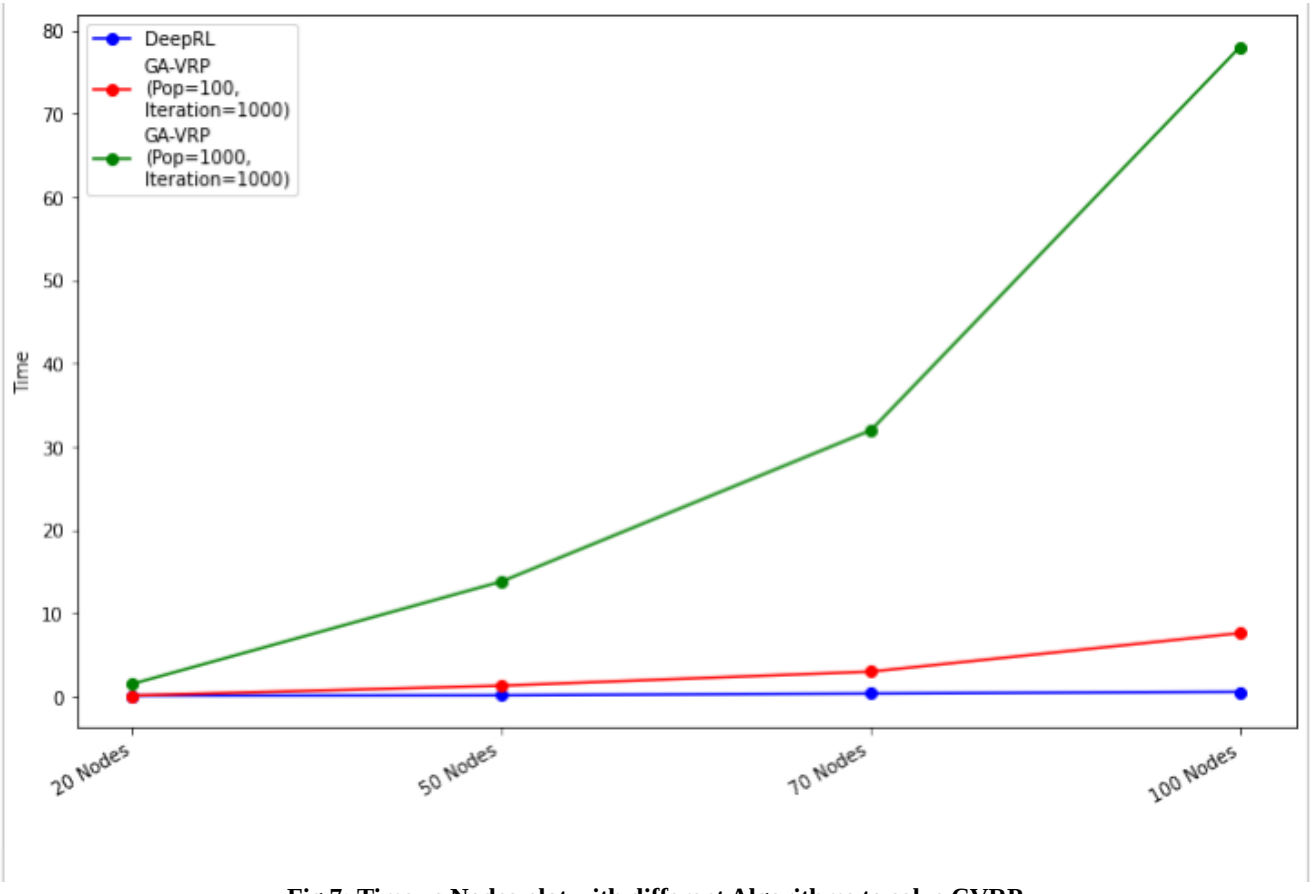


Fig.7: Time vs Nodes plot with different Algorithms to solve CVRP

No of Nodes	Cost (RL-VRP)	Time Taken (Sec)	Cost (GA-VRP, Pop=100, Iteration =1000)	Time Taken (Sec)	Cost (GA-VRP, Pop=1000, Itera-tion=1000)	Time Taken (Sec)
20 Nodes	6.413115	0.13227	7.429141	8.8206 7308	6.61718 4412	92.4954 0444
50 Nodes	13.48943	0.17408	19.92717	79.616 6889	17.3899 509	830.737 4334
70 Nodes	18.00218	0.38383	30.76816	180.86 43137	26.6052 6445	1919.96 6506
100 Nodes	23.6285	0.58476	44.56011	458.35 58464	39.1876 1302	4683.23 8334

Table 2. Results comparison of "Deep Learning-Reinforcement Learning (DeepRL) approach" and "Genetic Algorithm (GA) approach" with 10 Graphs for 20, 50, 70 and 100 customer nodes.

Note: These numbers are average of 10 Graphs for 20, 50, 70 and 100 nodes.

Refer Fig 6 and Fig 7.

Report Summary:

i. Time taken by DeepRL is very less and almost remains constant for different graph sizes.

ii. Graph generated by DeepRL and GA-VRP for 20 Nodes have almost same Cost and Time.

iii. Time Taken by GA-VRP increases significantly with increase in number of nodes or increase in population size (mentioned as POP in figure 6 & 7). It means DeepRL is comparably very efficient when number of customer nodes increases.

iv. Graph generated by DeepRL are more optimized as we increase the number of Nodes when compared with GA-VRP.

5. Future Work

In this work, we have used the Deep Reinforcement Learning algorithm to solve the CVRP. There are two important functionalities to be added to the codebase, as mentioned below.

i) Extending "Single depots Multiple customer nodes capacity planning problem" to "Multiple depots Multiple customer nodes capacity planning problem". In the real world, there can be multiple depot nodes, and the solution has to be optimized considering; the possibilities of any depot to any customer nodes with 2 constraints, and they are:

a) The demand of each customer node should not be more than the depot node capacity

b) Each customer node must be traversed only once

ii) Enriching the solution to take customer nodes and depot nodes as geographic locations. This technique will aid to optimize the routes between cities in the real-world with certain constraints. Moreover, using Euclidean distances is may is not the most reliable way to compute the distances between different cities. There are multiple factors like the traffic situation, roadway conditions, etc., that lead to the failure of VRP/CVRP using distances as evaluation measures. Next up, we will include "travel time" as a measure to evaluate the performance.

References

[1] Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning.*arXiv preprint arXiv:1611.09940*.
[2] Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. In *Advances in neural information processing systems* (pp. 2692-2700).
[3] Nazari, M., Oroojlooy, A., Snyder, L., & Takác, M. (2018). Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems* (pp. 9839-9849).
[4] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning.*Machine learning*, *8*(3-4), 229-256.
[5] Sutton, R. S., & Barto, A. G. (2017). Reinforcement learning: An introduction,(complete draft).
[6] Chunyu REN (2012) Applying Genetic Algorithm for Capacitated Vehicle Routing Problem
[7] Abdul Kadar, Abdul Kadar Muhammad Masum, Md. Faisal Faruque, & Mohammad Shahjalal (2011) Solving the Vehicle Routing Problem using Genetic Algorithm
[8] Rani, K., & Kumar, V. (2014). Solving travelling salesman problem using genetic algorithm based on heuristic crossover and mutation operator. *International Journal of Research in Engineering & Technology*, *2*(2), 27-34.

A Noninvasive model to detect Dengue based on symptoms using Artificial Intelligence and Machine Learning

Ruban S
PG Dept of Software Technology
St Aloysius College(Autonomous)
Mangalore, India
ruban@staloysius.ac.in

Naresha
PG Dept of Big Data Analytics
St Aloysius College(Autonomous)
Mangalore, India
nareshbhat333@gmail.com

Sanjeev Rai
Chief Research Officer
Father Muller Medical College
Mangalore, India
cmsfmci@fathermuller.in

***Abstract*— Artificial Intelligence has been transforming various sectors ranging from Finance, Entertainment, sports, Healthcare etc. The role of AI in healthcare will have an impact in all our lives owing to the change it brings to the Patientcare system, changing the traditional way of handling illness and diseases. Most of the AI based applications use Machine Learning algorithms that use data. Hence the source of Data and the nature of Data holds the key in developing effective AI based solutions for many health issues in the society. Though Data is available in all the hospitals and medical care facilities for many years now. They cannot be used directly to develop AI based applications until and unless they are transformed and made it into a format, on which machine learning algorithms can work. In this research paper, we discuss the process of developing an AI based application to predict Dengue, one of the vector borne diseases based on the symptoms. Our work was done on the data collected from the clinical notes of a 1500 bed hospital in the coastal district of Karnataka. We have implemented few of the machine algorithms like Logistic regression, Support vector machine and Decision Tree classifier. As the dataset is highly imbalanced (1:50), we applied over sampling techniques (Random over sampling, SMOTE) to overcome this problem. We compared the over sampling techniques and find that combination of SMOTE and Decision Tree classifier gave the best result (98% F1 micro score) compared to the other algorithms that we used in this study.**

Keywords*— *Artificial Intelligence, Machine Learning, Health Care, Medical Care, Dengue, Vector Borne disease

I. Introduction

The impact of Artificial Intelligence in health care sector is very evident, in recent times [1]. As it is defined traditionally AI is about developing machines with intelligence in contrast to the intelligence of human beings [2]. With more and more advances happening in the collection of data, processing and computing, intelligent systems are now assisting in these various tasks that once depended on human intervention. From Finance to Medical care [2] scenarios are changing drastically, in a way people never imagined before. However, all these advantages do come with various challenges. The challenges ranges from the algorithms, hardware implementation, development of application etc. AI involves developing systems that exhibit cognitive aptitude that uses technologies such as Machine Learning [3]. Every instance of the role of AI that we hear about, and its applications [4] in Health care takes advantage of the Data. Despite the digital revolution, most of the medical data are still handwritten [5]. Problems arise when other stake holders are involved either for interpretation or study. Poor handwritten clinical notes [6] poses a serious threat for researchers who are involved in data analysis. Dakshina kannada is one among the coastal districts of Karnataka, and reports many Dengue cases in a year. Dengue is one of the important vector borne diseases globally [7]. Few studies have been done to analyze the trends of vector borne diseases[8-9]. Artificial Intelligence (AI) has been used as a surveillance and prediction tool to predict vector borne diseases in different parts of the world [10]. The researchers of the above study came out with a system, that could predict the outbreak of dengue much earlier taking advantage of various data and parameters that were stored in different silos. Similar studies have also been done in other places as well [11]. Few such works are carried out in our country. [12–14]. However, in Indian scenario, there is hardly any study that is done in a deeper level involving clinical notes digitization. Many works take the demographic details and analyze. So an attempt was made to study the trends, symptoms, treatments of Dengue patients from the hospital records who were admitted in the span of four years (2015-2018) in Father Muller Medical College Hospital, Mangalore, Karnataka state, India. The study was conducted after the permission from the Research committees of the medical college. The medical data are maintained by the Hospital Medical Records Department (MRD). Section 2 elaborates the methodologies that were used and the following section discusses the results that we obtained from this study followed by a conclusion.

This study results have helped to understand the Dengue fever dynamics in this region, and can be used to predict the type of fever based on symptoms and hence probably can be used to assist the doctors for treating their patients quickly and effectively. Since the dataset that was generated for this study was highly imbalanced, we studied

the impact of oversampling and Synthetic Minority Oversampling Technique (SMOTE)[15]. Our experiments reveal that, in comparison of the other machine learning models, Decision Tree classifier gave the best result (98% F1 micro score) in this study.

.

II. MATERIALS AND METHODS

A. Data Sources

The Real Time Data Collection was done primarily in two locations - the DHO office in Mangalore, and the Father Muller Medical College. The Data from the DHO Office were gathered from different records, files and also by visiting different primary Health centers (PHC) and National Urban Health Mission centers (NUMC) in and around Mangalore. The data that were gathered from the PHC and NUMC did not have case sheets rather only basic demographic details. Hence the Data related to Dengue from Father Muller Medical College was accessed after getting the approval from the scientific and Ethics committee of the Father Muller Medical College, Mangalore.

B. Data related to Dengue

A patient suffering from Dengue presents few Mild symptoms such as fever, aches and pains. However most common symptom of dengue is fever with any of the following: eye pain, headache, muscle pain, rash, bone pain, nausea/vomiting, joint pain [16]. Symptoms of dengue typically last 2–7 days. Most people will recover after about a week. The positive confirmed dengue cases that were treated in Father Muller Medical college hospital during the year 2015-2018 were taken for the study. The individual patient medical records were accessed. The Data were available in two departments. The Registration department had patient details such as their IP number, name, sex, age, city, date of admission and date of discharge. The MRD department maintains a huge repository of case sheets where they are organized based on the ICD code [17].

TABLE I. Dengue Data Format maintained in Registration Department

Fields	Sample Data 1	Sample Data 2
IP Number	54xxxx45	87xxxx41
Patient Name	Xxxxxxx	Xxxxxxx
Age	45	54
Sex	Male	Male
City	Mangalore	Chickmangalur
DOA	20-11-2014 /09:15	10-11-2015 /10:15
Discharge Date	25-11-2014 / 01:31	17-11-2015 / 02:31
Primary Code	A90	A90
Primary Code Description	Dengue Fever	Dengue Fever

C. Data gathered from the Medical Records Department

The Data related to Dengue from Father Muller Medical College was accessed after getting the required permission from scientific and Ethical committee of the Father Muller Medical College, Mangalore. The Data related to Dengue were stored as Electronic Medical Records (EMR). The case sheets were scanned and stored in the Medical Records Department repository. However, the analysis of data was difficult. The corresponding patient history was accessed through the IP number, that acts as a unique identifier for the data that is stored in the Medical Records Department and the data that is stored in the Registration department.


DIAGNOSIS
DENGUE FEVER WITH THROMBOCYTOPENIA

CHIEF COMPLAINTS
FEVER- 6 DAYS
HEADACHE SINCE 6 DAYS
BODYPAIN SINCE 6 DAYS

HISTORY OF PRESENTING ILLNESS
Patient came with c/o fever since 6 day, intermittent ,high grade, asssociated with chills and rigo
h/o nausea and vomiting present
h/o headache present
h/o bodypain since 6 day.
h/o malena.
no h/o cough/expectoration/abdominal pain/diarrhoea/sore throat/burning micturition

PAST HISTORY
No history of Diabetes Mellitus, Hypertension, Tuberculosis, Asthma or IHD

FAMILY HISTORY
No history of Diabetes Mellitus, Hypertension, Tuberculosis, Asthma or IHD in the Family

PERSONAL HISTORY
Diet: Mixed.
Appetite: normal
Sleep: Adequate.
Bowel and Bladder: normal

Fig. 1. *Sample of Raw Data (Discharge Summary) that was extracted from the MRD*

D. Data Pre-processing

Major portion of the time in this research study was spend in this Data pre-processing step. All the health data thus collected go through Data pre-processing i.e., cleaning process where unnecessary information was removed. With the pre-processed data, we started finding patterns. We collected N-grams from the data using speechPyspellchecker package, which uses Levenshtein distance algorithm. The following steps were performed over the real time data collected from various Data sources related to Vector Borne diseases. In the initial phase, we dealt with various data quality issues. The initial data gathered is raw and usually not in a format to run the required analysis. It contains missing entries, inconsistencies, and semantic errors. After gathering the data, we clean and transform the data by manually editing it in the spreadsheet or by using Python. This step though does not give us much meaningful patterns or insight, however, consistently helps us to figure it out the right assumptions that should be made. This helps us to apply right

models that will assist in the important step of analysis. Data after re-formatting can be converted to JSON, CSV or any other format that makes it easy to load into one of our tools.

Exploratory data analysis forms an integral part at this stage, as the summarization of the clean data can help identify outliers, anomalies, and patterns that can become usable in the subsequent steps. This is the step that answer the question of the purpose for which data was collected. This phase consists of four primary sub steps: Data Cleaning, Data Integration, Data Transformation and Data Reduction.

1) Data Cleaning: Data cleaning helps in pre-processing. This helps to handle missing data, noisy data, detection and removal of outliers, minimizing duplication and computed biases within the data.

2) Data Integration: Different health organizations give us different data sources. These data sources must be integrated, to a single data point which is uniform that can be analyzed by the computer. In this case there were two types of data provided from two departments. One from the Registration department which gave us the demographic information about the patient and other data from the medical records department which gives information about the treatment that was given.

3) Data Transformation: The data we collected was in formats that are not optimal for processing. For example, if dates are involved, the data must be formatted from text to date format. In this state, we convert raw data into a useful format that can be processed with mathematical libraries. In this project the date of admission and date of discharge fields are used to compute the number of days the patient was admitted.

4) Data Reduction : Redundant data is identified and removed. Any unnecessary data is removed. This ensures that only valid data is used for processing.

E. Data Processing

The initial idea was to take screenshots of the patient discharge sheets and to extract text from those images. Each image which contained patient information from the day of his/her arrival to the day of discharge was recorded. In order to extract data from the images we used a python tool called Python-Tesseract. Python-Tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images. Python- tesseract is a wrapper for Google's Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others. All the images where run through the modified python program and the image files where converted into text files which contained all the textual information got from the images. Still there was a challenge with respect to the extracted files. Some of them had noise in them so the python program couldn't recognize the words in them and some of the extracted data was wrong.

Data Dictionary is developed as part of the Data processing in this project. This Data Dictionary was decided based on the Domain expertise. As this project was dealing with vector borne diseases such as Dengue and Malaria. The physicians in the medical college hospital were discussed about the Data to be captured from the clinical history that is recorded in the medical records. All the data parameters that has to be captured from the clinical history of the case files were extracted based on the domain experts' guidance. It acts as a Metadata. The following Data Dictionary was created for extracting information from the clinical files.

TABLE II. Features in the Data Dictionary

Diagnosis_Discharged_Improved	Joint Pain
Diagnosis_Dengue	Burning micturition
Fever	Vomiting
Cold	Chills
Cough	Loose stools
Headache	Nausea
Substance Abuse	Pallor
Clubbing	Lymphadenopathy
Breaths Per Minute	Heart Beats Per Minute
Abdominal Pain	IHD
Decreased Appetite	Malaria
Diabetes Mellitus	Diet
Hypertension	Sleep
Tuberculosis	Appetite
Asthma	Bowel & Bladder
Icterus	Cyanosis
Oedema	Blood Pressure
Temperature	-

III. RESULTS AND DISCUSSION

In this section the results that were derived out of the experiment are explained. The Exploratory Data Analysis was performed over the data that were collected. Few of the results are presented below that were derived from the Data that was collected from the Father Muller Medical College, Mangalore.

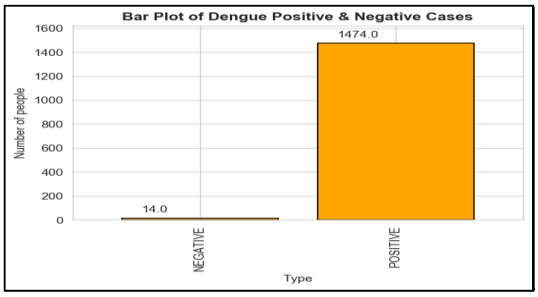


Fig. 2. Dengue positive and negative cases

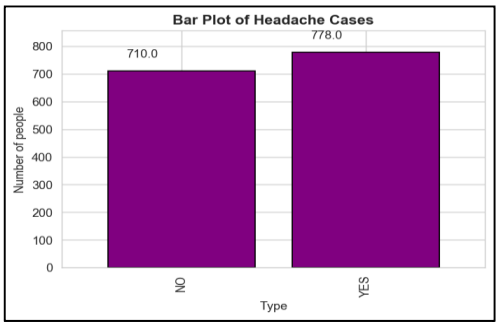


Fig. 3. Dengue cases with symptom of Headache

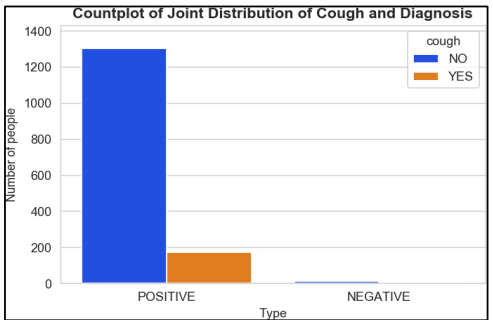


Fig. 4. Dengue cases with symptom of Cough

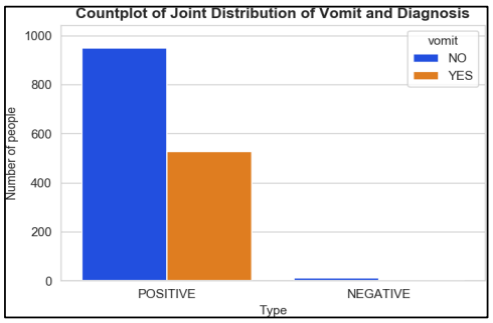


Fig. 5. Dengue cases with symptom of Vomiting

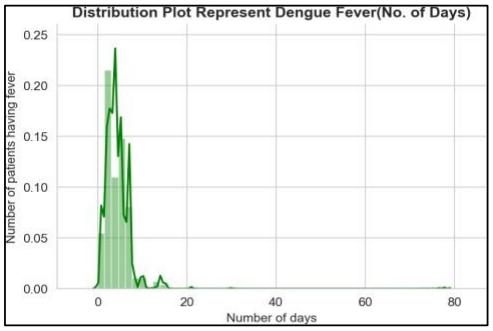


Fig. 6. Dengue cases with symptom of Fever

The data has now been enabled to perform any machine learning tasks such as classification or prediction or regression based on the need. There are various algorithms for each of the tasks that are mentioned above. Since the task that we have been trying to solve is a classification, we tried to find out the different algorithms that can be used for building a model. Before we start deciding the algorithm that should be used, we split the dataset into two parts. Machine Learning algorithms, or any algorithm for that matter, has to be first trained on the data distribution available and then validated and tested before it can be deployed to deal with real-world data. We tried using Logistic Regression, Support Vector Machine, KNeighborsClassifier and DecisionTreeClassifier for training the model. The various values that were generated for different metrics such as Accuracy, Precision, Recall and F-Measure are displayed below.

	Model Name	Train Accuracy	Test Accuracy	Precision	Recall	F1 score
3	GaussianNB	0.20	0.19	0.96	0.19	0.31
0	LogisticRegression	0.99	0.99	0.99	1.00	0.99
1	KNeighborsClassifier	0.99	0.99	0.99	1.00	0.99
2	DecisionTreeClassifier	0.99	0.99	0.99	1.00	0.99
4	SVC	0.99	0.99	0.99	1.00	0.99

Fig. 7. Performance comparison of different machine learning algorithms without oversampling.

	Model Name	Train Accuracy	Test Accuracy	Precision	Recall	F1 score
3	GaussianNB	0.91	0.83	0.99	0.84	0.91
4	SVC	0.94	0.90	0.99	0.91	0.95
0	LogisticRegression	0.96	0.94	0.99	0.95	0.97
1	KNeighborsClassifier	0.96	0.96	0.99	0.97	0.98
2	DecisionTreeClassifier	1.00	0.98	0.99	0.99	0.99

Fig. 8. Performance comparison of different machine learning algorithms with oversampling

	Model Name	Train Accuracy	Test Accuracy	Precision	Recall	F1 score
4	SVC	0.95	0.90	0.99	0.91	0.95
0	LogisticRegression	0.97	0.94	0.99	0.95	0.97
3	GaussianNB	0.96	0.95	0.99	0.96	0.97
1	KNeighborsClassifier	0.98	0.96	0.99	0.97	0.98
2	DecisionTreeClassifier	1.00	0.98	0.99	0.99	0.99

Fig. 9. Performance comparison of different machine learning algorithms with SMOTE

IV. CONCLUSION

This research study based on clinical notes of the patient, treated for Dengue, provides an insight into the types of symptoms prior to hospital admission. It also explores the efficiency of diagnostic treatment for Dengue. The quicker a physician assesses based on the symptoms, more effective the treatment tends to be. This study was done with data collected from one specific location. More data from different hospital setting and

different places would increase the efficiency of the System. However, the same steps that were performed in the preprocessing stages can be repeated for any hospital setting to gather data and transform the raw clinical data into a meaningful data over which effective AI based model can be built. Since the dataset that was generated for this study was highly imbalanced, we studied the impact of oversampling and Synthetic Minority Oversampling Technique (SMOTE). Our experiments reveal that, in comparison of the other over sampling techniques, Decision Tree classifier gave the best result (98% F1 micro score).

ACKNOWLEDGMENT

The authors would like to acknowledge, that this work was done in the lab funded by Vision Group of science and Technology (VGST), Government of Karnataka, under the Grant scheme K- FIST(L2)-545 and the data was collected from Father Muller Medical College Hospital, based on the Ethics committee approval via protocol no: 126/19(FMMCIEC/CCM/149/2019) on 12.06.2019.

REFERENCES

[1] Guogang Rong, Arnaldo Mendez, Elie Bou Assi, Bo Zhao, Mohamad Sawan. Artificial Intelligence in Healthcare: Review and prediction case studies. Engineering (2020), 6(3), 291-301.

[2] J.Weng, J. McClelland, A.Pentland, O.Sporns, I. Stockman, M.Sur, et al. Autonomous mental development by robots and animals. Science (2020), 291(5504), 599-600.

[3] Limitations of Artificial Intelligence. [Accessed on September 27, 2020 at https://www.analyticsinsight.net/top-5-limitations-artificial-intelligence].

[4] G.Huang, G.G.Huang, S.Song, K.You. Trends in extreme learning machines: a review. Neural Network (2015), 61, 32-48.

[5] Y.Guo,Y.Liu, A.Oerlemans,S.Lao,S.Wu, M.S. Lew. Deep Learning for visual understanding: a review. Neurocomputing (2016), 187, 27-48.

[6] F. Javier Rodriquez – vera Y Marin, A Sanchez, C Borrachero, E pujal. Illegible handwriting in medical records. Journal of the royal society of medicine (2002), 95, 545-546.

[7] Bhatt S, Gething PW,Brady OJ, The global distribution and burden of dengue. Nature (2013).496, 504-507.

[8] Rashmi Sharma,"Epidemiological Investigation Of Malaria Outbreak In Village Santej, District Gandhi Nagar (Gujarat)",Indian J. Prev. Soc. Med. Vol. 37 No. 3& 4 , 2006

[9] George T, Jakribetta RP, Yesudhas S,Thaliath A,Pais MLJ, Abraham S, Baliga MS. Trend analysis of dengue in greater mangalore region of karnataka india:observations from a tertiary care hospital. International Journal of Applied Research (2018),4(6),92-96.

[10] Sundram BM, Raja DB, Mydin F, Yee TC, Raj K. Utilizing Artificial Intelligence as a Dengue Survellance and prediction tool. J Appl Bioinforma Comput Biol (2019), 8:1.

[11] Laureano-Rosario AE, Duncan AP, Mendez-Lazaro PA, Garcia-Rejon JE,Gomez-carro S,Farfan-Ale J, Savic DA,Muller-Karger FE. Application of Artificial Neural Networks for dengue fever outbreak predictions in the Northwest Coast of Yucaton,Mexico and San Juan, Puerto Rico. Trop. Med. Infect. Dist(2018)3-5.

[12] Baruah J, Ananda S, Arun kumar G.Incidence of dengue in a tertiary care centre-Kasturba Hospital, Manipal. Indian J Pathol Microbiol(2006).49(3),462-3.

[13] Pai Jakribettu R, Boloor R, Thaliath A, Yesudasan George S, George T, Ponadka Rai M et al. Correlation of Cinicohaematological parameters in paediatric Dengue: A retrospective study. J Trop Med (2015).6(47), 162.

[14] Damodar T, Dias M, Mani R, Shipla KA, anand AM, Ravi V et al. clinical and laboratory profile of dengue viral infections in and around mangalore. Indian J Med Microbiol(2017), 35(2), 256-261.

[15] Blagus and Lusa: SMOTE for high-dimensional classimbalanced data. BMC Bioinformatics 2013 14:106.

[16] Symptoms of Dengue [Accessed on October 3rd 2020,https://www.cdc.gov/dengue/symptoms/index.html]

[17] ICD code for Dengue [Accessed on September 21st 2020, https://icd.codes/icd10cm/A90] [18]Symptoms of Malaria [Accessed on September 24th 2020, https://www.healthline.com/health/malaria#diagnosis].

Efficient and Optimal Deep Learning Inference for Computer Vision Applications

Venkatesh Wadawadagi
Solution Consultant - AI/ML, Engineering and Analytics
Sahaj Software Solutions

Bengaluru, India
venkateshw@sahaj.ai

***Abstract*— Journey of a cognitive solution is meaningful when it's put to use or can actually solve business problems in real time through inference. Deep Learning model Inference is as important as model training and especially when it comes to deploying cognitive solutions on the edge, inference becomes a lot more critical as it also controls the performance and accuracy of the implemented solution. For a given computer vision application, once the deep learning model is trained, the next step would be to ensure it is deployment/production ready, which requires application and model to be efficient and reliable. It's very essential to maintain a healthy balance between model performance/accuracy and inference time. Inference time decides the running cost for "on the cloud" solutions and cost optimal "on the edge" solutions come with processing speed and memory constraints, so it's important to have memory optimal and real time (lower processing time) deep learning models. With the rising use of Augmented Reality, Facial Recognition, Facial Authentication and Voice assistants that require real time processing, developers are looking for newer and more effective ways of reducing the size/memory and amount of compute required for the application of neural networks.**

Keywords*— *Accuracy, Inference time, On the edge

I. INTRODUCTION

Recent machine learning methods use increasingly large Deep Neural Networks(DNNs) to achieve state of the art results in various tasks. The gains in performance come at the cost of a substantial increase in computation and storage requirements. This makes real-time implementations on limited resources hardware a challenging task. DNNs proved to be extremely effective in solving a broad variety of problems in computer vision. Deep learning methods are usually evaluated only according to their accuracy over a given task. This criterion leads to the development of architectures with constantly increasing computational complexity and memory requirements. Thus, performing inference on low power System on a Chip (SoCs) used in smartphones or IoT devices is a significant challenge, due to the limited available memory and computational resources.

Large-scale datasets, high-end modern GPUs and new network architectures allow the development of unprecedented large CNN models. For instance, from AlexNet [1], VGGNet [2] and GoogleNet [3] to ResNets [4], the ImageNet Classification Challenge winner models have evolved from 8 layers to more than 100 layers. However, larger CNNs, although with stronger representation power, are more resource-hungry. For instance, a 152-layer ResNet [14] has more than 60 million parameters and requires more than 20 Giga float-point-operations (FLOPs) when inferencing an image with resolution 224×224. This is unlikely to be affordable on resource constrained platforms.

Over the years several approaches have been proposed in order to make DNNs less resource demanding. These approaches include Quantization of neural networks, Network pruning and Matrix factorisation via low-rank approximation of neural networks. Other ways of running optimal deep learning inferences include leveraging DNN inference accelerator frameworks and libraries. There are tools and libraries that have been developed to cater to run optimal inference on the edge and mobile devices.

II. BACKGROUND

Convolutional neural network (CNN) is a type of artificial neural network that has been successfully applied in many areas, especially in visual imagery [24]. A convolutional neural network consists of three building blocks: convolutional layer, pooling layer and fully-connected layer. A simple convolutional neural network is shown in Fig 1.

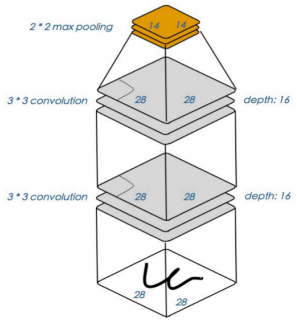


Figure 1: An illustration of convolutional neural networks.

Convolutional layer is a major building block of CNNs. It is used to extract features from images. In each convolutional layer we have a set of filters. During the forward pass, we slide each filter across the image and compute dot products

between the filter and the local receptive field. The output of the convolutional layer is called activation map that gives the response of each filter. Given an image I and a m x n filter F, an element $a_{i,j}$ in the activation map can be computed as,

$$a_{i,j} = \sum_{a=1}^{m} \sum_{b=1}^{n} I_{i+a-1,j+b-1} \times F_{a,b} \qquad (1)$$

The convolution operation is computationally very expensive. For example, the total time complexity of all convolutional layers can be expressed as $O(\sum_{l=1}^{d} n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2)$ [25]. Here l is the index of a convolutional layer and d is the number of convolutional layers. n_l is the number of filters in the l-th layer. n_{l-1} is the number of input channels of the l-th layer. s_l is the spatial size of the filter. m_l is the spatial size of the activation map. The computational cost of the convolutional layer motivates us to use low bit-width filters and inputs. With low bit-width filters and inputs, the dot product can be efficiently implemented by bitwise operations which can greatly accelerate the computation.

III. QUANTIZATION OF NEURAL NETWORKS

Quantization is recognised as one of the most effective approaches to satisfy the extreme memory requirements that deep neural network models demand. Instead of adopting 32-bit floating point format to represent weights, quantized representations store weights using more compact formats such as integers or even binary numbers. Despite a possible degradation in predictive performance, quantization provides a potential solution to greatly reduce the model size and the energy consumption.

Quantizing neural networks dates back to the 1990s [5;6;7;8]. In the early days, the main reason to quantize these models is to make it easier for digital hardware implementation. Recently, the research of quantizing neural networks has revived due to the success of deep neural networks and their huge sizes. A slew of new quantization methods and methodologies have been proposed. These efforts have enabled the quantized neural networks to have the same accuracy level as their full-precision counterparts.

Quantization methods attempt to reduce the precision of the NN parameters and/or activations from single precision (32 bit floating point, or FP32) to lower bit representations. Several benefits of low-bit precision can be exploited by deep learning accelerators. The storage requirement for a low-bit precision model can be diminished substantially, as well as the power consumption. Similarly, the memory bandwidth requirements can be significantly reduced. Since the multiply accumulate (MAC) operations are performed on low-bit processing engines, the computational complexity can be reduced as well. Perhaps the most important benefit of low bit representation is the saving of chip area. For instance, 8 bits integer (INT8) operations can save up to 30x energy and up to 116x area compared to FP32 operations [9], allowing significantly better computational throughput. However, low-bit precision inference often causes loss of the task accuracy, which is usually compensated with the help of heavy full retraining, mixed precision or non-uniform quantization.

Using a rounding function is an easy way to convert real values into quantized values. However, the network performance may drop dramatically after each rounding operation. It is necessary to keep the real values as reference during training which increases the memory overhead. Meanwhile, since the parameter space is much smaller if we use discrete values, it is harder for the training process to converge. Finally, rounding operation cannot exploit the structural information of the weights in the network.

There are two main quantization scenarios. The first one is the full training of a given model to a desired lower bit precision. With this approach, the weights, the activations and even the gradients can be quantized to very low precision, enabling potentially fast training and inference [10]. The major problem with the training approach above arises from the discreetness of the parameters, wherein the back-propagation approach is not well defined. The ”straight-through estimator” [11] has been used in [12, 13, 14] in order to estimate the gradient of a stochastic neuron. [12] proposed to use stochastic quantization of the weights via random rounding, in order to inject regularising noise to the training process. [15] suggests to approximate solution using variational Bayes method where the weights can be restricted to discrete values assuming Gaussian distribution. Instead of seeking for appropriate derivatives, [16] assumed smooth approximation of parameters with defined gradients. Non-uniform quantization of NN parameters has been proposed in [17] where the param- eters are approximated using k-means algorithm. [18] pro- posed a high order quantization scheme of weights where the approximation residual is further processed allowing better refinement of full precision input. Estimation of the quantization parameters by solving constrained optimisation problem has been proposed for binary [13] and ternary weights [19].

Second quantization scenario targets direct quantization of a pre-trained FP32 network to a lower bit-depth precision without full training. INT8 quantization of parameters has been proven to be relatively robust to quantization noise even with simple uniform quantization of weights [20]. [21] proposed L2 error minimisation of weights via alternating optimisation in order to obtain a generalising ability during the training. Nevertheless, INT8 quantization of the network activations is more challenging because of real time constraints. Nvidia proposed in TensorRT [22] a quantization framework that searches for saturation threshold of the activations, based on the Kullback-Leibler divergence measure between the quantized activations and their full precision counterpart. Recently, [23] pro- posed to approximate activations, as if they were sampled from a known distribution in order to obtain, under some assumptions, analytically optimal threshold in the L2 sense. However, quantization of full precision weights and activations to less than 8-bits usually causes significant loss of accuracy, a problem that has not been solved yet. In order to overcome such degradation in performance, quantization frameworks resort to retraining procedures, mixed precision solutions or non-uniform quantization. These solutions make fast and easy deployment of quantized NNs impossible, especially on highly constrained HW such as mobiles or IoT devices.

A. Weight Quantization

The motivation to quantize weights is clear: to reduce model size and accelerate training and inference process. Most of the methods we talked above can be used to quantize weights. In this section, we introduce more weight quantization strategies that we did not cover before. [26] proposed a layer-wise quantization scheme to reduce the performance degradation. In [27], the authors adopted a two-step pipeline. In the first step, the weights are compressed into the range of $[-1, 1]$ and in the second step the compressed weights are used to initialize the parameters of a binary network. In [28], the authors proposed incremental network quantization (INQ) which consists of three steps: weight partition, group-wise quantization and re-training. They quantized the weights in a group-wise manner to allow some groups of weights to compensate the accuracy loss due to the quantization of other groups. The work in [29] extended this method to power-of-two setting.

In [30] the authors tried to find the optimal fixed point bit-width allocation across layers. They examined how much noise can be introduced by quantizing different layers. [30] approximated the full-precision weights with a linear combination of multiple binary bases. The results show that it is the first time that a binary neural network can achieve prediction accuracy comparable to its full-precision counterpart on ImageNet dataset. In [31], the authors studied how to develop energy efficient quantized neural network. The work in [32] introduced network sketching to quantize a pre-trained model. The idea is to use binary basis to approximate pre-trained filters. They first proposed a heuristic algorithm to find the binary basis and then provided a refined version to better approximation. In [33], the authors proposed an end-to-end training framework to optimize original loss function, quantization error and the total number of bits simultaneously. However, the accuracy is not comparable to other quantized neural networks.

There are few challenges associated with quantization of weights. Quantized weights make neural networks harder to con- verge. A smaller learning rate is needed to ensure the network to have good performance [34]. Determine how to control the stability of the training process in a quantized neural network with quantized weights is critical.

Quantized weights make back-propagation infeasible since gradient cannot back-propagate through discrete neurons. Approximation methods are needed to estimate the the gradients of the loss function with respect to the input of the discrete neurons. Developing low-variance, unbiased gradient estimates is essential for the success of weight quantization. It is known that the weights in neural networks often follow some general structures. For an approach that trains quantized networks from scratch, how to quantize the weights locally while maintain their global structure is an issue.

B. Activation Quantization

Quantized activations can replace inner-products with binary operations which can further speed up the network training. We can also reduce the much memory by avoiding full-precision activations. [35] quantized the activations to 8 bits. They used a sigmoid function which limits the activations to the range of [0, 1] and quantized the activations after training the network. In [12;13;10] the authors adopted a similar approach. They introduced a continuous approximation of the non-differentiable operator during back-propagation to enable the gradients can back-propagate through the discrete neurons. More recently, [36] proposed an half-wave Gaussian quantizer to approximate the ReLU unit. In the forward approximation, they used a half- wave Gaussian quantization function,

$$Q(x) = \begin{cases} q_i & \text{if } x \in (t_i, t_{i+1}], \\ 0 & x \leq 0, \end{cases} \tag{2}$$

If use mean squared error to measure the performance, the optimal quantizers can be found as follows,

$$Q^*(x) = \underset{Q}{\operatorname{argmin}} \mathbf{E}_x[(Q(x) - x)^2] \tag{3}$$

They used batch normalization [37] and Lloyd's algorithm to find the optimal solution. During back-propagation, they further introduced three possible approximation method to avoid the gradient vanishing problem.

In [Mishra et al., 2017], the authors proposed wide reduced precision networks (WRPN) to quantize activation and weights. They found that activations actually occupy more memory than weights. They adopted a strategy that increases the number of filters in each layer to compensate the accuracy degradation due to quantization.

There are some reasons that make the quantization of activations more difficult than that of weights [36]. The first one is that we need to back-propagate through the non-differentiable operators. Consider the back-propagation equation,

$$\frac{\partial L}{\partial w_{ij}} = g'(a_j) \sum_k w_{jk} \frac{\partial L}{\partial a_k} x_i \tag{4}$$

When we replace $g(a_j)$ with a binary operator, the derivative $g'(a_j)$ is almost zero everywhere which makes gradient descent algorithm infeasible. The quantized activations can lead to "gradient mismatch" problem [38] which means that there is a discrepancy between the quantized activation with the computed backward gradient.

C. Gradient Quantization

Gradient quantization is a new branch of research in quantization of neural networks. The motivation to quantize gradients is to reduce the communication cost during distributed stochastic gradient descent (SGD) training of large neural networks. The magnitude and sign of gradients are both important for updating the weights. To quantize gradients, we must address the question of how to take both factors into account. A naive way to quantize gradients may not work well in practice since it may violate the conditions needed for stochastic gradient descent algorithm to converge. More sophisticated methods are needed in this case.

IV. PRUNING OF NEURAL NETWORKS

While modern deep CNNs are composed of a variety of layer types, runtime during prediction is dominated by the evaluation of convolutional layers. With the goal of speeding up inference, we can prune feature maps so that resulting networks may be run efficiently. Pruning is a popular approach to reduce a heavy network to obtain a light-weight form by removing redundancy in the heavy network. Simply put, pruning is a way to reduce the size of the neural network through compression.

Pruning neural networks is an old idea going back to 1990 (with Yan Lecun's optimal brain damage work) and before. These early works [40;39] performed pruning using a second order Taylor approximation of the increase in the loss function of the network when a weight is set to zero.

The idea is that among the many parameters in the network, some are redundant and don't contribute a lot to the output.

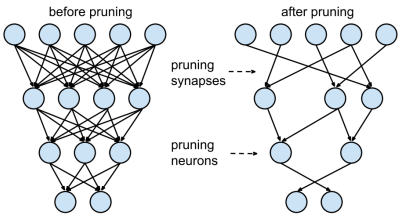


Figure 2: An illustration of neurons before and after pruning.

Neural networks generally look like the one on the left as shown in figure-2: every neuron in the layer below has a connection to the layer above, but this means that we have to multiply a lot of floats together. Ideally, we'd only connect each neuron to a few others and save on doing some of the multiplications; this is called a sparse network.

If you could rank the neurons in the network according to how much they contribute, you could then remove the low ranking neurons from the network, resulting in a smaller and faster network (the one on the right side as shown in figure-2) .

The ranking, for example, can be done according to the L1/ L2 norm of neuron weights. After the pruning, the accuracy will drop (hopefully not too much if the ranking is clever), and the network is usually trained-pruned-trained-pruned iteratively to recover. If we prune too much at once, the network might be damaged so much it won't be able to recover. So in practice, this is an iterative process — often called 'Iterative Pruning': Prune / Train / Repeat.

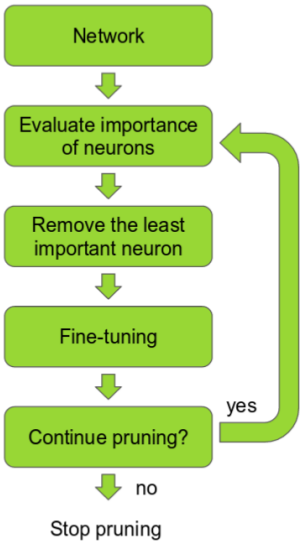


Figure 3: Network pruning as a backward filter [41].

A. *Unstructured Pruning*

These methods prune individual parameters. Doing so produces a sparse neural network, which although smaller in terms of parameter count may not be arranged in a fashion conducive to speedups using modern libraries and hardware. It is also called Weight Pruning as we set individual weights in the weight matrix to zero. This corresponds to deleting connections as in the figure-2 [42] [40]. Here, to achieve sparsity of *k%* we rank the individual weights in weight matrix *W* according to their magnitude, and then set to zero the smallest *k%*.

B. *Structured Pruning*

These methods consider parameters in groups, removing entire neurons, filters, or channels to exploit hardware and software optimized for dense computation. It is also called Unit/Neuron Pruning as we set entire columns in the weight matrix to zero, in effect deleting the corresponding output neuron.

In [43] method they advocate pruning entire convolutional filters. Pruning a filter with index *k* affects the layer it resides in, and the following layer. All the input channels at index *k*, in the following layer, will have to be removed, since they won't exist any more after the pruning.

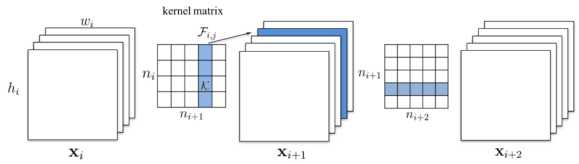


Figure 4: Pruning filters for efficient convnets [43]

In case the following layer is a fully connected layer, and the size of the feature map of that channel would be MXN, then MXN neurons be removed from the fully connected layer. The neuron ranking in this work is fairly simple. It's the L1 norm of the weights of each filter. At each pruning iteration they rank all the filters, prune the m lowest ranking filters globally among all the layers, retrain and repeat.

C. *Evaluating Pruning*

Pruning can accomplish many different goals, including reducing the storage footprint of the neural network, the computational cost of inference, the energy requirements of inference etc. Each of these goals favours different design choices and requires different evaluation metrics. For example, when reducing the storage footprint of the network, all parameters can be treated equally, meaning one should evaluate the overall compression ratio achieved by pruning. However, when reducing the computational cost of inference, different parameters may have different impacts. For instance, in convolutional layers, filters applied to spatially larger inputs are associated with more computation than those applied to smaller inputs.

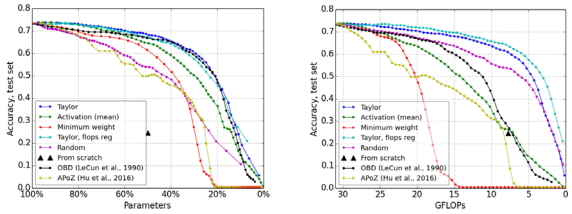


Figure 5: Pruning of feature maps in VGG-16 fine-tuned on the Birds-200 dataset [43]

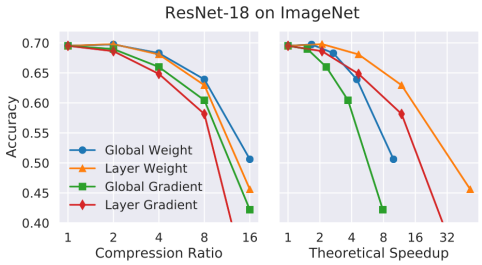


Figure 6: Pruning filters for efficient convnets [43]

Regardless of the goal, pruning imposes a tradeoff between model efficiency and quality, with pruning increasing the former while (typically) decreasing the latter. This means that a pruning method is best characterised not by a single model it has pruned, but by a family of models corresponding to different points on the efficiency-quality curve.

V. Low-rank Factorization

This method approximates weight matrix in neural networks with low-rank matrix using techniques like Singular value decomposition (SVD), QR decomposition with column pivoting, rank revealing QR factorization (RRQR), Interpolative decomposition etc. These methods work especially well on fully-connected layers, yielding ~3x model-size compression however without notable speed acceleration, since computing operations in CNN mainly come from convolutional layers. These techniques are very expensive ($O(n^3)$ operations are required for $n\,x\,n$ matrices). There are several randomized algorithms available in the literature which are not so expensive as the classical techniques (but the complexity is not linear in n). So, it is very expensive to construct the low rank approximation of a matrix if the dimension of the matrix is very large. There are alternative techniques like Cross/Skeleton approximation which gives the low-rank approximation with linear complexity in n.

In [44] the authors exploit the linear structure present within the convolutional filters in order to develop approximations which reduce the computations. The report the 2x speedup for a layer with 1% accuracy drop on classification task. In [45] the cross-channel or filter redundancy was exploited for construction the low-rank basis of filters. The developed methods showed the 4.5x compression with less than 1% drop in classification accuracy. Continuing to work in the similar direction, Lebedev et al. in their research [46] applied low-rank CPD on order-4 convolution kernel tensor. The experiments conducted in this paper demonstrated 8.5x CPU speedup with 1% accuracy drop for the 36-class character classification CNN and 4x speedup of second layer of the AlexNet[1] for ImageNet classification [47]. The paper [48] presents the one-shot whole CNN compression scheme based on the Tucker decomposition. Authors also describe the rank selection method with the help of variational Bayesian matrix factorization (VBMF) [49]. Another tensor decomposition - tensor-train decomposition - is used in [50] for converting the weight matrices of the fully-connected layers to the TT-format. TT-decomposition is also used in [51]. In this work both convolutional and fully-connected layers are compressed with TT-decomposition. The significant compression ratio of 80x is achieved with 1.1% accuracy drop on the CIFAR-10 classification. The most famous tensor decompositions are Canonical Polyadic (CPD) [52], Tucker decomposition [53] and tensor-train decomposition [54]. The Tucker tensor format is chosen for the low-rank approximation of convolutional weights.

VI. Inference Accelerators

A. *TensorRT*

If you want to get the best performance out of your GPUs, NVIDIA offers TensorRT, a model compiler for inference

deployment. It does additional optimizations to a trained model, and a full list is available on NVIDIA's TensorRT website. Key optimizations include quantization and graph fusion. Quantization reduces model precision from FP32 (single precision) to FP16 (half precision) or INT8 (8-bit integer precision). Graph fusion fuses multiple layers/ops into a single function call to a CUDA kernel on the GPU. This reduces the overhead of multiple function call for each layer/op.

Deploying with FP16 is straight forward with NVIDIA TensorRT. The TensorRT compiler will automatically quantize your models during the compilation step. To deploy with INT8 precision, the weights and activations of the model need to be quantized so that floating point values can be converted into integers using appropriate ranges. You have two options. In first option you need to perform quantization aware training, where-in the error from quantizing weights and tensors to INT8 is modelled during training, allowing the model to adapt and mitigate this error. This requires additional setup during training. Second option is about performing post training quantization. In post-quantization training, no pre-deployment preparation is required. You will provide a training model in full precision (FP32), and you will also need to provide a dataset sample from your training dataset that the TensorRT compiler can use to run a calibration step to generate quantization ranges.

	keras_gpu_8	trt_fp32_8	trt_fp16_8	trt_int8_8
instance_type	g4dn.xlarge	g4dn.xlarge	g4dn.xlarge	g4dn.xlarge
user_batch_size	8	8	8	8
accuracy	0.74956	0.74956	0.74968	0.74924
prediction_time	440.113	38.1336	38.0335	34.3497
wall_time	443.712	143.327	135.078	133.087
images_per_sec_mean	115.746	1666.69	1707.24	1895.03
images_per_sec_std	7.3476	960.928	1016.37	1086.22
latency_mean	70.418	6.10138	6.08536	5.49594
latency_99th_percentile	84.4612	13.797	14.1668	12.2826
latency_median	69.0285	5.91063	5.91636	5.27298
latency_min	62.314	1.36304	1.43266	1.44053

Figure 7: Comparison of accuracy and performance of TensorFlow ResNet50 inference.

VII. Inference Libraries For Edge/Mobile Devices

A. tkDNN

tkDNN[55] is a Deep Neural Network library built with cuDNN and tensorRT primitives, specifically thought to work on NVIDIA Jetson Boards. It has been tested on TK1(branch cudnn2), TX1, TX2, AGX Xavier, Nano and several discrete GPUs. The main goal of this project is to exploit NVIDIA boards as much as possible to obtain the best inference performance. It does not allow training.

Platform	Network	FP32, B=1	FP32, B=4	FP16, B=1	FP16, B=4	INT8, B=1	INT8, B=4
RTX 2080Ti	yolo4 320	118.59	237.31	207.81	443.32	262.37	530.93
RTX 2080Ti	yolo4 416	104.81	162.86	169.06	293.78	206.93	353.26
RTX 2080Ti	yolo4 512	92.98	132.43	140.36	215.17	165.35	254.96
RTX 2080Ti	yolo4 608	63.77	81.53	111.39	152.89	127.79	184.72
AGX Xavier	yolo4 320	26.78	32.05	57.14	79.05	73.15	97.56
AGX Xavier	yolo4 416	19.96	21.52	41.01	49.00	50.81	60.61
AGX Xavier	yolo4 512	16.58	16.98	31.12	33.84	37.82	41.28
AGX Xavier	yolo4 608	9.45	10.13	21.92	23.36	27.05	28.93
Xavier NX	yolo4 320	14.56	16.25	30.14	41.15	42.13	53.42
Xavier NX	yolo4 416	10.02	10.60	22.43	25.59	29.08	32.94
Xavier NX	yolo4 512	8.10	8.32	15.78	17.13	20.51	22.46
Xavier NX	yolo4 608	5.26	5.18	11.54	12.06	15.09	15.82
Tx2	yolo4 320	11.18	12.07	15.32	16.31	-	-
Tx2	yolo4 416	7.30	7.58	9.45	9.90	-	-
Tx2	yolo4 512	5.96	5.95	7.22	7.23	-	-
Tx2	yolo4 608	3.63	3.65	4.67	4.70	-	-
Nano	yolo4 320	4.23	4.55	6.14	6.53	-	-
Nano	yolo4 416	2.88	3.00	3.90	4.04	-	-
Nano	yolo4 512	2.32	2.34	3.02	3.04	-	-
Nano	yolo4 608	1.40	1.41	1.92	1.93	-	-

Figure 8: Inference FPS of yolov4 with tkDNN, average of 1200 images with the same dimension as the input size, on RTX 2080Ti (CUDA 10.2, TensorRT 7.0.0, Cudnn 7.6.5); Xavier AGX, Jetpack 4.3 (CUDA 10.0, CUDNN 7.6.3, tensorrt 6.0.1); Xavier NX, Jetpack 4.4 (CUDA 10.2, CUDNN 8.0.0, tensorrt 7.1.0); Tx2, Jetpack 4.2 (CUDA 10.0, CUDNN 7.3.1, tensorrt 5.0.6); Jetson Nano, Jetpack 4.4 (CUDA 10.2, CUDNN 8.0.0, tensorrt 7.1.0).

Steps needed to do inference on tkDNN with a custom neural network.

- Build and train a NN model with your favorite framework.
- Export weights and bias for each layer and save them in a binary file (one for layer).
- Export outputs for each layer and save them in a binary file (one for layer).
- Create a new test and define the network, layer by layer using the weights extracted and the output to check the results.
- Do inference.

B. TFLite

On-device machine learning (ML) offers a variety of benefits. The most apparent is the improved inference latency: By skipping the data upload to the server and wait- time for the inference result, the app can respond more quickly to the user's request.

TFLite is a fast inference engine that leverages the mobile GPU, a ubiquitous hardware accelerator on virtually every phone, we can achieve real-time performance for various deep network models. Figure-9 demonstrates that GPU has significantly more compute power than CPU.

Device	CPU (FP32)	GPU (FP16)
Samsung Galaxy S5	79	300
Samsung Galaxy S7	124	730
Samsung Galaxy S9	270	730

Figure 9: Example of available compute power on mobile in gigaflops (billion floating point instructions per second). FP16 and FP32 refer to 16-bit and 32-bit floating point arithmetic, respectively[56].

TFLite GPU leverages the mobile GPU with OpenGL ES for Android devices and Metal for iOS devices. The specific version requirements are OpenGL ES 3.1+ and iOS 9+ which are available for more than 52% of all Android devices [57]. One of the biggest strength of TFLite framework is that it employs open standards, *i.e.* is not limited by specific hardware vendor, and thus covers a wide range of devices.

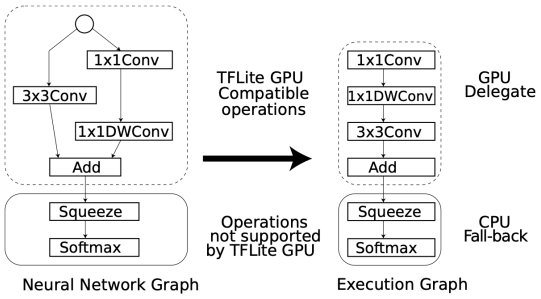


Figure 10: TFLite's delegate mechanism: Operations supported by the GPU delegate will run on the GPU, and the rest on the CPU[56].

The inference phase is fairly straightforward. The input tensors are reshaped to the PHWC4 format, if their tensor shape has channel size not equal to 4. For each operator, shader programs are linked by binding resources such the operator's input/output tensors, weights, etc. and dispatched, *i.e.* inserted into the command queue. The GPU driver then takes care of scheduling and executing all shader programs in the queue, and makes the result avail- able to the CPU by the CPU/GPU synchronization. There might be a final conversion from PHWC4 to HWC, if the output tensor has a channel size not equal to 4.

For maximum performance, one should avoid CPU/GPU synchronization at all cost, and preferably, never leave GPU context if real-time processing is needed. The most ideal scenario would be the following: A camera provides with RGBA texture that goes directly to TFLite GPU and the output of the network is then directly rendered to the screen.

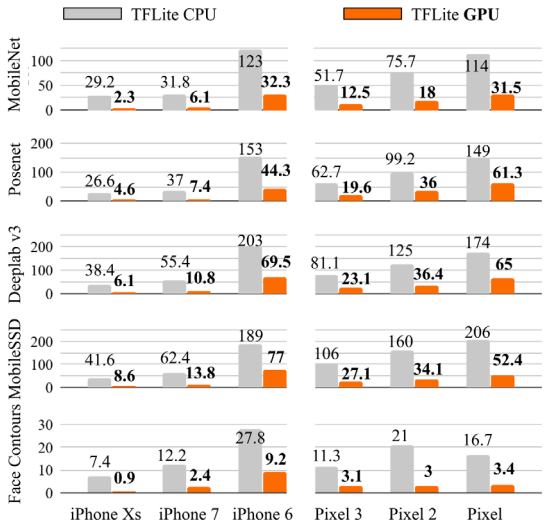


Figure 11: Average inference latency (in milliseconds) of TFLite GPU (orange) compared to CPU (gray) on various neural networks, run on a variety of smartphones (best viewed in color).

Figure-11 illustrates the performance of GPU inference compared to CPU inference in TFLite for various neural networks which generally demonstrates a 2–9× speedup. The first 10 warm-up runs were skipped for benchmarking and averages are based on the 100 subsequent inferences. This profiling revealed that TFLite GPU is often bound by memory bandwidth and we typically only see 20–40% ALU utilization. On iOS devices, we benefit from larger cache sizes that result in reduced memory I/O latency, and hence, better performance than the OpenGL backend.

VIII. Conclusion

As more and more applications find use with neural networks, lightweight algorithms are the need of the hour. The most recent example of this comes in the form of Apple's new products, which use neural networking to ensure a multitude of privacy and security features across products. Owing to the disruptive nature of the technology, it is easy to see its adoption by various companies.

References

1. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", 2012.
2. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.
3. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, et al. Going deeper with convolutions. In CVPR, pages 1–9, 2015.

4. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.
5. Emile Fiesler, Amar Choudry, and H John Caulfield, "Weight discretization paradigm for optical neural networks", 1990.
6. Wolfgang Balzer, Masanobu Takahashi, Jun Ohta, and Kazuo Kyuma, "Weight quantization in boltzmann machines", 1991.
7. Chuan Zhang Tang and Hon Keung Kwan, "Multilayer feedforward neural networks with single powers-of-two weights", 1993.
8. Michele Marchesi, Gianni Orlandi, Francesco Piazza, and Aurelio Uncini, "Fast neural networks without multipliers", 1993.
9. W.Dally.High-performance hardware for machine learning. In Tutorial in Advances in Neural Information Processing Systems, 2015.
10. S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv preprint arXiv:1606.06160, 2016.
11. Y.Bengio, N.Leonard, and A.Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation", 2013.
12. M. Courbariaux, Y. Bengio, and J.P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations", 2015.
13. M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In European Conference on Computer Vision, pages 525–542. Springer, 2016.
14. M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to + 1 or -1", 2016.
15. D. Soudry, I. Hubara, and R. Meir. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In Advances in Neural Information Processing Systems, pages 963–971, 2014.
16. O. Shayar, D. Levi, and E. Fetaya. Learning discrete weights using the local reparameterization trick. arXiv preprint arXiv:1710.07739, 2017.
17. Y. Gong, L. Liu, M. Yang, and L. Bourdev. Compress- ing deep convolutional networks using vector quantization. arXiv preprint arXiv:1412.6115, 2014.
18. Z. Li, B. Ni, W. Zhang, X. Yang, and W. Gao. Performance guaranteed network acceleration via high-order residual quantization. In 2017 IEEE International Conference on Computer Vision (ICCV), pages 2603–2611. IEEE, 2017.
19. F. Li and B. Liu. Ternary weight networks. CoRR, abs/1605.04711, 2016.
20. B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. arXiv preprint arXiv:1712.05877, 2017.
21. K. Hwang and W. Sung. Fixed-point feedforward deep neural network design using weights+ 1, 0, and- 1. In Signal Processing Systems (SiPS), 2014 IEEE Workshop on, pages 1–6. IEEE, 2014.
22. S. Migacz. 8-bits inference with tensorrt. In GPU Technology Conference, 2017.
23. R. Banner, Y. Nahshan, E. Hoffer, and D. Soudry. Aciq, "Analytical clipping for integer quantization of neural networks", 2018.
24. Yann LeCun, Le´on Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition", 1998.
25. Kaiming He and Jian Sun, "Convolutional neural networks at constrained time cost", 2015..
26. Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung, "Fixed point optimization of deep convolutional neural networks for object recognition", 2015.
27. Minje Kim and Paris Smaragdis, "Bitwise neural networks", arXiv:1601.06071, 2016.
28. Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen, "Incremental network quantization: Towards cnns with low-precision weights", 2017.
29. Denis A Gudovskiy and Luca Rigazio, "Shiftcnn: Generalized low-precision architecture for inference of convolutional neural networks", 2017.
30. Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy, "Fixed point quantization of deep convolutional networks", 2016.
31. Bert Moons, Koen Goetschalckx, Nick Van Berckelaer, and Marian Verhelst, "Minimum energy quantized neural networks", 2017.
32. Yiwen Guo, Anbang Yao, Hao Zhao, and Yurong Chen, "Network sketching: Exploiting binary structure in deep cnns", 2017.
33. Graham W. Taylor Sek Chai Mohamed Amer, Aswin Raghavan, "Bit-regularized optimization of neural nets", 2018.
34. Wu Shuang, Li Guoqi, Shi Luping, and Chen Feng, "Training and inference with integers in deep neural networks", 2018.
35. Vincent Vanhoucke, Andrew Senior, and Mark Z Mao. "Improving the speed of neural networks on cpus", 2011.
36. Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization", 2017.
37. Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", 2015.
38. Darryl D Lin and Sachin S Talathi. "Overcoming challenges in fixed point training of deep con- volutional networks", 2016.
39. Babak Hassibi and David G Stork, "Second order derivatives for network pruning: Optimal brain surgeon", In NIPS, 1993.
40. Yann LeCun, John S Denker, and Sara A Solla, "Optimal brain damage", In NIPS, 1990.
41. 1611.06440 Pruning Convolutional Neural Networks for Resource Efficient Inference.
42. Song Han, Huizi Mao, and William J Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding", arXiv preprint arXiv:1510.00149, 2015.
43. Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, Hans Peter Graf ,"Pruning filters for efficient convnets", 2016.
44. E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," 2014.
45. M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," 2014.
46. V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," 2014.
47. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," 2014.
48. Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," 2015.
49. S. Nakajima, M. Sugiyama, and S. D. Babacan, "Global solution of fully-observed variational bayesian matrix factorization is column-wise independent", 2011.
50. A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov, "Tensorizing neural networks," 2015.
51. T. Garipov, D. Podoprikhin, A. Novikov, and D. Vetrov, "Ultimate tensorization: compressing convolutional and fc layers alike," 2016.
52. F. L. Hitchcock, "Multiple invariants and generalized rank of a p-way matrix or tensor," Journal of Mathematics and Physics, vol. 7, no. 1-4, pp. 39–79, 1928.
53. L. R. Tucker, "Some mathematical notes on three-mode factor analysis," Psychometrika, vol. 31, pp. 279–311, 1966c.
54. I. Oseledets, "Tensor-train decomposition," SIAM J. Scientific Computing, vol. 33, pp. 2295–2317, 01 2011.
55. Verucchi et al., "A Systematic Assessment of Embedded Neural Networks for Object Detection", 2020.
56. Juhyun Lee et al., "On-Device Neural Net Inference with Mobile GPUs", 2019.
57. Carole-JeanWu et al., "Machine Learning at Facebook: Understanding Inference at the Edge. In IEEE International Symposium on High-Performance Computer Architecture", 2019.

Classification of different plant leaf diseases using multiple convolutional neural network and image pre-processing

Tharani D

Student: Department of Computer Science and Engineering

SA Engineering College
Chennai 600077, India

tharanidurairaj@gmail.com

Preetha M

Associate Professor: Department of Computer Science and Engineering
SA Engineering College
Chennai 600077, India

preetha@saec.ac.in

***Abstract* - The recognizable proof of plant sickness is the reason of the counteraction of plant infection proficiently and definitely in the unpredictable climate. This causes huge degree demolition of harvests, reduces advancement and eventually prompts money related loss of farmers. In view of quick improvement in grouping of ailments and adequate data on farmer, recognizing verification and treatment of the ailment has become a huge test. The leaves have surface and visual resemblances which credits for conspicuous verification of sickness type. Therefore, PC vision used with significant learning gives the best way to deal with tackle this issue. This paper proposes a significant learning-based model which is readied using Plantvillage dataset containing pictures of strong and undesirable yield leaves. The model serves its objective by organizing pictures of leaves into unfortunate class reliant on the case of flaw. Utilizing a Plantvillage dataset of 3900 pictures of ailing and solid plant leaves gathered under controlled conditions, we train a profound convolutional neural organization to distinguish 11 yield species and 26 illnesses (or nonappearance thereof). Batch Normalization is performed to forestall network over-fitting while at the same time upgrading the heartiness of the model. Prelu activation function and Adam optimizer are utilized to improve both assembly and exactness. The prepared model accomplishes a precision of 98.74% on a held-out test set, showing the achievability of this methodology.**

Index Terms - Plantvillage dataset, CNN, Batch Normalization, Activation function, Adam optimizer.

I. INTRODUCTION

The issue of productive plant sickness assurance is firmly identified with the issues of manageable agribusiness and environmental change. Examination results show that environmental change can adjust stages and paces of microorganism advancement; it can likewise alter have opposition, which prompts physiological changes of host-microbe cooperation's. The circumstance is additionally confounded by the way that, today, sicknesses are moved internationally more effectively than any other time. New illnesses can happen in spots where they were beforehand unidentified and, inalienably, where there is no neighbourhood mastery to battle them.

Unpractised pesticide use can cause the improvement of long-haul opposition of the microorganisms, seriously diminishing the capacity to retaliate. Opportune and exact finding of plant sicknesses is one of the mainstays of exactness horticulture. It is vital to forestall pointless misuse of monetary and different assets, along these lines accomplishing more beneficial creation, by tending to the drawn-out microbe obstruction improvement issue and moderating the negative impacts of environmental change.

In this evolving climate, proper and ideal illness distinguishing proof including early avoidance has never been more significant. There are a few different ways to recognize plant pathologies. A few illnesses don't have any obvious manifestations, or the impact gets observable past the point where it is possible to act, and in those circumstances, a refined investigation is compulsory. Nonetheless, most infections create some sort of indication in the noticeable range, so the unaided eye assessment of a prepared proficient is the prime method embraced practically speaking for plant sickness discovery. To accomplish exact plant infection diagnostics a plant pathologist ought to have great perception abilities with the goal that one can recognize trademark indications. Varieties in manifestations demonstrated by ailing plants may prompt an inappropriate determination since beginner cultivators and specialists could have a bigger number of challenges deciding it than an expert plant pathologist. A robotized framework intended to help recognize plant sicknesses by the plant's appearance and visual side effects could be of extraordinary assistance to novices in the cultivating cycle and furthermore prepared experts as a check framework in infection diagnostics.

II. LITRATURE SURVEY

[1]YANG ZHANG, CHENGLONG SONG, AND DONGWEN ZHANG proposed Faster RCNN calculation to distinguish ailing tomato leaves, which can both perceive tomato infections and distinguish tomato leaf areas. To make the anchors in the calculation closer to the ground reality of dataset, utilized the k-means calculation to group the bouncing boxes of tomato infection pictures and improve the anchors dependent on the outcomes. ResNet101 is used for feature extraction, which can separate the profound highlights of tomato illness. The test results show that strategy can

viably distinguish and perceive tomato infections and has higher location precision of 98.54%.

[2] JIANG HUIXIAN proposed plants image recognition 50 plant leaf information bases are endeavored and separated and KNN-based zone gathering, Kohonen network dependent on self- sorting out part organizing assessment and SVM- based help vector machine. At the same time, the leaves of 7 different plants were looked at and it was discovered that ginkgo leaves were simpler to perceive. For leaf pictures under complex foundation, phenomenal confirmation influence has been refined. Picture preliminary of the test set are input into the learning model to get entertainment bungles. The class name of the test set can be obtained by re-trying the critical learning model with the most modest fumble set. The outcomes show that this strategy has the most brief acknowledgment time and the most elevated right acknowledgment rate.

[3]MUHAMMAD ATTIQUE KHAN, M IKRAMULLAH LALI, MUHAMMAD SHARIF,KASHIF JAVED, KHURSHEED AURANGZEB, SYED IRTAZA HAIDER, ABDULAZIZ SAUD ALTAMRAH, TALHAAKRAM proposed an improved computerized PC based strategy and approved for acknowledgement of apple infections. The injury spot contrast extending, sore division, and conspicuous highlights determination and acknowledgment steps are used. The difference of contaminated spot is improved and division is performed by the proposed SCP technique. The presentation of the proposed SCP technique is further enhanced by EMI approach. At that point, various highlights are extricated and melded by utilizing an equal strategy. A Genetic calculation is applied to choose the best highlights, which are later used by M-SVM for arrangement. The M-SVM accomplished an arrangement exactness of 92.9%, 94.30%, and 97.20% for without highlight determination (Test 1), PCA based component decrease (Test 2), and proposed highlights determination technique (Test 3), individually. The outcomes show that the determination strategy gives better execution as far as exactness and execution time.

[4]Guoxiong Zhou, Wenzhuo Zhang, Aibin Chen, MingfangHe proposed a strategy for identifying quick rice illness dependent on FCM-KM and Faster R-CNN combination is proposed to address different issues with the rice illness pictures, for example, commotion, obscured picture edge, huge foundation impedance and low identification exactness. Initially, the technique utilizes a two-dimensional sifting veil joined with a weighted staggered middle channel (2DFM-AMMF) for noise reduction, and utilizes a quicker two- dimensional Otsu limit division calculation (Faster 2D-Otsu) to lessen the impedance of complex foundation with the discovery of target edge in the picture. At that point the dynamic populace firefly calculation dependent on the disarray hypothesis just as the most extreme and least distance calculation is applied for advancement of the K- Means clustering calculation (FCM-KM) to decide the ideal bunching class k value while tending to the inclination of the calculation to fall into the neighborhood ideal issue. Joined with the R-CNN calculation for the distinguishing proof of rice illnesses, FCM-KM investigation is led to decide the various sizes of the Faster R-CNN target outline. As uncovered by the application consequences of 3010 pictures, the precision and time needed for location of rice impact, bacterial scourge and curse were 96.71%/0.65s, 97.53%/0.82s and 98.26%/0.53s, respectively.

[5]SIVASUBRAMANIAMJANARTHAN,SELVARAJAHTHUSEETHAN,SUTHARSHANRAJASEGARAR,QIANGLYU, YONGQIANG ZHENG AND JOHN YEARWOOD proposed a lightweight, quick, and exact profound measurement learning-based design for citrus illness recognition from scanty information. propose a fix based grouping network that contains an implanting module, a bunch model module, and a basic neural organization classifier, to identify the citrus infections precisely. Assessment of proposed approach utilizing freely accessible citrus leafy foods dataset uncovers its productivity in precisely identifying the different infections from leaf pictures. Further, the speculation capacity of methodology is shown utilizing another dataset, to be specific the tea leaves dataset. Correlation examination of methodology with existing best in class calculations exhibit its prevalence as far as discovery exactness (95.04%), the quantity of boundaries needed for tuning (under 2.3 M), and the time effectiveness in recognizing the citrus diseases(less than 10 ms) utilizing the trained model.

III.PROPOSED SYSTEM

This paper proposes a significant learning-based model which is readied using Plantvillage dataset containing pictures of strong and undesirable yield leaves. The model serves its objective by organizing pictures of leaves into unfortunate class reliant on the case of flaw. Utilizing a Plantvillage dataset of 3900 pictures of ailing and solid plant leaves gathered under controlled conditions, we train a profound convolutional neural organization to distinguish 11 yield species such as maize, cherry, tomato, potato, bell pepper, apple, strawberry, grape, raspberry, peach, orange and 26 illnesses (or nonappearance thereof). Different convolutional neural organization bunch with deliberation and weight delivering abilities goes about as a proficient element extraction model. The proficient extractions of highlights from the CNNs render the best contribution for the profound learning model to foresee the leaf sickness from leaf dataset pictures. Thus, the profound learning model gives a non-intrusive, modest and tech neighbourly for farmers and in the field of exactness agriculture. Forecasts utilizing different boundaries, for example, fine buildup, leaf spot early curse, late scourge and so on.Batch Normalization is performed to forestall network over-fitting while at the

same time upgrading the heartiness of the model. Prelu activation function and Adam optimizer are utilized to improve both assembly and exactness.

IV.MATERIALS AND METHODOLOGY

The sample image arrangements and its meta-information conveyance in the Plantvillage dataset have a critical part in the proficient planning and working of the proposed model. The morphological highlights, shading, shape, and surface based highlights will straightforwardly affect the exhibition and exactness of the profound neural organization to be created. Absolute of 3900 sample images are utilized.

Table1.Description of Plantvillage dataset leaf parameters

Type of leaf	**Disease types**
Apple	Marssonia leaf blotch, Blackrot canker, Collar rot, Powdery mildew, Sooty blotch, Fly speck
Peach	Taphrina deformans
Orange	Sooty mold, marginal leaf burn, citrus greening, cigar leaf curling
Cherry	Leaf spot, powdery mildew
Tomato	Bacterial spot, early blight, grey leaf spot, late blight, leaf mold
Potato	bacterial wilt, Septoria leaf spot, late blight, early blight, common scab, black canker
Bell pepper	Bacterial leaf spot, southern blight, powdery mildew
Grape	Downy mildew, powdery mildew, anthracnose, leaf blight
Strawberry	Scorch, spot, blight
Maize	Bacterial wilt, anthracnose, northern corn leaf blight, southern corn leaf blight, common rust
Raspberry	Cane blight, grey mold, spur blight

A. Building Convolutional Neural Network

The fundamental thought of multi-scale convolution depends on deciding how an ideal neighborhood scanty structure in a convolutional vision organization can be approximated and covered by a promptly accessible thick part. To forestall the fix arrangement issues, current manifestations of the design are limited to channel sizes 1x1,3x3 and 5x5. It implies that the recommended design is a blend of every one of those layers with their yield channel bank linked into a solitary yield vector giving the contribution to the following layer**.**

B. 2D Convolution layer

Numerous layers of convolutional layers are utilized in a consecutive mix to remove the low and significant level highlights in the information [16]. The convolution network layer's cycle is characterized in the condition no.1. In this, the

$$Xlj = f\left(\sum i\epsilon MjXl - 1i * klij + blj\right)$$

Going before layers of the organization are appeared with Xl−1i, the bits for learning are klij and the predisposition term is blj. Mj is for the guide area coordinating cycle.

C. Activation layer

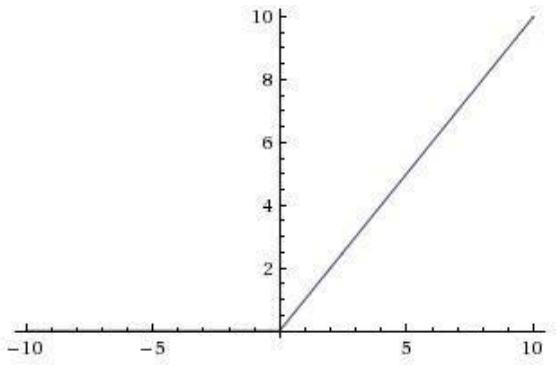
Relu work shows a quicker union speed than Sigmoid and tanh work, and the inclination won't be immersed. It is conceivable to cause the marvel of "angle vanishing" in the preparation of neural organization. An issue is that, the inclination after a neuron is consistently zero and stops to react to any information. To determine the issues as referenced above, PRelu actuation work as opposed to Sigmoid and Relu enactment work is used. Its numerical articulation is appeared as follows

PRelu (x) = {x ifx>0

{ax if x≤0

where x shows the yield estimation of a neuron, and a means the hyper boundary, which is set to 0.25 in our investigations. The contrast between PRelu capacity and Relu work is that, when the info signal is under 0, the estimation of PRelu work is a capacity with a more modest incline, which changes the circulation of information and holds a few estimations of the negative hub x.

Fig1.Graphically RELU looks like



D. Batch Normalization

The highlights of various leaf illness are unpredictable and variable. The neural organization learning speed is low or even hard to learn. In the interim, as the neural organization structure keeps on developing, the conveyance of concealed layer information has gone through critical changes and even vacillations which will negatively affect the strength of the organization. In our proposed framework pooling layer is utilized. This is proposed to guarantee information solidness and makes it simpler and more steady to prepare profound organization models while improving the capacity of organization speculation.

E. Flatten layer

Flatten is the capacity that changes over the pooled include guide to a solitary section that is passed to the completely associated layer.

F. Dense Layer

Dense layer is the ordinary profoundly associated neural organization layer. It is generally normal and often utilized layer. Dense layer does the underneath procedure on the info and return the yield.

yield = activation (dot (input, kernel) + bias)

where, input represent the data information

kernel represent the weight information

dot represent numpy speck result of all data and its comparing loads

bias represent to a one-sided esteem utilized in AI to upgrade the model activation represent to the actuation work

Table No.2 Model Summary

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_1(Conv2D)	(None, 256, 256, 32)	896
activation_1(Activation)	(None, 256, 256, 32)	0
batch_normalization_1	(None, 256, 256, 32)	128
max_pooling2d_1(MaxPooling2	(None, 85, 85, 32)	0
dropout_1 (Dropout)	(None, 85, 85, 32)	0
conv2d_2 (Conv2D)	(None, 85, 85, 64)	18496
activation_2 (Activation)	(None, 85, 85, 64)	0
batch_normalization_2	(None, 85, 85, 64)	256
conv2d_3 (Conv2D)	(None, 85, 85, 64)	36928
activation_3 (Activation)	(None, 85, 85, 64)	0
batch_normalization_3 (Batch	(None, 85, 85, 64)	256
max_pooling2d_2 (MaxPooling2	(None, 42, 42, 64)	0
dropout_2 (Dropout)	(None, 42, 42, 64)	0
conv2d_4 (Conv2D)	(None, 42, 42, 128)	73856
activation_4 (Activation)	(None, 42, 42, 128)	0
batch_normalization_4 (Batch	(None, 42, 42, 128)	512
conv2d_5 (Conv2D)	(None, 42, 42, 128)	147584
activation_5 (Activation)	(None, 42, 42, 128)	0
batch_normalization_5 (Batch	(None, 42, 42, 128)	512
max_pooling2d_3 (MaxPooling2	(None, 21, 21, 128)	0
dropout_3 (Dropout)	(None, 21, 21, 128)	0
flatten_1 (Flatten)	(None, 56448)	0
dense_1 (Dense)	(None, 1024)	57803776
activation_6 (Activation)	(None, 1024)	0
batch_normalization_6 (Batch	(None, 1024)	4096
dropout_4 (Dropout)	(None, 1024)	0

dense_2 (Dense)	(None, 39)	39975
activation_7 (Activation)	(None, 39)	0

Total params: 58,127,271
Trainable params: 58,124,391
Non-trainable params: 2,880

V. Training and testing of the proposed Robust CNN model

At the point when the neural association model is arranged and made for the gauge of leaf affliction it should be readied. Presently key characteristics for hyper limits like learning rate, group size, outright number of cycles, number of test pictures to be set up in each accentuation should be settled. After that an estimation for development must be picked for updation and back inciting of model burdens. In the assessment, experimentation procedure is used to choose the assessments of the learning rate, outright number of cycles, and hyper limits. Progression is done using the ADAM analyzer. From the outset the planning age consider is fixed 25. The most noteworthy accentuation per age is restricted to 1584. The learning rate for each trade is set to 0.1. The absolute dataset was isolated into two segments as the arrangement and test sets with 80% and 20% rates, exclusively. The proposed CNN model is arranged and attempted. By then the significant features eliminated from the proposed CNN model are considered. A mix of significant component extraction and AI procedures are utilized to achieve an anticipated and Simultaneously the plantvillage dataset (3900 data) is in like manner dealt with 80:20 train and test.

The dataset is preprocessed, for example, Image reshaping, resizing and transformation to a cluster structure. Comparable handling is likewise done on the test picture. A dataset comprising of around 11 distinctive plant leaf infections is gotten, out of which any picture can be utilized as a test picture for the software. The train dataset is utilized to prepare the model (CNN) so that it can distinguish the test picture and the sickness it has CNN has various layers that are Dense, Dropout, Initiation, Flatten, Convolution2D, and MaxPooling2D. After the model is prepared effectively, the product can recognize the illness if the plant species is contained in the dataset. After fruitful preparing and preprocessing, correlation of the test picture and prepared model takes spot to anticipate the illness.

Fig No.2 Architecture Flow Diagram

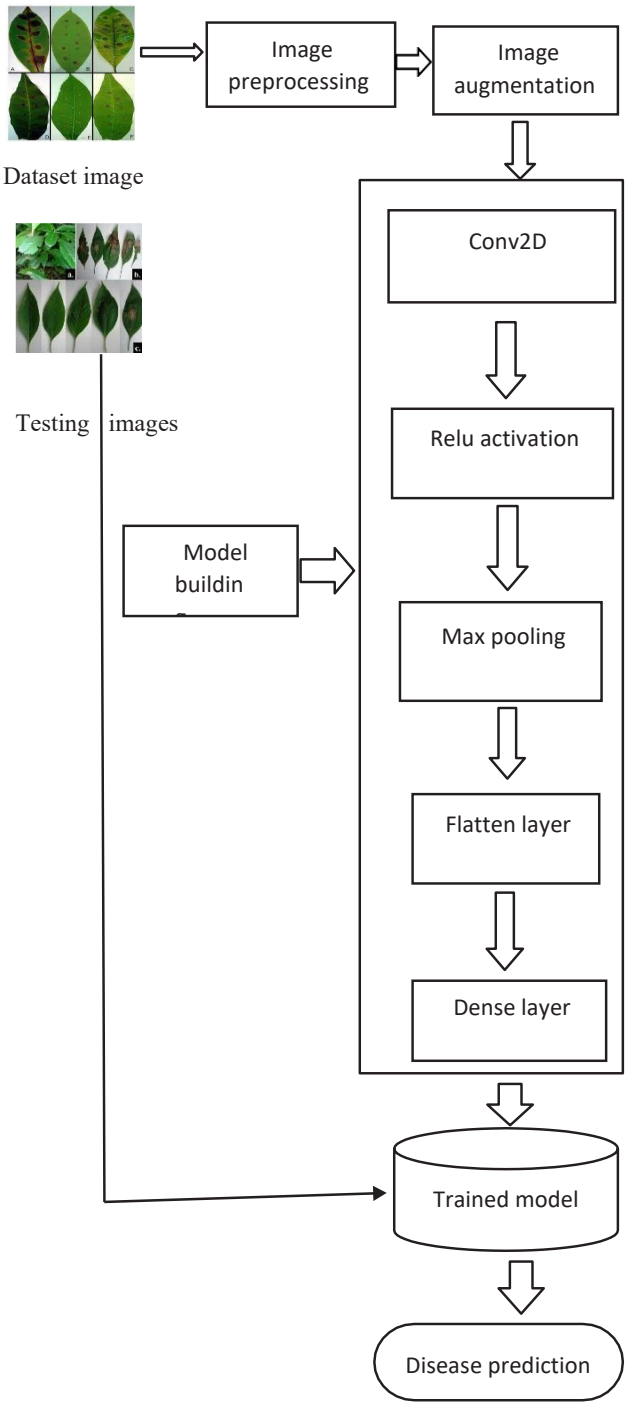


Fig 3. Training loss curve

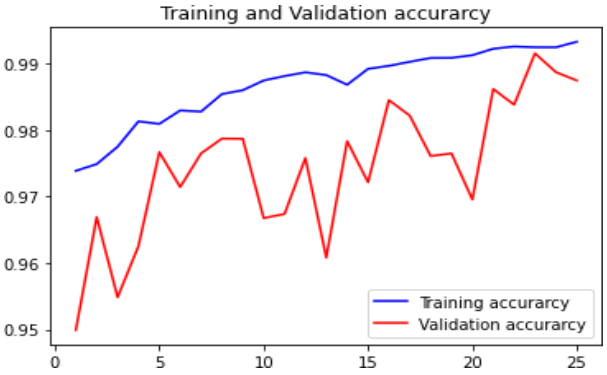
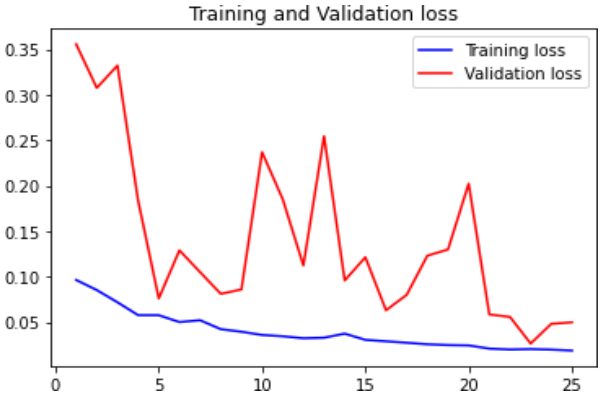


Fig 4. Training accuracy curve



VI. RESULTS AND DISCUSSIONS

In this investigation, we have proposed and summed up the Robust Deep Neural Network- based methodology and the contemporary basic discoveries. From the investigations led and results got there's a reasonable sign that utilizing profound Convolutional neural organizations will have brilliant impacts in contriving programming for highlight extraction, finding, identification, and foreseeing subtleties concerning forecast of plant leaf infection with the precision of 98.74%It is engaged in how picture from given dataset (arranged dataset) in field and past educational assortment used predict the case of plant infections using CNN model. This brings a part of the accompanying encounters about plant leaf affliction figure. As most noteworthy sorts of plant leaves will be covered under this system, farmer may turn out to be more familiar with about the leaf which may never have been created and runs through all possible plant leaves, it enables the farmer in unique of which to respect create. In like manner, this structure ponders the past production of data which will empower the farmer to get understanding into the premium and the cost of various plants in market. Cultivating division needs to automate the perceiving the yield crops from capability measure (certifiable time).To robotize this cycle by demonstrating the desire bring about web application or work territory application. To improve the work to execute in Artificial Intelligence atmosphere.

REFERENCES

[1]YANGZHANG,CHENGLONG SONG, AND DONGWEN ZHANG,“ Deep Learning-Based Object Detection Improvement for Tomato Disease,” Digital Object Identifier, vol 8,pp. 56607-56614, March 31, 2020.

[2]JIANG HUIXIAN,”The Analysis of Plants Image Recognition Based on Deep Learning and Artificial Neural Network,” Digital Object Identifier, vol.8, 2020, pp. 68828-68841, April 23, 2020.

[3]MUHAMMAD ATTIQUE KHAN, M IKRAMULLAH LALI, MUHAMMAD SHARIF, KASHIF JAVED, KHURSHEED AURANGZEB, SYED IRTAZA HAIDER, ABDULAZIZ SAUD ALTAMRAH, and TALHA AKRAM,”An Optimized Method for Segmentation and Classification of Apple Diseases based on Strong Correlation and Genetic Algorithm based Feature Selection,”Digital Object Identifier,vol.7,2019,pp.46261 – 46277, 28 March 2019.

[4]Guoxiong Zhou, Wenzhuo Zhang, Aibin Chen, MingfangHe,”Rapid Detection of Rice Disease Based on FCM-KM and Faster R-CNN Fusion,”Digital Object Idendifier,vol.7,2019,pp.143190 – 143206, 24 September 2019.

[5]SIVASUBRAMANIAMJANARTHAN,SELVARAJAHTHUSEETHAN,SUTHARSHANRAJASEGARAR,QIANGLYU,YONGQIANG ZHENG AND JOHN YEARWOOD,” Deep Metric Learning Based Citrus Disease Classification With Sparse Data,”Digital Object Idendifier,vol.8,2020,pp. 162588- 162600, September 17, 2020.

[6]SHUANGJIEHUANG, GUOXIONG ZHOU, MINGFANG HE, AIBIN CHEN, WENZHUO ZHANG, AND YAHUI HU,“Detection of Peach Disease Image Based on Asymptotic Non-Local Means and PCNN-IPELM” Digital Object Identifier, vol. 8, 2020, pp.136421-136433, August 5, 2020.

[7]CHAOWANG, PINGPING WANG,SHUGUANG HAN, LIDA WANG,YUMING ZHAO,AND LIRAN JUAN,“FunEffector-Pred: Identification of Fungi Effector by Activate Learning and Genetic Algorithm Sampling of Imbalanced Data” Digital Object Identifier, vol.8, 2020, pp. 57674-57683, April 1, 2020.

[8]XUAN NIE, LUYAO WANG, HAOXUAN DING, AND MIN XU,”Strawberry Verticillium Wilt Detection Network Based on Multi-Task Learning and Attention” Digital Object Identifier, vol.7, 2019, pp. 170003-170011, December 9,
2019.

[9] XUAN WANG, WEIKANG DU, FANGXIA GUO, AND SIMIN HU,”Leaf Recognition Based on Elliptical Half Gabor and Maximum Gap Local Line Direction Pattern” Digital Object Identifier, vol.8, 2020, pp. 39175-39183, March 4, 2020.

[10]DAWEI LI, YAN CAO, GUOLIANG SHI, XIN CAI, YANG CHEN, SIFAN WANG, AND SIYUAN YAN,”An Overlapping-Free Leaf Segmentation Method for Plant Point Clouds” Digital Object Identifier, vol.7, 2019, pp. 129054- 129070, September 23, 2019.

[11]Draško Radovanović, Slobodan Đukanović,” Image-Based Plant Disease Detection: A Comparison of Deep Learning and Classical Machine Learning Algorithms” 24th International Conference on Information Technology (IT) Zabljak, pp.1-4,18 – 22 February 2020.

[12]BAODONG MA, RUILIANG PU, SONG ZHANG, AND LIXIN WU,”Spectral Identification of Stress Types for Maize Seedlings Under Single and Combined Stresses,” Digital Object Identifier, vol.6, 2018, pp. 13773-13782, March 28, 2018.

[13]Ding Jiang, Fudong Li, Yuequan Yang, Song Yu,”A Tomato Leaf Diseases Classification Method Based on Deep Learning,” Digital Object Identifier,vol.8,2020, pp. 1446-1450,august 24,2020.

Classification of Weed Species Using Deep Learning

Pokala PranayKumar
Woxsen University
Sangareddy
pranaykumar.pokala_2022@woxsen.edu.in
Prof. Raul Villamarin Rodriguez
Dean- School of Business
Woxsen University
Sangareddy
raul.rodriguez@woxsen.edu.in

ABSTRACT:
Automatic identification and classification of weed species are essential in the agricultural field for controlling weed species. Weeds are an undesirable and unfortunate plant that meddles with the usage of land and water assets and along these lines unfavourably influence crop creation and human government assistance. So, identification and classification of weed are important for farmers to protect the crop field and to maintain the productivity and quality of the crop. But it takes a long time and huge human-effort to manually identify and classify weed species. Technology advancement has made complex problems to solve more efficiently and reduces manpower and lower the costs. With technological advancement, many methods have been introduced. Using deep learning methods such as neural networks on agricultural data has increased enormous consideration lately. The evolution of deep learning made it easy for identifying and classifying the weed type. This paper uses publicly available large multiclass image datasets of weed species obtained from Australian rangelands for classification. In this paper, we are using the Transfer Learning technique with a pretrained network called resnet18 to classify the type of weed from the images present in the dataset and also calculating performance metrics like accuracy, sensitivity, recall, precision, etc. This helps in controlling weed species in the crop field.
Keywords: transfer learning, deep learning, weeds, convolutional neural network (CNN), resnet-18, pretrained networks.

1. INTRODUCTION:

Man has designated his food crops from the numerous thousand plant species that exist for his or her nutritionary and flavour characteristic instead of through their ability to contend.

Weeds grow on the soil in conjunction with crop plants. Weeds are unwanted and undesirable plants that interfere with the use of land and water resources and so adversely they will even be referred to as plants out of place. Weeds contend with the helpful and desired vegetation in croplands, forests, aquatic systems, etc. furthermore, presents pleasant downside in non-trimmed regions like mechanical destinations, street/rail lines, runways, scene plantings, water tanks and streams in which, and so on., anyway this development of weed.

These unsought plants eat up the nutrients, water, and house assigned for the meant crop, and eventually cause a large reduction in crop yield. Some weeds unleash harmful substance that affects crop growth. In conjunction with this, weeds have an effect on and interfere with the management of all the terrestrial and aquatic resources. The growing of crops, as a part of agriculture for hundreds of years, has modified the natural vegetation. Weeds, in the crop field, scale back input potency, interfere with agricultural operations, impair the quality and act as alternate hosts for many insect pests and diseases. The plain impact of those traits is that the hike in the price of cultivation by many folds. The animals that rely upon this native multifariousness for his or her survival also are obtaining affected. Weeds have a task among agroecosystems in supporting multifariousness additional typically.

1.1TYPES OF WEEDS:

There are many types, but in our dataset, we are considering 9 types of weeds.

1. **CHINESE APPLE:**

   The scientific name of the Chinese apple is Ziziphus mauritiana. It is spread around the northern part of Australia and the central part of Queensland. It can grow up to 8meters tall and 10meters in covering diameter and the arms are heavy, crickled. These species are also restricted according to the biosecurity act 2014 [3].

2. **LANTANA:**
   Lantana spreads in two different ways. Layering is a type of vegetative multiplication where stems send roots into the dirt, permitting it to rapidly shape

exceptionally thick stands and spread short separations. Additionally, flying creatures and different creatures, for example, foxes expand also, pass the seed in their droppings, possibly spreading it over very enormous separations. The germination pace of new seed is commonly low, yet improves after being processed. Lantana can develop in high-precipitation zones with tropical, subtropical, and mild atmospheres. [4].

3. **PARKINSONIA:**
   Parkinsonia duplicates by seeds. Develop trees normally produce around 5000 seeds every year, except can create in the overabundance of 13,000. The unit's drift and can be conveyed huge separations downstream from upper catchment pervasions, particularly during floods. Seed can likewise be moved away from the parent plant in mud appended to creatures or hardware. In Australia, pervasions happen chiefly all through the waterfront, focal and western Queensland, focal and northern Territory, and the Kimberley what's more, Pilbara locales of Western Australia. [5].
4. **PARTHENIUM:**
   Parthenium is a yearly herb with a profound taproot and an erect stem that gets woody with age. It attacks upset exposed territories and fields. Parthenium costs Australia's hamburger industry $16.5 million every year and trimming enterprises a few million dollars for each year. Parthenium weed is a confined intrusive plant under the Biosecurity Act 2014. Its scientific name is Parthenium hysterophorus [6].
5. **PRICKLY ACACIA:**
   Prickly acacia is a little, prickly, spreading tree commonly developing to around 4–5m high and every so often to 10 m. It is as a rule single-stemmed. The bark of youthful trees has a tinge of orange as well as green. More seasoned trees have dull, unpleasant bark and tend to lose the greater part of their thistles. The green, plant-like leaves are 30–40 mm long. Each leaf is comprised of 10–25 sets of very little (3–6 mm) flyers along its length. [7].
6. **RUBBER VINE:**
   Rubber vine is a Weed of National Noteworthiness. It is viewed as one of the most exceedingly awful weeds in Australia as a result of its intrusiveness, the potential for spread, and financial and ecological effects. Rubber vine has impacts on peaceful also, protection regions of north-eastern Australia. Its primary effect on pastoralism is the loss of brushing nation, which in 1995 was evaluated to cost the Queensland meat industry $18 million [8].
7. **SIAM WEED:**
   Siam weed is one of the world's most noticeably terrible weeds, with a sensational development rate and enormous seed creation. Plants can arrive at 10 meters by scrambling through contiguous vegetation. It structures impervious bushes to three meters tall in open locales, for example, waterway banks and fields. It can cover tropical organic product crops, youthful ranger service manors, and fields [9].
8. **SNAKE WEED:**
   'Snakeweed' is the term used to depict various bushes from the Stachytarpheta family.
   Local to the tropical Americas, snakeweeds are clustering lasting bushes with intense, fanned stems and woody roots. Snakeweed species can attack the side of the road, upset regions, and wet fields. In the Pacific district, 8 snakeweed species have become weeds in tropical territories. Snakeweed species are found along the Queensland coast [10].
9. **NEGATIVES:**
   In this, the images are not related to describe weeds. These are might be other than described weeds or this can be a non-weed [1].

Using artificial intelligence day-by-day the technology growth has been increased and easily we are analyzing the images using different machine learning and deep learning techniques. In our experiment, we are taking Australia deep weed dataset for image classification using transfer learning. In Robotics many agricultural scientists are trying to make useful devices and techniques for farming and also for the farmers. The latest technology is used for farming will reduce time and also easily identify the diseased plant or crop. This makes us take appropriate action on these diseases before it spreads to remaining crops. It also reduces the manpower and also predicts with best accuracies. In deep learning, the CNN model is used for classification which classifies the image better and with fast execution.

2. MATERIALS AND METHODS

2.1 MATERIALS

2.1.1 DATASET DESCRIPTION:

This dataset encloses with 9 classes which are (**a**) Chinee apple, (**b**) Lantana, (**c**) Parkinsonia, (**d**) Parthenium, (**e**) Prickly acacia, (**f**) Rubber vine, (**g**) Siam weed, (**h**) Snakeweed and (**i**) Negatives. Total images are 17,509. These images are gathered from multiple regions around Australia. The chose weed species are nearby to peaceful meadows over the territory of Queensland. They consist of: Chinee apple, Snakeweed, Lantana, Prickly acacia, Siam weed, Parthenium, Rubber vine, and Parkinsonia. The pictures were gathered from weed pervasions at the accompanying locales across Queensland: Black River, Charters Towers, Cluden, Douglas, Hervey Range, Kelso, McKinlay, and Paluma. The table beneath separate the dataset by weed, area [2].

2.2 METHODS

2.2.1 CONVOLUTIONAL NEURAL NETWORK:

Convolutional neural networks (CNNs) rose out of the investigation of the mind's visual cortex, and they have been utilized in picture acknowledgment since the 1980s. In the last not many a long time, on account of the expansion in computational force, the measure of accessible preparing information. CNN's have man matured to accomplish superhuman execution on some complex visual errands. The power picture search administrations, self-driving autos, programmed video order frameworks, and more. Besides, CNNs are not confined to visual observation: they are likewise effective at different undertakings, for example, voice acknowledgment or common language preparing (NLP); These investigations of the visual cortex motivated the neocognitron, presented in 1980, which continuously advanced into what we currently call convolutional neural systems. This design makes them fabricate squares that you know, for example, completely associated layers and sigmoid enactment capacities, however, it additionally presents two new structure squares: convolutional layers and pooling layers.

2.2.1.1 Convolution layer:

The most significant structure square of a CNN is the convolutional layer. Neurons in the first convolutional layer are not associated with every pixel in the info picture, however just to pixels in their responsive fields. Thusly, every neuron in the second convolutional layer is associated uniquely with neurons situated inside a little square shape in the main layer. This design permits the system to focus on low-level highlights in the principal concealed layer, at that point gather them into more elevated level highlights in the following concealed layer, etc. This various levelled structure is basic in certifiable pictures, which is one reason why CNNs work so well for picture acknowledgment.

2.2.1.2 Filters:

A neuron's loads can be spoken to as a little picture of the size of the responsive field. There are two potential arrangements of loads called channels. Right now, two sorts of channels. 1. Vertical filter: for instance, 7 × 7 frameworks loaded with 0s aside from the focal section, which is brimming with 1s. neurons utilizing these loads will disregard everything in their open field aside from the focal vertical line (since all information sources will get increased by 0, except for the ones situated in the focal vertical line).2.horizontal filter: once more, neurons utilizing these loads will disregard everything in their open field except for the focal even line. A layer loaded with neurons utilizing a similar channel gives you a feature map, which features the regions in a picture that are generally like the filter. During preparing, a CNN finds the most valuable channels for its assignment, and it figures out how to consolidate them into increasingly complex examples (e.g., a cross is a territory in a picture where both the vertical channel and the even channel are dynamic).

2.2.1.3 Multiple feature maps:

Till now we showed for convolutional 2D layer, but in the real world, there are several feature maps of the same sizes shown as a 3D layer. Input pictures are additionally made out of various sublayers: one for each shading channel. There are commonly three: red, green, and blue (RGB). Grayscale pictures have only one station, yet a few pictures may have considerably more—for instance, satellite pictures that catch additional light frequencies, (for example, infrared). A neuron situated in push I, section j of the element map k in a given convolutional layer l is associated with the yields of the neurons in the past layer l – 1, situated in lines $I \times s_w$ to $I \times s_w + f_w - 1$ and segments $j \times s_h$ to $j \times s_h + f_h - 1$, overall element maps (in layer l – 1). Note that all neurons situated in a similar line I and segment j yet in various component maps are associated with the yields of precisely the same neurons in the past layer.

2.2.1.4 Pooling Layer:

They will probably subsample (i.e., recoil) the information picture to diminish the computational burden, memory utilization, and the number of parameters (accordingly constraining the danger of overfitting). Reducing the picture size likewise causes the neural system to endure a smidgen of picture move (area invariance). Much the same as in convolutional layers, every neuron in a pooling layer is associated with the yields of a set number of neurons in the past layer, situated inside a little rectangular responsive field. You should characterize its size, the walk, and the cushioning type. a pooling neuron has no loads; everything it does is total the information sources utilizing a total capacity, for example, the maximum or mean. max-pooling layer, which is the most well-known sort of pooling layer. Right now, we utilize a 2 × 2 pooling portion, a stride of 2, and no padding. Note that lone the maximum info esteem in every piece makes it to the following layer. Different information sources are dropped. A pooling layer normally chips away at each info channel freely, so the yield profundity is equivalent to the information profundity.

After all these the final layers are fully connected layer with the output layer (eg: softmax layer) for prediction.

2.2.2 TRANSFER LEARNING:

Transfer learning is a Machine Learning technique whereby a model is trained and developed for one assignment and is then re-used on a second related task. It refers back to the situation wherein what has been learned in one setting is exploited to enhance optimization in another setting [11]. The objective of TL is to help enhance the prediction feature in gaining knowledge of the target undertaking the usage of the information from the source area with the supply target [12].
There are two casual approaches are as follows:

Develop a Model Approach
Pre-trained Model Approach

1. Develop a Model Approach

1. Select Source Task. You should select an associated predictive modeling hassle with an abundance of information where there may be some relationship inside the input information, output statistics, and/or ideas learned for the duration of the mapping from entering to output information.
2. Develop Source Model. Next, you ought to broaden a skillful model for this first venture. The version has to be higher than a naive version to make certain that some feature gaining knowledge has been performed.
3. Reuse Model. The version suit at the source assignment can then be used as the place to begin for a model at the second assignment of interest. This may additionally involve the usage of all or elements of the model, relying on the modeling technique used.
4. Tune Model. Optionally, the version might also need to be adapted or delicate on the enter-output pair records available for the undertaking of interest.

2. Pre-trained Model Approach

1. Select the Source Model. A pre-skilled source model is chosen from available models. Many studies establishments launch models on massive and challenging datasets that may be included in the pool of candidate models from which to pick out from.
2. Reuse Model. The version pre-skilled version can then be used as the start line for a version on the second undertaking of interest. This may additionally involve the usage of all or components of the version, depending on the modeling method used.
3. Tune Model. Optionally, the version can also want to be tailored or subtle on the enter-output pair data to be had for the task of interest.

This second kind of transfer learning is common within the area of deep learning [13].

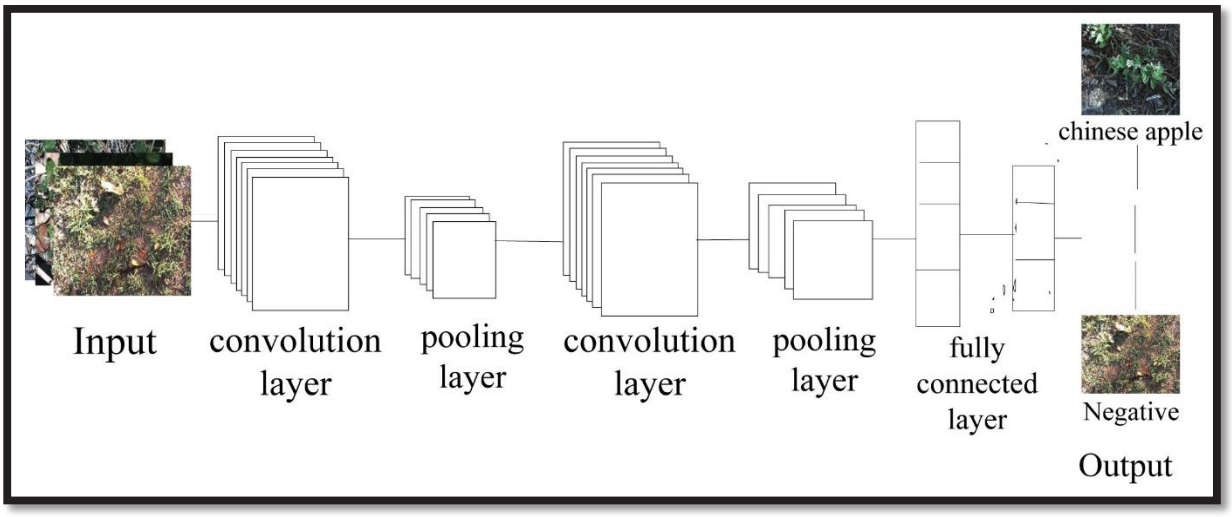


Fig 10. Deep convolutional neural network

2.2.3 RESNET (RESIDUAL NETWORK):

Deep residual studying framework
for picture classification tasks.
Which supports numerous architectural
configurations, allowing to gain an appropriate rati
o between the velocity of work and quality.

2.2.3.1 Resnet-18

ResNet-18 is a convolutional neural network that is 18 layers profound. You can load a pretrained model of
the community educated on more than a million pics from the ImageNet database. The pretrained organization can arrange pics into 1,000 article classes, for example, a console, mouse, pencil, and numerous creatures. As a result, the network has discovered prosperous fea
ture representations for a large variety of images. The community has a photo enter dimension of 224-by-224[14].

3. EVALUATION AND EXPERIMENT RESULTS:

3.1 EVALUATION PROCESS:

In this process, we used resnet18 as a pretrained network in the transfer learning method. The options we used for the training process are we used an initial learning rate of 0.004 with 10 epochs. The dataset is divided into train, validation, and test. The divided ratios are for train-80%, for validation-10%, for test-10%. We used an sgdm optimizer. The results describe the accuracy after training progress. We displayed validation accuracy and test accuracy. Besides, we calculated the confusion matrix along with performance metrics like F-score, recall, sensitivity, etc.

3.2 EXPERIMENT RESULTS

In this section, we are presenting the classification results of the deep weed dataset. In this experiment we are using Matlab as a platform for execution and the programming language is also Matlab. For this training progress, we used GPU. In this, we are providing performance metrics for our model. The results show in table 2. In this, the images are taken from several regions of Australia. The table describes the metrics for each class. Overall performance metrics for each class.

Table 2. Performance metrics

Classname	accuracy	sensitivity	specificity	precision	recall	F-score
ChineeApple	0.972648432	0.802083333	0.984319316	0.77777778	0.8020833	0.78974359
Lantana	0.980653769	0.813186813	0.991477273	0.86046512	0.8131868	0.83615819
Negative	0.942628419	0.961538462	0.922114047	0.93052109	0.9615385	0.94577554
Parkinsonia	0.991994663	0.943181818	0.995038979	0.92222222	0.9431818	0.93258427

Parthenium	0.988659106	0.863636364	0.996456414	0.9382716	0.8636364	0.89940828
Pricklyacacia	0.986657772	0.923076923	0.990767045	0.86597938	0.9230769	0.89361702
Rubbervine	0.989993329	0.860465116	0.997876858	0.96103896	0.8604651	0.90797546
Siamweed	0.993328886	0.945652174	0.99644634	0.94565217	0.9456522	0.94565217
Snakeweed	0.975983989	0.701149425	0.992917847	0.85915493	0.7011494	0.7721519

The validation accuracy is 91.21±% and test accuracy is 91.13±%. The confusion matrix is given below.

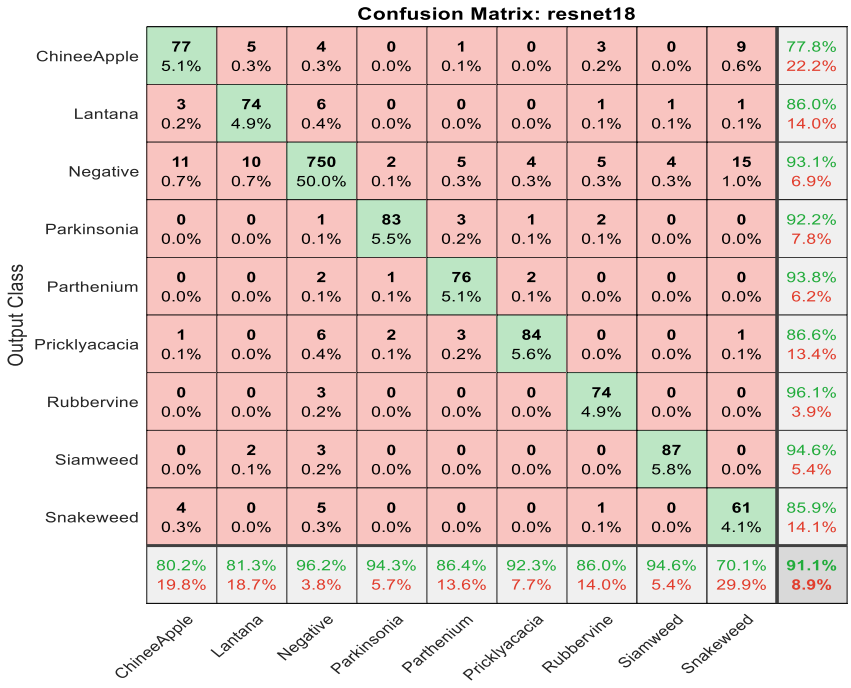


Fig 11. Confusion matrix

The overall performance metrics graph is shown below. The average accuracy of all classes is 98%.

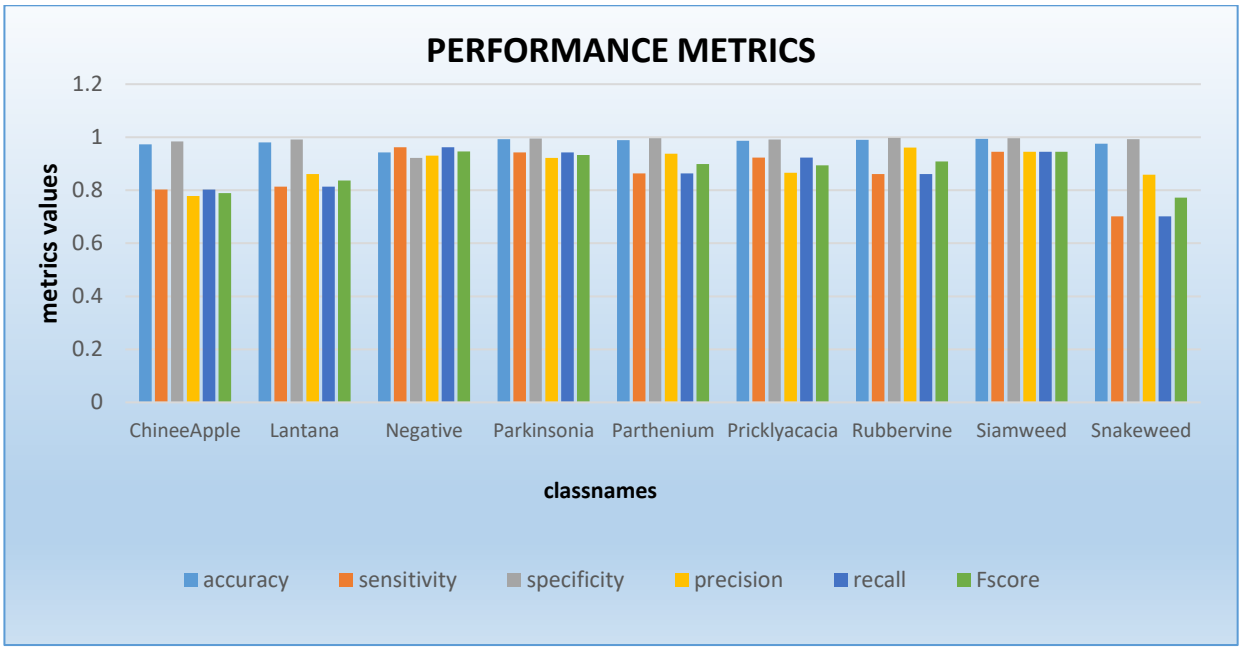


Fig 12. Describes the performance metrics for each class.

The resultant images as shown below are the random images displayed after testing is done. The images are showing how much the probability percentage of the data is matched with the training data.

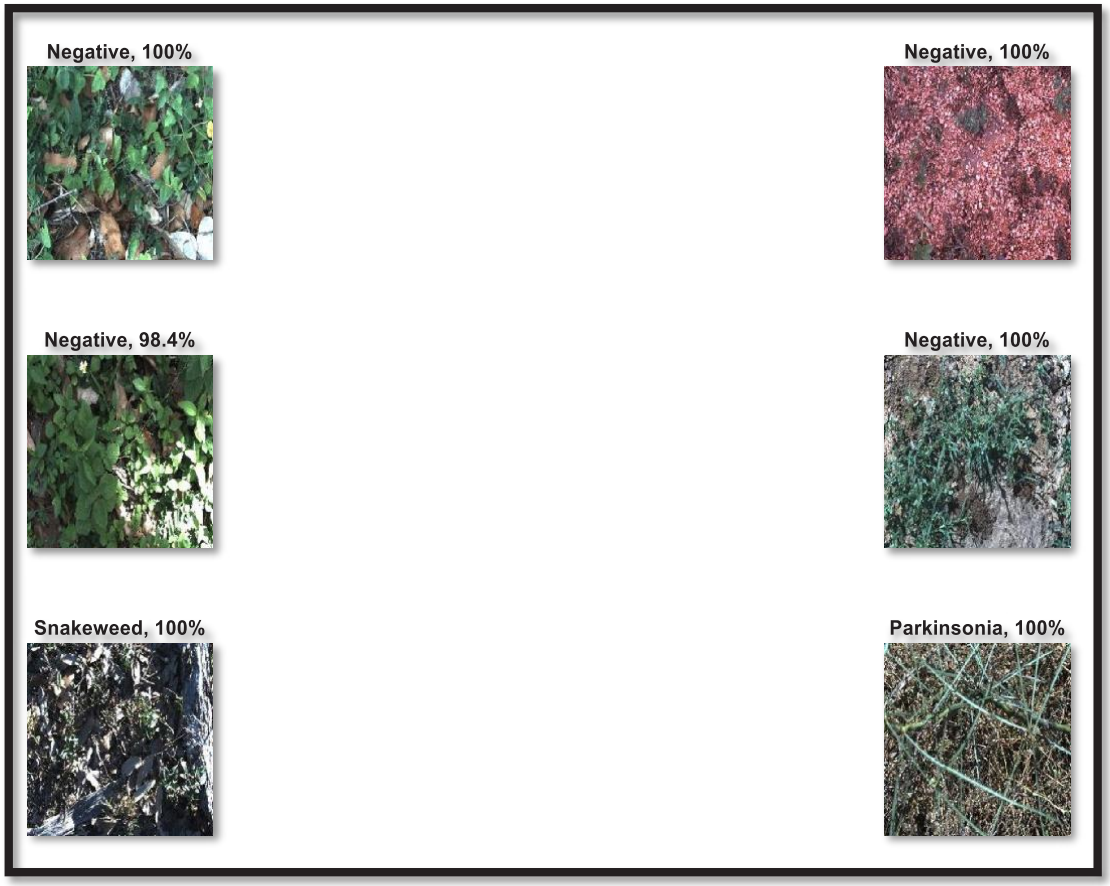


Fig 13. Random test result images

The results analyses give a well-defined performance with a new pretrained network as a new idea. This analysis provides a way to find weed in crop fields. This experiment is very useful to agriculture as farmers can identify the weed type with remedies to remove from farming land. This experiment can be developed as a mobile application. This makes farmers easy to take pictures from their devices and upload them into the application which gives the details about the weed.

4. ACKNOWLEDGMENT

The author wants to thanks Prof. S. Phani Kumar and Dr.Raul V. Rodriguez for providing the knowledge to do this project and also to the dataset collectors.

5. CONCLUSION AND FUTURE WORK

In this paper, we described the methods on a deep weed dataset using the transfer learning method. This is a multi-class classification. This could be a base for future experiments as we discussed this can be used as a mobile application or this can be installed in drones as software that detects the weed crop with the scanner in the drone. The researchers can use this as a reference and make predictions using different techniques as well as using different pretrained networks.

6. REFERENCES

1. Alex Olsen, Dmitry A. Konovalov, Bronson Philippa, Peter Ridd, Jake C. Wood, Jamie Johns, Wesley Banks, Benjamin Girgenti, Owen Kenny, James Whinney, Brendan Calvert, Mostafa Rahimi Azghadi & Ronald D. White "DeepWeeds: A Multiclass Weed Species Image Dataset for Deep Learning".
2. Dataset,https://github.com/AlexOlsen/DeepWeeds.
3. The Australian government website, https://www.environment.gov.au/biodiversity/invasive/weeds/

4. Brisbane City Council weed identification tool, https://weeds.brisbane.qld.gov.au/weeds/chinee-apple
5. Environment.gov,https://www.environment.gov.au/biodiversity/invasive/weeds/publications/guidelines/wons/pubs/l-camara.pdf
6. Environment.gov,https://www.environment.gov.au/biodiversity/invasive/weeds/publications/guidelines/wons/pubs/p-aculeata.pdf
7. Queensland Government, https://www.business.qld.gov.au/industries/farms-fishing forestry/agriculture/land-management/health-pests-weeds-diseases/weeds Diseases/invasive-plants/restricted/parthenium
8. Environment.gov,https://www.environment.gov.au/biodiversity/invasive/weeds/publications/guidelines/wons/pubs/a-nilotica.pdf
9. Brisbane City Council weed identification tool, https://weeds.brisbane.qld.gov.au/weeds/rubber-vine
10. The Australian government, depart of agriculture, water, and environment,https://www.agriculture.gov.au/biosecurity/australia/naqs/naqs-target-lists/siam-weed
11. Queensland Government, https://www.business.qld.gov.au/industries/farms-fishing-forestry/agriculture/land-management/health-pests-weeds-diseases/weeds-diseases/invasive-plants/other/snakeweeds
12. Yuqing Gao, Khalid M. Mosalam, Deep Transfer Learning for Image-Based Structural Damage Recognition, First published:16 April 2018
13. A Gentle Introduction to Transfer Learning for Deep Learning by Jason Brownlee on December 20, 2017, in Deep Learning for Computer Vision Tweet Share.
14. A Study on CNN Transfer Learning for Image Classification Mahbub Hussain, Jordan J. Bird, and Diego R. Faria
15. Mathworks, https://in.mathworks.com/help/deeplearning/ref/resnet18.html

About ADASCI

The Association of Data Scientists is the premier global professional body of data science & machine learning professionals.

ADaSI serves the scientific and professional needs of data science Professionals including educators, scientists, students, managers, analysts, and consultants. It serves as a focal point for data science, permitting them to communicate with each other and reach out their professional goal, as well as the varied clientele of the profession's research and practice.

It provides services such as publishing peer reviewed scholarly journals that describe the latest data science methods and applications, organizing national and international conferences for academics and professional, providing analytics certification and continuing education to assist members and others in furthering their career

PUBLISHED BY:

ADaSci | THE ASSOCIATION OF DATA SCIENTISTS

ONLINE CONTENTS AVAILABLE: EVERY QUARTER ON **www.adasci.org/lattice**