Dissertation on
Multi-class Classification of Diabetic Foot Ulcer using Faster R-CNN based on Wagner Grading Scale

Thesis submitted towards partial fulfilment
of the requirements for the degree of

Master of Technology in IT (Courseware Engineering)

Submitted by
RITUPARNA *CHATTERJEE*

EXAMINATION ROLL NO. M1CWE22011
UNIVERSITY REGISTRATION NO. 160374 of 2021-22

Under the guidance of
Dr SASWATI MUKHERJEE

School of Education Technology
Jadavpur University

Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata-700032
India

2023

MTech. IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled **"Multi-class Classification of Diabetic Foot Ulcer using Faster R-CNN based on Wagner Grading Scale"** is a bonafide work carried out by **RITUPARNA CHATTERJEE** under our supervision and guidance for partial fulfilment of the requirements for the degree of **Master of Technology in IT (Courseware Engineering)** in **School of Education Technology**, during the academic session 2022-2023.

Dr SASWATI MUKHERJEE
SUPERVISOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

DIRECTOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

DEAN - FISLM
Jadavpur University,
Kolkata-700 032

MTech. IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF APPROVAL

This foregoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned does not endorse or approve any statement made or opinion expressed or conclusion drawn therein but approves the thesis only for the purpose for which it has been submitted.

Committee of final examination for evaluation of the Thesis

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of her **Master of Technology in IT (Courseware Engineering)** studies.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by this rule and conduct, I have fully cited and referenced all materials and results that are not original to this work.

NAME	:	RITUPARNA CHATTERJEE
EXAMINATION ROLL NUMBER	:	M1CWE22011
REGISTRATION NUMBER	:	160374 of 2021-22
THESIS TITLE	:	MULTI-CLASS CLASSIFICATION OF DIABETIC FOOT ULCER USING FASTER R-CNN BASED ON WAGNER GRADING SCALE

SIGNATURE: DATE:

Acknowledgment

I feel fortunate while presenting this dissertation at **the School of Education Technology, Jadavpur University, Kolkata,** in the partial fulfilment of the requirement for the degree of **M. Tech in Information Technology (Courseware Engineering).**

I hereby take this opportunity to show my gratitude towards my mentor, **Dr Saswati Mukherjee,** who has guided and helped me with all possible suggestions, support, aspiring advice, and constructive criticism along with illuminating views on different issues of this dissertation which helped me throughout my work.

I would like to express my warm thanks to **Prof. Matangini Chattopadhyay, Director of the School of Education Technology** for his timely encouragement, support, and advice. I would also like to thank **Mr Joydeep Mukherjee** for their constant support during my entire course of work. My thanks and appreciation goes to my classmates from M. Tech in Information Technology (Courseware Engineering) and Master in Multimedia Development.

I do wish to thank all the departmental support staff and everyone else who has different contributions to this dissertation.

Finally, my special gratitude to my parents who have invariably sacrificed and supported me and made me achieve this height.

Date:
Place: Kolkata

Rituparna Chatterjee
Examination Roll Number: M1CWE22011
M.Tech in IT (Courseware Engineering)
School of Education Technology
Jadavpur University
Kolkata:70003

Contents

Executive Summary

Diabetic Foot Ulcers (DFU) are a major complication of Diabetes Mellitus (DM), that affects the lower extremities. It has been estimated that patients with diabetes have a lifetime risk of 15% to 25% in developing foot ulcers contributing to up to 85% of lower limb amputation due to failure to recognize and treat the foot ulcers properly. An extensive review of computerized techniques for the recognition of diabetic foot ulcers has been performed to associate the work done so far in this field. While performing the studies, it became clear that computerized analysis of diabetic foot ulcers is a relatively emerging field which is why related literature and research works are limited. There is also a lack of a standardized public database of diabetic foot ulcers and other wound-related pathologies.

This dissertation report aims to develop an automated deep learning-based method for localizing and classifying foot ulcers of diabetic patients in different grades/scales. The classification of multi-class foot ulcers is based on the Wagner diabetic foot ulcer grade system using the Faster Region-based Convolutional Neural Network (Faster R-CNN) algorithm.

The proposed automated classification process helps to take prompt therapeutic strategy and subsequent clinical treatment.

In this approach, images of diabetic foot ulcers (DFU) are collected from a Primary Care Hospital in Kolkata. The annotation of DFU images is performed by the domain expert for localization and the annotated images are further classified for prediction. Faster R-CNN-based two architectures namely, resnet-101 and mobilenet-v3 network are employed for the prediction. The training of the deep learning model is validated using 5-fold cross-validation technique. A comparative performance analysis of resnet-101 and mobilenet-v3 is performed to validate the efficacy of the deep learning models. The performance of the model with each network is compared. The results demonstrate that the proposed predictive model has accurately classified the foot ulcers in different grades which eventually helps in disease diagnosis and treatment.

1. Introduction

1.1 Overview

Diabetic Mellitus (DM), commonly known as diabetes is a lifelong condition resulting from high blood sugar levels (hyperglycemia) due to deficiency and action of insulin, can cause serious life-threatening complications like cardiovascular disease, kidney failure, blindness, and lower limb amputation, which is frequently preceded by diabetic foot ulcer (DFU) [1]. In 2016, according to the global report on Diabetes by WHO, 422 million people were suffering from DM in 2014, compared to 108 million people in 1980. Among adults that are over 18 years of age, the global prevalence has gone up from 4.7% in 1980 to 8.5% in 2014 [2]. Annually, more than 1 million diabetic patients are subjected to amputation due to delays in the treatment of DFU [3]. A diabetic patient with a "high-risk" foot requires ongoing, expensive medication, routine doctor visits, and hygienic personal care to prevent the adverse effects of foot loss. As a result, it places a heavy financial burden on the patients and their families, particularly in underdeveloped nations where the expense of treating this illness can be as high as 5.7 years of a family's annual income. In developed countries, healthcare systems are also extremely costly [4].

The worldwide adoption of information and communication technologies presents challenges as well as opportunities for the creation of modern healthcare systems. There are fewer computer methods developed for the assessment of diabetic foot diseases considering the study of DFU with computerized methods is still a relatively new topic. These approaches use traditional machine learning and basic image processing.
In recent years computer vision algorithms have greatly improved, particularly in the area of medical imaging. The advancement of deep learning methods has effectively found anomalies in various types of medical imaging.
Generally, from a computer vision and medical image perspective, three different tasks are performed to detect anomalies in medical images: classification, localization, and segmentation [5].

Hence, developing robust methods that can also detect, localized, and classify the foot ulcer is an important advancement in the computerized analysis of DFU.

1.2. The Wagner Diabetic Foot Ulcers Grade Classification System

Foot ulcers are caused by damaged skin tissues under the big toes and on the plantar surface of the foot. Foot ulcers are a consequence of untreated diabetes that expose the skin's underlying layer and damage the foot to the bone. Recognizing the variables that determine the rate of DFU healing and the risk of amputation requires an understanding of infection and ischemia.
The medical categorization systems for DFU are used to categorize the DFU based on numerous factors such as size, area, neuropathy, ischemia, and infection to predict the prognosis of DFU. These systems are all currently based on observations made by the physician and clinical judgments.

The Wagner classification and Texas classification are the two main systems used to categorize diabetic foot conditions clinically. Meggitt initially suggested the Wagner classification in 1976. The Wagner classification system [6], which is based on the depth of penetration, the existence of osteomyelitis or gangrene, and the amount of tissue necrosis, is one of the most frequently accepted categorization systems. (Table 1 for Wagner classification system). the Wagner classification method's clinical applicability still relies on its simplicity and usefulness. Wagner classification is the foundation for the target detection-based classification standard for image recognition.

Table 1: Wagner diabetic foot ulcer grade classification system

Ulcer Grading	**Description**
Grade0	Though there is a possibility of foot ulceration, none has yet developed.
Grade1	Foot ulceration that is only superficial and stands out as a neurotic ulcer with no evidence of infection
Grade2	No myelitis or deep abscess, but a deep ulcer frequently associated with soft tissue inflammation
Grade3	Deep ulcer with osteomyelitis or an abscess
Grade4	Ischemic gangrene, which is present in localized gangrene (toe, heel, or dorsal forefoot), is typically linked to neuropathy.
Grade5	The whole foot gangrene

The researchers' task of detecting DFUs is difficult due to (i) lighting conditions, (ii) considerable similarity in inter- and intraclass changes, and (iii) foot ulcers that emerge in a variety of sizes, shapes, and locations. For a precise diagnosis of DFU, a thorough medical history, physical examination, bacteriological investigation, blood tests, and in-depth research on the leg blood vessels are all necessary. Most of the time, these tools and examinations are not accessible worldwide.

Many Computerized techniques are required to classify and localize the DFU at an early stage to get beyond these constraints.

In this thesis, deep learning algorithms are examined for categorization and localization of the DFU by considering existing literature.

1.3. Problem Statement

Multi-class classification of diabetic foot ulcer using Faster R-CNN-based on Wagner Grading System.

1.4. Objectives

The objectives are:

I. To develop a deep learning-based prediction method for classifying diabetic foot ulcer (DFU) based on the Wagner foot ulcer classification system.

II. To localize and classify foot ulcers in different classes which helps in prompt clinical decision making and treatment.

III. To localize the Region of Interest (ROI) and augment the classification process, DFU images are annotated.

IV. To prevent generalization errors, model validation is performed for classification accuracy.

V. To achieve optimal performance of the classification model, hyperparameter adjustments are done during model-training.

2. Background Concepts

Deep learning methods for object localization task in the computer vision and medical imaging field is drawing a lot of attention from both researchers and developers. In the last few years, the accuracy of algorithms on public object localization datasets is significantly improved with the introduction of Convolutional Neural Networks (CNNs).
Object identification or localization in medical images is the process of locating items in an image and classifying them according to their characteristics. There are currently two types of object detection based on deep learning: single-stage detection architecture and two-stage detection architecture. While the latter views object detection as a regression or classification problem, adopting an integrated process to achieve final results (categories and locations) directly, which are relatively fast but less accurate, compared to the former, the former first generates region proposals and then classifies each proposal into different object categories, and its advantage is accuracy [7,8]. Two-stage architectures include R-CNN, fast R-CNN, faster R-CNN, and feature pyramid network (FPN) while one-stage architectures include YOLO, EfficientDet, and SSD.

However, these region-based CNNs, such as Faster R-CNN and Mask R-CNN, are commonly used for diabetic foot ulcer localization. These models combine object detection and classification by identifying regions of interest (foot ulcers) within an image and predicting their class labels. Region-based CNNs enable both localization and classification of foot ulcers in a single framework.

2.1. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN), is a deep learning method that can take an image as input, assign importance that are learnable biases and weights to numerous objects in the image, and then segregate them.
Compared to other classification approaches, CNNs require less pre-processing. With relevant training, CNNs are capable of grasping the distinctive characteristics of filters, which are often hand-engineered in traditional procedures [9,10].

The architecture of CNN is comparable regarding how real brain neurons are connected. It was brought on by a phenomenon known as the Visual Cortex Association, in which certain neurons respond to stimuli in a small area of the visual (receptive) field. One of these fields encompasses the entire visual field.

CNN Layers

The various CNN layers are shown in Figure 1.

- Input Layer: It contains visual data that is shown as a 3-D matrix.
- Convo Layer: It is a feature extractor layer where image features are extracted. An image fragment is attached to this layer to accomplish this procedure. The dot product between the filter and the receptive field is calculated. The operation's output is a single integer volume. [10].

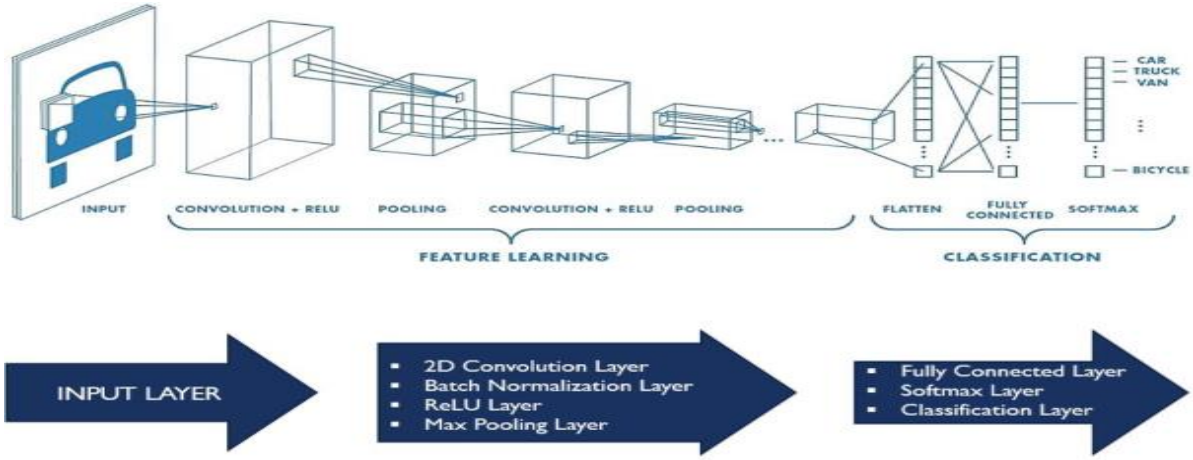


Figure 1: Feature Detection and Classification using CNN

- Batch Normalization Layer: With the assistance of this layer, each layer of the network can learn more independently. The activations in normalization ascend the input layer. Additionally, it is used to normalize the output of preceding layers.

- Pooling Layer: It is used to decrease the 3-D volume of image input afterwards convolution. If FC is applied next to the Convo layer without max pooling, it is affluent. Therefore, the only means to lessen the spatial volume of the image input - is max pooling.

- Fully Connected Layer: Biases, neurons, and weights are all part of FC. A layer's neurons are connected to those in another layer. It is used for segmenting images by training them into many categories.
- Logistic (Softmax) Layer: It operates at the FC layer's ending. While logistics is used in binary classification, the softmax layer is employed in multi-classification.
- Output Layer: This has one hot encoding for the label.

2.2. Transfer Learning

Deep CNNs are pre-trained to classify the different collections of images, which makes this process time efficient than the others. Transfer learning refers to the process of gathering knowledge from one model and applying it to another. In DL applications, it is commonly employed. This method is shown in Figure 2.

It is laborious to train a network from scratch using weights that are arbitrarily chosen. Transfer learning makes network fine-tuning considerably simpler and faster. This is because it employs a pre-trained network as a starting point for learning a new skill [10]. Less training images can easily transfer learned features to a new assignment.
Region-based CNN detectors used a pre-trained convolution neural network (CNN) as the basis for an object detection model.

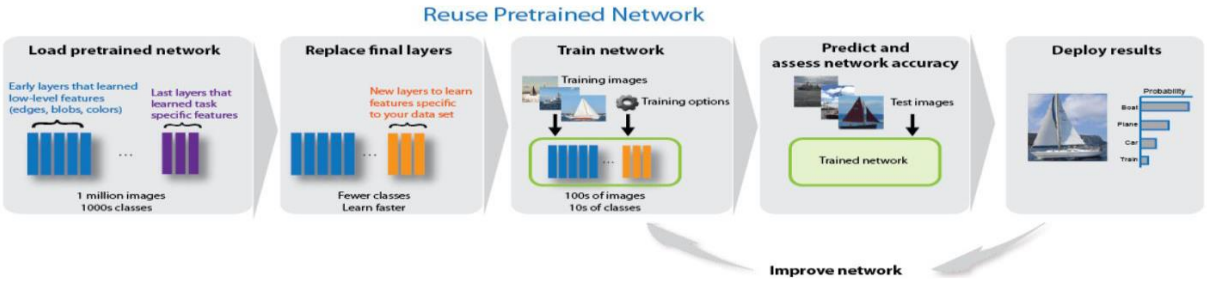


Figure 2: Transfer Learning

2.3. Object Localization Models

The descriptions of Object Localization Models are as follows.

R-CNN

Regions with convolutional neural networks (R-CNN), combine rectangular region proposals with convolutional neural network features. R-CNN is a two-stage detection algorithm. The first stage identifies a subset of regions in an image that might contain an object. The second stage classifies the object in each region.
R-CNN detector first generates the region proposals using an algorithm such as Edge Boxes. The proposal regions are cropped out of the image and resized. CNN then classifies the cropped and resized regions. Finally, the region proposal bounding boxes are refined by a support vector machine (SVM) that is trained using CNN features.

Fast R-CNN

As in the R-CNN detector, the Fast R-CNN detector also uses an algorithm like Edge Boxes to generate region proposals. Unlike the R-CNN detector, which crops and resizes region proposals, the Fast R-CNN detector processes the entire image. Whereas an R-CNN detector must classify each region, Fast R-CNN pools CNN features corresponding to each region proposal. Fast R-CNN is more efficient than R-CNN because in the Fast R-CNN detector, the computations for overlapping regions are shared

Faster R-CNN

After the Fast R-CNN version of the R-CNN's object detection network architecture was developed. Although the network's training and detection times are greatly reduced, the network is not quick enough to be employed as a real-time system because it takes about two seconds to create output on an input image. A selective search algorithm is the architecture's bottleneck. As a result, a novel architecture known as Faster R-CNN was suggested. Instead of using selective search, Region Proposal Network was proposed which another algorithm for generating region proposals. Faster R-CNN is a two-stage object identification architecture that was introduced by Ren et al. in 2015 [11].

The foundation of faster R-CNN is CNN architecture, which creates feature maps by extracting features from the input image. The region proposal networks (RPNs) were introduced to take the convolution feature map as input, and then to output a series of proposing regions with objectness score generated by a sliding window convolution applied on the input feature map. The object detection network used in Faster R-CNN is very much similar to that used in Fast R-CNN. In terms of a backbone network, it is likewise compatible with VGG-16. Additionally, it employs twin layers for the softmax classifier and the bounding box regressor, which are also employed in the prediction of the object and its bounding box, as well as the region of interest (ROI) pooling layer for creating a region proposal of a fixed size. Additionally, the feature map is shared by the RPN and ROI pooling layer, which decreases the number of parameters and prediction time.

The faster-R CNN detection process is shown in Figure 3, which is mainly composed of three stages.

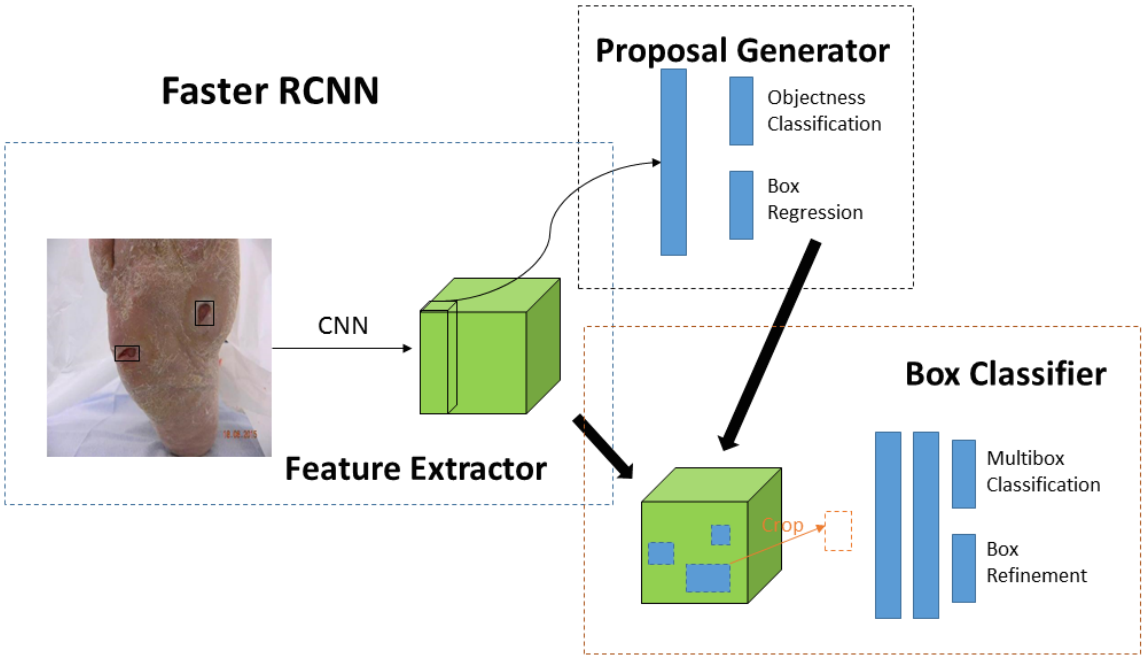


Figure 3: Faster R-CNN Architecture for DFU localization [5].

CNN as a feature extractor

In Stage 1, the convolutional layers of a standard CNN, such MobileNet, retrieve the features from the input images as feature maps. With a focus on DFU zones as seen in Figure. 4, these feature maps are used to identify the objects in the image. The generation of proposals in the second stage, as well as the classification and regression of RoI in the third stage, both use these feature maps as input.

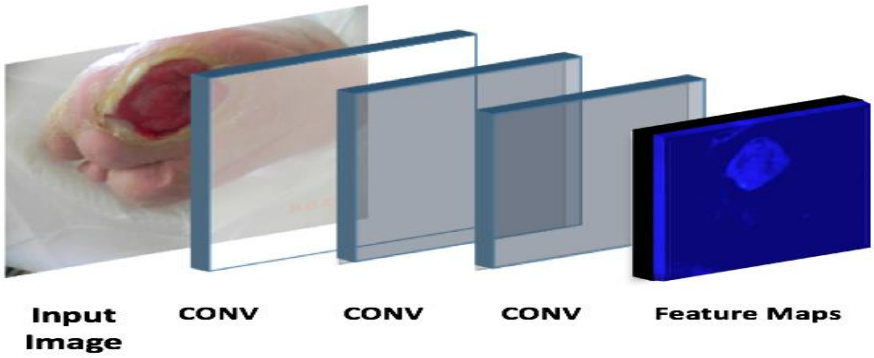


Figure 4: Feature map extraction for object localization [5].

Generation of proposals and refinement

In Stage 2, using the feature map that was extracted in Stage 1, the network does a sliding-window scan of the image to identify specific areas that contain the objects to be identified. These regions, which include various boxes dispersed around the image, are referred to as proposals. To include as many things in the image as feasible, approximately 200,000 proposals of various sizes and aspect ratios are often found. Faster R-CNN generates this many anchors using GPU in 10ms [12]. Each proposal receives two outputs from Stage 2:

- Proposal Class: It can be either foreground or background. The foreground class means there is likely an object in that proposal and it is also known as a positive proposal.
- Proposal Refinement: A positive proposal might not perfectly capture the object. So, the network estimates a delta (% change in x, y, width, height) for refinement of the proposal box to center the object better as illustrated in Figure 5.

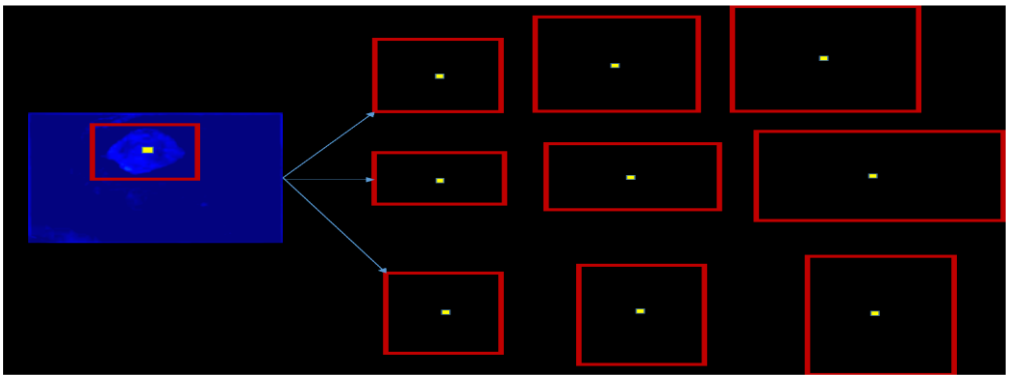


Figure 5: Detection of proposal boxes with respect to translate/scale operation [5].

Loss function of Region Proposal Network (RPN)

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}\,(p_i\,, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i L_{reg}\,(t_i, t_i^*)$$

Here, *i* represents the index of anchor in a mini-match, and

p_i represents the prediction probability that anchor *i* belongs to the target.

p_i^* represents the prediction probability of the corresponding real box GT (Ground Truth).

t_i is a vector representing the four-parameter coordinates of the Bounding box,

t_i^* is the coordinate vector of the real box GT corresponding to the anchor box.

L_{cls} is the classifier loss that is binary log loss over two classes,

L_{reg} represents the regression loss,

N_{cls} is the Normalization parameter of mini-batch size which is approx.256,

N_{reg} is the Normalization parameter of regression (equal to the number of anchor locations which is approx. 2400),

λ = 10 is generally considered to make both parameters equally weighted.

RoI classifier and bounding box regressor

Stage 3 entails classifying the RoI boxes Stage 2 provided and further refining them, as shown in Figure 6. As RoI boxes might have varied sizes, all of them are first fed into the RoI pooling layer to be resized into a fixed input size for the classifier. It generates two outputs for each ROI, similar to Stage 2:

- RoI Class: The softmax layer provides the classification of regions to specific classes (if more than one class). If the RoI is classified as a background class, it is discarded.
- Bbox Refinement: Its purpose is to refine the location of RoI boxes.

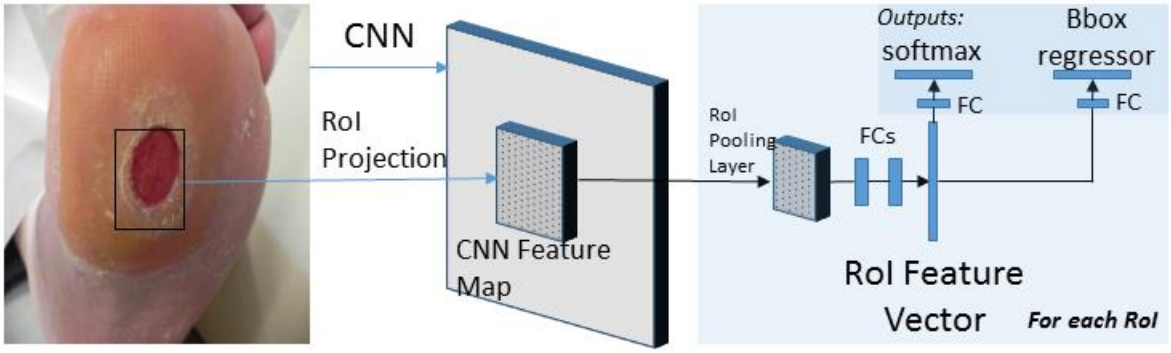


Figure 6: The classification and box refinement of RoI boxes.

Softmax and bounding box regression layer

The feature map of size *(7 * 7 * D)* produced by RoI pooling is subsequently sent to two fully connected layers, which flatten the feature maps and then transmit the output into two parallel fully connected layers, each of which is given a separate task:

- The first layer predicts the objects in the region proposal using a softmax layer with N+1 output parameters (N is the number of class labels and background).
- A bounding box regression layer with 4* N output parameters makes up the second layer. This layer regresses the position of the object's bounding box within the image.

For bounding box regression, Faster R-CNN adopts the following method:

$$t_x = (x - x_a)/w_a, \qquad t_y = (y - y_a)/h_a,$$

$$t_w = \log(w/w_a), \qquad t_h = \log(h/h_a),$$

$$t_x^* = (x^* - x_a)/w_a, \qquad t_y^* = (y^* - y_a)/h_a,$$

$$t_w^* = \log(w^*/w_a), \qquad t_h^* = \log(h^*/h_a),$$

Faster R-CNN architecture which is an object localization model used state-of-the-art classification algorithms like MobileNet, and ResNet101, to get the anchor boxes from the features maps, and finally, classify these anchors into different classes.

3. Literature Survey

Goyal et al. [13], proposed a novel deep learning framework, DFUNet, to distinguish between normal (healthy) skin and abnormal (DFU) skin. DFUNet consisted of parallel convolution blocks to concatenate features from convolution operations and conventional single convolutional layers. Standard CNN architecture like LeNet, AlexNet, and GoogleNet have been employed to extract high-level features, which are fed to ML classifiers for DFU identification. 10-fold cross-validation is used with an AUC score of 0.961. DFUNet lacks sufficient transition layers between parallel convolutional blocks, thus hindering abstraction from multi-layer feature map concatenations.

A novel CNN architecture DFU QUTNet [14] extracts multi-level features, and these features were finally fed to Support Vector Machine (SVM) and K-Nearest Neighbors (KNN) classifiers. This novel architecture was designed by increasing the network width without reducing computation costs. The DFU QUTNet, based features, and SVM classifier outperformed the traditional CNN models (GoogleNet, AlexNet, and VGG16) in terms of classification results by achieving the F1 score of 94.5%. Despite the DFU QUTNet model's remarkable speed, this technique suffers from efficiency because it uses identical kernel sizes (3 × 3) in the convolutional layers instead of heterogeneous kernels that improve the receptive field of the network.

Faster-RCNN combined with Inception-v2 and Resnet101, to identify the DFU cases among the normal cases. The combined model achieved an F1-score of 0.7424 and a mean average precision of 0.6940 [15]. The model locates the areas of interest in the image and classifies them into two categories, ulcer, and non-ulcer. Increase in the number of false positives values which leads to a reduction in the precision value.

Oliveira et al. [16] proposed a tool for detecting diabetic foot ulcers based on original faster R-CNN object detection. Monte Carlo cross-validation technique employed for validation of training set. The method has a mean average precision of 91.4%, an F1-score of 94.8%, and an average detection speed of 332ms. The precision value is higher and the detection time is less.

An ensemble strategy [17] is proposed that combines the five modified convolutional neural networks, i.e., VGG-16, VGG-19, Resnet-50, InceptionV3, and Densenet-201, to classify the DFU images. The combination of the five CNNs improved the classification rates. After five-fold cross-validation, an average accuracy of 95.04% and a Kappa index of over 91.85% were achieved. Although the ensemble model increases the computational complexity, data requirement, and overfitting risk, it still provides improved performance and robustness.

A reliable model [18] is presented to predict the probability of in-hospital amputation in DFU patients. A multi-class classification model was created using the light gradient boosting machine (LightGBM) to forecast the three outcomes that are minor amputation, non-amputation, and major amputation outcomes. The SHapley Additive exPlanations 72 (SHAP) method was utilized to evaluate the model's predictions and obtained AUCs of 85%, 90%, and 73.86% for minor amputation, non-amputation, and major amputation outcomes, respectively.

Khandakar et al. [19] compared numerous state-of-the-art convolutional neural networks (CNNs) to a machine learning-based scoring technique, on foot thermogram images using feature selection and optimization techniques, as well as learning classifiers, and provided a reliable solution to diagnose the diabetic foot. The AdaBoost Classifier used 10 features and obtained an F1 score of 97%, and MobilenetV2 only produced an F1 score of 95% percent for a two-foot thermogram image-based classification.

In [20], an end-to-end CNN-based deep learning architectures, i.e., AlexNet, VGG16,19, GoogLeNet, ResNet50,101, MobileNet, SqueezeNet, and DenseNet, for infection and ischemia categorization using the benchmark dataset DFU2020 are presented. The weight has been fine-tuned to overcome the data-scarcity and to reduce the computational cost. The affine transformation is used to augment the input data. The results indicate that the ResNet50 has achieved an accuracy of 99.49% and 84.76% for ischemia and infection respectively.

Liu et al. [21] proposed a Faster-RCNN target detection model to intelligently identify diabetic foot ulcers. The classification criteria are strictly under the Wagner level. The detection model is based on two model schemes of the resnet101 network and VGG16 network respectively. The experiment shows that the precision of the Faster R-CNN model based on resnet101 is higher than that of VGG16. The use of method of changing the contrast of the picture, adding random Gaussian noise, and flipping to enhance the data for uneven distribution of the number of foot images in some Wagner grades.

4. Proposed Approach

In the proposed automated system, the images of foot ulcers are annotated by the domain expert which helps to localize the region of interest. After the localization is completed, a prediction model is developed that classifies the foot ulcers into different grades following the Wagner grading system. The severity grades range from 0-5 where, grade 0 → pre-or post-ulcerative lesion, grade 1 → partial or full thickness ulcer, grade 2 → ulcer probing to tendon or capsule, grade3 → deep ulcer with osteitis, grade 4 → partial foot gangrene and grade 5 → whole foot gangrene.

The localization and classification model employed Faster R-CNN which intelligently locate the ulcer in the diabetic foot image.
At the very onset, diabetic foot ulcer images are collected from Primary Care Hospital in Kolkata, followed by image annotation that creates a bounding box around the ROI. The foot ulcer images are further arranged in VOC format where the description of the bounding box is given in XML file format. The dataset is then augmented to avoid overfitting the model, and validated using a 5-fold cross-validation technique before training the model. Pre-trained models are used with the MS-COCO dataset for transfer learning. The Faster R-CNN model uses two different CNN backbone networks namely resnet-101 and mobilenet-V3.

4.1. Dataset Description:

In this work, the foot images of diabetic patients were collected from the Primary Care Hospital in Kolkata. The identity of all the patients is removed to maintain confidentiality by assigning a unique identification number in the dataset. The foot images of diabetic patients that were collected contain a total of 1,052 images, including all six classes, which are grade 0, grade 1, grade 2, grade 3, grade 4, and grade 5, each corresponding to the respective Wagner's grade. The frequency count of ulcer images as per the Wagner grades is shown in Table 2.

Table 2: DFU images distribution into different classes of Wagner grade.

Grades	***Ulcer Description***	***Number of images***
grade 0	Pre-or post-ulcerative lesion	120
grade 1	Partial or full-thickness ulcer	236
grade 2	Ulcer probing to tendon or	188

	capsule	
grade 3	deep ulcer with osteitis	158
grade 4	partial foot gangrene	147
grade 5	whole foot gangrene	36

All the collected ulcer foot images lie in varied pixel dimensions which ranged from 1600 x 1200 to 4160 x 3120. To avoid computational complexity all the images were reduced to 640 x 640. The images of foot ulcers of diabetic patients are graded by the domain expert as per the Wagner grading scale. The expert professionals draw a rectangular box around the area of interest (i.e., the ulcer and its surroundings) in the ulcer foot images using the Computer Vision Annotation Tool (CVAT) (https://www.cvat.ai/), a graphical image annotation tool. The object bounding boxes were tagged to all the diabetic foot images. Since grade 0 in the DFU dataset does not significantly differ from healthy feet, it is considered a complete foot. A total of 976 ground truths were represented by medical professionals (some of the images in the diabetic foot collection have more than one DFU). An extensible Markup Language (XML) file is created after the DF image is designated with the appropriate class and bounding box, as shown in Fig. 7.

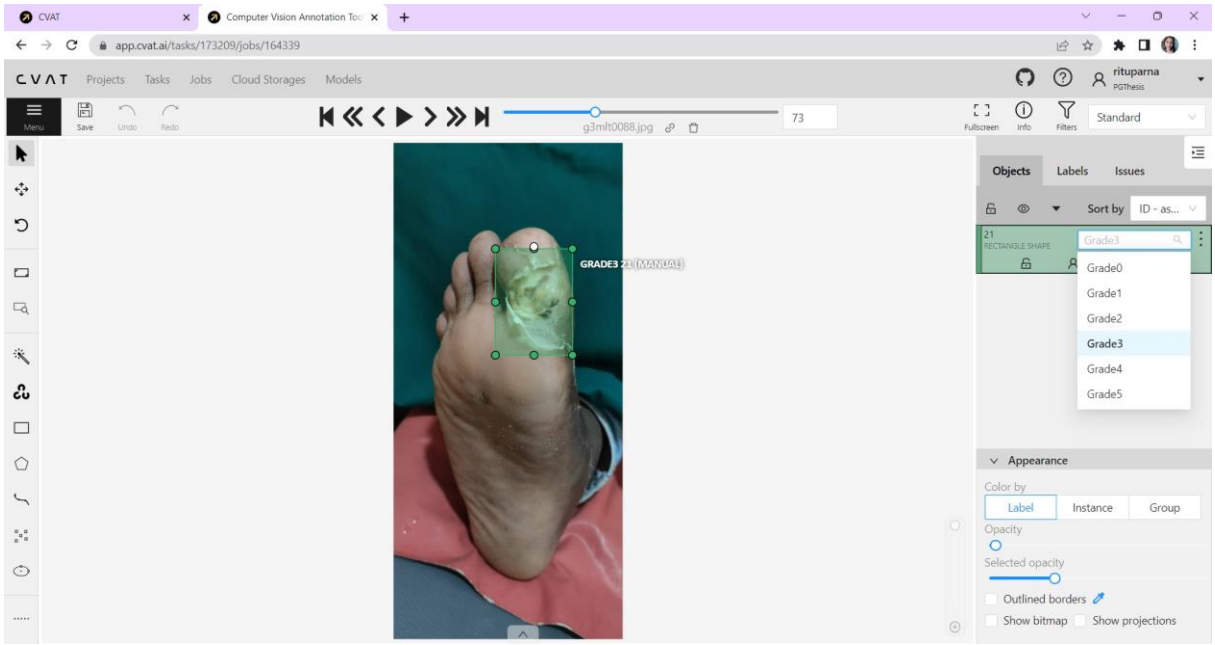


Figure 7: Example of depicting the ground truth on the DFU dataset using CVAT.

In Fig. 8 the diabetic foot ulcer images of each Wagner grade are shown. For each image in the DFU dataset, there is only one bounding box for maximum images in one class.

Figure 8: Illustration of high-resolution Wagner ulcer classification images.

4.2. Data Pre-processing

I. Data Augmentation

The DFU dataset has only 1,052 images, hence avoid over-fitting during model training augmentation is performed. The operation used in the augment process is flip, rotation, random scale, random crop, and translation. the augmentation operations help to locate the Region of interest of foot ulcer images. As the area of the foot ulcers is smaller in size than

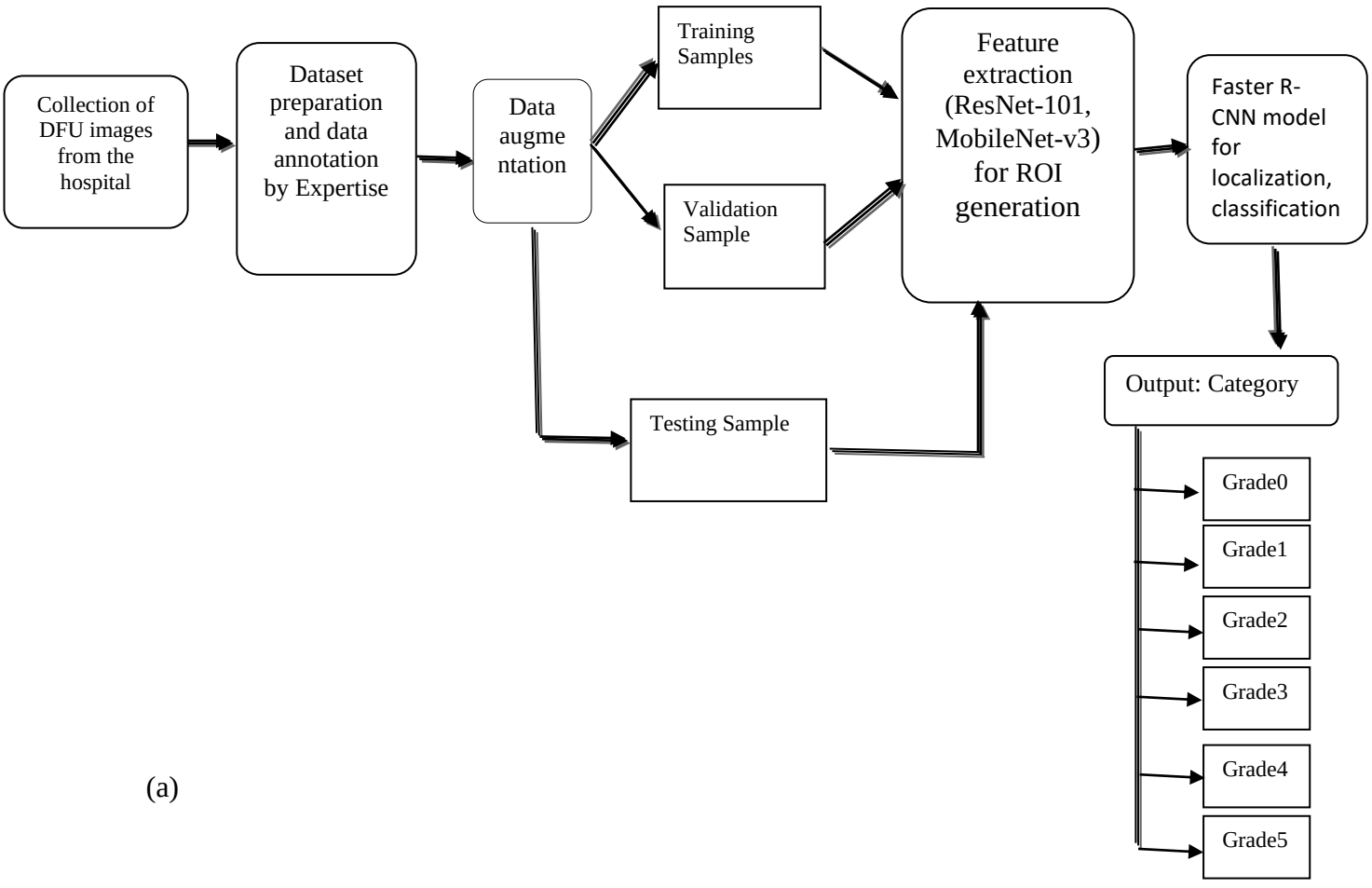
the entire foot image so there is a probability of missing the ROI of DFU images. The augmentation technique helps the predictive model to clearly pinpoint the ROI of foot images and thereafter the strong features are extracted from the localized area.

II. Cross-validation

The validation is performed on the training samples to assess the predictive analysis of the model. 5-fold cross-validation is applied on the training samples where the entire training samples are split into 5 equal folds. In each of the 5 folds, 70% of the dataset are preserved for training, 10% for validation, and 20% for testing. For each fold, the total dataset is split into three sets: a train set of 736 images, a validation set of 106 images, and a test set of 210 images. The dataset is validated/tested for five times and in each of the processes of execution, a random set is chosen.

4.3. Methodology

The workflow of the proposed classification model is shown in Figure 9. Once the dataset of diabetic foot images is arranged with their respective labelled bounding boxes, the pre-processing steps are performed.



(a)

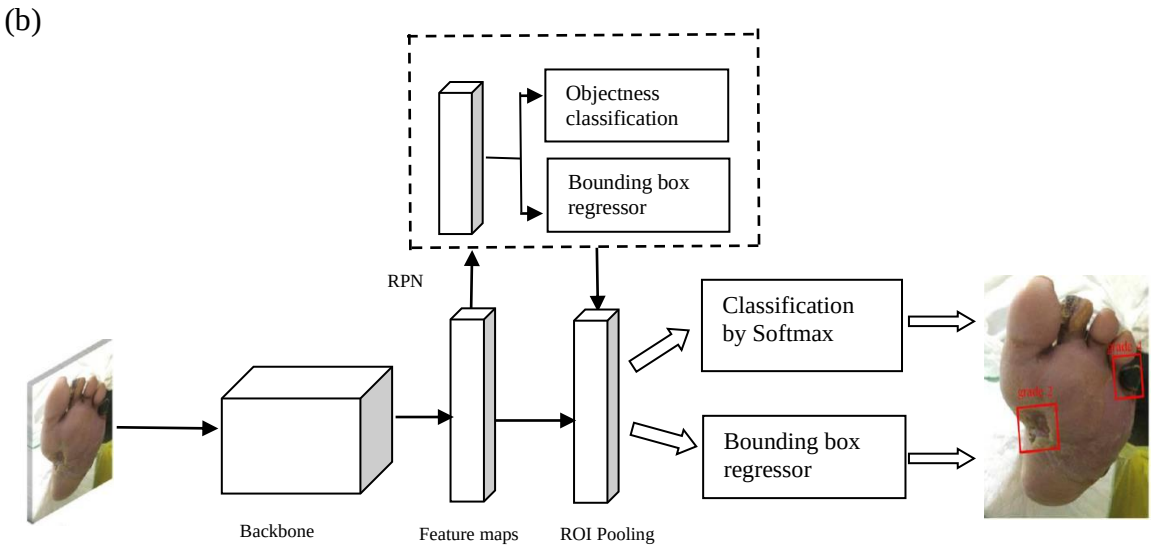


Figure 9: (a) workflow of proposed classification and (b) Faster R-CNN localization and classification of DFU images

DFU dataset is validated using a 5-fold cross-validation technique. The entire dataset is randomly split into 5 test sets (20% of each) for 5-fold cross-validation. This is to ensure that the dataset is evaluated on the test sets. For each test set (20%), the remaining images are randomly split into 7:3 ratio of train and validation set.

Faster R-CNN model is applied for the localization and classification of foot ulcers. The two CNN architectures namely ResNet-101 and MobileNet-V3 are employed for feature extraction and classification. CNNs required a considerable amount of data for performance scaling. As the medical imaging data are diverse and limited so the concept of transfer learning is considered. Both the models are pre-trained by employing transfer learning that uses datasets of common objects not related to the medical domain such as ImageNet and MS-COCO datasets. In the proposed approach, the network is initialized with pre-trained weights considering the MS-COCO dataset, which consists of 82,783 images in the train set with 90 classes.

Faster R-CNN-based ResNet-101, and MobileNet-v3, are the convolutional layers that composed of standard CNN. The CNN-architectures are utilized as feature extractors to retrieve features from the input images as feature maps, and to locate the ulcers in the input foot-image with strict reference to the ground truth area. The sole objective of MobileNet-v3 architecture is to reduce the latency while considering the size of the model into account. ResNet-101, also known as Residual Networks 101 focuses on precision in comparison to the MobileNet-v3 network and generates a high number of parameters after training.

The learning rate (LR) hyper parameter varies from 0.0001 to 0.005 and it uses a step scheduler to ensure that the results of the model-training are optimal. Weight decay of 0.0005 is used to prevent overfitting which add a penalty to the loss function for large weights. To minimize both training and validation loss, the number of training epochs considered is 80.

5. Results and Analysis

In this work, Python (version 3.10) is used with the required built-in libraries. For the implementing Faster R-CNN model, PyTorch is utilized, an open-source deep learning framework due to its flexibility and computational power. It uses GPU- accelerated libraries i.e., CUDA (torch version 2.0.1+cu118). TorchVision (version 0.15.2+cu118) is used for employing pre-trained models.

Two different backbone architectures of Faster R-CNN that is, ResNet-101 and MobileNet-V3 are tested, for locating the ulcer sites as well as classifying them into the respective grades. The mean average precision (mAP) and the F1-score of each detector are shown in Table 3. The mAP measures the overlap percentage of the prediction and ground truth, and is defined as the average of average precision for all classes:

$$mAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}$$

where a class represents the occurrence of a DFU, Q is the number of queries in the set (testing set image), and AveP(q) is the average precision for a given query, q.
This metric needs an overlap criterion that specifies the minimum value of the intersection over union (IoU) for correct detection. The value of IoU is chosen as 0.5. F1-score is a metric defined by the harmonic mean of precision and recall. The precision, recall, and F1 score can be calculated using the followings expressions:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

where, TP represents the number of true positive, FP denotes the number of false positives, and FN denotes the number of false negatives. The size of the model depends the framework used, specific implementation details, and the storage format. For the model Faster R-CNN with ResNet-101 backbone, it has 170 million parameters, whereas the size of MobileNet-v3 ranges from several megabytes to a few tens of megabytes.

Table 3: Performance of DFU detection model

Technique	mAP	F1-score	Speed (ms)	Size of Model (MB)
Faster R-CNN - ResNet-101	90.32	89.75	101	181.02
Faster R-CNN - MobileNet-v3	87.2	87.56	48	74.4

The performance evaluation of the object localization model on 5-fold cross-validation is shown in Table 3. Overall, the two CNN architecture of the Faster R-CNN model achieves promising localization results with high confidence in the DFU dataset. A few instances of accurate ulcer localization results by both the trained models are demonstrated in Figure 10.

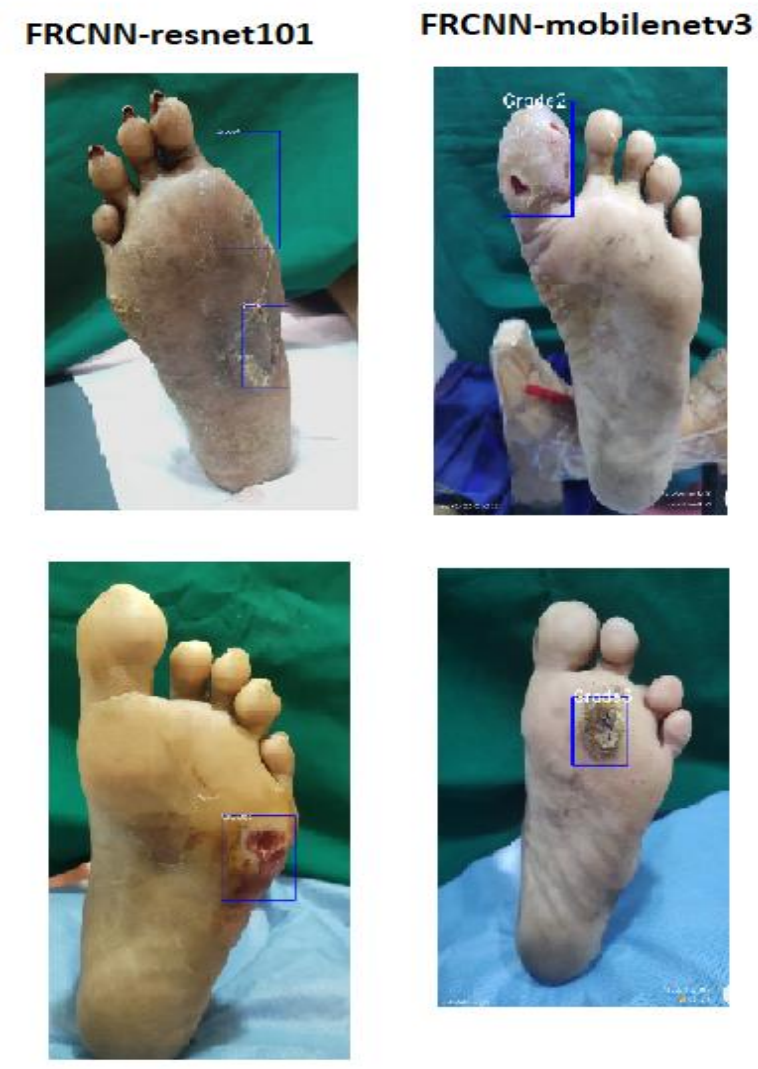


Figure 10: Accurate localization for performance comparison of ResNet-101 and MobileNet-v3

The model Faster R-CNN with ResNet-101 achieved highest mAP of 90.32 and F1 score of 89.75. while the other network, Faster R-CNN with MobileNet-v3 achieved the lowest value in speed of 48ms and size of 74.4 MB.

In Figure 11, the confusion matrix of the testing results is demonstrated. The confusion matrix provides a more complete display of the performance of our algorithm. By comparing the results from Figure 11, the classification accuracy of the DFU grade 0-5 is overall higher.

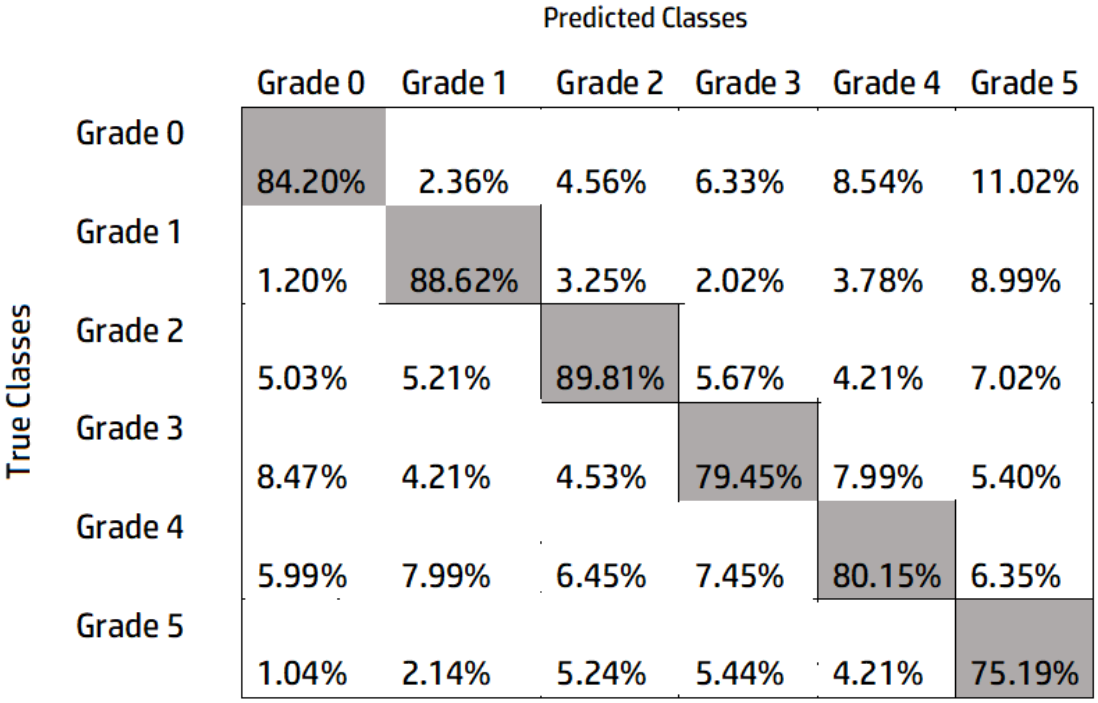


Figure 11: Confusion matrix

6. Conclusions and Future Scope

The proposed approach locates and classifies the diabetic foot ulcers in different grades as per the Wagner's classification system. Six different stages of ulcers are detected and analysed which aids in prompt clinical decision making for a podiatrist. Faster R-CNN model uses two different CNN feature extractors, ResNet-101 and MobileNet-v3. The performance of both Faster R-CNN with ResNet-101 and Faster R-CNN with MobileNet-v3 are measured in the same DFU dataset which is augmented, and validated. The hyperparameter optimization is performed to improve the overall accuracy of the prediction model. The comparative analysis shows that ResNet-101 outperforms MobileNet-v3 in accuracy and mean average precision, whereas with respect to model-size and speed, MobileNet-v3 outperforms ResNet-101. Although, the choice between MobileNet-V3 and ResNet-101 depends on specific requirements such as model size, inference speed, and desired accuracy. MobileNet-V3 is generally favoured for applications with limited resources or real-time constraints, while ResNet-101 is preferred when higher accuracy is crucial and computational resources are more abundant.

In future, this work can be further extended for designing a real-time Wagner Grade detector, by employing other detection models with MobileNet-v3 architecture.

7. References

[1] S. Wild, G. Roglic, A. Green, R. Sicree, and H. King, "*Global prevalence of diabetes estimates for the year 2000 and projections for 2030"*, Diabetes care, vol. 27, no. 5, pp. 1047-1053, 2004.

[2] World Health Organization, "Global report on diabetes", who, Geneva, 2016.

[3] D. G. Armstrong, L. A. Lavery, and L. B. Harkless, "Validation of a diabetic wound classification system: the contribution of depth, infection, and ischemia to risk of amputation", Diabetes care, vol. 21, no.5, pp. 855–859, 1998.

[4] P. Cavanagh, C. Attinger, Z. Abbas, A. Bal, N. Rojas, and Z. R. Xu, "Cost of treating diabetic foot ulcers in five different countries", Diabetes/metabolism research and reviews, vol. 28, pp. 107– 111, 2012.

[5] M. Goyal, N. D. Reeves, S. Rajbhandari, and M. H. Yap, "Robust methods for real-time diabetic foot ulcer detection and localization on mobile devices", IEEE journal of biomedical and health informatics, vol. 23, no. 4, pp. 1730–1741, 2018.

[6] F. W. Wagner, "The diabetic foot" Orthopaedics, vol. 10, no. 1, pp. 163-172, 1987.

[7] R. Yang, and Y. Yu, "Artificial convolutional neural network in object detection and semantic segmentation for medical imaging analysis", Front Oncol, vol.11, pp. 6381-6382, 2021.

[8] Z.Q. Zhao, P. Zheng, S. Xu, and X. Wu, "Object detection with deep learning: A review", IEEE Trans Neural Networks Learn System, vol. 30, no. 11, pp. 3212–3232,2019.

[9] L. A. Lavery, D. G. Armstrong, and L. B. Harkless, "Classification of diabetic foot wounds", The Journal of Foot and Ankle Surgery, vol. 35, no. 6, pp. 528-531, 1996.

[10] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6.

[11] Q. Liu and J. Zhao, "The Classification of Diabetic Foot Based on Faster," 2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL), Chongqing, China, 2020, pp. 585-591.

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks", In Advances in neural information processing systems, pp. 91-99, 2015.

[13] M. Goyal, N. D. Reeves, A. K. Davison, S. Rajbhandari, J. Spragg, and M. H. Yap, "DFUNet: Convolutional Neural Networks for Diabetic Foot Ulcer Classification", IEEE Trans. on Emerging Topics in Computational Intelligence, pp. 1-12, 2018.

[14] L. Alzubaidi, M. A. Fadhel, S. R. Oleiwi, O. Al-Shamma, and J. Zhang, “DFU_QUTNet: diabetic foot ulcer classification using novel deep convolutional neural network”, Multimedia Tools Application, vol. 79, no. 21, pp. 15655–15677, 2020.

[15] B. Cassidy, N. D. Reeves, P. Joseph, D. Gillespie, C. O’Shea, S. Rajbhandari, A. G. Maiya, E. Frank, A. Boulton, and D. Armstrong, “Dfuc2020: Analysis towards diabetic foot ulcer detection”, 2020.

[16] A. Oliveira, A. Britto de Carvalho, and D. Dantas, “Faster R-CNN Approach for Diabetic Foot Ulcer Detection”, International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, vol. 4, pp. 677-684, 2021.

[17] E. Santos, F. Santos, J. Dallyson, K. Aires, J.M.R. Tavares, R. Veras, “Diabetic Foot Ulcers Classification using a fine-tuned CNNs Ensemble”, IEEE 35th International Symposium on Computer-Based Medical Systems (CBMS), Shenzen, China, pp. 282–287, July 2022.

[18] P. Xie, Y. Li, B. Deng, C. Du, S. Rui, W. Deng, M. Wang, J. Boey, D.G. Armstrong, and Y. Ma, “An explainable machine learning model for predicting in-hospital amputation rate of patients with diabetic foot ulcer”, International Wound Journal, vol. 19, pp. 910–918, 2022.

[19] A. Khandakar, M.E. Chowdhury, M.B.I. Reaz, S.H.M. Ali, M.A. Hasan, S. Kiranyaz, T. Rahman, R. Alfkey, A.A.A. Bakar, and R.A. Malik, “A machine learning model for early detection of diabetic foot using thermogram images”, Computer in Biology and Medicine, vol. 137, pp. 104838, 2021.

[20] M. Ahsan, S. Naz, R. Ahmad, H. Ehsan and A Sikandar, “A Deep Learning Approach for Diabetic Foot Ulcer Classification and Recognition”, Information, vol. 14, 2023.

[21] Q. Liu and J. Zhao, "The Classification of Diabetic Foot Based on Faster," International Conference on Computer Vision, Image and Deep Learning (CVIDL), Chongqing, China, 2020, pp. 585-591, 2020.

Appendix

DFU localisation and classification model using Faster R-CNN with ResNet-101 architecture.

```
# importing libraries
import os
import collections
import pandas as pd
import numpy as np
import functools
import matplotlib.pyplot as plt
import cv2
from sklearn import preprocessing
import xml.etree.ElementTree as ET

import albumentations as A
from albumentations.pytorch.transforms import ToTensorV2

import torch
import torchvision

from torchvision.models.detection.faster_rcnn import FastRCNNPredictor
from torchvision.models.detection import FasterRCNN
from torchvision.models.detection.rpn import AnchorGenerator

from torch.utils.data import DataLoader, Dataset,subset
from torch.utils.data import SequentialSampler
from sklearn.model_selection import KFold


# mounting drive
from google.colab import drive
drive.mount('/content/drive')
BASE_PATH = "/content/drive/MyDrive/dfu_pvoc"
XML_PATH = os.path.join(BASE_PATH, "Annotations")
IMG_PATH = os.path.join(BASE_PATH, "JPEGImages")
XML_FILES = [os.path.join(XML_PATH, f) for f in os.listdir(XML_PATH)]

# retriving the values from XML file
class XmlParser(object):

    def __init__(self,xml_file):

        self.xml_file = xml_file
        self._root = ET.parse(self.xml_file).getroot()
        self._objects = self._root.findall("object")
        # path to the image file as describe in the xml file
```

```
        self.img_path = os.path.join(IMG_PATH,
self._root.find('filename').text)
        # image id
        self.image_id = self._root.find("filename").text
        # names of the classes contained in the xml file
        self.names = self._get_names()
        # coordinates of the bounding boxes
        self.boxes = self._get_bndbox()

    def parse_xml(self):
        """Parse the xml file returning the root."""

        tree = ET.parse(self.xml_file)
        return tree.getroot()

    def _get_names(self):

        names = []
        for obj in self._objects:
            name = obj.find("name")
            names.append(name.text)

        return np.array(names)

    def _get_bndbox(self):

        boxes = []
        for obj in self._objects:
            coordinates = []
            bndbox = obj.find("bndbox")
            coordinates.append(np.int32(np.float32(bndbox.find("xmin").
text)))
            coordinates.append(np.int32(np.float32(bndbox.find("ymin").
text)))
            coordinates.append(np.int32(np.float32(bndbox.find("xmax").
text)))
            coordinates.append(np.int32(np.float32(bndbox.find("ymax").
text)))
            boxes.append(coordinates)

        return np.array(boxes)

# converting the XML file to Dataframe
def xml_files_to_df(xml_files):

    """Return pandas dataframe from list of XML files."""

    names = []
    boxes = []
```

```
    image_id = []
    xml_path = []
    img_path = []
    for file in xml_files:
        xml = XmlParser(file)
        names.extend(xml.names)
        boxes.extend(xml.boxes)
        image_id.extend([xml.image_id] * len(xml.names))
        xml_path.extend([xml.xml_file] * len(xml.names))
        img_path.extend([xml.img_path] * len(xml.names))
    a = {"image_id": image_id,
         "names": names,
         "boxes": boxes,
         "xml_path":xml_path,
         "img_path":img_path}

    df = pd.DataFrame.from_dict(a, orient='index')
    df = df.transpose()

    return df

df = xml_files_to_df(XML_FILES)
df.head()
# check values for per class
df['names'].value_counts()
```

```
Grade2  206
Grade4  206
Grade1  205
Grade0  112
Grade3  108
Grade5   47
Name: names, dtype: int64
```

```
# remove .jpg extension from image_id
df['img_id'] = df['image_id'].apply(lambda x:x.split('.')).map(lambda
x:x[0])
df.drop(columns=['image_id'], inplace=True)
df.head()

# Make dataset by Dataset Module
class DFUDataset(Dataset):

    def __init__(self, dataframe, image_dir, transforms=None):
        super().__init__()

        self.image_ids = dataframe['img_id'].unique()
        self.df = dataframe
        self.image_dir = image_dir
        self.transforms = transforms
```

```
    def __getitem__(self, index: int):
        image_id = self.image_ids[index]
        records = self.df[self.df['img_id'] == image_id]

        image = cv2.imread(f'{self.image_dir}/{image_id}.jpg',
cv2.IMREAD_COLOR)
        image = cv2.cvtColor(image,
cv2.COLOR_BGR2RGB).astype(np.float32)
        image /= 255.0
        rows, cols = image.shape[:2]

        boxes = records[['xmin', 'ymin', 'xmax', 'ymax']].values


        area = (boxes[:, 3] - boxes[:, 1]) * (boxes[:, 2] - boxes[:,
0])
        area = torch.as_tensor(area, dtype=torch.float32)

        label = records['labels'].values
        labels = torch.as_tensor(label, dtype=torch.int64)

        # suppose all instances are not crowd
        iscrowd = torch.zeros((records.shape[0],), dtype=torch.int64)

        target = {}
        target['boxes'] = boxes
        target['labels'] = labels
        # target['masks'] = None
        target['image_id'] = torch.tensor([index])
        target['area'] = area
        target['iscrowd'] = iscrowd

        if self.transforms:
            sample = {
                'image': image,
                'bboxes': target['boxes'],
                'labels': labels
            }
            sample = self.transforms(**sample)
            image = sample['image']

            target['boxes'] = torch.stack(tuple(map(torch.tensor,
zip(*sample['bboxes'])))).permute(1,0)

            return image, target

    def __len__(self) -> int:
        return self.image_ids.shape[0]
```

```
# 5-fold cross validation
num_folds = 5
indices = list(range(len(dataset)))
kfold = KFold(n_splits=num_folds, shuffle=True)

# Initialize lists to store the data loaders for each fold
train_loaders = []
val_loaders = []

# Split the dataset into folds
for fold, (train_indices, val_indices) in
enumerate(kfold.split(indices)):
    print(f"Fold {fold+1}:")

    # Create subset for training data
    train_subset = Subset(dataset, train_indices)

    # Create subset for validation data
    val_subset = Subset(dataset, val_indices)

    # Create data loaders for training and validation
    train_loader = DataLoader(train_subset, batch_size=32,
shuffle=True)
    val_loader = DataLoader(val_subset, batch_size=32, shuffle=False)
    train_loaders.append(train_loader)
    val_loaders.append(val_loader)

# Data Augmentation
def get_transform_train():
    return A.Compose([
        A.Resize(640, 640, p=1, always_apply=True)
        A.HorizontalFlip(p=0.5),
        A.RandomBrightnessContrast(p=0.2),
        ToTensorV2(p=1.0)
    ], bbox_params={'format':'pascal_voc', 'label_fields': ['labels']})

def get_transform_valid():
    return A.Compose([
        ToTensorV2(p=1.0)
    ], bbox_params={'format': 'pascal_voc', 'label_fields':['labels']})

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Load a model; pre-trained on COCO
from torchvision.models.detection.backbone_utils import
resnet_fpn_backbone
backbone = resnet_fpn_backbone(backbone_name = "resnet101", pretrained
= True)
```

```
num_classes = 7 #(+1 for background)

fasterrcnn = FasterRCNN(backbone = backbone,num_classes=num_classes)
model =
torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)

# get number of input features for the classifier
in_features = model.roi_heads.box_predictor.cls_score.in_features

# replace the pre-trained head with a new one
model.roi_heads.box_predictor = FastRCNNPredictor(in_features,
num_classes)

model.to(device)
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.005, weight_decay=0.0005)
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=5,
gamma=0.1)

# Download modules for model training
!pip install -U
'git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI'

# cloning the vision
!git clone https://github.com/pytorch/vision.git
!cd vision;cp references/detection/utils.py ../;cp
references/detection/transforms.py ../;cp
references/detection/coco_eval.py ../;cp references/detection/engine.py
../;cp references/detection/coco_utils.py ../

from engine import train_one_epoch, evaluate
import utils
```

```
# train the model
num_epochs = 80

for epoch in range(num_epochs):
    # train for one epoch, printing every 10 iterations
    train_one_epoch(model, optimizer, train_data_loader, device, epoch,
print_freq=10)
    # update the learning rate
    lr_scheduler.step()
    # evaluate on the test dataset
    evaluate(model, valid_data_loader, device=device)
```

```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ]
 = 0.678
```

```
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ]
 = 0.788
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ]
 = 0.896
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ]
 = 0.850
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ]
 = 1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ]
 = 0.878
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ]
 = 0.668
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ]
 = 0.743
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ]
 = 0.852
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ]
 = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ]
 = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ]
 = 0.852
```

```
# Saving the model weight
torch.save(model.state_dict(), 'faster_rcnn_state.pth')
```

```
# test model
# load a model;
model =
torchvision.models.detection.fasterrcnn_resnet101(pretrained=False,
pretrained_backbone=False)

WEIGHTS_FILE = "./faster_rcnn_state.pth"

num_classes = 7

# get number of input features for the classifier
in_features = model.roi_heads.box_predictor.cls_score.in_features

# replace the pre-trained head with a new one
model.roi_heads.box_predictor = FastRCNNPredictor(in_features,
num_classes)

# Load the traines weights
model.load_state_dict(torch.load(WEIGHTS_FILE))

model = model.to(device)
```

```
# Function to locate the DFU
```

```
def DFU_locator(img):
    img = cv2.imread(img, cv2.IMREAD_COLOR)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB).astype(np.float32)


    img /= 255.0
    img = torch.from_numpy(img)
    img = img.unsqueeze(0)
    img = img.permute(0,3,1,2)

    model.eval()

    detection_threshold = 0.70

    img = list(im.to(device) for im in img)
    output = model(img)

    for i , im in enumerate(img):
        boxes = output[i]['boxes'].data.cpu().numpy()
        scores = output[i]['scores'].data.cpu().numpy()
        labels = output[i]['labels'].data.cpu().numpy()

        labels = labels[scores >= detection_threshold]
        boxes = boxes[scores >= detection_threshold].astype(np.int32)
        scores = scores[scores >= detection_threshold]

        boxes[:, 2] = boxes[:, 2] - boxes[:, 0]
        boxes[:, 3] = boxes[:, 3] - boxes[:, 1]

    sample = img[0].permute(1,2,0).cpu().numpy()
    sample = np.array(sample)
    boxes = output[0]['boxes'].data.cpu().numpy()
    name = output[0]['labels'].data.cpu().numpy()
    scores = output[0]['scores'].data.cpu().numpy()
    boxes = boxes[scores >= detection_threshold].astype(np.int32)
    names = name.tolist()

    return names, boxes, sample
```

```
pred_path = "/content/drive/MyDrive/dfu_pvoc/JPEGImages"
pred_files = [os.path.join(pred_path,f) for f in os.listdir(pred_path)]

plt.figure(figsize=(20,60))
for i, images in enumerate(pred_files):
    if i > 19:break
    plt.subplot(10,2,i+1)
    names,boxes,sample = obj_detector(images)
    for i,box in enumerate(boxes):
        cv2.rectangle(sample,
```

```
                        (box[0], box[1]),
                        (box[2], box[3]),
                        (0, 220, 0), 6)
        cv2.putText(sample, classes[names[i]], (box[0],box[1]-
5),cv2.FONT_HERSHEY_COMPLEX ,2,(220,0,0),3,cv2.LINE_AA)

    plt.axis('off')
    plt.imshow(sample)
#     plt.savefig('save_image.png', bbox_inches='tight')  # if you want
to save result
```


```
#Confusion Matrix
confusion_matrix = torch.zeros(len(classes), len(classes))
```

```
records = self.df[self.df['img_id'] == image_id]
boxes = torch.tensor(boxes, dtype=torch.float32)
label = records['labels'].values
labels = torch.as_tensor(label, dtype=torch.int64)
    # Run the image through the model
outputs = model([image])

    # Get the predicted labels
predicted_labels = outputs[0]['labels']

    # Update the confusion matrix
for true_label, predicted_label in zip(labels, predicted_labels):
    confusion_matrix[true_label, predicted_label] += 1

# Print the confusion matrix
print(confusion_matrix)
```

```
([[0.842 , 0.0236, 0.0456, 0.0633, 0.0854, 0.11 ],
  [0.012 , 0.8862, 0.0325, 0.02 , 0.0378, 0.0899],
  [0.0503, 0.0521, 0.898 , 0.0567, 0.0421, 0.0702],
  [0.0847, 0.0421, 0.0453, 0.7945, 0.0799, 0.054 ],
  [0.0599, 0.0799, 0.0645, 0.0745, 0.8015, 0.0635],
  [0.0104, 0.0214, 0.0524, 0.0544, 0.0421, 0.7519]])
```

……THE END……