Computers in Biology and Medicine 135 (2021) 104596

Contents lists available at ScienceDirect

Computers in Biology and Medicine

journal homepage: www.elsevier.com/locate/compbiomed

Deep learning in diabetic foot ulcers detection: A comprehensive evaluation

Moi Hoon Yap [a,*], Ryo Hachiuma [b], Azadeh Alavi [c], Raphael Brüngel [d,g], Bill Cassidy [a], Manu Goyal [e], Hongtao Zhu [f], Johannes Rückert [d], Moshe Olshansky [c], Xiao Huang [f], Hideo Saito [b], Saeed Hassanpour [e], Christoph M. Friedrich [d,g], David B. Ascher [c], Anping Song [f], Hiroki Kajita [h], David Gillespie [a], Neil D. Reeves [a], Joseph M. Pappachan [i], Claire O'Shea [j], Eibe Frank [k]

[a] Faculty of Science and Engineering, Manchester Metropolitan University, John Dalton Building, Chester Street, Manchester, M1 5GD, UK
[b] Keio University, Yokohama, Kanagawa, Japan
[c] Baker Heart and Diabetes Institute, 20 Commercial Road, Melbourne, VIC, 3000, Australia
[d] Department of Computer Science, University of Applied Sciences and Arts Dortmund (FH Dortmund), Emil-Figge-Str. 42, 44227 Dortmund, Germany
[e] Department of Biomedical Data Science, Dartmouth College, Hanover, NH, USA
[f] Shanghai University, Shanghai, 200444, China
[g] Institute for Medical Informatics, Biometry and Epidemiology (IMIBE), University Hospital Essen, Hufelandstr. 55, 45122, Essen, Germany
[h] Keio University School of Medicine, Shinanomachi, Tokyo, Japan
[i] Lancashire Teaching Hospitals, Chorley, UK
[j] Waikato Diabetes Health Board, Hamilton, New Zealand
[k] Department of Computer Science, University of Waikato, Hamilton, New Zealand

ARTICLE INFO

Keywords:
Diabetic foot ulcers
Object detection
Machine learning
Deep learning
DFUC2020

ABSTRACT

There has been a substantial amount of research involving computer methods and technology for the detection and recognition of diabetic foot ulcers (DFUs), but there is a lack of systematic comparisons of state-of-the-art deep learning object detection frameworks applied to this problem. DFUC2020 provided participants with a comprehensive dataset consisting of 2,000 images for training and 2,000 images for testing. This paper summarizes the results of DFUC2020 by comparing the deep learning-based algorithms proposed by the winning teams: Faster R–CNN, three variants of Faster R–CNN and an ensemble method; YOLOv3; YOLOv5; EfficientDet; and a new Cascade Attention Network. For each deep learning method, we provide a detailed description of model architecture, parameter settings for training and additional stages including pre-processing, data augmentation and post-processing. We provide a comprehensive evaluation for each method. All the methods required a data augmentation stage to increase the number of images available for training and a post-processing stage to remove false positives. The best performance was obtained from Deformable Convolution, a variant of Faster R–CNN, with a mean average precision (mAP) of 0.6940 and an F1-Score of 0.7434. Finally, we demonstrate that the ensemble method based on different deep learning methods can enhance the F1-Score but not the mAP.

1. Introduction

According to the International Diabetes Federation [39], in 2019 there were approximately 463 million adults with diabetes worldwide. This number is expected to grow to 700 million by 2045. A person with diabetes has a 34% lifetime risk of developing a diabetic foot ulcer (DFU). In other words, 1 in every 3 people with diabetes will develop a DFU in their lifetime [1]. Infection of a DFU frequently leads to limb amputation, causing significant morbidity, psychological distress and reduced quality of life and life expectancy. This research is the first step of a future diabetic foot care project. Periodic monitoring of foot ulcers is important to assess the progress of ulcer healing, which is currently performed manually by clinicians. Many foot clinics take photographs of ulcers during initial evaluation and subsequent reviews for comparison of various stages of ulcer progression to boost the visual memory of clinicians. The current research aims to develop artificial

* Corresponding author.
E-mail addresses: m.yap@mmu.ac.uk, moihoon@gmail.com (M.H. Yap).

https://doi.org/10.1016/j.compbiomed.2021.104596
Received 17 March 2021; Received in revised form 17 June 2021; Accepted 17 June 2021
Available online 23 June 2021

intelligence-based deep learning algorithms for detection of ulcers without direct clinical intervention. This is especially important in the current COVID-19 climate, where social distancing is of paramount importance. Technologies developed to enhance ulcer diagnostics and care plans have the potential to revolutionise diabetic foot care.

Detection tasks can be challenging when taking into account the numerous environmental elements in real-world settings. Examples of some observations include:

- Newly acquired and subtle early stages of ulceration can be easily missed by care personnel during visual assessment due to time constraints
- Low-quality images with poor focus, motion blur, occlusion, inadequate lighting, and backlight are common in wound documentation due to time constraints associated with treatment and documentation, even when performed by trained personnel
- Malformed toenails, deep rhagades, folded amputation scars, and fresh epithelialization are examples for false positive detections that require manual correction, which can be time consuming when documenting DFU
- Very small, very large and curved ulcers are problematic for certain detectors, but are common in typical wound care documentation

It is essential to develop a technological solution capable of transforming current screening practices that has the potential to significantly reduce clinical time burdens.

With the emerging growth of deep learning, automated analysis of DFU has become possible. However, deep learning requires large-scale datasets to achieve results comparable with those of human experts. Currently, medical imaging researchers are working in isolation and the majority of their research is not reproducible. To bridge the gap and to motivate data sharing amongst researchers and clinicians, Yap et al. [50,51] proposed the diabetic foot ulcer challenges. This paper presents an overview of the state-of-the-art computer methods in DFU detection, provides an overview of the publicly available datasets, presents a comprehensive evaluation of the popular object detection frameworks on DFU detection, proposes an ensemble method and Cascade Attention DetNet for DFU detection, and conducts a comprehensive evaluation of the deep learning algorithms trained on the DFUC2020 dataset.

2. Related work

The growing number of reported cases of diabetes has resulted in a corresponding growth in research interest in DFU. Early attempts in training deep learning models in this domain have shown promising results. Previous research [14,16,17] trained models capable of classification, localization and segmentation. These models reported high levels of mean average precision (mAP), sensitivity and specificity in experimental settings. The existing method on localization was trained using Faster R–CNN with Inception v2 and two-tier transfer learning from the Microsoft Common Objects in Context (MS COCO) dataset. However, despite the high scoring performance measures, these models were trained and evaluated on small datasets (<2000 images), therefore the results cannot be regarded as conclusive evidence of their efficacy in real-world settings.

Brown et al. [4] created the MyFootCare mobile app which was designed to encourage patient self-monitoring using diaries, goals and notifications. The app stores a log of patient foot images and is capable of semi-automated segmentation. This novel solution to maintaining foot records utilises a method of automatic photograph capture where the phone is placed on the floor and the patient is guided using voice feedback. However, this particular function of the system was not tested during the actual experiment, so it is not known how well it performed in real-world settings.

Wang et al. [45,46] devised a method of consistent DFU image capture using a box with a glass surface containing mirrors which reflect the image back to a camera or mobile device. Cascaded two-stage support vector classification was used to ascertain the DFU region, followed by a two-stage super-pixel classification technique used for segmentation and feature extraction. Despite being highly novel, this method exhibited a number of limitations, such as risk of infection due to physical contact between wound and capture box. The design of the capture box also limited monitoring to DFU that are present on the plantar surface of the foot. The sample size was also statistically insignificant, with only 35 images from real patients and 30 images of wound moulds.

3. Datasets

The DFU datasets provided by The Manchester Metropolitan University and Lancashire Teaching Hospitals NHS Trust [10,14,15] are digital DFU image datasets with expert annotations. The aim of the publication of this data is to encourage more researchers to work in this domain and to conduct reproducible experiments. There are three types of datasets made publicly available for researchers. The first dataset consists of foot skin patches for wound classification [14]; the second dataset contains regions of interest for infection and ischaemia classification [15]; and the third is the most recently published dataset for DFU detection [10]. The third dataset is the largest dataset to date, and increased usage of this data is the driving force for the organisers of the DFU challenges. The researchers involved in organising the yearly DFU challenges [50,51], in conjunction with the MICCAI conferences, aim to attract wider participation to improve the diagnosis/monitoring of foot ulcers and to raise awareness of diabetes and DFU. There are numerous aspects to take into account in the development of accurate detection algorithms. As is the case with other medical imaging research fields, increasing the number of images is only one of them. The Diabetic Foot Ulcers Grand Challenge (DFUC2020) dataset consists of 2,000 training images, 200 validation images and 2,000 testing images [10,16]. The data consists of 2,496 ulcers in the training set and 2,097 ulcers in the testing set. In an attempt to promote model robustness, some of the images in the testing set do not exhibit DFUs. The details of the dataset are described in Ref. [10]. To improve the performance of the deep learning methods and to reduce computational costs, all images were resized to 640 × 480 pixels.

Since the release of the DFUC2020 training dataset on the 27th April 2020, we received requests from 39 international institutions from 20 countries, as shown in Fig. 1. There are a total of 31 submissions to the challenge from 11 teams. In this paper, we report the top scores from each team and discuss their methods according to the object detection approaches they implemented.

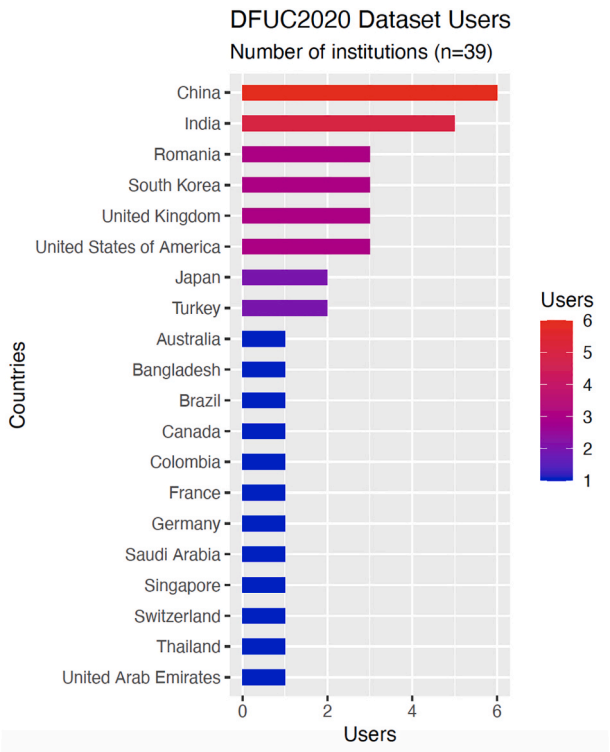

Fig. 1. Summary of DFUC2020 participants across the world, 39 institutions in 20 countries have licensed the dataset for participation in the challenge.

4. DFU detection methods

This section presents a comprehensive description of the DFU detection methods used, grouped according to the popular deep learning object detection algorithms they apply, which include Faster R–CNN, YOLOv3, YOLOv5, and EfficientDet. We also include descriptions of an ensemble method and a new Cascade Attention DetNet (CA-DetNet).

4.1. Faster R–CNN

Faster R–CNN [38] is one of the two-stage object detection models, which generates a sparse set of candidate object locations using a Region Pooling Network (RPN) based on shared feature maps, which then classifies each candidate proposal as the foreground or background class. After extracting shared feature maps with a CNN, the first stage RPN takes shared feature maps as an input and generates a set of bounding box candidate object locations, each with an "objectness" score. The size of each anchor is configured using hyperparameters. Then, the proposals are used in the region of interest pooling layer (RoI pooling) to generate subfeature maps. The subfeature maps are converted to 4,096 dimensional vectors and fed forward into fully connected layers. These layers are then used as a regression network to predict bounding box offsets, with a classification network used to predict the class label of each bounding box proposal.

The RoI pooling layer quantizes a floating-number RoI to the discrete granularity of the feature map. This quantization introduces misalignments between the RoI and the extracted features. Therefore, the model evaluated in this paper employs a RoIAlign layer, which is introduced in Mask R–CNN [18], instead of the RoI pooling layer. This removes the harsh quantization of the RoI pooling layer, properly aligning the extracted features with the input.

Additionally, the Feature Pyramid Network (FPN) [27] is employed as the backbone of the network. FPN uses a top-down architecture with lateral connections to build an in-network feature pyramid from a single-scale input. Faster R–CNN with an FPN backbone extracts RoI features from different levels of the feature pyramid according to their scale, with the remainder of the approach being similar to ResNet. Using a ResNet-FPN backbone for feature extraction with Mask R–CNN gives excellent gains in both accuracy and speed. Specifically, we employ ResNeXt101 [47] with the FPN feature extraction backbone to extract the features.

4.1.1. Data augmentation

In this challenge, the images in the dataset were captured from different viewpoint angles, cameras with different focal lengths and varying levels of blur. Also, the training dataset contains only 2, 000 images, which could be considered small for training deep learning models. Therefore, we employ various data augmentation techniques for robust prediction. Specifically, we employ the following augmentations:

- HSV and RGB: As the lighting conditions vary between dataset images, we apply random RGB and HSV shift to the images. Especially, we randomly add/subtract from 0 to 10 RGB values and 0 to 20 HSV values in the images.
- Blurring: As the dataset contains images captured from different focal lengths, some images are blurred and contain camera noise. Therefore, we apply Gaussian and median blur filters with the filter size set to 3. The filters are applied with the probability of 0.1.
- Affine transformation: As the images are captured from different camera angles, we apply random affine transformations. Specifically, we apply random shift, scaling (0.1) and rotation (90°).
- Brightness: As the images are captured in various environments, we employ brightness and contrast data augmentation. More specifically, we randomly change the brightness and contrast in a scale from 0.1 to 0.3, with probability set to 0.2.

4.1.2. Model training and implementation

For training, we fine-tune a model pretrained on MS-COCO [28]. We employ Stochastic Gradient Descent Optimizer with a momentum of 0.9 and weight decay set to 0.0001. During training, we employ a warm up learning rate scheduling strategy, using lower learning rates in the early stages of training to overcome optimization difficulties. More specifically, we linearly increase the learning rate to 0.01 in the first 500 iterations, then multiply by 0.1 at epochs 6, 12 and 30. We implemented the methods based on the detection repository.[1]

4.1.3. Variants of faster R–CNN

Several papers have proposed variants of Faster R–CNN. In this paper, we implement Faster R–CNN, three variants of Faster R–CNN and ensemble the results. The three variants of Faster R–CNN are as follows:

- Cascade R–CNN [8]: this variant implements a different architecture for the ROI head (the module that predicts the bounding boxes and the category label). Cascade R–CNN builds up a cascade head based on Faster R–CNN [38] to refine detection progressively. Since the proposal boxes are refined by multiple box regression heads, Cascade R–CNN is suitable for more precise localization of objects.
- Deformable Convolution [57]: in this variant, the basic architecture of the network is the same as Faster R–CNN. However, we replace the convolution layer with a deformable convolution layer [56] at the second, third and fourth ResNeXt blocks of the feature extractor. The deformable convolution adds 2D offsets to the regular grid sampling locations in the standard convolution, enabling free-form deformation of the sampling grid. The offsets are learned from the feature maps, via additional convolutional layers. Thus, the deformation is conditioned on the input features in a local, dense and adaptive manner.

[1] https://github.com/open-mmlab/mmdetection.

- Prime Sample Attention [9] (PISA): PISA is motivated by two considerations: samples should not be treated as independent and equally important, and the classification and localization are correlated. Thus, a ranking strategy is employed that places the positive samples with highest IoUs around each object, and the negative samples with highest scores in each cluster at the top of the ranked list. This directs the focus of the training process via a simple re-weighting scheme. It also employs a classification-aware regression loss to jointly optimize the classification and regression branches.

4.1.4. Post-processing

At test time, we employ a test-time augmentation scheme: we augment the test image by applying two resolutions (640 × 480 and 800 × 600), and we also flip the image. As a result, we augment a single image to four images and merge the predictions obtained for the four images. We employ soft NMS (non maximum suppression) [3] with a confidence threshold of 0.5 as the post-processing of predicted bounding boxes.

4.1.5. Ensemble method

Combining predictions from different models can improve generalization and usually yields more accurate results compared to a single model. During the post-processing stage for Faster R-CNNs, we employ soft NMS [3] to select the predicted bounding boxes for each method. Such methods work well on a single model, but they only select the boxes and cannot produce averaged localization of predictions combined from various models effectively. Therefore, after predicting the bounding boxes for each method, we ensemble these predicted bounding boxes using Weighted Boxes Fusion [40]. Unlike NMS-based methods that simply exclude part of the predicted bounding boxes, the Weighted Boxes Fusion algorithm uses the confidence scores of all proposed bounding boxes to form the average boxes. The reader is referred to Ref. [40] for further details of the algorithm. We ensemble four models (pure Faster R–CNN, Cascade R–CNN, Faster R–CNN with Deformable Convolution and Faster R–CNN with Prime Sample Attention model). We set equal weights when fusing the predicted bounding boxes of each model.

4.2. YOLO

You-Only-Look-Once (YOLO) [35] is a unified, real-time object detection algorithm that reformulates the object detection task to a single regression problem. YOLO employs a single neural network architecture to predict bounding boxes and class probabilities directly from full images. Hence, when compared to Faster R–CNN [38], YOLO provides faster detection.

Over time, improvements of YOLO were implemented and released as distinct and independent software packages by the originators [35–37]. As a result of increased publicity and popularity, a model zoo containing further YOLO adaptations emerged. Subsequently, further maintainers continued to improve the DarkNet[2]-based versions, and [2] created ports for other machine learning libraries such as PyTorch[3] [32].

In this paper, two approaches are selected for DFU detection using the DFUC2020 dataset: YOLOv3 and YOLOv5. We discuss the networks and present descriptions of our implementation in the following subsections.

4.2.1. YOLOv3

YOLOv3 [37] was developed as an improved version of YOLOv2 [36]. It employs multi-scale schema, predicting bounding boxes at different scales. This allows YOLOv3 to be more effective for detecting smaller targets when compared to YOLOv2.

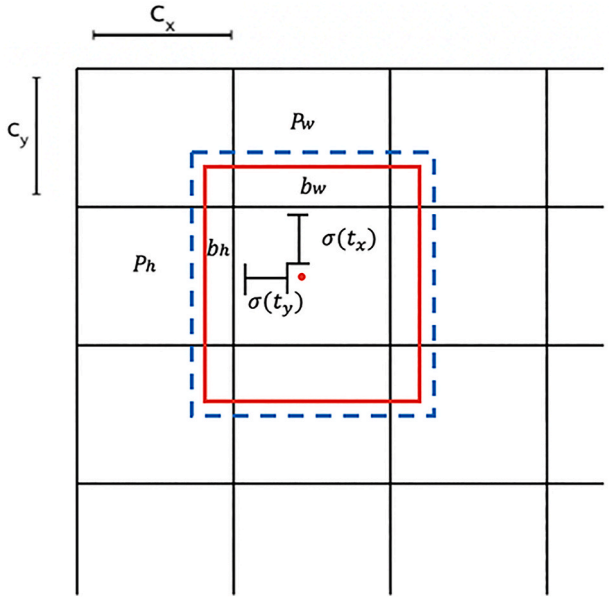

Fig. 2. Illustration of bounding boxes, dimension priors and location prediction. The red dot represents the bounding box center with the coordinates (b_x, b_y). Adapted from Ref. [37].

YOLOv3 uses dimension clusters as anchor boxes in order to predict bounding boxes around the desired objects in given images. Logistic regression is used to predict the objectness score for a given bounding box. Specifically, as illustrated in Fig. 2, the algorithm predicts the four coordinates of the bounding box (t_x, t_y, t_h, t_w) as in Equation (1), Equation (2), Equation (3), and Equation (4) [37].

$$b_x = \sigma(t_x) + c_x \tag{1}$$

$$b_y = \sigma(t_y) + c_y \tag{2}$$

$$b_h = p_w e^{t_w} \tag{3}$$

$$b_w = p_h e^{t_h} \tag{4}$$

Table 1
The architecture of DarkNet-53 used in YOLOv3. Adapted from Ref. [37].

Type		Filters	Size
Convolutional		32	3 × 3
Convolutional		64	3 × 3/2
Convolutional		32	1×1
Convolutional	1 ×	64	3 ×3
Residual			
Convolutional		128	3 ×3/2
Convolutional		64	1 ×1
Convolutional	2 ×	128	3 × 3
Residual			
Convolutional		256	3 × 3/2
Convolutional		128	1 × 1
Convolutional	8 ×	256	3 × 3
Residual			
Convolutional		512	3 × 3/2
Convolutional		256	1 × 1
Convolutional	8x	512	3 × 3
Residual			
Convolutional		1024	3 × 3/2
Convolutional		512	1 × 1
Convolutional	4 ×	1024	3 × 3
Residual			
Avgpool Connected Softmax		Global 1000	

[2] DarkNet GitHub repository: https://github.com/pjreddie/darknet (accessed 2020-08-29).
[3] PyTorch website: https://pytorch.org/(accessed 2020-08-29).

where (c_x, y_y) are offsets from the top left corner of the image, and (p_w, p_h) are bounding box prior height and weight. The k-means clustering algorithm is used to determine bounding box priors, while the sum of squared errors is used for training the network. Let $\widehat{t_*}$ be the ground truth for some coordinate prediction, and t_* be the network prediction during training. Then, the gradient is $\widehat{t_*} - t_*$.

4.3. Model pipeline

The backbone of YOLOv3 is a hybrid model called Darknet-53 (as shown in Table 1), which is used for feature extraction. As the name indicates, DarkNet-53 is made of 53 convolutional layers that also take advantage of shortcut connections.

As the detection algorithm is required to detect only one type of object, the complexity of the problem is reduced from multi-class detection to single object detection. Hence, for the purpose of detecting diabetic foot ulcers, we have employed a simplified version of YOLOv3.

4.3.1. Training

We employ transfer learning by using the pre-trained DarkNet weights which are provided by Ref. [37]. Then, we train our detector in 2 steps, using the following settings: Adam optimizer with learning rate 1e-3, number of epochs = 100, batch size = 32 and using 20% of the data for validation.

First, we start by freezing the top DarkNet-53 layers and train the algorithm with the above settings. Then, we retrain the entire network to improve performance. Similar to the original YOLOv3, our trained network extracts features from 3 different pre-defined scales, which is a similar concept to feature pyramid networks [27]. We then use the trained network for detecting diabetic foot ulcers in blind test images.

4.3.2. Post-processing

As observed from Fig. 3, in rare cases, the resulting algorithm may produce double detections or false positives. To reduce such examples, we include a post-processing stage.

Our post-processing steps consist of two stages. First, we identify double detections by flagging the detected bounding boxes with more than 80% overlap. Among the overlapping detected boxes we only keep the box with the highest confidence result. Finally, we further post-process the results by removing any detection with a confidence score <0.3, with the aim of reducing the rate of false positive detections.

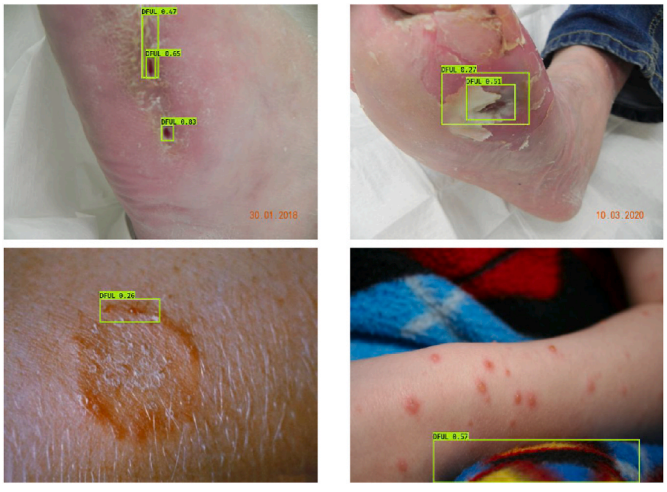

Fig. 3. Illustration of two types of false positives. The top row shows false-positive examples from double detections; the bottom row shows single false-positive detections caused by a non-DFU condition and a background object.

4.3.3. YOLOv5

YOLOv5 was first published on GitHub[4] in May 2020 in v1.0 [21]. The maintainer is already well known for a YOLOv3 [37] port for PyTorch[5] [22]. The maintainer named the network YOLOv5 to avoid naming conflicts due to the prior release of YOLOv4 [2]. However, YOLOv5 is not to be confused with a descendent of the original DarkNet-based[6] YOLO-series. A scientific paper reporting on the improvements in YOLOv5 has not yet been published, but is currently pending.[7] YOLOv5 is currently under active development, with the latest version being v5.0 [23] at the time of writing.

New features and improvements in YOLOv5 are mainly focused on the incorporation of the state-of-the-art for deep learning networks, such as activation functions and data augmentation. These were partly adopted from YOLOv4[8] such as the CSPNet backbone [43] with other elements originating from prior YOLOv4 contributions by the YOLOv5 maintainer. One of the most notable data augmentation aspects is the mosaic loader in which four images are altered and combined to form a new image. This allows detection of objects outside of their normal context and at smaller sizes, which reduces the need for large mini-batch sizes. YOLOv5 reports high inference speed and small model sizes, allowing a convenient translation to mobile use cases via model export.

The approach on DFU detection via YOLOv5 described in the following is based on the early version v1.0[9] [21] commit `a1c8406`[10] from 14th July 2020 that still exhibited several issues.

4.3.4. Pre-processing

Initially, image data of the training dataset was analyzed via AntiDupl[11] in version `2.3.10` to identify duplicate images, yielding a set of 39 pair findings. A spatial analysis of duplicate pair annotation data was performed, utilizing the R language[12] [34] in version `4.0.1` and the Simple Features for R (sf) package[13] [33] in version `0.9-2`. Originally, none of the duplicate pair images showed bounding box intersections by themselves. After joining duplicate pair annotations, several intersections were detected with a maximum of two involved bounding boxes. These represented different annotations of the same wound in two duplicate images, now joint in one image. To resolve these, each pair of intersecting bounding boxes BBox_1 and BBox_2 was merged into a single bounding box $\widehat{\mathrm{BBox}}$ by using their outer boundaries, as shown in Equ. 5.

$$\widehat{\mathrm{BBox}} \begin{cases} \widehat{\mathrm{xmin}} &= \min(\mathrm{xmin}_1, \mathrm{xmin}_2) \\ \widehat{\mathrm{ymin}} &= \min(\mathrm{ymin}_1, \mathrm{ymin}_2) \\ \widehat{\mathrm{xmax}} &= \max(\mathrm{xmax}_1, \mathrm{xmax}_2) \\ \widehat{\mathrm{ymax}} &= \max(\mathrm{ymax}_1, \mathrm{ymax}_2) \end{cases} \tag{5}$$

The applied duplicate cleansing and annotation merging strategy resulted in $n = 1,961$ images with $k = 2,453$ annotations in the cleansed

[4] YOLOv5 GitHub repository: https://github.com/ultralytics/yolov5/releases/tag/v1.0 (accessed 2021-04-28).
[5] Ultralytics' YOLOv3 GitHub repository: https://github.com/ultralytics/yolov3 (accessed 2020-08-29).
[6] YOLOv4 GitHub repository: https://github.com/AlexeyAB/darknet (accessed 2020-08-29).
[7] YOLOv5 question on scientific paper: https://github.com/ultralytics/yolov5/issues/2847 (accessed 2021-04-28).
[8] YOLOv5 question on adopted YOLOv4 features: https://github.com/ultralytics/yolov5/issues/370 (accessed 2021-04-28).
[9] YOLOv5 v1.0: https://github.com/ultralytics/yolov5/releases/tag/v1.0 (accessed 2020-09-12).
[10] YOLOv5 GitHub commit `a1c8406`: https://github.com/ultralytics/yolov5/commit/a1c8406 (accessed 2020-08-29).
[11] AntiDupl GitHub repository: https://github.com/ermig1979/AntiDupl (accessed 2020-08-29).
[12] R language website: https://www.r-project.org/(accessed 2020-08-29).
[13] Simple Features for R (sf) GitHub repository: https://github.com/r-spatial/sf (accessed 2020-08-29).

training dataset. Boundaries of merged bounding boxes were checked for consistency. Finally, annotation data was converted to the resolution-independent format used by YOLO implementations.

Reviewing image data of all dataset parts (training, validation and test), showed pronounced compression artifacts and color noise due to a high compression rate and downscaling to a low resolution. As both compression artifacts and color noise had derogatory effects on the detection performance, images were enhanced using a fast implementation of the non-local means algorithm [6] for color images, utilizing the Python language[14] in version `3.6.9` with the OpenCV on

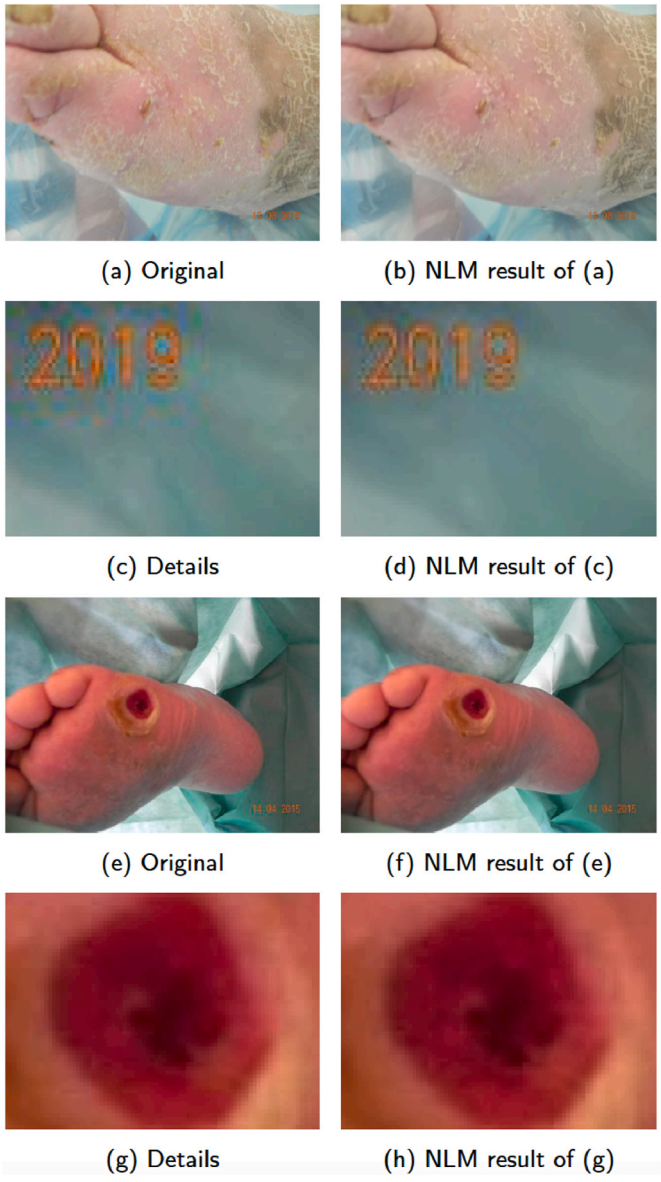

Fig. 4. Effects of the non-local means (NLM) algorithm are shown for two example images (a) and (e) from the training dataset in (b) and (f). At a macroscopic level the changes are not obvious. At a detail level borders of compression artifacts on homogeneous areas and color noise of (c) are visibly reduced in (d). Vague textures of (g) are also more pronounced in (h).

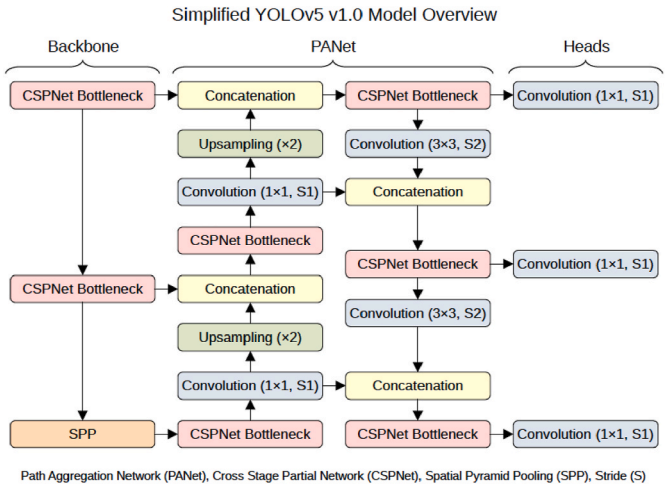

Fig. 5. The architecture of YOLOv5 v1.0, adapted from community-driven discussions on the model representation (https://github.com/ultralytics/yolov5/issues/280 (accessed 2021-04-28)), verified by the maintainer.

Wheels (opencv-python)[15] package in version `4.2.0.34`. The algorithm parameters were set to `h` = 1 (luminance component filter strength) and `hColor` = 1 (color component filter strength) with `templateWindowSize` = 7 (template patch size in pixels) and `searchWindowSize` = 21 (search window size in pixels).

Resulting images show less definitive compression artifact borders and notably reduced color noise. Some textures are also more pronounced. Examples of results at a macroscopic and a detail level are shown in Fig. 4.

4.3.5. Data Augmentation

YOLOv5 in v1.0 implements three sets of data augmentation techniques. The first set comprises alterations of colorspace components (hue, saturation, value), the second set comprises geometric distortions (random scaling, rotation, translation and shearing), and the third set is represented by the mosaic loading of images.

A normalized fraction of 0.014 images received hue augmentation, 0.68 received saturation augmentation and 0.36 received value augmentation. Scaling was applied in a normalized range of ± 0.5. Rotation, translation and shearing were disabled. Settings for colorspace component alterations and geometric distortions are definitions for distributions, generated during runtime by a random sampler for the augmentation function.[16] Using this approach, no image is presented more than once during training.

Mosaic data augmentation is comparable to CutMix, but takes four images instead of two and does not overlap them. Image parts are placed as quadrants in a new image with random ratios, thereby allowing the model to detect objects in different contexts and at different sizes. This reduces the need for large mini-batch sizes. However, the mosaic loader had to be disabled in the presented approach due to a bug, leading to invalid bounding boxes in resulting predictions.

4.3.6. Model

YOLOv5 includes four different models ranging from the smallest YOLOv5s with 7.5 million parameters (plain 7 MB, COCO pre-trained 14 MB) and 140 layers to the largest YOLOv5x with 89 million parameters and 284 layers (plain 85 MB, COCO pre-trained 170 MB). In the approach considered in this paper, the pre-trained YOLOv5x model is

[14] Python language website: https://www.python.org/(accessed 2020-08-29).

[15] OpenCV on Wheels GitHub repository: https://github.com/skvark/opencv-python (accessed 2020-08-29).

[16] YOLOv5 question on data augmentation: https://github.com/ultralytics/yolov5/issues/2164 (accessed 2021-04-28).

used. The general YOLOv5 v1.0 architecture is displayed in Fig. 5. Different model sizes s, m, l and x vary in set depth and width factors for the model and its layer channels, which are 1.33 and 1.25 for the YOLOv5x model.

The YOLOv5x model uses a detector that consists of a Cross Stage Partial Network (CSPNet) [43] backbone trained on MS COCO [28], and a model head using a Path Aggregation Network (PANet) [29] for instance segmentation. The backbone further incorporates a Spatial Pyramid Pooling (SPP) network [19], which allows for dynamic input image size and is robust against object deformations.

4.3.7. Training

The hardware setup used for the experiment comprised a single NVIDIA® V100[17] tensor core graphics processing unit (GPU) with 16 GB memory as part of an NVIDIA® DGX-1[18] supercomputer for deep learning. YOLOv5 was set up using a provided Docker container,[19] executed via Nvidia-Docker[20] in version `19.03.5`.

Training was organized in two stages: Initial training and self-training. The initial training stage uses the original available training data to train a model. The self-training approach, also called pseudo-labelling, extends available training data by inferring detections on images for which originally no annotation data is available [25]. This is realized using the model resulting from the initial training stage; yielded detections are then used as pseudo-annotation data. Resuming the initial training in the self-training stage with the extended training data generalizes detection capabilities of the model.

A five-fold cross-validation was performed for each training stage to approximate training optima. Both stages used the default set of hyperparameters (including parameters related to the data augmentation procedures): `optimizer` = SGD, `lr0` = 0.01, `momentum` = 0.937, `weight_decay` = 0.0005, `giou` = 0.05, `cls` = 0.58, `cls_pw` = 1.0, `obj` = 1.0, `obj_pw` = 1.0, `iou_t` = 0.2, `anchor_t` = 4.0, `fl_gamma` = 0.0, `hsv_h` = 0.014, `hsv_s` = 0.68, `hsv_v` = 0.36, `degrees` = 0.0, `translate` = 0.0, `scale` = 0.5, and `shear` = 0.0. A default seed value of 0 was used for model initialization. Both training stages were performed in the single-class training mode, with mosaic data augmentation deactivated due to issues regarding bounding box positioning in the current YOLOv5 implementation.

During the initial training stage, a base model was trained on the pre-processed training dataset for 60 epochs with a batch size of 30. This base model was initialized with weights from the MS COCO pre-trained YOLOv5x model. For the self-training approach, the base model was then used to create the extended training dataset for self-training. Pseudo-annotation data was inferred for the validation and test datasets, using the best-performing epoch automatically saved at epoch 58. The resulting extended training dataset contained 4,161 images, of which 3,963 included 4,638 wound annotations.

During the self-training stage, the base model training was resumed at its latest epoch, but trained further on the extended training dataset with a batch size of 20. Three final training states were created: (1) after an additional 30 epochs, (2) after an additional 40 epochs, and (3) after an additional 60 epochs of self-training (referred to as `E60_SELF90`, `E60_SELF100`, and `E60_SELF120`).

4.3.8. Post-processing

The minimum confidence threshold for detection was set to 0.70, so that only highly certain predictions were exported. This applies for pseudo-annotation data of the extended training dataset created for self-training as well as for the final predictions.

Predictions for our experiments were inferred via the final training states `E60_SELF90`, `E60_SELF100`, and `E60_SELF120`, using the best epochs 88, 96 and 118 respectively. An additional experiment was conducted based on the training state `E60_SELF100` involving the built-in test-time augmentation and non-maxima suppression (NMS) features of YOLOv5 for inference.

Test-time augmentation (TTA) is a data augmentation method which involves several augmented instances of an image that are presented to the model. For each instance, predictions are made which provide an ensemble of instance predictions. This can enable a model to detect objects it may not be able to detect in a "clean" image. However, TTA may also cause multiple distinct detections for the same object that can harm evaluation scores. To tackle these, NMS was applied to collapse multiple intersecting detections into a single bounding box. The intersection over union (IoU) threshold was set to IoU $\geq$ 0.30, as images with multiple wounds a distinct spatial demarcation was usually given. Thus, the risk of interfering detections of different wounds was low.

4.4. EfficientDet

The EfficientDet architecture [42] is an object detection network created by the Google Brain team, and utilises the EfficientNet ConvNet [41] classification network as its backbone. EfficientDet uses feature fusion techniques in the form of a bidirectional feature pyramid network (BiFPN) which combines representations of input images at different resolutions. BiFPN adds weights to input features which enables the network to learn the importance of each feature. The outputs from the BiFPN are then used to predict the class of the detected object and to generate bounding boxes using bounding box regression. The main feature of EfficientDet is its ability to utilise compound scaling, which allows all parts of the network to scale in accordance to the target hardware being used for training and inference [42]. An overview of the EfficientDet architecture is shown in Fig. 6.

4.4.1. Pre-processing

The dataset was captured with different types of camera devices under various lighting conditions. To counter variations in noise and lighting found in the dataset images, the Shades of Gray (SoG) color constancy algorithm was used [31]. Examples of pre-processed DFU images using SoG are shown in Fig. 7.

4.5. Data augmentation

Data Augmentation techniques have been proven to be an important tool in improving the performance of deep learning algorithms for various computer vision tasks [13,49]. For the application of EfficientDet, we augmented the training data by applying identical transformations to the images and associated bounding boxes for DFU detection. Random rotation and shear transformations were used to augment the DFUC2020 dataset. Shearing involves the displacement of the image at its corners, resulting in a skewed or deformed output. Examples of these types of data augmentation are shown in Fig. 8.

4.5.1. Model

EfficientDet algorithms achieved state-of-the-art accuracy on the popular MS-COCO [28] object detection dataset. EfficientDet pre-trained weights are classed from D0 to D7, with D0 having the fewest number of parameters and D7 having the highest number of parameters. Tests on the MS-COCO dataset indicate that training using weights with more parameters results in better network accuracy. However, this comes at the cost of significantly increased training time. Given that the DFUC2020 dataset images were resized to 640 $\times$ 480, we selected the EfficientDet-D1 pre-trained weights for DFU detection [12].

[17] NVIDIA® V100: https://www.nvidia.com/en-us/data-center/v100/ (accessed 2020-08-30).
[18] NVIDIA® DGX-1: https://www.nvidia.com/en-us/data-center/dgx-1/ (accessed 2020-08-30).
[19] YOLOv5 Docker Hub container: https://hub.docker.com/r/ultralytics/yolov5 (accessed 2020-08-30).
[20] Nvidia-Docker GitHub repository: https://github.com/NVIDIA/nvidia-docker (accessed 2020-08-30).

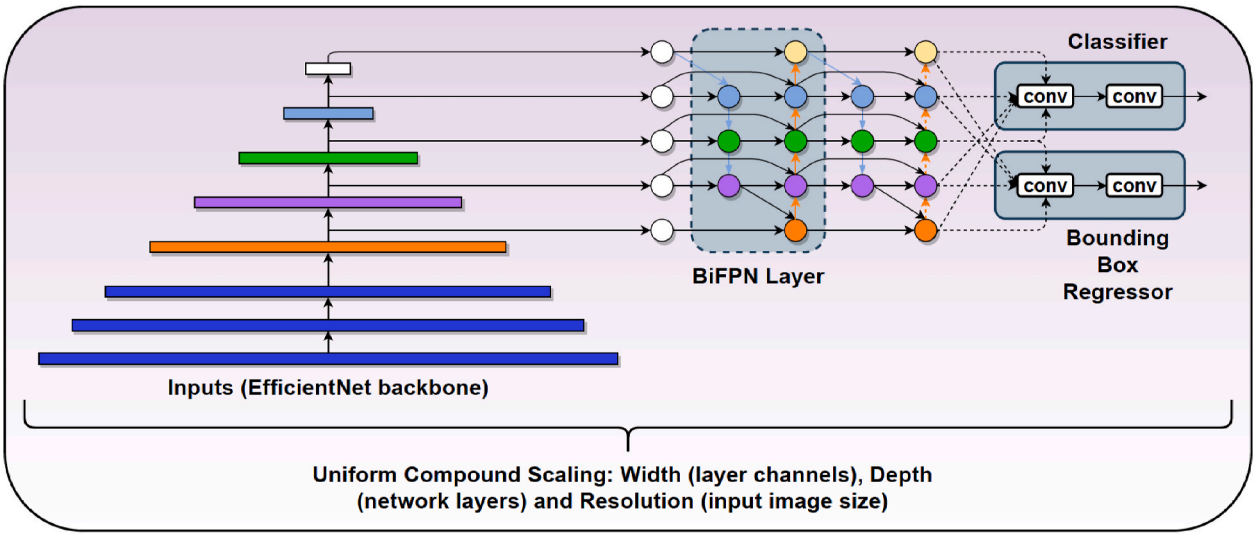

Fig. 6. The architecture of EfficientDet. Adapted from Ref. [42].

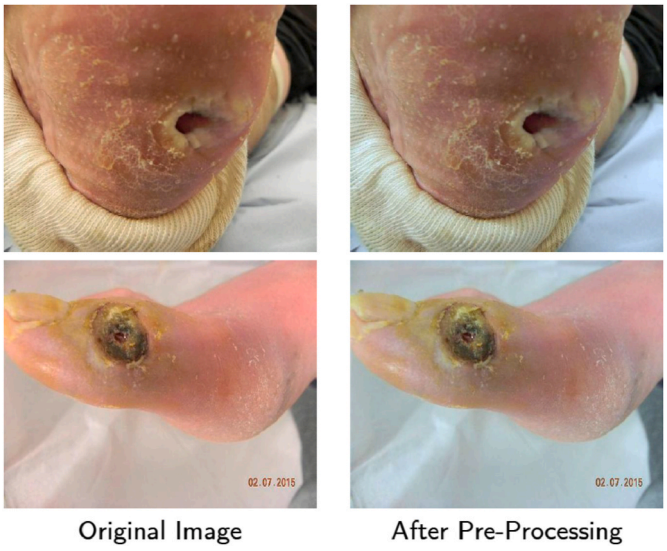

Fig. 7. Shades of gray algorithm for pre-processing of the DFUC2020 dataset: The left column shows the original images; the right column shows results of pre-processed images.

4.5.2. Training

We trained the EfficientDet-D1 method on an NVIDIA Quadro RTX 8000 GPU (48 GB) with a batch-size of 16, SGS optimizer with a learning rate of 0.00005, momentum of 0.9 and number of epochs set to 50. We used the validation accuracy with early stopping to select the final model for inference.

4.5.3. Post-processing

We further refined the EfficientDet architecture with a score threshold of 0.5 and removed overlapping bounding boxes to minimize the number of false positives. The scores were compared between the overlapping bounding boxes, with the bounding box with the highest score used as the final output.

4.6. Cascade Attention DetNet

4.6.1. Data augmentation

Given that the DFUC2020 dataset has only 2,000 images for training, we use several data augmentation methods to complement the dataset in order to avoid over-fitting when training models. A more generalized model can be obtained through data augmentation in order to make it adapt to the complex clinical environment. We use common data augmentation methods including horizontal and vertical image flipping, random noise and a central scaling method (which scales with ground truth as the center). Additionally, we increase the number of training images by using the visually coherent image mixup method [52]. The original purpose of this method is to overcome the problem of disturbance rejection. Since Zhang et al. [53] introduced this method into object detection, many researchers have used it in data augmentation to enhance network robustness. The principle of this algorithm involves the random selection of two sample images which are then used to generate a new sample image according to Equation (6) and Equation (7).

$$\hat{x} = \lambda x_i + (1-\lambda)x_j \tag{6}$$

$$\hat{y} = \lambda y_i + (1-\lambda)y_j \tag{7}$$

where (x_i, y_i), (x_j, y_j) are the points of two sample images and $\lambda \in [0,1]$, which is randomly generated by the Beta(alpha, alpha) distribution. The new sample $(\hat{x}, \hat{y})$ is used for training. As shown in Fig. 9, two images of DFU are mixed in a certain ratio. We use Beta(1.5,1.5) for the images' synthesis.

DFU detection can be challenging in complex environments, such as clinical settings, due to the large number of objects that might be present. To improve the accuracy of detection, we use the mobile fuzzy method for data augmentation, as shown in Fig. 10.

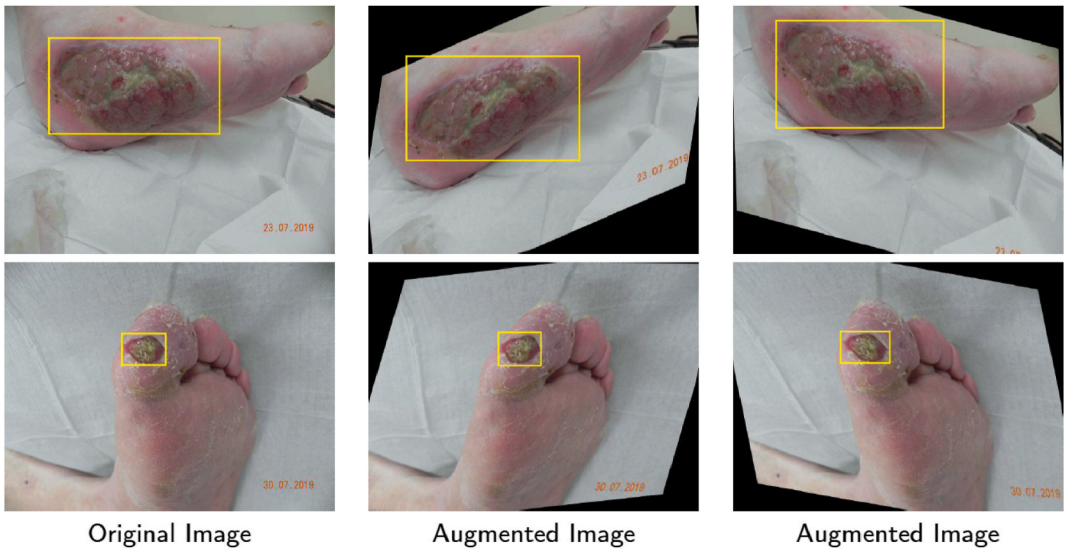

Fig. 8. Bounding box data augmentation on the DFUC2020 dataset.

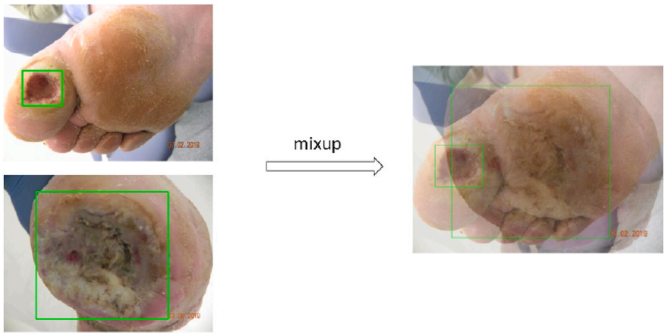

Fig. 9. The effect of the visually coherent image mixup method.

Fig. 10. The effect of the mobile fuzzy method. (a) shows the original image, and (b) shows the image after blurring with the mobile fuzzy method.

4.6.2. Model

The Cascade R–CNN [7] is the first cascaded object detection model. Due to the superior performance of the cascade structure, it is widely used in the field of object detection [54]. We use the cascade structure in conjunction with DetNet [26], which is designed to address the problems incurred by down-sampling repeatedly, as such a process reduces the accuracy of positioning. DetNet makes full use of dilated convolutions to enhance the receptive field instead of down-sampling repeatedly. The overall framework of our method, Cascade Attention DetNet (CA-DetNet) is shown in Fig. 11.

The detection of DFU is different from common object detection tasks. For common object detection tasks, objects can appear anywhere in the image. For the detection of DFU, the wounds can only appear on the foot, which is a good fit for applying an attention mechanism, which we added into the DetNet by adopting the mask branch of the Residual Attention Network [44].

The Attention DetNet (A-DetNet) is composed of 6 stages. The first stage consists of a 7×7 convolution layer (with a stride of 2) and a max-pooling layer. The second, third and fourth stages contain an A-Resbody, with the fifth and sixth stages containing an A-Detbody. The A-Resbody and A-Detbody are similar to those in the original DetNet. The difference between A-DetNet and the original DetNet is the addition of an attention branch into the Resbody and Detbody. The attention branch is similar to the mask branch of the Residual Attention Network, while we take other parts from the original Resbody or Detbody as the trunk. The attention branch of the Resbody is comprised of two zoom structures, which consist of a max-pooling layer and an up-sampling layer, followed by two 1×1 convolution layers activated by sigmoid functions.

Given that the five times down-sampling results in a feature map that is too small to recover the original size by upsampling, we only add one zoom structure into the attention branch of the A-Detbody. The feature map from the trunk is multiplied by the mask from the attention branch. To avoid consuming the value of the feature and breaking the identity mapping, we refer to the Residual Attention Network and add one to the mask.

4.6.3. Training

For the cascade structure, we set the total number of cascade stages to 3, with the intersect over union (IoU) threshold set to 0.5, 0.6 and 0.7 for each of the three stages. During training we use DetNet pre-trained model, which has been trained on the ImageNet dataset, to accelerate model convergence. We train on a single GPU (NVIDIA Tesla P100) for 60 epochs, with a batch size of 4 and a learning rate of 0.001. The learning rate decreases 10 times at the 10th epoch, and then decreases another 10 times at the 20th epoch. We optimize the model with the Adam optimizer.

4.6.4. Post-processing

Noise from the external environment can lead to many low confidence bounding boxes. These bounding boxes will reduce the performance of the detector, so we adopt a special threshold suppression method to suppress bounding boxes with low thresholds except when the detector detects only one bounding box. We set the threshold to 0.5.

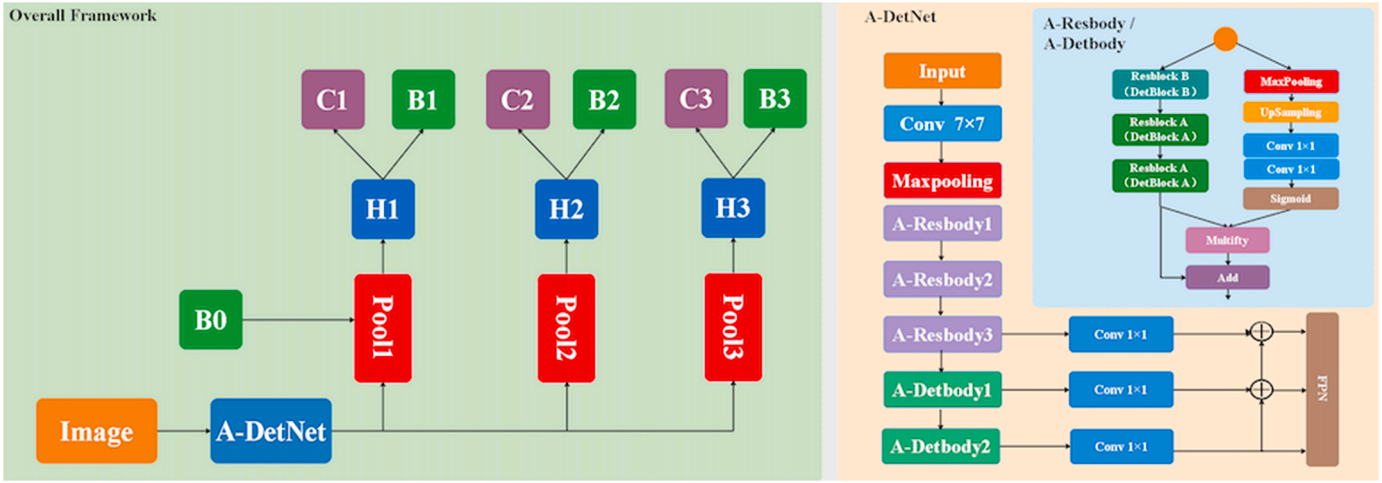

Fig. 11. The architecture of CA-DetNet. "Image" is an input image. "A-DetNet" is a backbone network. "Pool" represents region-wise feature extraction. "H" is a network head. "B" is a bounding box and "C" represents classification. "B0" is the proposal in all architectures. The structure of the A-DetNet is based on the DetNet. The attention mechanism is applied in Resbody and Detbody. Different bottleneck blocks in the Detbody or Resbody are similar to those in the DetNet.

Table 2
Faster R–CNN. The first row shows the results of pure Faster R–CNN, the second row shows the results of Cascade R–CNN, the third row shows the results of Faster R–CNN with Deformable Convolution v2, the fourth row shows the results of Faster R–CNN with Prime Sample Attention, and the last row shows the results of the ensemble method.

Method	TP	FP	Recall	Precision	F1-Score	mAP
Faster	1512	683	0.7210	0.6888	0.7046	0.6338
Cascade	1483	649	0.7072	0.6956	0.7014	0.6309
Deform	**1612**	628	**0.7687**	0.7196	**0.7434**	**0.6940**
PISA	1495	444	0.7129	0.7710	0.7408	0.6518
Ensemble	1447	**394**	0.6900	**0.7860**	0.7349	0.6353

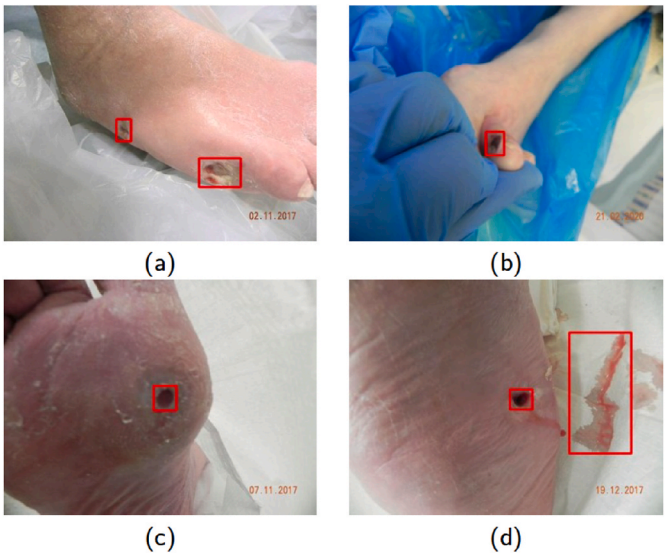

Fig. 12. The qualitative results of Faster R–CNN with Deformable Convolution, which shows the best performance among Faster R–CNN based methods. It is noted that the network is able to detect small ulcers as shown in (a),(b) and (c). An example of a FP generated by the network is shown in (d).

5. Results and analysis

We report and analyse the results obtained using the methods described above. The evaluation metrics are the number of true positives (TP), the number of false positives (FP), recall, precision, F1-Score and mAP, as described in the diabetic foot ulcer challenge 2020 [10]. For the common object detection task, mAP is used as the main evaluation metric. However, in this DFU task, miss-detection (a false negative) has potentially severe implications as it may affect the quality of life of patients. An incorrect detection (a false positive) could increase the financial burden on health services. Therefore, we regard F1-Score as equally important as mAP for performance evaluation.

5.1. Faster R–CNN

Table 2 summarizes the quantitative results of pure Faster R–CNN, its variants, and the final ensemble model. From the table, the performance of pure Faster R–CNN is on par with Cascade R–CNN. In contrast, employing the Deformable convolution or PISA module significantly improves the performance. After we ensemble the model, we reduce FP substantially, with a reduction in TP also observed. Although the ensemble method improves the precision of DFU detection, it does not improve the overall score. Therefore, the best result is achieved by Deformable Faster R–CNN, with a mAP of 0.6940 and F1-Score of 0.7434.

The qualitative results of Faster R–CNN with Deformable Convolution is summarized in Fig. 12. It can be seen that our model successfully detected the wounds in the images, even in cases with small wound sizes (top-left, bottom-left and bottom-right images) or the images are blurred (top-right image). However, we observed the miss-detection as in the bottom-right image. In this image, the background texture of the blood was incorrectly detected as a DFU. To improve prediction accuracy, the training data should be captured in various environments so that the network is better able to discern between DFU and background objects.

5.2. YOLOv3

Table 3 shows the final results of the proposed YOLOv3 method on the testing dataset. The results are reported for two different batch sizes, with and without post-processing.

As the results indicate, using a batch size of 50 leads to a better overall performance compared to using a batch size of 32. It also demonstrates that removing the overlaps leads to an improvement in both F1-score and Precision, while resulting in slight decreases to both mAP and Recall. As the gain overpowers the loss, we conclude that removing overlaps results in better overall performance.

While removing the detections with less than 0.3 confidence results

Table 3
YOLOv3: Results of different settings, post-processing and adding extra copyright free foot images. B50 and B32: compares the performance of the method with batch size 50 and 32. Overlap-Removed: indicates the performance of the method with overlap removal post processing. conf0.3: shows the impact of ignoring predictions with <0.3 confidence. Extra: demonstrates the effect on performance of adding extra images of healthy feet.

Method	Settings			Metrics					
	Base	Coefficient	Overlap-Removed	TP	FP	Recall	Precision	F1-Score	mAP
B50	50	0	×	**1572**	676	**0.7496**	0.6993	0.7236	**0.6560**
B50_Overlap	50	0	✓	1553	618	0.7406	0.7153	0.7277	0.6500
B32	32	0	×	1452	605	0.6929	0.7060	0.6994	0.6053
B32_Overlap	32	0	✓	1433	551	0.6834	0.7223	0.7023	0.5998
B32_Overlap_conf	32	0.3	✓	1386	**490**	0.6609	**0.7388**	0.6977	0.5835
B50_Exact	50	0	×	1563	616	0.7454	0.7173	0.7311	0.6548
B50_Overlap_Extra	50	0	✓	1543	565	0.7358	0.7320	**0.7339**	0.6484

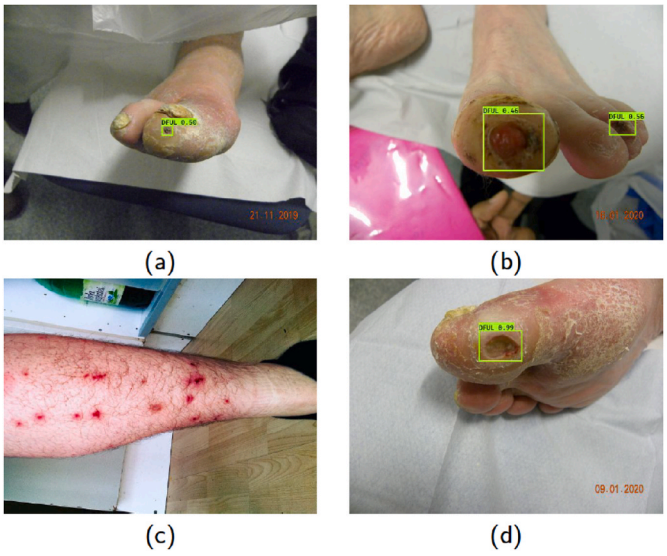

Fig. 13. Examples of final detection outputs of trained YOLOv3, after post-processing.

in slightly better precision, it reduces recall, F1-score and mAP. Therefore, unless precision is the priority, removing the low confidence detections would not lead to an improvement. Examples of final detections for YOLOv3 are presented in Fig. 13.

Additionally, we added 60 copyright-free images of healthy feet[21] to the training set to observe the effect on detection performance. As shown in Table 3, this results in an improvement of F1-Score, but reduces mAP.

5.3. YOLOv5

Table 4 summarizes the results of YOLOv5. Fewer additional self-training epochs in method E60_SELF90 achieved better results than E60_SELF100 and E60_SELF120. However, the application of TTA with NMS on E60_SELF100 achieved the best results in E60_SELF100_TTA_NMS. Examples of detections with E60_SELF100_TTA_NMS on the test set are shown in Fig. 14, Fig. 15 shows additional examples of false negative and false positive cases.

5.4. EfficientDet

Table 5 shows the results of the EfficientDet model on the DFUC2020 testing set both with and without post-processing. The results indicate that the number of both TP and FP cases are reduced with the post-processing method. However, with the post-processing method, the percentage of TP cases (from 1,626 to 1,593) is 2.02% compared to FP cases (from 720 to 594), which is 17.50%. Hence, the post-processing method results in an important improvement in both Precision (67.86%–72.84%) and F1-score (72.38%–74.37%), with a slight decrease in both mAP (57.82%–56.94%) and Recall (77.44%–75.97%). The EfficientDet with post-processing method achieved the highest F1-Score and Precision (least number of FP cases) in DFUC2020. Examples of final outputs by the refined EfficientDet architecture are shown in Fig. 16.

5.5. Cascade Attention DetNet

Table 6 summarizes the results of the Cascade Attention DetNet on the DFUC2020 testing dataset. The results are reported for two different data augmentation methods, two different backbones and with or without a pre-trained model.

From the results, we observe that CA-DetNet with two data augmentation methods and the pre-trained model achieves the best result. It achieves the highest score of 63.94% on mAP and 70.01% on F1-Score. The C-DetNet achieves the highest score of 74.11% on Recall, while the CA-DetNet with the mobile fuzzy method achieves the highest score of 66.67% on Precision.

From the analysis, we observe that the mobile fuzzy data augmentation method brings about a striking effect and improves 1.46% on mAP and 1.03% on F1-Score. However, we note that using the single mixup method in data augmentation did not enhance the performance. The results suggest that the mobile fuzzy method allows the model to adapt to the noise from the external environment, while the mixup method is detrimental. The attention mechanism contributes to the improved performance of detection and increases mAP by 0.02% and F1-Score by 0.03%. Moreover, training with a pre-trained model can accelerate the convergence of the model and improve its ability to detect DFU.

Our approach was effective for the vast majority of the detected cases, as shown in Fig. 17. However, due to the visual complexity of clinical environments, there are also some failure cases in our approach. From our observations, such failures are generally due to the false identification of toenails, interference from the external environment and low image quality. For the false identification of toenails, we believe that the appearance of leuconychia is similar to wounds and some cases of DFU are located on or around the toenail. Background objects may also sometimes interfere with detection results. We use the attention mechanism to deal with this problem to some extent. For image quality, we observe that there are several images which are blurry. We use data augmentation methods like the mobile fuzzy method to partially address this problem. We speculate that a two-stage architecture with an initial stage to detect and segment the relevant foot area could be used to address this issue. However, additional labeled data may be required to achieve this goal.

5.6. Comparison of the challenge results

The results from the popular deep learning object detection methods

[21] Freepik website: https://www.freepik.com/(accessed 2020-08-29).

Table 4
YOLOv5: Results of different submitted runs. The settings state epochs for base and self-training as well as the use of test-time augmentation (TTA) and non-maximum suppression (NMS). Best results are highlighted bold.

Method	Settings			Metrics					
	Base	Self-training	TTA + NMS	TP	FP	Recall	Precision	F1-Score	mAP
E60_SELF90	60	30	×	1504	**474**	0.7172	**0.7604**	**0.7382**	0.6270
E60_SELF100	60	40	×	1496	485	0.7134	0.7552	0.7337	0.6165
E60_SELF100_TTA_NMS	60	40	✓	**1507**	498	**0.7187**	0.7516	0.7348	**0.6294**
E60_SELF120	60	60	×	1502	478	0.7163	0.7586	0.7368	0.6201

and the proposed CA-DetNet are comparable. Table 7 shows the overall results when evaluated on the DFUC2020 testing set, where we present the best mAP from each object detection method. Considering the ranking based on mAP, the best result is achieved by the variant of Faster R–CNN using Deformable Convolution, with 0.6940. This method achieves the highest TP and the best Recall. It is noted that YOLOv5 achieved the lowest number of FP, but it has lower mAP and F1-Score.

In Table 8, the ranking according to F1-Score shows the highest F1-Score of 0.7437 obtained by EfficientDet, however, this network reports the lowest mAP at 0.5694. On the other hand, the Faster R–CNN approach achieves a comparable F1-Score of 0.7434 with the highest

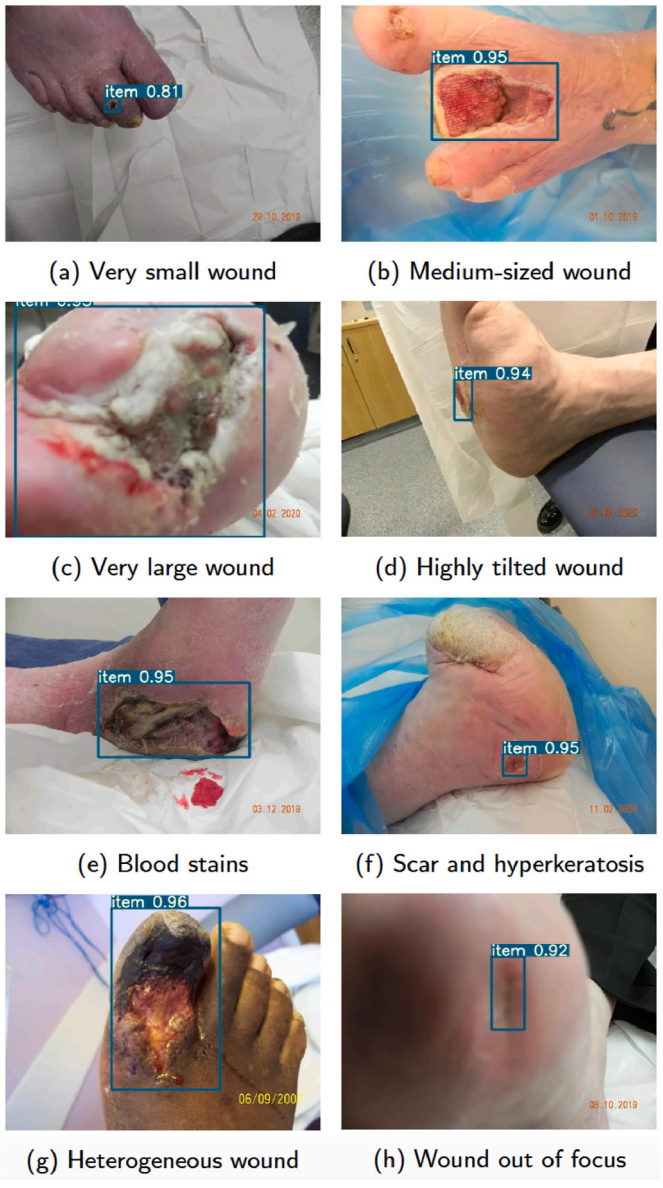

Fig. 14. Examples for adequate predictions with YOLOv5 for different DFU sizes and compositions: (a) to (c) different wound sizes, (d) partially visible wound, (e) non-detected blood stain on dressing, (f) non-detected scar and hyperkeratosis, (g) heterogeneous wound composition, (h) detected wound out of focus.

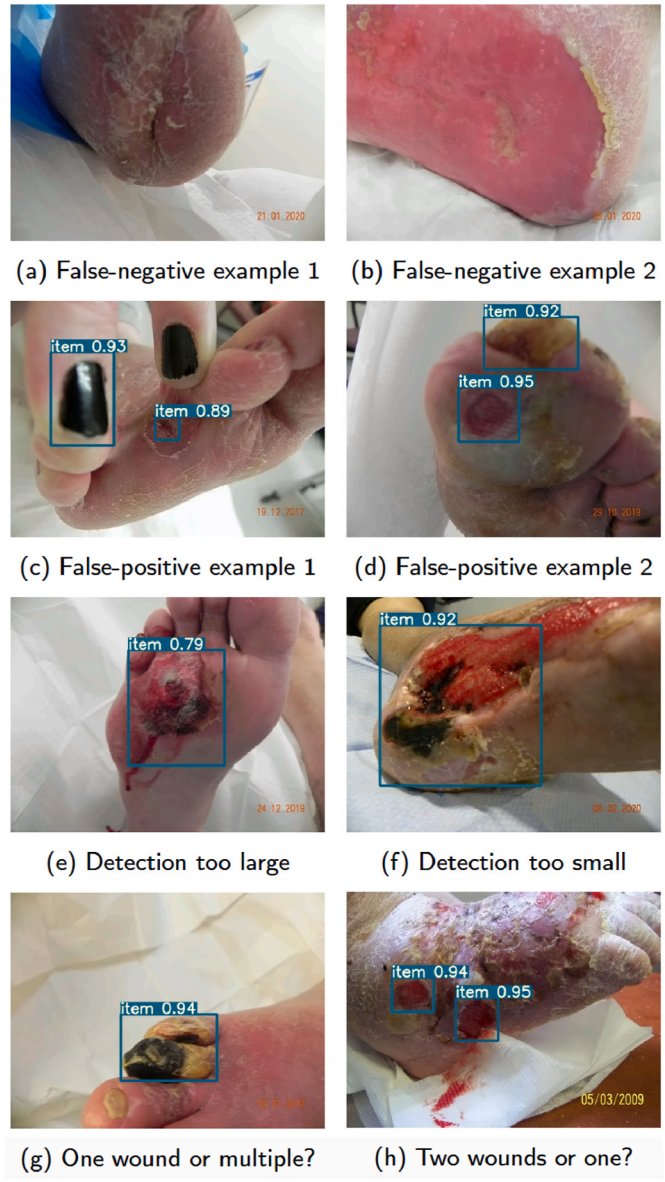

Fig. 15. Examples of false negative, false positive, inadequate and questionable YOLOv5 predictions: (a) and (b) non-detected wounds, (c) and (d) painted finger nail and malformed toe nail, (e) and (f) too large and too small, (g) and (h) unclear detections (one, two, many?).

Table 5
EfficientDet. 'Before' is the result of EfficientDet without post-processing and 'After' is the result with post-processing.

Methods	TP	FP	Recall	Precision	F1-Score	mAP
Before	**1626**	770	**0.7754**	0.6786	0.7238	**0.5782**
After	1593	**594**	0.7597	**0.7284**	**0.7437**	0.5694

mAP of 0.6940.

Fig. 18 visually compares the detection results on DFUs with less visible appearances. In Fig. 18(a), the ulcer was detected by all the methods. However, in Fig. 18(b), only Faster R–CNN and EfficientDet detected the ulcer. Fig. 18(c) is another challenging case and was detected by CA-DetNet and Faster R–CNN. In Fig. 18(d), we demonstrate a case where only Faster R–CNN successfully localised the ulcer.

5.7. Further analysis with ensemble method

We conduct further analysis of the performance of the proposed methods. In Section 5.1, we demonstrate that the ensemble method using Weighted Boxes Fusion did not improve the results of four Faster R–CNN approaches. This observation suggests that additional experiments based on different deep learning approaches should be investigated. We ran experiments based on combinations of two approaches (Faster R–CNN + (CA-DetNet/EfficientDet/YOLOv3/YOLOv5)), three approaches and a combination of all approaches, as summarized in Table 9. From our observation, the ensemble methods reduce the number of TPs and FPs, i.e., the more approaches used, the lower the number of TPs and FPs. This approach did not improve mAP, but in the majority of the ensembles there are notable improvements in precision,

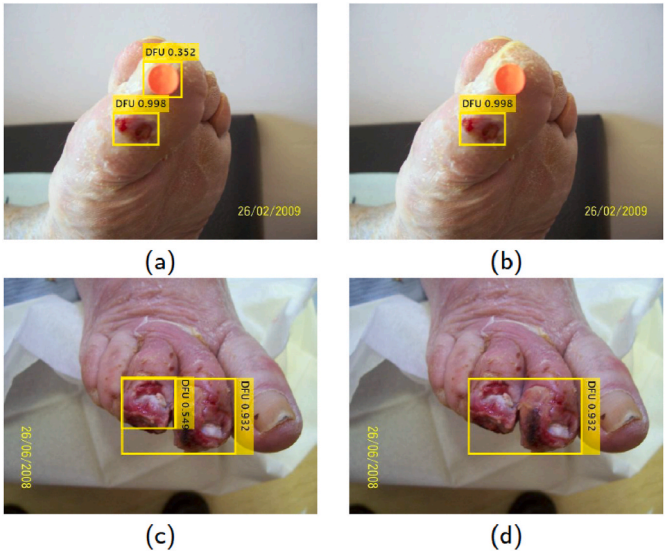

Fig. 16. The results of EfficientDet. (a) and (c) are the results of EfficientDet without post-processing; (b) and (d) are the results obtained with post-processing.

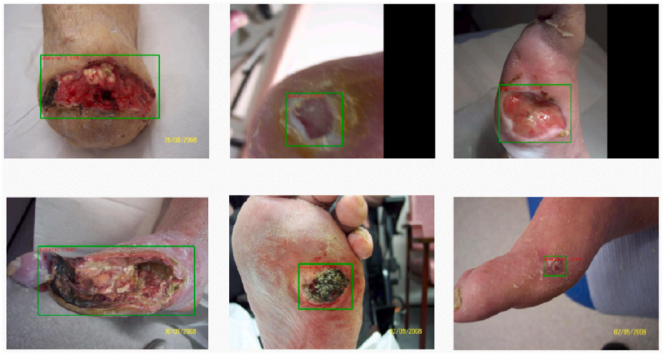

Fig. 17. The results of CA-DetNet: Illustration of successful DFU detections. Note the variety of DFU sizes detected by the network, ranging from small (bottom-right), medium (top-middle and bottom-middle), large (top-left, top-right) and very large (bottom-left).

Table 7
A summary based on the mAP ranking from each object detection method when evaluated on the DFUC2020 testing set.

Methods	TP	FP	Recall	Precision	F1-Score	mAP
Faster R–CNN	**1612**	628	**0.7687**	0.7196	0.7434	**0.6940**
YOLOv3	1572	676	0.7496	0.6993	0.7236	0.6560
CA-DetNet	1554	788	0.7411	0.6635	0.7002	0.6394
YOLOv5	1507	**498**	0.7187	**0.7516**	0.7348	0.6294
EfficientDet	1593	594	0.7597	0.7284	**0.7437**	0.5694

Table 8
A summary based on F1-Score ranking from each object detection method when evaluated on the DFUC2020 testing set.

Methods	TP	FP	Recall	Precision	F1-Score	mAP
EfficientDet	1593	594	0.7597	0.7284	**0.7437**	0.5694
Faster R–CNN	**1612**	628	**0.7687**	0.7196	0.7434	**0.6940**
YOLOv5	1504	474	0.7172	**0.7604**	0.7382	0.6270
YOLOv3	1543	**565**	0.7358	0.7320	0.7339	0.6484
CA-DetNet	1554	788	0.7411	0.6635	0.7002	0.6394

which led to an improvement in F1-Score. The best F1-Score for the ensemble method is 0.7617, achieved by ensembling Faster R–CNN with Deformable Convolution and EfficientDet.

5.8. Comparison with existing methods

DFU detection is still in its infancy, and research in this field is limited. As reported by Cassidy et al. [10], the organisers of DFUC2020 have provided the baseline results for the challenge, which include the approach used in Goyal et al. [16] on Faster R–CNN, but without the post-processing stages. When comparing our overall best results (*Proposed* in Table 10) with the benchmark algorithms, the new method (based on Faster R-CNN with the configuration from Table 7) outperformed them in every category of the performance metrics.

Apart from fine-tuning each deep learning method to maximise

Table 6
Results for each of the Cascade Attention DetNets.

Backbone	Settings			Metrics					
	pre-trained	mobile fuzzy	mixup	TP	FP	Recall	Precision	F1-Score	mAP
C-DetNet	✓	✓	✓	1554	789	**0.7411**	0.6633	0.7000	0.6391
CA-DetNet	×	×	×	1493	1089	0.7120	0.5782	0.6382	0.5963
CA-DetNet	✓	×	×	1523	820	0.7263	0.6500	0.6860	0.6204
CA-DetNet	✓	×	✓	1431	961	0.6824	0.5982	0.6376	0.5749
CA-DetNet	✓	✓	×	1528	764	0.7287	**0.6667**	0.6963	0.6350
CA-DetNet	✓	✓	✓	1554	788	0.7411	0.6635	**0.7002**	**0.6394**

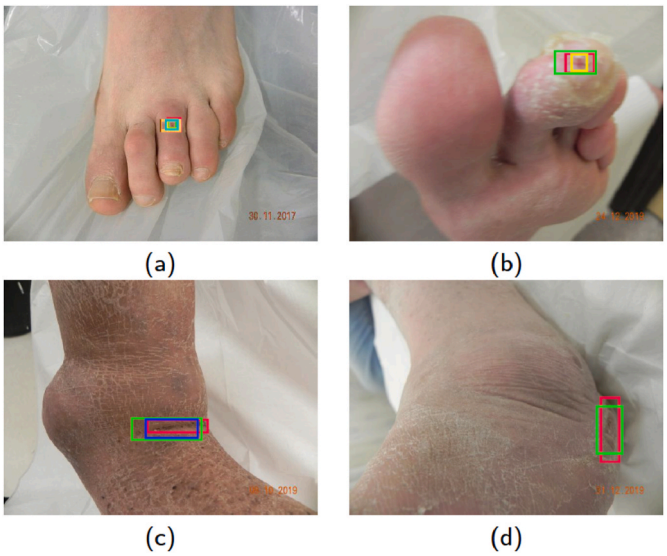

Fig. 18. Visual comparison of object detection methods when compared to the ground truth (in red): (a) An easy case where all methods detected the ulcer; (b) A more challenging case detected by Faster R–CNN (green) and EfficientDet (yellow); (c) A challenging case detected by Faster R–CNN (green) and CA-DetNet (blue); and (d) A challenging case only detected by Faster R–CNN (green).

Table 9
A comparison of ensemble methods with different combinations of object detection frameworks, where FRCNN is Faster R–CNN, DetNet is CA-DetNet, EffDet is EfficientDet and 'ALL methods' represents an ensemble method based on Faster R–CNN, CA-DetNet, EfficientDet, YOLOv3 and YOLOv5.

Methods	TP	FP	Recall	Precision	F1-Score	mAP
FRCNN + DetNet	**1510**	426	**0.7201**	0.7800	0.7488	**0.6619**
FRCNN + EffDet	1502	345	0.7163	0.8132	**0.7617**	0.6425
FRCNN + YOLOv3	1423	310	0.6786	0.8211	0.7431	0.6205
FRCNN + YOLOv5	1453	350	0.6929	0.8059	0.7451	0.6421
FRCNN + YOLOv5+EffDet	1396	252	0.6657	0.8471	0.7455	0.6109
FRCNN + YOLOv5+DetNet	1384	295	0.6600	0.8243	0.7331	0.6132
FRCNN + DetNet + EffDet	1435	270	0.6843	0.8416	0.7549	0.6229
ALL methods	1277	**198**	0.6090	**0.8658**	0.7150	0.5642

performance, the methods are highly dependent on the pre-processing stage, selection of data augmentation, post-processing methods and ensemble method. We address the limitations and future challenges of our work in the following section.

6. Discussion

In this section, we discuss the performance of each object detection method and future work to improve DFU detection. Whilst most of the results show an F1-Score $>$ 70%, there are many challenges ahead to enable the use of deep learning algorithms in real-world settings.

Faster R–CNN based approaches detected DFU in the DFUC2020 testing set with high mAP and F1-Scores. In addition, the variants of Faster R–CNN largely improve the performance of the original Faster R–CNN. After ensembling the results of four models, we reduced the number of false positives, however, the overall performance was less than the individual variants of Faster R–CNN. The reason may be that even though we are fusing the prediction of four models into one prediction, similar results are predicted among these four models because all models are based on Faster R–CNN. Therefore, in future work, a one-stage object detection method such as CenterNet [55] could potentially

Table 10
Performance of the benchmark algorithms on the testing set reported by Cassidy et al. [10] without fine-tuning and/or post-processing stages. FRCNN represents Faster R–CNN and *Proposed* represents the overall best result of DFUC2020.

Benchmark Algorithms	Recall	Precision	F1-Score	mAP
FRCNN R–FCN	0.7511	0.6186	0.6784	0.6596
FRCNN ResNet101	0.7396	0.5995	0.6623	0.6518
FRCNN Inception-v2-ResNet101	0.7554	0.6046	0.6716	0.6462
YOLOv5	0.7244	0.6081	0.6612	0.6304
EfficientDet	0.6939	0.6919	0.6929	0.6216
Proposed	**0.7687**	**0.7196**	**0.7434**	**0.6940**

be included in the ensemble method to produce more accurate results.

The YOLOv3 algorithm is able to reliably detect DFU and ranked third place in both mAP and F1-Score ranking. We have observed that post-processing (by removing overlaps), along with the removal of low confidence detections, leads to an improvement in precision but at the expense of the number of true positives and recall. Additionally, our analysis indicates that adding additional images of healthy feet, along with post-processing, can result in a higher F1-score. We aim to further investigate the results of pre-processing, as well as studying a more effective post-processing method.

The YOLOv5 approach demonstrated reliable detection performance with an overall high precision over the different model configurations. Application of the NLM algorithm for image enhancement and generalization via self-training helped to further increase precision. Improvements via duplicate cleansing and bounding box merging were marginal due to the limited number of cases, but could prove beneficial on larger datasets. Use of TTA with NMS further increased true-positive detections at the cost of increased false-positive cases, yet increased the mAP and F1-Score. For the presented approach, several optimizations may be possible. The least self-trained model performed best, indicating that models with less self-training epochs may perform better. Model Ensembling[22] might provide further performance improvements when fusing different specialized models. In addition, investigation of Hyperparameter Evolution[23] allows general hyperparameter optimization, given the required resources.

As the presented results were obtained with the initial release v1.0 [21] of YOLOv5, the resulting performance is limited compared to that achievable with matured up-to-date versions of the network [5]. Since its release, YOLOv5 has been improving rapidly and its full potential could not be taken advantage of during the DFUC2020. E.g., in v1.0 the novel Mosaic data augmentation method was not functioning correctly on custom data. At the time of writing, the matured version v5.0[24] [23] is available, featuring numerous bug fixes, improvements and novelties. E.g., the activation function changed from Leaky ReLU [30] in v1.0 (used here) to Sigmoid Linear Unit (SiLU) [20] (since v4.0) [24], further increasing detection performance. Due to its reasonable performance and mobile-focus, YOLOv5 may prove to be useful when performing DFU detection tasks directly on mobile devices.

The refined EfficientDet algorithm is able to detect DFU with a high recall rate. The pre-processing stage using the Shades of Gray algorithm improved the color consistency of the images in the dataset. We extensively used data augmentation techniques to learn the subtle features of DFUs of various sizes and severity. The post-processing stage we implemented refined the inference capability of the original EfficientDet method by removing overlapping bounding boxes. Due to low mAP, further work will focus on investigating the larger EfficientDet network

[22] YOLOv5 GitHub repository tutorial on Model Ensembling: https://github.com/ultralytics/yolov5/issues/318 (accessed 2020-09-28).
[23] YOLOv5 GitHub repository tutorial on Hyperparameter Evolution: https://github.com/ultralytics/yolov5/issues/607 (accessed 2020-09-28).
[24] YOLOv5 v5.0: https://github.com/ultralytics/yolov5/tree/v5.0 (accessed 2021-04-26).

architectures, particularly EfficientDet-D7.

The performance of Cascade Attention DetNet on the DFUC2020 testing set is competitive but not entirely satisfactory. We evaluated our model on 10% of the DFUC2020 training set and it achieved an mAP of 0.9. We analyzed the possible reasons and speculate that the model may be over-fitting, to which ensemble learning may provide a possible solution. We further aim to use appropriate data augmentation methods to improve the robustness of the model.

The ensemble methods based on the fusion of different backbones reduced the number of predicted bounding boxes substantially. Faster R–CNN with Deformable Convolution predicted 2,240 bounding boxes. However, after ensembling with EfficientDet, only 1,847 bounding boxes were predicted. The number of predicted bounding boxes dropped to 1,475 when we ensembled the results from all five networks. Consequently, the ensemble methods reduced the number of TPs and FPs. It is crucial for future research to focus on true positives, i.e., correctly locate the DFUs. One of the aspects required to overcome this issue is to understand the threshold setting of IoU. Our experiments used IoU $\geq$0.5, which is the guideline set by object detection for natural objects. However, some medical imaging studies [11,48] used an IoU (or Jaccard Similarity Index) threshold of 0.4. When we evaluated the performance of the best ensemble method, the number of TPs increased to 1,594, with IoU $\geq$0.3 the number of TPs increased to 1,668. With Faster R–CNN with Deformable Convolution, the number of TPs increased to 1,743 and 1,883 for IoU thresholds of 0.4 and 0.3, respectively.

Currently, most clinicians involved in DFU care solely use a visual assessment for detection of ulcers, taking photographs at the diagnosis stage and periodically re-evaluating wound states on subsequent patient clinic appointments. This method is time-consuming and thus economically costly. Further, manual comparison of a patient's wound photograph history over different stages only enables a rough assessment of the healing progress. Other changes in wound condition may also go unrecognized, such as the subtle features that occur prior to the emergence of a DFU. In addition, the effects of intra and inter-observer variability for manual wound monitoring do not support an objective measurement with consistent results.

An accurate method capable of detecting DFU at all stages of development is the first step towards a fully-automated assessment tool, enabling objective, cost and time-efficient DFU documentation and analysis. However, due to the high variability of features present in DFU cases, the problem of detection is non-trivial, as highlighted by our reported results. Color, texture of tissue types, location, size, shape and depth of DFU present a highly variable set of features. Pathological phenomena like infections, ischaemia, hyperkeratosis, and stasis dermatitis, can further alter the appearance of a DFU and its surrounding area. Additional complexity is added by altered anatomy due to deformation or partial amputation. This study has presented a diverse set of deep learning-based solutions to address the problems associated with accurate DFU detection.

7. Conclusion

We conduct a comprehensive evaluation of the performance of deep learning object detection networks for DFU detection. Deformable convolutions appear to work well in DFU detections and contribute to the improvement to the best performing method. While the overall results show the potential of localising DFUs using CNNs, the number of false positive results is significant, and the networks are not always able to effectively discriminate ulcers from other skin conditions. The introduction of a second classifier based on a negative dataset may provide a possible solution to this issue when training future networks. However, in reality, it may prove impossible to gather all possible negative examples for supervised learning algorithms. This approach could also impact network size and complexity, which could negatively impact inference speed. Segmenting the foot from its surroundings might provide another possible solution to this problem, so that trained models do not have to account for objects in real-world environments. Other future research challenges include:

- Increasing the size of the existing dataset with clinical annotations which would include metadata indicating the development stage of each DFU. However, there are still barriers in data sharing and clinical annotation is expensive and time consuming. It will be important to encourage the co-creation of such datasets via machine learning and clinical experts to foster a better understanding of the annotated data. While increasing the number of images may benefit the training process, other aspects such as ulcer location should be considered.
- Creating self-supervised and unsupervised deep learning algorithms for DFU detection. These methods have been developed and implemented for natural object detection tasks but remain under-explored in medical imaging.
- Accurate delineation of an ulcer and its surrounding skin can help to measure the healing progress of the ulcer. Goyal et al. [17] developed an automated segmentation algorithm for DFU, however, they experimented using a small dataset. Future work will potentially enable a larger scale of experimentation.
- The use of DFU classification systems that can be used by clinicians to analyse ulcer condition. Automated analysis and recognition of DFU can help to improve the diagnosis of DFUs. With this goal in mind, the next DFU challenge (DFUC2021 [50]) will focus on multi-class DFU pathology recognition.
- Detection and quantification of tissue types within DFU wounds as a means of monitoring healing status over time.
- Ensuring that future DFU detection models are trained using datasets that consist of cases taken from a variety of ethnicities. DFUs are an especially pertinent problem in developing countries, where cheap remote detection systems may be most useful.

With the growth in the number of people diagnosed with diabetes, remote detection and monitoring of DFU can help to reduce the burden on health services. Research in the optimization of CNNs for remote monitoring is another active research area that has the potential to change the healthcare landscape globally.

The DFUC2020 dataset contains many examples of images which include a variety of artifacts, including malformed toenails, rhagades and hyperkeratosis. Without an accurate DFU detection algorithm, accurate segmentation and wound size estimation is not possible. A reliable detection method that performs well on typical wound care images, created under uncontrolled (non-laboratory) conditions, remains the first and cardinal problem.

This work brought together researchers from around the world with the aim of advancing research efforts in diabetic foot ulcer detection. The latest state-of-the-art methods were implemented in a series of experiments that will provide future researchers with vital data that can be built upon to address the many obstacles present in this problem domain. The results of this work help to define the larger scope of the challenges inherent in automated DFU detection, and provide important indicators for possible avenues for future work.

Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation for the use of GPUs during the challenge and for sponsoring our event. A.A., D.B.A. and M.O. were supported by the National Health and Medical Research Council [GNT1174405] and the Victorian Government's OIS Program. The work of R.B. was partially funded by a PhD grant from the University of Applied Sciences and Arts Dortmund, 44227 Dortmund, Germany.

References

[1] D.G. Armstrong, A.J. Boulton, S.A. Bus, Diabetic foot ulcers and their recurrence, N. Engl. J. Med. 376 (2017) 2367–2375, https://doi.org/10.1056/nejmra1615439.

[2] A. Bochkovskiy, C.Y. Wang, H.Y.M. Liao, YOLOv4: Optimal Speed and Accuracy of Object Detection, 2020 arXiv:2004.10934.

[3] N. Bodla, B. Singh, R. Chellappa, L.S. Davis, Soft-NMS — improving object detection with one line of code, in: 2017 IEEE International Conference on Computer Vision (ICCV), IEEE, 2017, pp. 5561–5569, https://doi.org/10.1109/iccv.2017.593.

[4] R. Brown, B. Ploderer, L.S. Da Seng, P. Lazzarini, J. van Netten, MyFootCare: a mobile self-tracking tool to promote self-care amongst people with diabetic foot ulcers, in: Proceedings of the 29th Australian Conference on Computer-Human Interaction (OzCHI'17), Association for Computing Machinery, 2017, pp. 462–466, https://doi.org/10.1145/3152771.3156158.

[5] R. Brüngel, C.M. Friedrich, DETR and YOLOv5: exploring performance and self-training for diabetic foot ulcer detection, in: 2021 IEEE 34th International Symposium on Computer-Based Medical Systems (CBMS), IEEE, 2021, pp. 148–153, https://doi.org/10.1109/CBMS52027.2021.00063.

[6] A. Buades, B. Coll, J.M. Morel, A non-local algorithm for image denoising, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), IEEE, 2005, pp. 60–65, https://doi.org/10.1109/cvpr.2005.38.

[7] Z. Cai, N. Vasconcelos, Cascade R-CNN: delving into high quality object detection, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2018, pp. 6154–6162, https://doi.org/10.1109/cvpr.2018.00644.

[8] Z. Cai, N. Vasconcelos, Cascade R-CNN: high quality object detection and instance segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 43 (2021) 1483–1498, https://doi.org/10.1109/tpami.2019.2956516.

[9] Y. Cao, K. Chen, C.C. Loy, D. Lin, Prime sample attention in object detection, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2020, pp. 11580–11588, https://doi.org/10.1109/cvpr42600.2020.01160.

[10] B. Cassidy, N.D. Reeves, P. Joseph, D. Gillespie, C. O'Shea, S. Rajbhandari, A. G. Maiya, E. Frank, A. Boulton, D. Armstrong, B. Najafi, J. Wu, M.H. Yap, The DFUC 2020 Dataset: Analysis towards Diabetic Foot Ulcer Detection, vol. 17, 2021, pp. 5–11, https://doi.org/10.17925/EE.2021.1.1.5.

[11] K. Drukker, M.L. Giger, K. Horsch, M.A. Kupinski, C.J. Vyborny, E.B. Mendelson, Computerized lesion detection on breast ultrasound, Med. Phys. 29 (2002) 1438–1446, https://doi.org/10.1118/1.1485995.

[12] M. Goyal, S. Hassanpour, A Refined Deep Learning Architecture for Diabetic Foot Ulcers Detection, 2020 arXiv:2007.07922.

[13] M. Goyal, S. Hassanpour, M.H. Yap, Region of Interest Detection in Dermoscopic Images for Natural Data-Augmentation, 2019 arXiv:1807.10711.

[14] M. Goyal, N.D. Reeves, A.K. Davison, S. Rajbhandari, J. Spragg, M.H. Yap, DFUNet: convolutional neural networks for diabetic foot ulcer classification, IEEE Transactions on Emerging Topics in Computational Intelligence 4 (2020) 728–739, https://doi.org/10.1109/TETCI.2018.2866254.

[15] M. Goyal, N.D. Reeves, S. Rajbhandari, N. Ahmad, C. Wang, M.H. Yap, Recognition of ischaemia and infection in diabetic foot ulcers: dataset and techniques, Comput. Biol. Med. 117 (2020) 103616, https://doi.org/10.1016/j.compbiomed.2020.103616.

[16] M. Goyal, N.D. Reeves, S. Rajbhandari, M.H. Yap, Robust methods for real-time diabetic foot ulcer detection and localization on mobile devices, IEEE Journal of Biomedical and Health Informatics 23 (2019) 1730–1741, https://doi.org/10.1109/JBHI.2018.2868656.

[17] M. Goyal, M.H. Yap, N.D. Reeves, S. Rajbhandari, J. Spragg, Fully convolutional networks for diabetic foot ulcer segmentation, in: 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2017, pp. 618–623, https://doi.org/10.1109/SMC.2017.8122675.

[18] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, in: 2017 IEEE International Conference on Computer Vision (ICCV), IEEE, 2017, pp. 2980–2988, https://doi.org/10.1109/ICCV.2017.322.

[19] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, IEEE Trans. Pattern Anal. Mach. Intell. 37 (2015) 1904–1916, https://doi.org/10.1109/tpami.2015.2389824.

[20] D. Hendrycks, K. Gimpel, Gaussian Error Linear Units (GELUs), 2020 arXiv:1606.08415 arXiv:1606.08415.

[21] G. Jocher, L. Changyu, A. Hogan, L. Yu, changyu98, P. Rai, T. Sullivan, Ultralytics/yolov5, Initial release (2020), https://doi.org/10.5281/zenodo.3908560.

[22] G. Jocher, Y. Kwon, guigarfr, J. Veitch-Michaelis, perry0418, Ttayu, Marc, G. Bianconi, F. Baltacı, D. Suess, T. Chen, P. Yang, idow09, WannaSeaU, W. Xinyu, T.M. Shead, T. Havlik, P. Skalski, NirZarrabi, LukeAI, LinCoce, J. Hu, IlyaOvodov, GoogleWiki, F. Reveriano, Falak, D. Kendall, ultralytics/yolov3: 43.1mAP@0.5: 0.95 on COCO2014, 2020, https://doi.org/10.5281/zenodo.3785397.

[23] G. Jocher, A. Stoken, J. Borovec, NanoCode012, A. Chaurasia, TaoXie, L. Changyu, V.A. Laughing, tkianai, yxNONG, A. Hogan, lorenzomammana, AlexWang1900, J. Hajek, L. Diaconu, Marc, Y. Kwon, oleg, wanghaoyang0106, Y. Defretin, A. Lohia, ml5ah, B. Milanko, B. Fineran, D. Khromov, D. Yiwei, Durgesh Doug, F. Ingham, ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 Models, AWS, Supervise.Ly and YouTube Integrations, 2021, https://doi.org/10.5281/zenodo.4679653.

[24] G. Jocher, A. Stoken, J. Borovec, NanoCode012, ChristopherSTAN, L. Changyu, Laughing, tkianai, yxNONG, A. Hogan, lorenzomammana, AlexWang1900, A. Chaurasia, L. Diaconu, Marc, wanghaoyang0106, ml5ah, Durgesh Doug, F. Ingham, Guilhen Frederik, A. Colmagro, H. Ye, Jacobsolawetz, J. Poznanski, J. Fang, J. Kim, K. Doan, L. Yu, ultralytics/yolov5: v4.0 - nn.SiLU( ) Activations, Weights & Biases Logging, PyTorch Hub Integration, 2021, https://doi.org/10.5281/zenodo.4418161.

[25] S. Koitka, C.M. Friedrich, Optimized convolutional neural network ensembles for medical subfigure classification, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction 8th International Conference of the CLEF Association, CLEF 2017, Lecture Notes in Computer Science (LNCS), Springer International Publishing, 2017, pp. 57–68, https://doi.org/10.1007/978-3-319-65813-1_5.

[26] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, J. Sun, DetNet: design backbone for object detection, in: European Conference on Computer Vision (ECCV) 2018, Springer International Publishing, 2018, pp. 339–354, https://doi.org/10.1007/978-3-030-01240-3_21.

[27] T.Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017, pp. 936–944, https://doi.org/10.1109/CVPR.2017.106.

[28] T.Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft COCO: common objects in context, in: European Conference on Computer Vision (ECCV) 2014, Springer International Publishing, 2014, pp. 740–755, https://doi.org/10.1007/978-3-319-10602-1_48.

[29] S. Liu, L. Qi, H. Qin, J. Shi, J. Jia, Path aggregation network for instance segmentation, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2018, pp. 8759–8768, https://doi.org/10.1109/cvpr.2018.00913.

[30] A.L. Maas, A.Y. Hannun, A.Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: Proceedings of the 30th International Conference on Machine Learning (ICML) 2013,, ICML, 2013.

[31] J.H. Ng, M. Goyal, B. Hewitt, M.H. Yap, The effect of color constancy algorithms on semantic segmentation of skin lesions, in: Medical Imaging 2019: Biomedical Applications in Molecular, Structural, and Functional Imaging, International Society for Optics and Photonics, SPIE, 2019, pp. 138–145, https://doi.org/10.1117/12.2512702.

[32] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: an imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems, vol. 32, Curran Associates, Inc., 2019, pp. 8024–8035 (NeurIPS 2019).

[33] E. Pebesma, Simple features for R: standardized support for spatial vector data, The R Journal 10 (2018) 439–446, https://doi.org/10.32614/RJ-2018-009.

[34] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2020. URL: https://www.R-project.org/. Retrieved on 2021-04-28.

[35] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016, pp. 779–788, https://doi.org/10.1109/cvpr.2016.91.

[36] J. Redmon, A. Farhadi, YOLO9000: better, faster, stronger, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017, pp. 7263–7271, https://doi.org/10.1109/cvpr.2017.690.

[37] J. Redmon, A. Farhadi, YOLOv3: an Incremental Improvement, 2018 arXiv:1804.02767.

[38] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, IEEE Trans. Pattern Anal. Mach. Intell. 39 (2017) 1137–1149, https://doi.org/10.1109/tpami.2016.2577031.

[39] P. Saeedi, I. Petersohn, P. Salpea, B. Malanda, S. Karuranga, N. Unwin, S. Colagiuri, L. Guariguata, A.A. Motala, K. Ogurtsova, J.E. Shaw, D. Bright, R. Williams, Global and regional diabetes prevalence estimates for 2019 and projections for 2030 and 2045: results from the international diabetes federation diabetes atlas, 9th edition. Diabetes Research and Clinical Practice 157 (2019) 107843, https://doi.org/10.1016/j.diabres.2019.107843.

[40] R. Solovyev, W. Wang, T. Gabruseva, Weighted boxes fusion: ensembling boxes from different object detection models, Image Vis Comput. 107 (2021) 104117, https://doi.org/10.1016/j.imavis.2021.104117.

[41] M. Tan, Q. Le, EfficientNet: rethinking model scaling for convolutional neural networks, in: Proceedings of the 36th International Conference on Machine Learning, 2019, pp. 6105–6114. URL, http://proceedings.mlr.press/v97/tan19a.html. Retrieved on 2021-04-28.

[42] M. Tan, R. Pang, Q.V. Le, EfficientDet: scalable and efficient object detection, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2020, pp. 10778–10787, https://doi.org/10.1109/CVPR42600.2020.01079.

[43] C.Y. Wang, H.Y. Mark Liao, Y.H. Wu, P.Y. Chen, J.W. Hsieh, I.H. Yeh, CSPNet: a new backbone that can enhance learning capability of CNN, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE, 2020, pp. 1571–1580, https://doi.org/10.1109/CVPRW50498.2020.00203.

[44] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, X. Tang, Residual attention network for image classification, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017, pp. 6450–6458, https://doi.org/10.1109/CVPR.2017.683.

[45] L. Wang, P.C. Pedersen, E. Agu, D.M. Strong, B. Tulu, Area determination of diabetic foot ulcer images using a cascaded two-stage SVM-based classification, IEEE (Inst. Electr. Electron. Eng.) Trans. Biomed. Eng. 64 (2017) 2098–2109, https://doi.org/10.1109/TBME.2016.2632522.

[46] L. Wang, P.C. Pedersen, D.M. Strong, B. Tulu, E. Agu, R. Ignotz, Smartphone-based wound assessment system for patients with diabetes, IEEE (Inst. Electr. Electron. Eng.) Trans. Biomed. Eng. 62 (2015) 477–488, https://doi.org/10.1109/TBME.2014.2358632.

[47] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017, pp. 5987–5995, https://doi.org/10.1109/CVPR.2017.634.
[48] M.H. Yap, E.A. Edirisinghe, H.E. Bez, A novel algorithm for initial lesion detection in ultrasound breast images, J. Appl. Clin. Med. Phys. 9 (2008) 181–199, https://doi.org/10.1120/jacmp.v9i4.2741.
[49] M.H. Yap, M. Goyal, F. Osman, R. Martí, E. Denton, A. Juette, R. Zwiggelaar, Breast ultrasound region of interest detection and lesion localisation, Artif. Intell. Med. 107 (2020) 101880, https://doi.org/10.1016/j.artmed.2020.101880.
[50] M.H. Yap, N. Reeves, A. Boulton, S. Rajbhandari, D. Armstrong, A.G. Maiya, B. Najafi, E. Frank, J. Wu, Diabetic Foot Ulcers Grand Challenge 2021, 2020, https://doi.org/10.5281/zenodo.3715020.
[51] M.H. Yap, N.D. Reeves, A. Boulton, S. Rajbhandari, D. Armstrong, A.G. Maiya, B. Najafi, E. Frank, J. Wu, Diabetic Foot Ulcers Grand Challenge 2020, 2020, https://doi.org/10.5281/zenodo.3715016.
[52] H. Zhang, M. Cisse, Y.N. Dauphin, D. Lopez-Paz, Mixup: beyond empirical risk minimization, in: 6th International Conference on Learning Representations (ICLR) 2018, ICLR., 2018. URL: https://openreview.net/forum?id=r1Ddp1-Rb. Retrieved on 2021-04-28.
[53] Z. Zhang, T. He, H. Zhang, Z. Zhang, J. Xie, M. Li, Bag of Freebies for Training Object Detection Neural Networks, 2019 arXiv:1902.04103.
[54] W. Zhao, H. Huang, D. Li, F. Chen, W. Cheng, Pointer defect detection based on transfer learning and improved cascade-RCNN, Sensors 20 (2020) 4939, https://doi.org/10.3390/s20174939.
[55] X. Zhou, D. Wang, P. Krähenbühl, Objects as Points, 2019 arXiv:1904.07850.
[56] J. Zhu, L. Fang, P. Ghamisi, Deformable convolutional neural networks for hyperspectral image classification, Geosci. Rem. Sens. Lett. IEEE 15 (2018) 1254–1258, https://doi.org/10.1109/LGRS.2018.2830403.
[57] X. Zhu, H. Hu, S. Lin, J. Dai, Deformable ConvNets V2: more deformable, better results, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2019, pp. 9300–9308, https://doi.org/10.1109/CVPR.2019.00953.