Deep Learning in Diabetic Foot Ulcers Detection: A Comprehensive Evaluation

Moi Hoon Yap[a,*], Ryo Hachiuma[b,1], Azadeh Alavi[c,1], Raphael Brüngel[d,1], Manu Goyal[e,1], Hongtao Zhu[f,1], Bill Cassidy[a], Johannes Ruckert[d], Moshe Olshansky[c], Xiao Huang[f], Hideo Saito[b], Saeed Hassanpour[e], Christoph M. Friedrich[d,g], David Ascher[c], Anping Song[f], Hiroki Kajita[h], David Gillespie[a], Neil D. Reeves[a], Joseph Pappachan[i], Claire O'Shea[j], Eibe Frank[k]

[a]*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 (FHDO), 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

Article history:

2000 MSC: 41A05, 41A10, 65D05, 65D17

Keywords: diabetic foot ulcers, object detection, machine learning, deep learning, DFUC2020

ABSTRACT

There has been a substantial amount of research on 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. With recent development and data sharing performed as part of the DFU Challenge (DFUC2020) such a comparison becomes possible: DFUC2020 provided participants with a comprehensive dataset consisting of 2,000 images for training each method and 2,000 images for testing them. The following deep learning-based algorithms are compared in this paper: 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 is obtained Deformable Convolution, a variant of Faster R-CNN, with a 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 enhanced the F1-Score but not the mAP. Our results show that state-of-the-art deep learning methods can detect DFU with some accuracy, but there are many challenges ahead before they can be implemented in real world settings.

1. Introduction

According to the International Diabetes Federation IDF (2019), there are approximately 463 million adults with diabetes worldwide. This number is expected to grow to 700 million in 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 Armstrong et al. (2017). Infection of a DFU frequently leads to limb amputation, causing significant morbidity, psychological distress, and reduced quality of life and life expectancy. Prevention of DFU is the optimal management pathway; however, current prevention strategies rely on patient and clinician vigilance and place a high burden on global health services. It is essential to develop a technological solution capable of transforming current screening practices and vastly reduce the clini-

*Corresponding author: Tel.: +44 161 247 1503;
e-mail: M.Yap@mmu.ac.uk (Moi Hoon Yap)
[1]Authors with equal contribution.

cal time burden.

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. (2020c,b) proposed 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 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. Goyal et al. (2018, 2017, 2019) trained models capable of classification, localisation and segmentation. These models reported high levels of mAP, sensitivity and specificity in experimental settings. The localisation model 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) so the results cannot be regarded as conclusive evidence of their efficacy in real-world settings.

Brown et al. (2017) 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. (2014, 2016) 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 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.

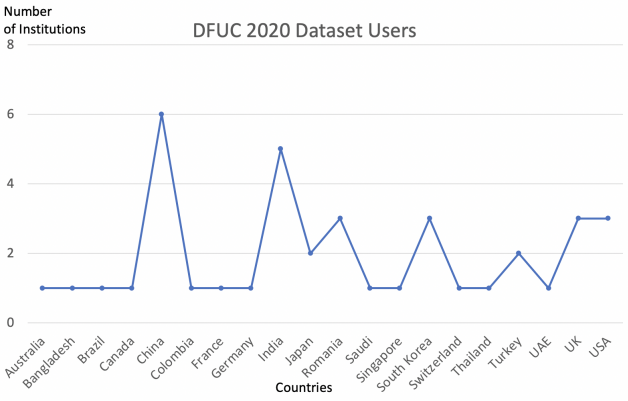

Fig. 1. The users of DFUC2020 dataset across the world.

3. Datasets

The DFU datasets provided by The Manchester Metropolitan University and The Lancashire Teaching Hospitals Goyal et al. (2018, 2020); Cassidy et al. (2020) 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 conduct reproducible experiments. There are three types of datasets made publicly available for researchers. The first dataset consists of foot skin patches for DFUNet classification Goyal et al. (2018); the second dataset contains regions of interests for infection and ischaemia classification Goyal et al. (2020); and the third is the most recently published dataset for DFU detection Cassidy et al. (2020). The third dataset is the largest dataset to date, and increased usage of this data is the driving force for the organisers of the DFUC contests. The researchers involved in organising the yearly DFU challenges Yap et al. (2020c,b), in conjunction with MICCAI conferences, aim to attract wider participation to improve the diagnosis/monitoring of foot ulcers and raise the awareness of diabetes and DFU. The Diabetic Foot Ulcers Grand Challenge (DFUC2020) datasets consist of 2,000 training images, 200 validation images and 2,000 testing images Cassidy et al. (2020); Goyal et al. (2019). The data consists of 2,496 ulcers in the training set and 2,097 ulcers in the testing set. To increase the level of challenge, some of the images in the testing set do not have DFU. The details of the dataset are described in Cassidy et al. (2020). To improve the performance of the deep learning methods and reduce the computational costs, the images were resized to 640×480. Since the release of the DFUC2020 training dataset on the 27th April 2020, we received requests from 39 international institutions, as shown in Table 1. There are a total of 31 submissions from 11 teams to the grand challenge. We report the top scores from each team and discuss their methods according to the object detection approaches involved.

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, i.e. 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 Ren et al. (2015) is one of the two-stage object detection models, which generates a sparse set of candidate object locations by 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 4096 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 He et al. (2017), instead of the RoI pooling layer. This removes the harsh quantization of the RoI pooling layer, properly aligning the extracted features with the input.

Also, the Feature Pyramid Network (FPN) Lin et al. (2017) 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, but otherwise the rest of the approach is similar to vanilla 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 Xie et al. (2017) 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 are different among images in the dataset, 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 to 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 transformation to the images. Specifically, we apply random shift, scaling (0.1), and rotation (90 degrees) to the images.

- 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

In this paper, we fine-tune a model pretrained on MS-COCO Lin et al. (2014). 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 epoch $6, 12$, and 30. We implemented the methods based on the mmdetection repository [2].

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 Cai and Vasconcelos (2019): Cascade R-CNN is similar to Faster R-CNN, but the architecture of the ROI head (the module that predicts the bounding boxes and the category label) is different. Cascade R-CNN builds up a cascade head based on Faster R-CNN Ren et al. (2015) to refine detection progressively. Since the proposal boxes are refined by multiple box regression heads, Cascade R-CNN is optimal for more precise localization of objects.

- Deformable Convolution Zhu et al. (2019): Here, the basic architecture of the network is the same as the one in Faster R-CNN. However, we replace the convolution layer with a deformable convolution layer Zhu et al. (2018) 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 so that it enables 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.

- Prime Sample Attention Cao et al. (2020) (PISA): Here, the basic network architecture is again the same as in

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

Faster R-CNN. PISA is motivated by two considerations: samples should not be treated as independent and equally important, and the classification and localization are correlated. Thus, it employs a ranking strategy 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, and we also flip the test 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) Bodla et al. (2017) with a confidence threshold of 0.5 as the post-processing of predicted bounding boxes.

4.1.5. Ensemble method

Generally, combining predictions from different models generalizes better and usually yields more accurate results compared to a single model. At the post-processing step of Faster R-CNNs, we employ soft NMS Bodla et al. (2017) 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 Solovyev et al. (2019). Unlike NMS-based methods that simply exclude part of the predicted bounding boxes, the Weighted Boxes Fusion algorithm uses confidence scores of all proposed bounding boxes to constructs the average boxes. The reader is referred to Solovyev et al. (2019) 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) Redmon et al. (2016) 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 Ren et al. (2015), YOLO provides faster detection.

Over time, improvements of YOLO were implemented and released as distinct and independent software packages by the originators Redmon et al. (2016); Redmon and Farhadi (2017, 2018). As an effect of increasing publicity and popularity, a model zoo containing further YOLO adaptations emerged. Subsequently, further maintainers continued to improve the DarkNet[3]-based versions, and Bochkovskiy et al. (2020) created ports for other machine learning libraries such as PyTorch[4] Paszke et al. (2019).

In this paper, two approaches are selected for DFU detection on 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 Redmon and Farhadi (2018) was developed as an improved version of YOLOv2 Redmon and Farhadi (2017). It employs multi-scale schema, predicting bounding boxes on different scales. This allows YOLOv3 to be more effective for detecting smaller targets when compared to YOLOv2.

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.

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_h &= p_w e^{t_w} \\ b_w &= p_h e^{t_h} \end{aligned} \tag{1}$$

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 $\hat{t_*}$ be the ground truth for some coordinate prediction, and t_* be the network prediction during training. Then, the gradient is $\hat{t_*} - t_*$, which can be easily computed by inverting equation 1.

Model Pipeline

The backbone of YOLOv3 is a hybrid model called Darknet-53 (as shown in Table 1), which is employed for the feature extraction part of the algorithm. 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 one type of object only, 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.

Training

We employ transfer learning by using the pre-trained DarkNet weights which are provided by Redmon and Farhadi (2018). Then, we train our detector in 2 steps, using the following settings: Adam optimizer with learning rate 1e-3, number

[3]DarkNet GitHub repository: `https://github.com/pjreddie/darknet` (accessed 2020-08-29)

[4]PyTorch website: `https://pytorch.org/` (accessed 2020-08-29)

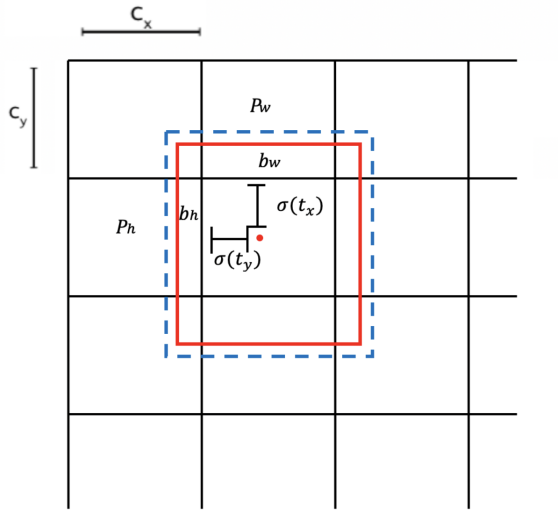

Fig. 2. Illustration of bounding boxes, dimension priors, and location prediction.

Table 1. The architecture of DarkNet-53 used in YOLOv3.

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	

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 for better 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 Lin et al. (2017). We then use the trained network for detecting diabetic foot ulcers in blind test images.

Post-processing

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

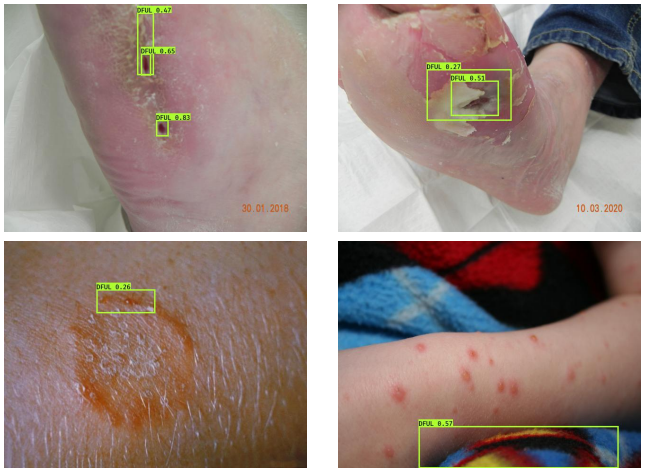

Fig. 3. Illustration of two types of false positives: (top row) false positives from double detection; and (bottom row) false positives of the network.

Our post-processing steps consist of two stages. First, we identify double detection by flagging the detected bounding boxes with more that 80% overlap. Among the overlapping detected boxes we only keep the one with highest confidence. Finally, we further post-process the results by removing any detection with confidence under 0.3, aiming to reduce the rate of false positive detections.

4.2.2. YOLOv5

YOLOv5 Jocher et al. (2020b) was first published in May 2020 by Glenn Jocher of Ultralytics LLC[5] on GitHub[6]. Originally, it was an improved version of their well known YOLOv3 implementation for PyTorch[7] Jocher et al. (2020a), based on the original YOLOv3 Redmon and Farhadi (2018). However, due to the release of YOLOv4 Bochkovskiy et al. (2020) for the DarkNet framework[8], which incorporated many improvements made in the PyTorch YOLOv3 implementation, the authors decided to name it YOLOv5 to avoid naming conflicts. Essentially, YOLOv5 can be labeled as "YOLOv4 for PyTorch". Unlike the original YOLOv3 and YOLOv4, there has not been a scientific paper published on the PyTorch port and its improvements yet. YOLOv5 is under active development with new features and releases appearing on a weekly basis. At the time of writing, the latest release is v3.0, published on 20 August 2020.

The new features and improvements in YOLOv4/YOLOv5 are mainly focused around incorporating state-of-the-art tech-

[5]Ultralytics LLC website: `https://www.ultralytics.com/` (accessed 2020-08-29)

[6]YOLOv5 GitHub repository: `https://github.com/ultralytics/yolov5/` (accessed 2020-08-29)

[7]Ultralytics' YOLOv3 GitHub repository: `https://github.com/ultralytics/yolov3` (accessed 2020-08-29)

[8]YOLOv4 GitHub repository: `https://github.com/AlexeyAB/darknet` (accessed 2020-08-29)

niques for activation functions, data augmentation, and post-processing into the established YOLO architecture to achieve the best possible object detection performance. One of the most notable new features is the novel mosaic loader data augmentation. Four images are combined to form a new image, allowing detection of objects outside of their normal context and at smaller sizes, and reducing the need for large mini-batch sizes. Another new data augmentation technique is self-adversarial training (SAT), where images are generated to deceive the network. YOLOv5 claims accelerated inference and smaller model files compared to YOLOv4, allowing easy translation to mobile use cases.

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

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] R Core Team (2020) in version `4.0.1` and the Simple Features for R (sf) package[13] Pebesma (2018) in version `0.9-2`. Originally, none of the duplicate pair images showed BBox intersections by themselves. After joining duplicate pair annotations, several intersections were detected with a maximum of two involved BBox. These represented different annotations of the same wound in two duplicate images, now joint in one image. To resolve these, each intersections of a BBox_1 and a BBox_2 were merged into $\widehat{\mathrm{BBox}}$ by using their outer boundaries, as shown in Equation 2.

$$\widehat{\mathrm{BBox}} \begin{cases} \widehat{\mathrm{xmin}} &= \min(\mathrm{xmin}_1, \mathrm{xmin}_2) \\ \widehat{\mathrm{ymin}} &= \min(\mathrm{ymin}_1, \mathrm{ymin}_2) \\ \widehat{\mathrm{xmax}} &= \max(\mathrm{xmax}_1, \mathrm{xmax}_2) \\ \widehat{\mathrm{ymax}} &= \max(\mathrm{ymax}_1, \mathrm{ymax}_2) \end{cases} \tag{2}$$

The applied duplicate cleansing and annotation merging strategy resulted in $n = 1961$ images with $k = 2453$ annotations in the cleansed training dataset. Boundaries of merged BBoxes were checked for consistency. Afterwards, annotation data was converted to the image 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 Buades et al. (2005) for color images, utilizing the Python language[14] in version `3.6.9` with the OpenCV on 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 pixel) and `searchWindowSize` = 21 (search window size in pixel).

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 Figure 4.

Fig. 4. Effects of the non-local means (NLM) algorithm are shown for two example images (a) and (e) of 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).

Data Augmentation

As mentioned in the introduction of YOLOv5, it is basically a port of YOLOv4 for PyTorch, adapting the novelties of YOLOv4. Hence, in the following, these novelties are explained but ascribed to YOLOv4. Nonetheless, the described techniques also apply for YOLOv5.

A key factor in the improved performance of YOLOv4 over YOLOv3 is data augmentation, where additional training data is artificially generated by manipulating or combining existing training images to improve the robustness of the trained model. In Bochkovskiy et al. (2020), these techniques are referred to as "bag of freebies", meaning that they can be applied at training time and do not affect inference speed.

A first set of data augmentation techniques in YOLOv4 are pixel-wise adjustments including photometric distortion. This involves adjustments of brightness, contrast, hue, saturation, and noise of images as well as geometric distortion, consisting of random scaling, cropping, flipping, and rotating. A second set of techniques tackles the problem of object occlusion.

[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)

[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)

Here, techniques like random erase or CutOut DeVries and Taylor (2017) select a rectangle in the image to be filled with random or zero values. Hide-and-seek and grid mask work similarly, but select several regions. Similar techniques can also be applied to the feature maps, where the techniques DropOut Srivastava et al. (2014), DropConnect Wan et al. (2013), and DropBlock Ghiasi et al. (2018) randomly drop certain values during propagation to improve model robustness. Third, there are techniques that combine several images of the training set into one image. MixUp Zhang et al. (2017a) superimposes two images by multiplying their pixel values with a coefficient and doing the same for the labels. CutMix Yun et al. (2019) takes one image and covers a random rectangle with a region of another image, adjusting the labels according to the size of the region.

Two novel data augmentation techniques, Mosaic and Self-Adversarial Training (SAT), were introduced in YOLOv4. Mosaic augmentation is similar to CutMix but takes four images instead of two. These are placed in the four corners of the 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. SAT generates deceiving images based on the response of the model to given images. Its goal is to get the model to not detect a previously detected object, and then adjust the network weights based on this new image. Mosaic data augmentation was disabled in this approach because it led to invalid bounding boxes (BBoxes).

Lastly, class label smoothing is applied to improve model robustness. Additional smoothing is based on relationships between categories, modelled through a label refinement network.

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 used. Its architecture is displayed in Table 2, derived from YOLOv5's model export[16].

The YOLOv5x model uses a two-stage detector that consists of a Cross Stage Partial Network (CSPNet) Wang et al. (2020) backbone trained on MS COCO Lin et al. (2014), and a model head using a Path Aggregation Network (PANet) Liu et al. (2018) for instance segmentation. Each BottleneckCSP unit consists of two convolutional layers with 1×1 and 3×3 filters. The backbone incorporates a Spatial Pyramid Pooling network (SSP) He et al. (2015), which allows for dynamic input image size and is robust against object deformations.

Training

The infrastructure used for training comprised a single NVIDIA® V100[17] tensor core graphical grocessing 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 originally 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 Koitka and Friedrich (2017). 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 training stages

Table 2. Architecture of the YOLOv5x model.

Type		Filters	Size
Backbone			
Focus		12	3×3
Convolutional		160	3×3
BottleneckCSP	4×	160	$1 \times 1 + 3 \times 3$
Convolutional		320	3×3
BottleneckCSP	12×	320	$1 \times 1 + 3 \times 3$
Convolutional		640	3×3
BottleneckCSP	12×	640	$1 \times 1 + 3 \times 3$
Convolutional		1280	3×3
SPP			
BottleneckCSP	4×	1280	$1 \times 1 + 3 \times 3$
Head			
Convolutional		640	1×1
Upsample		2	
BottleneckCSP	4×	640	$1 \times 1 + 3 \times 3$
Convolutional		320	1×1
Upsample		2	
BottleneckCSP	4× x	320	$1 \times 1 + 3 \times 3$
Convolutional		320	3×3
BottleneckCSP	4×	640	$1 \times 1 + 3 \times 3$
Convolutional		640	3×3
BottleneckCSP	4×	1280	$1 \times 1 + 3 \times 3$
Detection			

[16]YOLOv5 model export: `https://github.com/ultralytics/yolov5/issues/251` (accessed 2020-09-21)

[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)

used the default set of hyperparameters: `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 BBox positioning.

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 dataset, using the best-performing epoch 58 automatically saved by YOLOv5. The resulting extended training dataset held 4161 images of which 3963 held 4638 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: One after an additional 30 epochs, another one after an additional 40 epochs, and a final one after an additional 60 epochs of self-training (referred to as `E60_SELF90`, `E60_SELF100`, and `E60_SELF120`).

Post-processing

The minimum confidence threshold for detection was set to 0.70, so only quite 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 each. Another experiment was based on the training state `E60_SELF100` involving the built-in test-time augmentation (TTA) and non-maxima suppression (NMS) features of YOLOv5 for inference.

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; these predictions for the image 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 one BB. The intersection over union (IoU) threshold was set low to IoU $\geq$ 0.30, as in case of multiple wounds in an image usually a distinct spatial demarcation was given. Thus, the risk of interfering detections of different wounds was low.

4.3. EfficientDet

The EfficientDet architecture Tan (2019) is an object detection network created by the Google Brain team, and utilises the EfficientNet ConvNet Tan and Le (2019) 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 class and generate bounding boxes using bounding box regression. EfficientDet also utilises compound scaling, which allows all parts of the network to scale in accordance to the target hardware being used for training and inference Tan et al. (2020). An overview of the EfficientDet architecture is shown in Fig. 5.

4.3.1. Pre-processing

Since the dataset was captured with different types of camera devices and lighting conditions, a color constancy algorithm, Shades of Gray (SoG), was used to handle variations in noise and lighting from the different capture devices hua Ng et al. (2019). Examples of pre-processed DFU images using SoG are shown in Fig. 6.

4.3.2. Data Augmentation

Data Augmentation techniques have proven to be an important tool in improving the performance of deep learning algorithms for various computer vision tasks Goyal and Yap (2018); Yap et al. (2020a). 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. 7.

4.3.3. Model

EfficientDet algorithms achieved state-of-the-art accuracy on the popular MS-COCO Lin et al. (2014) 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 640x480, we selected to use the EfficientDet-D1 pre-trained weights for DFU detection Goyal and Hassanpour (2020).

4.3.4. 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 and early stopping to select the final model for inference.

4.3.5. Post-processing

We further refined the EfficientDet architectures with a score threshold of 0.5 and removed overlapping bounding boxes to

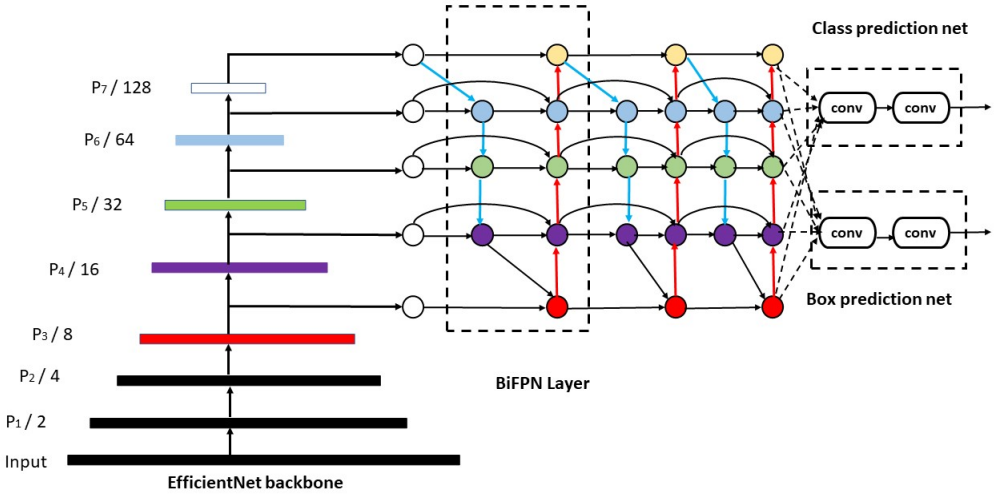

Fig. 5. (The architecture of EfficientDet, redrawn from Tan et al. (2020)

Original Image After Pre-Processing

Fig. 6. Shades of gray algorithm for pre-processing of DFUC2020 dataset: (To the left) original images; and (Right) pre-processed images.

minimize the number of false positives. The scores were compared between the overlapping bounding boxes, and the bounding box with the highest score was used as the final output.

4.4. Cascade Attention DetNet

4.4.1. Data Augmentation

Since the DFUC2020 dataset has only 2000 images for training, we use two 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 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 Zhang et al. (2017b). The original purpose of this method is to overcome the problem of disturbance rejection. Since Zhang et al. Zhang et al. (2019) introduced this method into object detection, many researchers have used it in data augmentation to enhance network robustness. The principle of this algorithm can be described as follows: We randomly select two sample images, and then generate a new sample image according to the mixup method of equation 3 and equation 4.

$$\hat{x} = \lambda x_i + (1 - \lambda)x_j \tag{3}$$

$$\hat{y} = \lambda y_i + (1 - \lambda)y_j \tag{4}$$

where (x_i, y_i), (x_j, y_j) are the points of two sample images, $\lambda \in [0,1]$, which is randomly generated by Beta(alpha, alpha) distribution. The new sample $(\hat{x}, \hat{y})$ is used for training. As shown in Fig. 8, two images of DFU are mixed in a certain ratio. We use Beta(1.5,1.5) for the images' synthesis.

Moreover, it is unsatisfactory to detect the DFU in a complex environment. To improve the ability for detection, we use the mobile fuzzy method for data augmentation. As shown in Fig. 9, we blur every image with the mobile fuzzy method to increase the number of images in the training set.

4.4.2. Model

The Cascade R-CNN Cai and Vasconcelos (2017) 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 Zhao et al. (2020). We use the cascade structure in conjunction with DetNet Li et al. (2018), 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 the 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. 10.

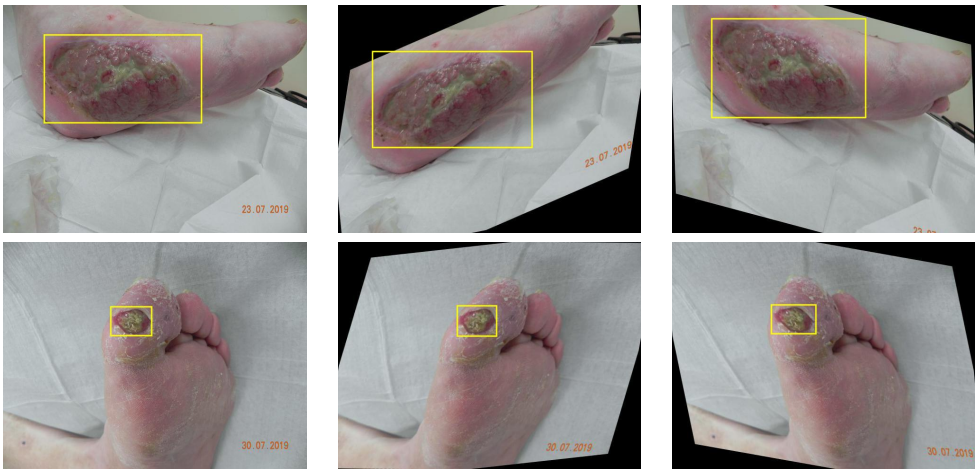

Fig. 7. Bounding box data augmentation on DFUC2020 dataset

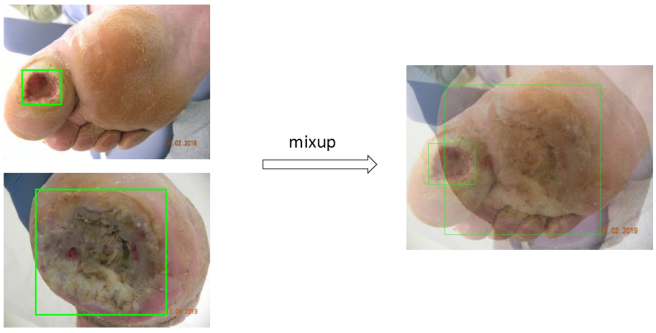

Fig. 8. The effect of visually coherent image mixup method.

Fig. 9. The effect of mobile fuzzy method. (a) is the original image, and (b) is the image after blurring with the mobile fuzzy method.

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, so we add an attention mechanism into the DetNet. To this end, we adopt the mask branch of the Residual Attention Network Wang et al. (2017).

The Attention DetNet (A-DetNet) is composed of six stages. The first stage consists of a 7*7 convolution (with stride 2) layer and a max-pooling layer. The second, third and fourth stages contain an A-Resbody, and the fifth and sixth stages contain 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 that we add an attention branch into the Resbody and Detbody. The attention branch is like 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 made up 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. Because down-sampling five times is not able to make the feature map (20×15) recover the original size by up-sampling, we only add one zoom structure into the attention branch of the A-Detbody. The feature map from the trunk will be 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.

4.4.3. Training

For the cascade structure, we set the total number of the cascade stages to 3. Considering the intersect over union (IOU) threshold, we set it to 0.5, 0.6 and 0.7 for each of the three stages. During training, we use a pre-trained model to accelerate model convergence. We use the pre-trained model of DetNet, which has been trained on the ImageNet dataset. We train on one GPU (NVIDIA Tesla P100) for 60 epochs, with a batch size of 4 and a learning rate of 0.001. The learning rate de-

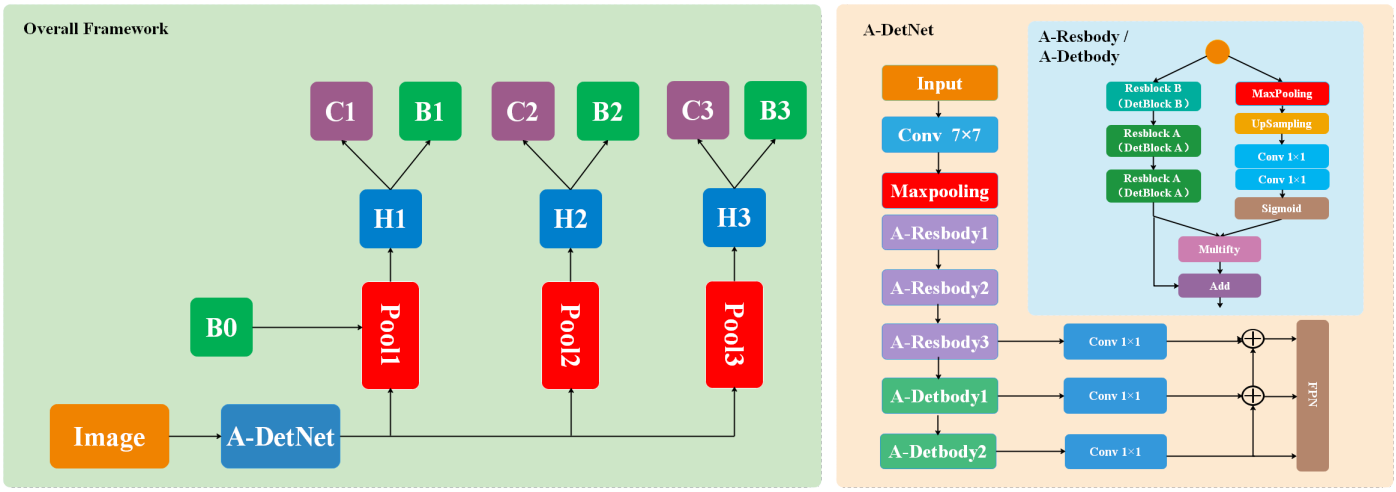

Fig. 10. The framework of CA-DetNet. "Image" is input image. "A-DetNet" is backbone network. "Pool" is region-wise feature extraction. "H" is network head. "B" is bounding box and "C" is classification. "B0" is proposals in all architectures. The structure of the A-DetNet is based on the DetNet.The attention mechanism is applied in Resbody and Detbody. Different bottleneck blocks in the Detbody or Resbody are similar to those in the DetNet.

creases 10 times at the 10th epoch, and then decreases another 10 times at the 20th epoch. We optimize the model with the Adam optimizer.

4.4.4. Post-processing

Noise from the external environment will 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 so that we suppress bounding boxes with low thresholds except when the detector detects only one bounding box. We set the threshold to 0.5.

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 mean average precision (mAP), as described in the diabetic foot ulcer challenge 2020 Cassidy et al. (2020). For the common object detection task, mAP is used as the main evaluation metric. However, in this DFU task, miss-detection (a false negative) potentially has severe implications as it may affect the quality of life of patients, and an incorrect detection (a false positive) could increase the financial burden on health services. Therefore, F1-Score is as important as mAP for performance evaluation.

5.1. Faster R-CNN

Table 3 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, but TP is also reduced. 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 mAP of 0.6940 and F1-Score of 0.7434.

Table 3. Faster R-CNN. The first column shows the results of pure Faster R-CNN, the second column shows the results of Cascade R-CNN, the third column shows the results of Faster R-CNN with Deformable Convolution v2, the fourth column shows the results of Faster R-CNN with Prime Sample Attention, and the last column shows the results of the ensemble method.

Methods	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

The qualitative results of Faster R-CNN with Deformable Convolution is summarized in Figure 11. It can be seen that our model successfully detected the defects in the image, even though the defects are small (bottom-right image) or the images are blurred (top-middle image). However, we observed the miss-detection as in the top-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 4 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

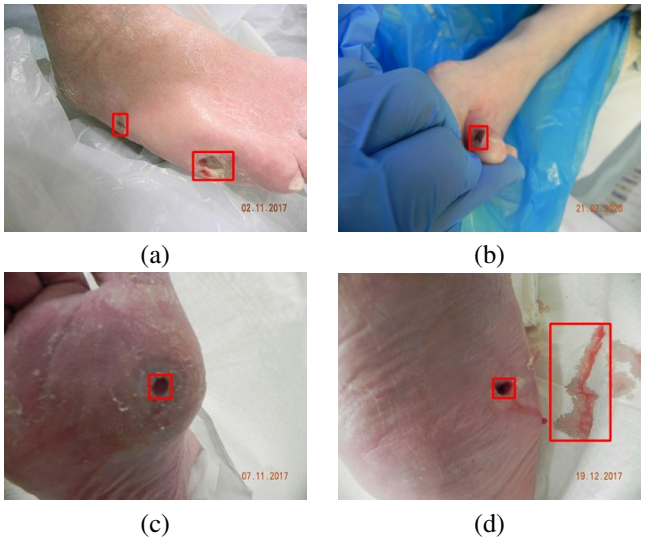

Fig. 11. The qualitative results of Faster R-CNN with Deformable Convolution, which shows the best performance among Faster R-CNN based methods. It is noted that the network is able to detect small ulcers as shown in (a),(b) and (c). However, it generates FP as demonstrated in (d).

also demonstrates that removing the overlaps leads to 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 in slightly better precision, it reduces recall, F1-score and mAP. Therefore, unless the precision is the priority, removing the low confidence detection would not lead to improvement. Examples of final detections for YOLOv3 are presented in Figure 12.

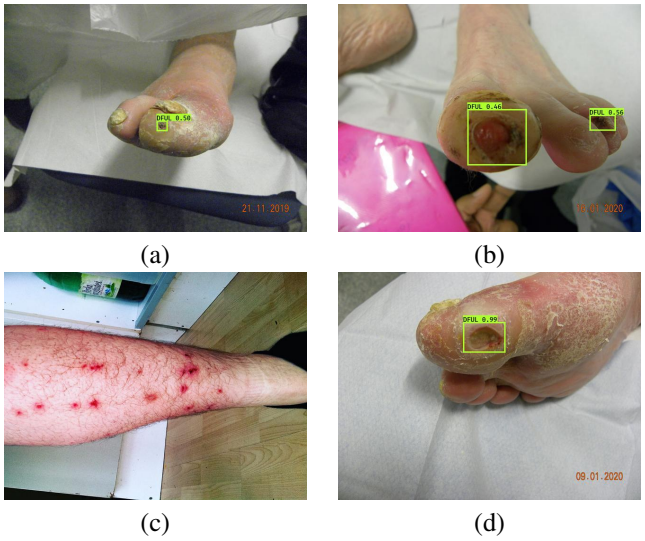

Fig. 12. Examples of final detection output of trained YOLOv3, after post-processing.

Additionally, we have added 60 copyright free images of healthy feet[21] to the training set to observe the effect on detection performance. As the results show, the above action results in improvement in F1-Score, but reduces the mAP.

5.3. YOLOv5

Table 5 summarizes the results of YOLOv5. Fewer additional self-training epochs in method E60_SELF90 achieved better results than E60_SELF100 and E60_SELF120. Yet, application of TTA with NMS on E60_SELF100 achieved the best results in E60_SELF100_TTA_NMS. Examples for detections of E60_SELF100_TTA_NMS on the test set are shown in Figure 13, Figure 14 shows additional examples of false negative and false positive cases.

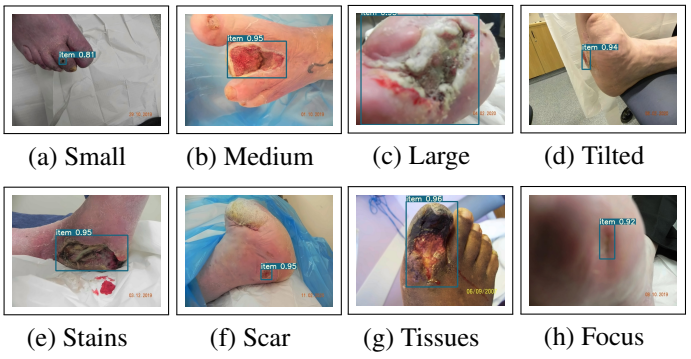

Fig. 13. Examples for adequate predictions of YOLOv5 for different DFU sizes and compositions: (a) to (c) different wound sizes, (d) highly tilted wound, (e) ignored blood stain on dressing, (f) ignored scar and hyperkeratosis, (g) heterogeneous wound composition, (h) detected wound out of focus.

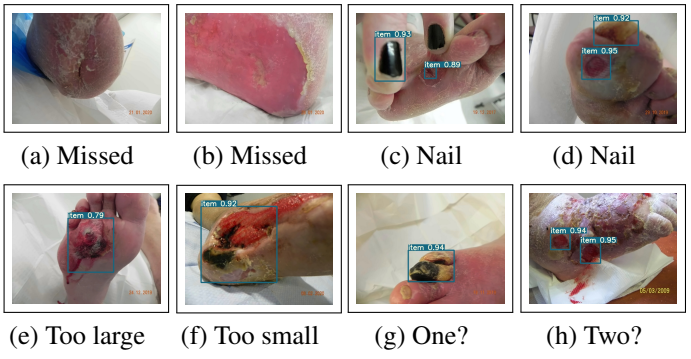

Fig. 14. Examples for false negative, false positive, inadequate, and questionable predictions of YOLOv5: (a) and (b) missed wounds, (c) and (d) painted finger nail and malformed toe nail, (e) and (f) too large and too small, (g) and (h) unclear detections (one, two, many?).

5.4. EfficientDet

Table 6 shows the results of EfficientDet on the DFUC2020 testing set both with and without post-processing. As the results indicated, the number of both TP and FP cases are reduced with the post-processing method. But, with post-processing method,

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

Table 4. 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 with 32. OverlapRemoved: indicates the performance of the method, with overlap removal post processing. conf0.3: shows the impact of ignoring any prediction 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

Table 5. 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-maxima suppression (NMS). Best results are highlighted bold, the winning method is highlighted gray.

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_SELF120	60	60	×	1502	478	0.7163	0.7586	0.7368	0.6201
E60_SELF100_TTA_NMS	60	40	✓	**1507**	498	**0.7187**	0.7516	0.7348	**0.6294**

Table 6. 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

the percentage of TP cases (from 1626 to 1593) is 2.02% compared to FP cases (from 720 to 594) is 17.50%. Hence, post-processing method lead to important improvement in both Precision (67.86% to 72.84%) and F1-score (72.38% to 74.37%), while the slightly decrease in both mAP (57.82% to 56.94%) and Recall (77.44% to 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. 15.

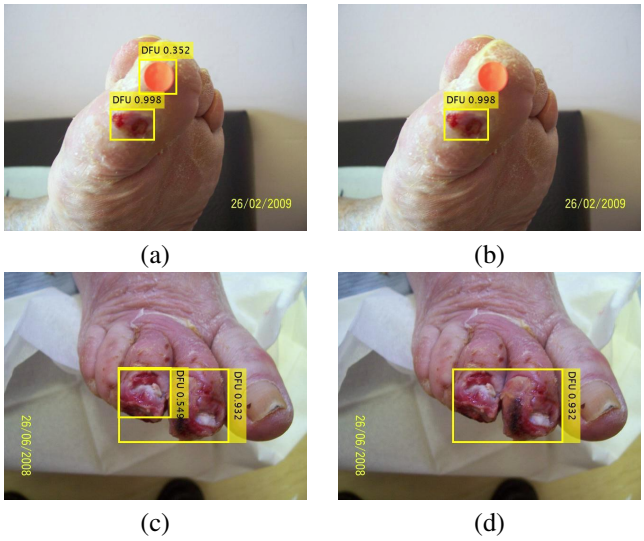

Fig. 15. The results of EfficientDet. (a) and (c) are the results of EfficientDet without post-processing; (b) and (d) are the results obtained with post-processing.

5.5. Cascade Attention DetNet

Table 7 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 see that the mobile fuzzy data augmentation method brings about a striking effect and improves 1.46% on mAP and 1.03% on F1-Score. At the same time, using the single mixup method in data augmentation did not enhance the performance. The results suggest that the mobile fuzzy method can make the model 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

Table 7. Cascade Attention DetNet.

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**

by 0.03%. Moreover, training with a pre-trained model can accelerate the convergence of the model and improve the ability to detect DFU.

Our approach was effective for the vast majority of the detected cases, as shown in Fig. 16. However, due to the complex clinical environment, there are also some failure cases in our approach. From our observation, 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 on the location of toenails. It is not easy to overcome this problem. For the interference from objects present in the external environment, we believe that the background can sometimes interfere with detection. 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 if a two-stage architecture whose first stage is to detect and segment the relevant area of feet is designed, the above problems could be solved. However, more labeled data may be required be achieve this goal.

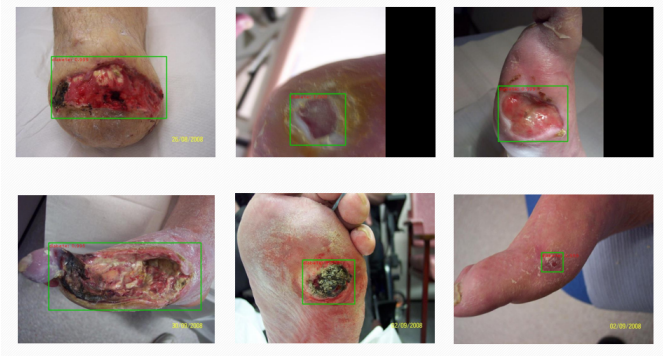

Fig. 16. The results of CA-DetNet: Illustration of successful DFUs detection.

5.6. Comparison

The results from the popular deep learning object detection methods and the proposed CA-DetNet are comparable. Table 8 shows the overall result when evaluated on DFUC2020 testing set, where we present the best mAP from each object detection method. Considering the ranking based on mAP, the best result

Table 8. 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 9. 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

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 9, the ranking according to F1-Score shows the highest F1-Score of 0.7437 obtained by EfficientDet, however, the mAP is only 0.5694. On the other hand, the Faster R-CNN approach achieves a comparable F1-Score of 0.7434 with a much higher mAP of 0.6940.

Fig. 17 visually compare the detection results on DFUs with less visible appearances. In Fig. 17(a), the ulcer was detected by all the methods. However, in Fig. 17(b), only Faster R-CNN and EfficientDet detected the ulcer. Fig. 17(c) is another challenging case and it was detected by CA-DetNet and Faster R-CNN. In Fig. 17(d), we demonstrate a case where only Faster R-CNN successfully localise the ulcer.

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 run some experiments based on combinations of two approaches (Faster R-CNN + (CA-DetNet / EfficientDet / YOLOv3 / YOLOv5)), three approaches and a combination of all approaches, as summarised in Table 10.

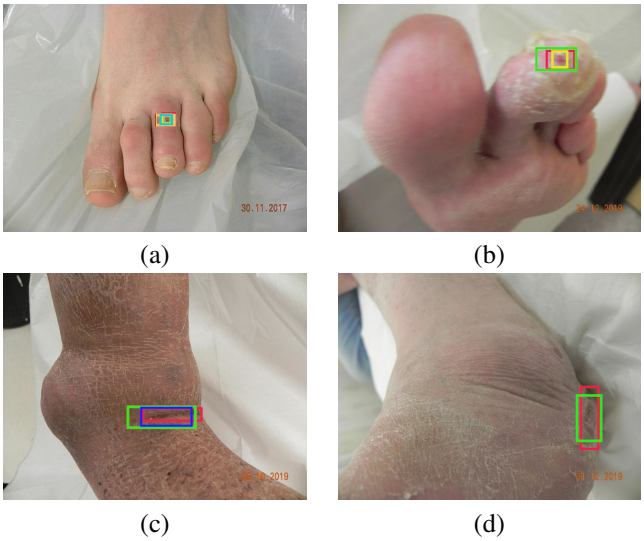

Fig. 17. Visual comparisons of object detection methods when compared to the ground truth (in red): (a) An easy case where all the methods detected the ulcer; (b) A more challenging case detected by Faster R-CNN (green) and EfficientDet (yellow); (c) A challenging case detected by Faster R-CNN (green) and CA-DetNet (blue); and (d) A challenging case only detected by Faster R-CNN (green).

Table 10. A Comparison of ensemble methods with different combinations of object detection framework, where FRCNN is Faster R-CNN, DetNet is CA-DetNet, EffDet is EfficientDet, and 'ALL methods' represent 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

From our observation, the ensemble methods reduce the numbers of TPs and FPs, i.e., the more approaches, the lower the numbers of TPs and FPs. It did not improve the mAP, but in the majority of the ensembles, there are notable improvement in precision, hence led to 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.

Apart from fine-tuning each deep learning method to achieve maximum 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 of greater than 70%, there is much work to do 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-Score. In addition, the variants of Faster R-CNN largely improve the performance of the original Faster R-CNN. After ensemble the results of four models, we managed to reduce the number of false positives, but the overall performance when compared to 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 Zhou et al. (2019) could potentially 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 removing low confidence detections, leads to 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 scheme.

The YOLOv5 approach also demonstrated a 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 notably increase precision further. Improvements by applied duplicate cleansing and BBox merging were marginal due to the limited number of cases, but could prove beneficial on larger datasets. Application of TTA with NMS helped to further increase true-positives at the cost of increased false positive detection, yet increased mAP and F1-Score.

However, the presented results may not be representative in regards to YOLOv5's actual capabilities. Surprisingly, the least self-trained model performed best, indicating optimization potential in the configurations considered. Models with fewer self-training epochs may perform better. In addition, an early version (v1.0) of the network was applied during the DFUC2020, whereby the Mosaic data augmentation was not functioning correctly on custom data. At the time of writing, the more developed version v3.0[22] is available, featuring numerous improvements and bug fixes. E.g., the activation function was changed from Leaky ReLU Maas et al. (2013) in versions v1.0 (used here) and v2.0 to hard-swish Howard et al. (2017), further increasing detection performance.

YOLOv5 is improving rapidly and its full potential could not be taken advantage of during the DFUC2020. E.g., Model Ensembling[23] could allow further performance increases when

[22] YOLOv5 v3.0: `https://github.com/ultralytics/yolov5/releases/tag/v3.0` (accessed 2020-09-28)

[23] YOLOv5 GitHub repository tutorial on Model Ensembling: `https://github.com/ultralytics/yolov5/issues/318` (accessed 2020-09-28)

fusing differently specialized models as well as investigation of Hyperparameter Evolution[24]. Hence, YOLOv5 could prove helpful for performing DFU detection tasks particularly when considering implementation directly on mobile devices.

The refined EfficientDet algorithm is able to detect DFU with a high recall rate. The pre-processing stage with the Shades of Gray algorithm improved the consistency of the images. We extensively used the data augmentation techniques to learn the subtle features of DFUs of various sizes and severity. The post-processing stage has refined the inference of the original EfficientDet method by removing overlapping bounding boxes. Due to low mAP, further work will focus on investigating other options of EfficentDet, particularly EfficientDet-D7.

The performance of Cascade Attention DetNet on the DFUC2020 testing set is not entirely satisfactory. We evaluated our model on 10% of the DFUC2020 training set and it achieved 0.9 on mAP. We analyzed the possible reasons and consider 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 fusion of different backbones have reduced the number of predicted bounding boxes substantially. Faster R-CNN with Deformable Convolution predicted 2240 bounding boxes, but after ensembled with EfficientDet, it only predicted 1847 bounding boxes. The predicted bounding boxes was dropping to 1475 when we ensemble the results from all the five networks. Consequently, the ensemble methods have 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 aspect to overcome this issue is to understand the threshold setting of IOU. Our experiment is using IOU $\geq$ 0.5, which is the guideline set by object detection for natural objects. However, in medical imaging studies Drukker et al. (2002); Yap et al. (2008), they used the IOU (or Jaccard Similarity Index) threshold of 0.4. When we evaluate the performance of the best ensemble method, the number of TPs increases to 1594, and with IOU $\geq$ 0.3, the number of TP increases to 1668. With Faster R-CNN with Deformable Convolution, the number of TPs increases to 1743 and 1883 for IOU threshold of 0.4 and 0.3, respectively.

7. Conclusion

We conduct a comprehensive evaluation of the performance of deep learning object detection networks for DFU detection. While the overall results show the potential of automatically localising the ulcers, there are many false positives, and the networks struggle to discriminate ulcers from other skin conditions. A possible solution to address this issue might be to introduce a second classifier in the form of a negative dataset to train future networks on. 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 complex environments. Future research challenges include:

- Gather a larger-scale dataset with clinical annotations. This is the best solution for supervised machine learning algorithms. However, in the real-world, there are still barriers in data sharing. Additionally, clinical annotation is expensive and time consuming. It is important to encourage co-creation by machine learning and clinical experts to foster better understanding of the annotated data.

- Create self-supervised and unsupervised deep learning algorithms for DFU detection. These methods were developed and implemented for natural object detection tasks and remain under-explored in medical imaging.

- For inspections of DFU, accurate delineation of an ulcer and its surrounding skin can help to measure the progress of the ulcer. Goyal et al. Goyal et al. (2017) developed an automated segmentation algorithm for DFU. However, they experimented on a small dataset only and 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. The next challenge (DFUC2021 Yap et al. (2020b)) will focus on multi-class DFU recognition.

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

Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation for the use of GPUs for this challenge and 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.

References

Armstrong, D.G., Boulton, A.J., Bus, S.A., 2017. Diabetic foot ulcers and their recurrence. New England Journal of Medicine 376, 2367–2375.

Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M., 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. URL: https://arxiv.org/abs/2004.10934, arXiv:2004.10934.

Bodla, N., Singh, B., Chellappa, R., Davis, L.S., 2017. Soft-nms–improving object detection with one line of code, in: Proceedings of the IEEE international conference on computer vision, pp. 5561–5569.

Brown, R., Ploderer, B., Da Seng, L.S., Lazzarini, P., van Netten, J., 2017. 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, pp. 462–466.

[24] YOLOv5 GitHub repository tutorial on Hyperparameter Evolution: https://github.com/ultralytics/yolov5/issues/607 (accessed 2020-09-28)

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

Cai, Z., Vasconcelos, N., 2017. Cascade r-cnn: Delving into high quality object detection .

Cai, Z., Vasconcelos, N., 2019. Cascade r-cnn: High quality object detection and instance segmentation. arXiv:1906.09756.

Cao, Y., Chen, K., Loy, C.C., Lin, D., 2020. Prime sample attention in object detection, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11580–11588.

Cassidy, B., Reeves, N.D., Joseph, P., Gillespie, D., O'Shea, C., Rajbhandari, S., Maiya, A.G., Frank, E., Boulton, A., Armstrong, D., et al., 2020. Dfuc2020: Analysis towards diabetic foot ulcer detection. arXiv preprint arXiv:2004.11853 .

DeVries, T., Taylor, G.W., 2017. Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552 URL: https://arxiv.org/abs/1708.04552, arXiv:1708.04552. URL: https://arxiv.org/abs/1708.04552 (accessed on 11 September 2020).

Drukker, K., Giger, M.L., Horsch, K., Kupinski, M.A., Vyborny, C.J., Mendelson, E.B., 2002. Computerized lesion detection on breast ultrasound. Medical physics 29, 1438–1446.

Ghiasi, G., Lin, T.Y., Le, Q.V., 2018. DropBlock: A regularization method for convolutional networks, in: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 31. Curran Associates, Inc., pp. 10727–10737.

Goyal, M., Hassanpour, S., 2020. A refined deep learning architecture for diabetic foot ulcers detection. arXiv preprint arXiv:2007.07922 .

Goyal, M., Reeves, N.D., Davison, A.K., Rajbhandari, S., Spragg, J., Yap, M.H., 2018. Dfunet: convolutional neural networks for diabetic foot ulcer classification. IEEE Transactions on Emerging Topics in Computational Intelligence , 1–12doi:10.1109/TETCI.2018.2866254.

Goyal, M., Reeves, N.D., Rajbhandari, S., Ahmad, N., Wang, C., Yap, M.H., 2020. Recognition of ischaemia and infection in diabetic foot ulcers: Dataset and techniques. Computers in Biology and Medicine , 103616.

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

Goyal, M., Yap, M.H., 2018. Region of interest detection in dermoscopic images for natural data-augmentation. arXiv preprint arXiv:1807.10711 .

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

He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask r-cnn, in: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2980–2988.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 37, 1904–1916. URL: https://doi.org/10.1109/tpami.2015.2389824, doi:10.1109/tpami.2015.2389824.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 .

IDF, 2019. International diabetes federation: Facts & figures. https://www.idf.org/aboutdiabetes/what-is-diabetes/facts-figures.html.

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

Jocher, G., Stoken, A., Borovec, J., NanoCode012, ChristopherSTAN, Changyu, L., Laughing, Hogan, A., lorenzomammana, tkianai, yxNONG, AlexWang1900, Diaconu, L., Marc, wanghaoyang0106, ml5ah, Doug, Hatovix, Poznanski, J., Yu, L., changyu98, Rai, P., Ferriday, R., Sullivan, T., Xinyu, W., YuriRibeiro, Claramunt, E.R., hopesala, pritul dave, yzchen, 2020b. ultralytics/yolov5: v3.0. URL: https://doi.org/10.5281/zenodo.3983579, doi:10.5281/zenodo.3983579.

Koitka, S., Friedrich, C.M., 2017. Optimized convolutional neural network ensembles for medical subfigure classification, in: 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, pp. 57–68. doi:10.1007/978-3-319-65813-1_5.

Li, Z., Peng, C., Yu, G., Zhang, X., Sun, J., 2018. Detnet: A backbone network for object detection .

Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S., 2017. Feature pyramid networks for object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2117–2125.

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft coco: Common objects in context, in: European conference on computer vision, Springer. pp. 740–755.

Liu, S., Qi, L., Qin, H., Shi, J., Jia, J., 2018. Path aggregation network for instance segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 8759–8768.

Maas, A.L., Hannun, A.Y., Ng, A.Y., 2013. Rectifier nonlinearities improve neural network acoustic models, in: Proc. ICML, p. 3. URL: http://robotics.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf. URL: http://robotics.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf (accessed on 11 September 2020).

hua Ng, J., Goyal, M., Hewitt, B., Yap, M.H., 2019. 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. p. 109530R.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. Pytorch: An imperative style, high-performance deep learning library, in: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 32. Curran Associates, Inc., pp. 8024–8035.

Pebesma, E., 2018. Simple Features for R: Standardized Support for Spatial Vector Data. The R Journal 10, 439–446. doi:10.32614/RJ-2018-009.

R Core Team, 2020. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria. URL: https://www.R-project.org/.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You Only Look Once: Unified, Real-Time Object Detection, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE. URL: https://doi.org/10.1109/cvpr.2016.91, doi:10.1109/cvpr.2016.91.

Redmon, J., Farhadi, A., 2017. YOLO9000: Better, Faster, Stronger, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE. URL: https://doi.org/10.1109/cvpr.2017.690, doi:10.1109/cvpr.2017.690.

Redmon, J., Farhadi, A., 2018. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 .

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks, in: Advances in neural information processing systems, pp. 91–99.

Solovyev, R., Wang, W., Gabruseva, T., 2019. Weighted boxes fusion: ensembling boxes for object detection models. arXiv:1910.13302.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research 15, 1929–1958.

Tan, M., Le, Q., 2019. Efficientnet: Rethinking model scaling for convolutional neural networks .

Tan, M., Pang, R., Le, Q.V., 2020. Efficientdet: Scalable and efficient object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10781–10790.

Tan, Mingxing, P.R.V.L.Q., 2019. Efficientdet: Scalable and efficient object detection. https://arxiv.org/pdf/1911.09070 64, 2098–2109.

Wan, L., Zeiler, M., Zhang, S., Cun, Y.L., Fergus, R., 2013. Regularization of neural networks using dropconnect, PMLR, Atlanta, Georgia, USA. pp. 1058–1066. URL: http://proceedings.mlr.press/v28/wan13.html.

Wang, C.Y., Mark Liao, H.Y., Wu, Y.H., Chen, P.Y., Hsieh, J.W., Yeh, I.H., 2020. Cspnet: A new backbone that can enhance learning capability of cnn, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 390–391.

Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., Tang, X.,

2017. Residual attention network for image classification .

Wang, L., Pedersen, P.C., Agu, E., Strong, D.M., Tulu, B., 2016. Area determination of diabetic foot ulcer images using a cascaded two-stage svm-based classification. IEEE Transactions on Biomedical Engineering 64, 2098–2109.

Wang, L., Pedersen, P.C., Strong, D.M., Tulu, B., Agu, E., Ignotz, R., 2014. Smartphone-based wound assessment system for patients with diabetes. IEEE Transactions on Biomedical Engineering 62, 477–488.

Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K., 2017. Aggregated residual transformations for deep neural networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5987–5995.

Yap, M.H., Edirisinghe, E.A., Bez, H.E., 2008. A novel algorithm for initial lesion detection in ultrasound breast images. Journal of Applied Clinical Medical Physics 9, 181–199.

Yap, M.H., Goyal, M., Osman, F., Marti, R., Denton, E., Juette, A., Zwiggelaar, R., 2020a. Breast ultrasound region of interest detection and lesion localisation. Artificial Intelligence in Medicine , 101880.

Yap, M.H., Reeves, N., Boulton, A., Rajbhandari, S., Armstrong, D., Maiya, A.G., Najafi, B., Frank, E., Wu, J., 2020b. Diabetic foot ulcers grand challenge 2021. URL: https://doi.org/10.5281/zenodo.3715020, doi:10.5281/zenodo.3715020.

Yap, M.H., Reeves, N.D., Boulton, A., Rajbhandari, S., Armstrong, D., Maiya, A.G., Najafi, B., Frank, E., Wu, J., 2020c. Diabetic foot ulcers grand challenge 2020. doi:http://doi.org/10.5281/zenodo.3715016.

Yun, S., Han, D., Chun, S., Oh, S.J., Yoo, Y., Choe, J., 2019. CutMix: Regularization strategy to train strong classifiers with localizable features, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE. URL: https://doi.org/10.1109/iccv.2019.00612, doi:10.1109/iccv.2019.00612.

Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D., 2017a. mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 URL: https://arxiv.org/abs/1710.09412, arXiv:1710.09412. URL: https://arxiv.org/abs/1710.09412 (accessed on 11 September 2020).

Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D., 2017b. mixup: Beyond empirical risk minimization .

Zhang, Z., He, T., Zhang, H., Zhang, Z., Xie, J., Li, M., 2019. Bag of freebies for training object detection neural networks. arXiv preprint arXiv:1902.04103 .

Zhao, W., Huang, H., Li, D., Chen, F., Cheng, W., 2020. Pointer defect detection based on transfer learning and improved cascade-rcnn. Sensors 20, 4939.

Zhou, X., Wang, D., Krähenbühl, P., 2019. Objects as points, in: arXiv preprint arXiv:1904.07850.

Zhu, J., Fang, L., Ghamisi, P., 2018. Deformable convolutional neural networks for hyperspectral image classification. IEEE Geoscience and Remote Sensing Letters 15, 1254–1258.

Zhu, X., Hu, H., Lin, S., Dai, J., 2019. Deformable convnets v2: More deformable, better results, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 9300–9308.