UNSUPERVISED LEARNING - CLUSTERING

Kamanis Fotios (aid24010)
Kouti Aikaterini Maria (aid24007)

University of Macedonia
MSc in Artificial Intelligence and Data Analysis

December 2023

Contents

1 Dataset: Diabetic Foot Ulcer Challenge 2020 (DFUC 2020)

The DFUC 2020 dataset is a specialized collection of images for the development and evaluation of diabetic foot ulcer detection systems. This dataset plays a crucial role in addressing the global health concern of diabetic foot ulcers. The following details offer a comprehensive understanding of the dataset:

- **Composition**: The dataset includes 2,000 training images. Additionally, a separate test dataset, comprises 2,000 images. This diversity and volume contribute to the dataset's robustness for machine learning applications.
- **Annotations**: Each image in the DFUC 2020 training dataset is annotated with the precise location of foot ulcers. These annotations are provided as coordinates (xmin, ymin, xmax, ymax). Importantly, these annotations were verified by healthcare professionals specializing in diabetic foot care.
- **Variety in Cases**: The dataset encompasses a wide range of cases, including different stages of ulcer healing, various foot conditions, and diverse environmental factors such as lighting and angle. This variety ensures the development of algorithms that are effective in real-world, non-medical settings.
- **Evaluation Metrics**: The primary metric for assessing algorithm performance in the DFUC 2020 challenge is the F1-score, which is particularly relevant due to the critical importance of both false positives and false negatives in medical diagnosis. The F1-score provides a balanced measure of precision and recall, essential for applications where both types of errors have significant consequences. Other metrics such as precision, recall, specificity, and mean average precision (mAP) are also considered for a comprehensive evaluation of the models.
- **Global Health Significance**: The dataset's focus on diabetic foot ulcers addresses a significant health issue faced by millions globally. By facilitating the development of advanced detection systems, the DFUC 2020 dataset contributes to improved diagnosis and treatment strategies, potentially reducing the rate of complications and amputations associated with diabetic foot ulcers.

The DFUC 2020 dataset stands as a pivotal resource for the advancement of medical imaging analysis, particularly in the domain of diabetic foot ulcer detection. Its comprehensive nature and clinical relevance position it as a valuable tool for researchers and healthcare professionals alike.

2 Methodology

The methodology for our study on the DFU_2020 dataset involves strategic dataset utilization and model evaluation. Our approach is outlined as follows:

1. **Dataset Splitting**: The DFU_2020 dataset, consisting of 2,000 training images with annotations, is meticulously divided into two subsets:
   - *Training Set*: Comprising 80% of the dataset (1,600 images), this set is used for training our models. It includes a variety of images with annotations that define the precise location of diabetic foot ulcers.
   - *Validation Set*: Making up the remaining 20% of the dataset (400 images), the validation set is crucial for evaluating model performance and fine-tuning. The annotations enable a comprehensive assessment of the models' capabilities in detecting ulcers accurately.
2. **Test Dataset Utilization**: The test dataset, containing 2,000 images without annotations, serves a unique purpose in our study. While it does not facilitate quantitative model evaluation, it plays a vital role in qualitative analysis:
   - The test set allows for the visualization of model predictions on unseen data. This enables us to explore the models' prediction capabilities in real-world scenarios.
   - By examining model outputs on the test set, we gain valuable insights into the models' performance, particularly in terms of their generalizability and reliability in detecting ulcers under diverse conditions.
3. **Evaluation Strategy**: Our primary focus lies in evaluating the models using the validation set. This approach enables us to:
   - Assess the models' precision, recall, and F1-score, thereby understanding their effectiveness in identifying true positives and minimizing false negatives and false positives.
   - Iterate and refine our models by fine-tuning parameters based on validation set performance, ensuring the development of robust and accurate ulcer detection systems.

This methodology underpins our research on the DFU_2020 dataset. Through strategic dataset splitting and a dual approach of quantitative and qualitative evaluation, we aim to develop and refine models that can effectively detect diabetic foot ulcers, contributing significantly to medical imaging analysis and patient care.

3 Annotations

Object detection tasks use various types of annotations to label and define the location and characteristics of objects within images. The main types of annotations include bounding boxes, masks, and keypoints, each with its own attributes and uses.

3.1 Bounding Boxes

- **Attributes:** Rectangular boxes defined by coordinates (x, y) for the top-left corner and (width, height) or two sets of coordinates representing opposite corners.
- **Use:** Suitable for tasks where the object's exact shape is less important than its location and scale. Widely used due to simplicity and effectiveness.
- **Differences:** Less precise in capturing the exact shape of objects compared to masks. Faster to create and require less computational resources.

3.2 Masks (Pixel-wise Annotations)

- **Attributes:** Detailed labeling of each pixel as part of the object (object mask) or background, resulting in pixel-wise segmentation.
- **Use:** Essential for tasks requiring high precision in object shape and boundary delineation, such as medical image analysis.
- **Differences:** More complex and time-consuming to create than bounding boxes. Provide a more accurate representation of object shape but require more computational power.

3.3 Keypoints

- **Attributes:** Specific locations within an object, marked as x, y coordinates, identifying notable parts of an object.
- **Use:** Crucial in applications like human pose estimation and facial feature recognition, where understanding the structure or pose of an object is important.
- **Differences:** Focus on accurately pinpointing specific parts of an object rather than providing information about the object's area or shape.

3.4 Annotations in the DFU 2020 Dataset

In our specific case, the DFU 2020 dataset contains only bounding box annotations and these are provided solely for the training set. This limits our ability to use models that require masks or keypoints for object detection. Therefore, our object detection methodologies are constrained to techniques that work within the scope of bounding box annotations.

4 Evaluation Metrics

4.1 Average Precision

AP combines both precision and recall into a single metric by analyzing the area under the precision-recall curve.

- **Precision-Recall Curve**: Plots precision (y-axis) against recall (x-axis) for various threshold levels, providing a comprehensive view of the model's performance.
- **Calculation of AP**:
  - AP is the area under the precision-recall curve.
  - It is calculated by taking the average of the maximum precision values at each recall level, ensuring the model is effective across various thresholds.
- **Significance of AP**:
  - AP reflects both the detection (recall) and the accuracy of these detections (precision).
  - It balances the trade-off between precision and recall, requiring a model to perform well in both aspects to achieve a high AP.

4.1.1 Average Precision Variants

- **AP / mAP_0.5:0.95 (Average Precision)**: Often a shorthand for $mAP_{0.5:0.95}$. Represents the mean of the Average Precision values calculated at different Intersection over Union (IoU) thresholds ranging from 0.5 to 0.95 (in steps of 0.05). This metric provides a comprehensive view of the model's performance across various levels of localization accuracy.
- **AP50 / mAP_0.5 (AP at IoU = 0.5)**: Essentially $mAP_{0.5}$. It measures the Average Precision at an IoU threshold of 0.5. It reflects the model's ability to correctly detect and reasonably localize objects with at least 50% overlap with the ground truth. It is less strict and often used in scenarios where moderate overlap is sufficient. It indicates how well the model performs in terms of both detection and localization at a moderate IoU threshold.
- **AP75 (AP at IoU = 0.75)**: Measures the Average Precision at a stricter IoU threshold of 0.75. It is more stringent than AP50 and shows how precisely the model can localize objects. Higher performance in this metric indicates better localization accuracy of the model.
- **APs (AP for small objects)**: The Average Precision for small objects in the dataset. It specifically measures how well the model performs on smaller-scale objects.
- **APm (AP for medium objects)**: Measures the Average Precision for medium-sized objects in the dataset. It's a measure of the model's performance on objects that are neither too small nor too large.
- **APl (AP for large objects)**: The Average Precision for large objects. Models sometimes perform better on larger objects as they are easier to detect and localize due to their size.
- **APs, APm, APl**: These metrics provide insights into how well the model performs across different object sizes. They are crucial for datasets with a wide range of object scales.

4.2 Precision and Recall

- **Precision**: Indicates the proportion of correctly identified objects among all the detections made by the model.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP) + False Positives (FP)}} \tag{1}$$

  High precision implies a low rate of false positives, crucial in applications like medical imaging to avoid unnecessary treatments.

- **Recall**: Measures the proportion of actual objects that were correctly identified by the model.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP) + False Negatives (FN)}} \tag{2}$$

  High recall is essential to ensure that most of the real objects are detected, reducing the risk of missing critical findings.

4.3 F1 Score

The F1 score is a commonly used metric in binary classification and object detection tasks, representing the harmonic mean of precision and recall. In the context of object detection using the COCO evaluation metrics, precision is represented by Average Precision (AP) and recall by Average Recall (AR).

For our model evaluation, we use the metrics AP and AR calculated over a range of Intersection over Union (IoU) thresholds (from 0.50 to 0.95) and across all area sizes. The specific metrics chosen for F1 score calculation are:

- AP @[IoU=0.50:0.95 — area= all — maxDets=100] = 0.067
- AR @[IoU=0.50:0.95 — area= all — maxDets=100] = 0.397

The F1 score is calculated using the formula:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

Substituting the values, we get:

$$F1 = 2 \times \frac{0.067 \times 0.397}{0.067 + 0.397} \tag{4}$$

The parameter `maxDets` in the COCO metrics stands for the maximum number of detections per image that are considered for evaluating the model. In this case, `maxDets=100` implies that the evaluation considers up to 100 detected objects per image. This parameter is crucial for assessing the model's performance in scenarios with varying object counts per image. A higher `maxDets` value allows the evaluation to capture the model's behavior in images with many objects, which is particularly relevant in dense object detection tasks.

4.4 Summary

Mean Average Precision (mAP) is an absolute metric that is not a function of confidence threshold, so for this reason it is the default metrics for comparison in object detection models. Mean Average Precision (mAP) is a prevalent metric for evaluating object detection models due to several reasons:

1. **Combines Precision and Recall:** mAP accounts for both precision and recall by computing the area under the precision-recall curve, reflecting the model's accuracy and coverage.
2. **Threshold Agnostic:** It is calculated over various Intersection over Union (IoU) thresholds, making it robust against different detection thresholds and capturing the model's performance across a range of acceptance criteria.
3. **Class Averages:** mAP averages the performance over all classes, which is crucial for multi-class detection tasks to ensure comprehensive performance across the board.
4. **Comparability:** As a standard metric, mAP enables the comparison of different models with varying configurations, architectures, or datasets, providing a common ground for evaluation.
5. **Balance Between Different Errors:** mAP takes into account both false positives and false negatives, offering a balanced view of a model's predictive performance.

While mAP is highly informative, it's not the only metric to consider. In specific contexts, the cost of false positives or the importance of not missing a positive instance may necessitate a focus on precision or recall, respectively. The F1 score, which provides the harmonic mean of precision and recall, is also useful, particularly in scenarios with class imbalances.

4.4.1 Task-Specific Metric Considerations

Depending on the specific application requirements, metrics like precision, recall, and the F1 score may also be crucial:

- **Precision:** Paramount in applications where false positives carry high costs.
- **Recall:** Critical when it is vital not to miss any positive detections.
- **F1 Score:** Useful as a single metric to balance precision and recall, especially in datasets with class imbalances.

In conclusion, while mAP provides a solid foundation for model comparison, it should be accompanied by additional metrics aligned with the particular needs of the task. Benchmarks such as COCO and PASCAL VOC have standardized the use of AP for these reasons, with variations like AP50, AP75, and APs/APm/APl providing insights into model performance across different object scales and localization accuracies.

5 YOLOv5

5.1 Introduction

This report presents an analysis of a YOLOv5 model trained for the task of ulcer detection in medical images. The model's performance was evaluated over 25, 50, and 100 epochs to assess the progression of learning and identify any signs of overfitting.

5.2 Methodology

The model was trained using a dataset comprising images annotated with the presence of ulcers. A confusion matrix was used to visualize the model's performance, and various loss components were tracked to monitor learning progression.

5.3 Training Overview

The YOLOv5 model was trained using a dataset split into training and validation subsets. The model's loss components, precision, recall, and mean average precision (mAP) were recorded. These metrics serve as indicators of the model's ability to accurately identify and localize ulcers within the images.

5.4 Results

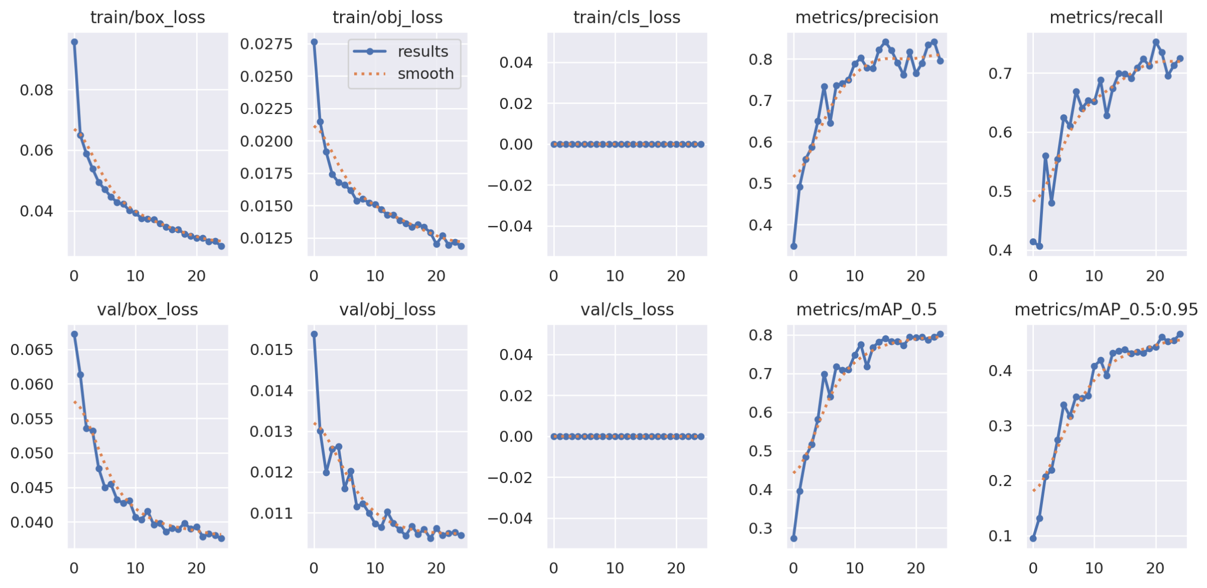

Figure 1: Results Over 25 Epochs

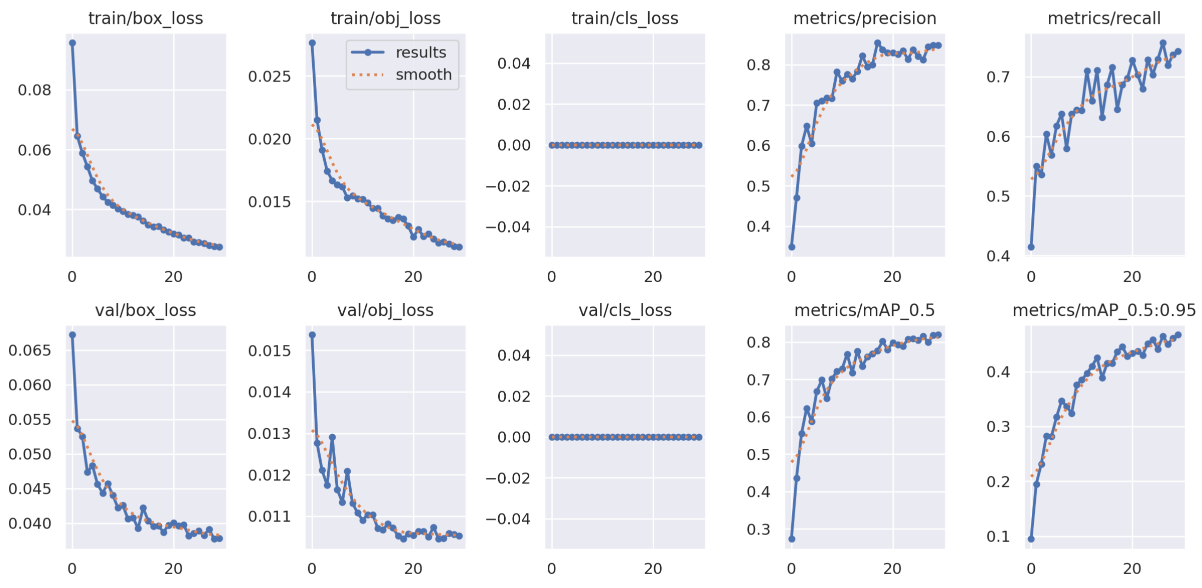

Figure 2: Results Over 30 Epochs

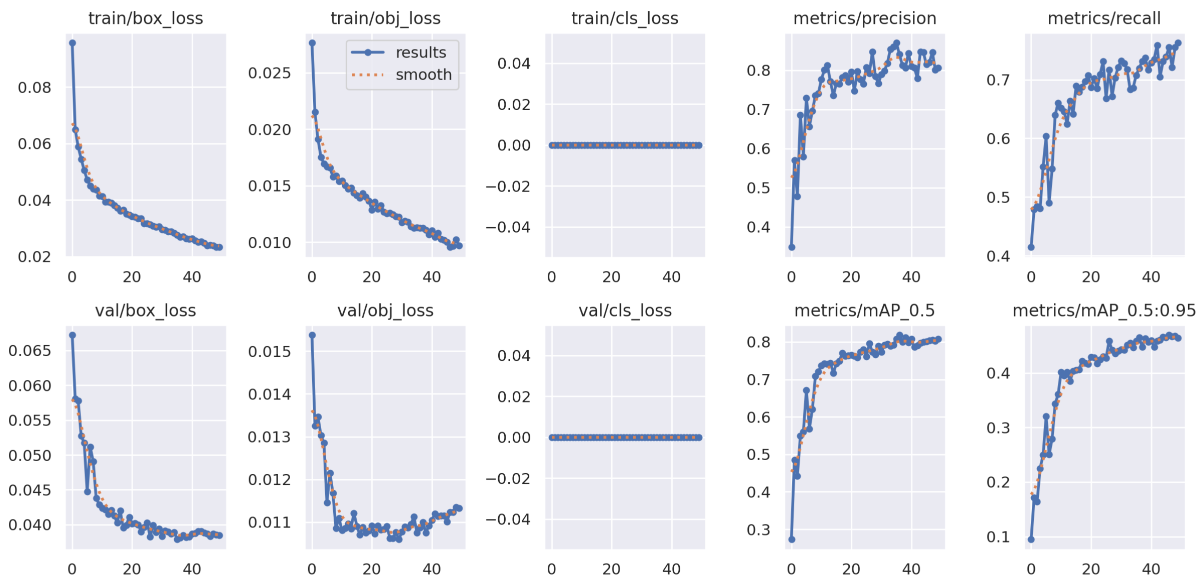

Figure 3: Results Over 50 Epochs

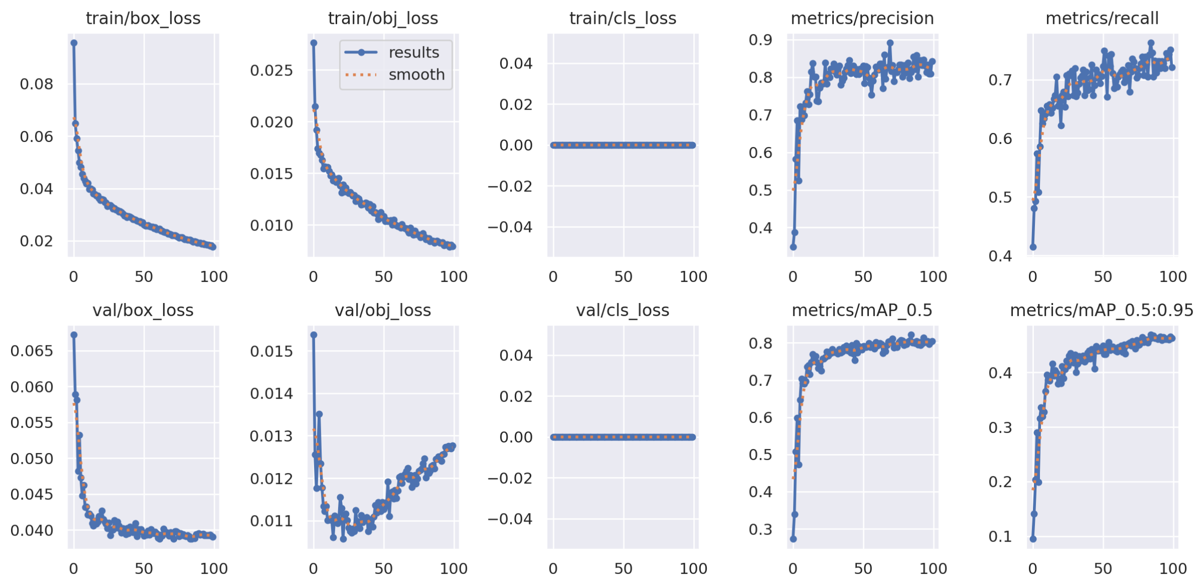

Figure 4: Results Over 100 Epochs

5.4.1 Objective Loss

The validation objective loss (val/obj_loss) was observed to decrease sharply during the initial epochs, indicating rapid learning. However, after approximately 30-40 epochs, an increase in loss was noted, suggesting a potential overfitting scenario where the model learns noise and dataset-specific features that do not generalize well to unseen data.

5.4.2 Precision and Recall

The precision and recall metrics exhibited an improvement over the epochs, with precision increasing notably. This improvement is a positive indicator of the model's capability to accurately predict the presence of ulcers while minimizing false positives.

5.4.3 Mean Average Precision

The mAP scores at IoU thresholds of 0.5 and the averaged interval between 0.5 to 0.95 displayed an overall upward trend, signifying enhanced model detection capabilities over time. The results after 50 epochs showed a better balance between precision and recall, with no significant improvement observed after 100 epochs.

5.5 Discussion

The analysis of the model's training process over 25, 50, and 100 epochs suggests that while the model's performance in terms of mAP improved, the increase in validation loss after 30-40 epochs indicates that the model may have begun overfitting to the training data. This is corroborated by the fact that there was no significant gain in performance metrics after 50 epochs.

5.6 Overfitting

The observed increase in validation loss after an initial decrease is a classical sign of overfitting. The model's increasing complexity might be capturing noise and dataset-specific details, thereby losing its generalization capabilities.

5.7 Recommendations

Based on the observed trends, it is recommended to implement early stopping around 30 epochs to prevent overfitting. Additionally, employing techniques such as learning rate scheduling, regularization, and data augmentation could further enhance model performance and generalizability.

5.8 Conclusion

The YOLOv5 model demonstrated substantial learning capability for ulcer detection. However, the training process revealed a tendency for the model to overfit after a certain number of epochs. Future work should focus on optimizing the number of training epochs and implementing strategies to mitigate overfitting.

6 Detectron2

Detectron2 is a software system for object detection and segmentation that is built on top of PyTorch. While it is not PyTorch itself, it is a library developed by Facebook AI Research (FAIR) that extends PyTorch's capabilities for specific tasks related to object detection and segmentation.

Detectron2 is an open-source software system created by Facebook AI Research (FAIR) that implements state-of-the-art object detection algorithms. It's a complete rewrite of the original Detectron, developed by FAIR, and is built on top of PyTorch, a popular machine learning framework. Below are some of the key features and aspects of Detectron2:

- **Object Detection Framework:** Detectron2 is primarily used for object detection, which involves identifying and classifying objects within an image. It also performs tasks like instance segmentation, delineating object boundaries, and panoptic segmentation, combining instance and semantic segmentation.
- **PyTorch Integration:** Built on PyTorch, Detectron2 takes advantage of PyTorch's dynamic computation graph for flexibility and efficiency in research and experimentation.
- **State-of-the-Art Models:** It includes implementations of several cutting-edge models for object detection and segmentation, such as Faster R-CNN, Mask R-CNN, and RetinaNet.
- **Extensibility and Customization:** Detectron2 is designed to be easily extendable, allowing for experimentation with new ideas in object detection and segmentation.
- **Pre-trained Models:** A range of pre-trained models trained on datasets like COCO (Common Objects in Context) are available. These can be used for inference or fine-tuned on custom datasets.
- **Efficient and Scalable:** Optimized for performance, Detectron2 can scale across different hardware configurations, suitable for both research and production deployments.
- **Active Community and Support:** As an open-source project, it benefits from a large community of developers and researchers who contribute to its continual improvement and expansion.

Detectron2 is a powerful and versatile toolkit for anyone working in the field of computer vision, especially in object detection and segmentation. Its integration with PyTorch and support for state-of-the-art models make it a popular choice for both academic research and industrial applications.

6.1 Based on PyTorch

Detectron2 leverages PyTorch's features for tensor operations, neural network layers, and optimization techniques. It operates within the PyTorch ecosystem, making it familiar to those already versed in PyTorch.

6.2 Specialization

The library is specifically designed for object detection, instance segmentation, keypoint detection, and panoptic segmentation. It includes state-of-the-art models such as Faster R-CNN, Mask R-CNN, RetinaNet, among others.

6.3 Model Zoo and Pre-trained Models

Detectron2 provides a model zoo with pre-trained models ready for inference or fine-tuning. These models, trained on standard datasets like COCO, can be used as starting points for various computer vision tasks.

6.4 Extension and Customization

The library allows for easy customization and extension. Users can define their own models, modify existing ones, create custom datasets, and utilize various utilities and hooks provided by Detectron2.

6.5 Community and Ecosystem

Being part of the PyTorch ecosystem, Detectron2 benefits from the large community support and a wide range of complementary tools and libraries available within the PyTorch framework.

In summary, Detectron2, built upon PyTorch, offers specialized functionalities for object detection and segmentation, aiding the development and deployment of advanced computer vision applications.

7 Faster R-CNN with ResNet-50 Backbone and Feature Pyramid Network (FPN) Pre-Trained on COCO (3x Schedule)

The configuration line in Detectron2:

```
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml")
```

specifies the use of a specific model in the object detection domain. This model, part of the Detectron2 model zoo, is characterized by the following features:

- **Base Architecture - Faster R-CNN:** A widely-used model for object detection, known for efficiently predicting object bounding boxes and class scores.
- **Backbone - ResNet-50:** The model utilizes a 50-layer Residual Network (ResNet-50) as its backbone. This network is acclaimed for its efficiency in feature extraction, which is fundamental in object detection tasks.
- **Enhancement - Feature Pyramid Network (FPN):** Incorporated with FPN, the model can effectively detect objects at various scales. FPN enhances the standard backbone by integrating a top-down architecture with lateral connections, creating rich, multi-scale feature maps.
- **Training Schedule - 3x:** The '3x' notation indicates an extended training period, roughly three times longer than the standard training duration. This extended training allows the model to better learn from the data, often resulting in enhanced performance.
- **Pre-Training - COCO Dataset:** The model is pre-trained on the COCO dataset, a comprehensive collection for object detection, segmentation, and captioning. This pre-training offers a robust foundational understanding of diverse object detection tasks.

This configuration is particularly well-suited for a broad range of object detection challenges, offering a commendable balance between detection accuracy and computational efficiency. The choice of ResNet-50 as a backbone, coupled with FPN, makes it adept at handling varied scales and complexities in object detection scenarios.

8 Why not dimensionality reduction?

Dimensionality reduction is a technique used to reduce the number of features or variables in a dataset. While it can be beneficial in some scenarios, especially when dealing with high-dimensional data, its application depends on the nature of your dataset and the specific requirements of your object detection task.

In the context of object detection on images of diabetic feet, dimensionality reduction might not be a typical preprocessing step. Image data is often represented in high-dimensional spaces, and techniques like convolutional neural networks (CNNs) are well-suited to handle such data directly.

Here are some considerations:

Convolutional Neural Networks (CNNs): CNNs are commonly used for image-related tasks, including object detection. They are designed to automatically learn hierarchical features from images, and their architecture inherently handles the high-dimensional nature of image data.

Transfer Learning: Consider using pre-trained models for object detection, such as those based on architectures like ResNet, VGG, or MobileNet. These models have already been trained on large image datasets, and you can fine-tune them on your specific dataset containing diabetic foot images.

Data Augmentation: Instead of reducing dimensionality, consider applying data augmentation techniques to artificially increase the size of your training dataset. Techniques like rotation, flipping, and zooming can help improve the generalization ability of your model.

Image Preprocessing: Focus on appropriate image preprocessing steps, such as resizing, normalization, and cropping. These steps are often more relevant for image data than traditional dimensionality reduction techniques.

Feature Engineering: While traditional dimensionality reduction techniques may not be directly applicable to image data, you can explore feature engineering techniques specific to your domain. For instance, you might extract texture features or use region-based features depending on the characteristics of diabetic foot ulcers.

In summary, for object detection on images, particularly with modern deep learning techniques, dimensionality reduction might not be the primary concern. Focus on using CNNs, transfer learning, data augmentation, and relevant image preprocessing techniques to effectively train a model for detecting ulcers in diabetic foot images.

9 Data Augmentation Decision for DFU_2020 Dataset

In the context of our object detection project using the DFU_2020 dataset, the decision not to employ data augmentation is based on careful considerations related to the dataset characteristics and task requirements. The following rationale outlines the key factors influencing this decision:

9.1 Dataset Size

The DFU_2020 dataset comprises 2000 training images, which, while not extremely small, provides a moderate-sized dataset for training an object detection model. The decision not to use data augmentation takes into account the reasonable size of the training set.

9.2 Dataset Variability

An analysis of the DFU_2020 dataset reveals that the images already encapsulate a diverse set of scenarios, encompassing various object orientations, lighting conditions, and other relevant factors. The inherent variability within the dataset contributes to a sufficient range of examples for the model to learn from, diminishing the immediate need for extensive data augmentation.

9.3 Task-specific Considerations

The complexity of the object detection task associated with the DFU_2020 dataset has been considered. If the objects of interest exhibit a wide range of poses, scales, and orientations in real-world scenarios, data augmentation becomes crucial for the model to generalize well. However, based on our assessment, the dataset adequately captures these variations without the need for additional augmentation.

9.4 Computational Resources

Consideration is given to the available computational resources and training time constraints. While data augmentation is a valuable technique for improving model robustness, it also introduces an increased computational load during training. In our case, the decision not to use data augmentation is aligned with the available resources and the desire to maintain a reasonable training time.

9.5 Conclusion

In conclusion, the decision not to employ data augmentation for the DFU_2020 dataset is grounded in a thoughtful evaluation of dataset characteristics, task requirements, and practical considerations. The inherent diversity within the dataset, combined with the moderate dataset size and computational constraints, collectively support the decision to proceed with training the object detection model without augmentation.